A Probabilistic Analysis of 2-Machine Flowshops

Amar Ramudhin * John J. Bartholdi, III
James M. Calvin John H. Vande Vate Gideon Weiss

1993; revised April 3, 2003

Abstract

We study a 2-machine flowshop in which all processing times are independently and identically distributed, with values known to the scheduler. We are able to describe in detail the expected behavior of the flowshop under optimal and under heuristic schedules. Our results suggest that minimizing makespan might be a superfluous objective: Scheduling the jobs randomly requires less information and yet is asymptotically optimal. Moreover, random schedules require significantly less intermediate storage between the machines.

Key words: flowshop, sequencing, scheduling, heuristic, probabilistic analysis

*Faculté des Sciences de l'Administration, Université Laval, Québec, Canada G1K 7P4. The other authors are at the School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Georgia 30332 USA.

1 Introduction

Let $J = \{1, \ldots, n\}$ represent a set of jobs to be processed through a 2-machine flowshop with unlimited intermediate storage. Each job requires processing on each of the two machines M_1 and M_2, in that order, and the processing times of job j are a_j on machine M_1 and b_j on M_2. We assume that all processing times are drawn independently from the same uniform distribution $U[l, u]$, so that the flowshop is balanced; as we point out later, the unbalanced case seems easier.

We also restrict attention to permutation schedules, in which the jobs are processed according to the same sequence on each machine. Again, this is not a significant restriction since it is well-known that some minimal makespan schedule is also a permutation schedule (Johnson, 1954).

When the processing times of all jobs are known in advance, a minimum makespan schedule is produced by Johnson's algorithm (Johnson, 1954): Job i precedes job j if and only if $\min\{a_i, b_j\} \leq \min\{a_j, b_i\}$. We will find it convenient to consider the algorithm in its equivalent, imperative form: Partition the jobs J into two groups; set J_1 consists of those jobs j for which $a_j < b_j$ (smaller processing times on M_1) and set J_2 consists of the remaining jobs; sequence the jobs in J_1 in non-decreasing order of a_j and those in J_2 in non-increasing order of b_j to give subsequences S_1^* and S_2^* respectively; append S_2^* to the end of S_1^* to get an optimal sequence S^*. The idea behind Johnson's algorithm is to fill the buffer between the two machines quickly so that the second machine is never starved. If idle time on M_2 is minimized, then the machines are working in parallel to the greatest extent possible and the makespan will be minimized.

We show that for large n, the expected minimum makespan is $n\mu + \sigma\sqrt{n/\pi} + o(\sqrt{n})$, where μ is the expected processing time and σ is the standard deviation of the processing time of one job on one machine. The expected total work content (processing time) of the jobs is $2n\mu$, so an optimally scheduled flowshop gives sufficient parallelism to reduce one of these factors of n to $\sqrt{n}$. However, the cost for this speedup is that intermediate storage (the buffer between

machines) must have capacity to hold $O(n)$ jobs.

In contrast, schedules with some randomness require less information and computation, but are asymptotically optimal with respect to makespan. Moreover, random schedules require significantly less intermediate storage capacity than Johnson's schedule. These properties are evident in simulations with as few as 20–30 jobs.

The random schedules we study are all approximations of Johnson's schedule. They include the following, listed in order of increasing randomness.

Partial Johnson's As in Johnson's algorithm, divide the jobs into J_1, those jobs that have shorter processing time on the first machine, and J_2, the remaining jobs. Now within J_1 schedule only the $\log n$ jobs with shortest processing times a_j as in Johnson's rule; the remaining jobs of J_1 follow in random order. Similarly, within J_2 schedule only the $\log n$ jobs with shortest processing times b_j last as in Johnson's rule; the remaining jobs of J_2 precede in random order.

Johnson's Partition Process all jobs of J_1 in random order and then process all jobs of J_2 in random order.

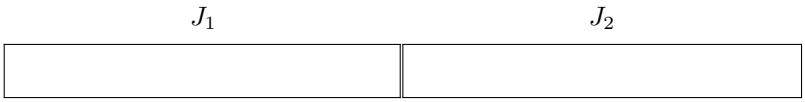
Random Process all jobs in random order.

Figure 1 illustrates the structures of the sequences constructed by each of these algorithms.

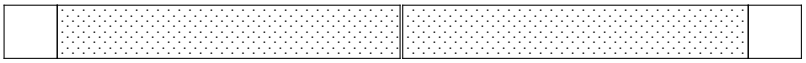
We show that as the sequence of jobs becomes increasingly random, the expected makespan increases negligibly, but the required intermediate storage decreases. Even scheduling jobs completely at random is asymptotically optimal and yet requires significantly less intermediate storage than Johnson's sequence ($\Theta(\sqrt{n})$ versus $\Theta(n)$).

This suggests that the randomized schedules might be practical alternatives to Johnson's schedule, especially when processing times are not known exactly.

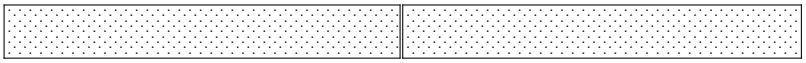
As a byproduct our work suggests an explanation for the puzzling computational results of Dannenbring (1977), wherein the performance of random



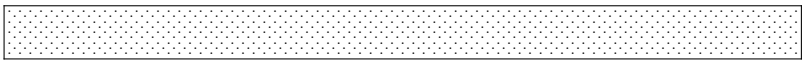
(a) Johnson's sequence



(b) Partial Johnson's



(c) Johnson's Partition



(d) Random

Figure 1: Structures of the job sequences constructed by the four algorithms. Shaded areas show where a sequence is random.

schedules improved as the number of jobs increased until, for the largest sets of jobs, scheduling at random was better than all but one of eleven competing heuristics. We believe that this is an artifact of the asymptotic optimality of random schedules, which we conjecture to hold even for permutation flowshops with arbitrarily many machines.

2 Related work

For deterministic flowshops a fast heuristic is known that produces asymptotically optimal permutation schedules: Barany (1981) gave a polynomial-time algorithm that produces a schedule whose makespan cannot exceed the minimum by more than a constant that depends on the largest processing time and the number of machines, but does not depend on the number of jobs. This is asymptotically optimal for large numbers of jobs, fixed number of machines, and bounded processing times.

Our work has somewhat the same flavor in that scheduling at random is asymptotically optimal when the tail of the probability density function from which processing times are drawn decreases sufficiently rapidly. We prove this here only for the special case of a 2-machine flowshop, and we have worked out the details only for the uniform distribution; but we indicate how this might be done for more general distributions.

Papadimitriou and Kanellakis (1980) gave a crisp explication of deterministic flowshop scheduling and observed that for 2-machine flowshops there are efficient algorithms to minimize the makespan when the intermediate storage capacity is either zero or unbounded; all intermediate cases (finite storage capacity) are NP-complete. They also suggested a heuristic to minimize makespan while observing limited intermediate storage. Our approach is in a sense dual to theirs in that we do not limit storage a priori, but instead determine how much will be required.

Our assumptions about the availability of information differ from those of Ku and Niu (1986). They studied a 2-machine flowshop in which all processing

times on each machine are independent random variables that are *not* known in advance. They gave conditions under which job i should precede job j if and only if $\mathrm{E}[\min\{a_i, b_j\}] \leq \mathrm{E}[\min\{a_j, b_i\}]$. To implement this scheduling rule apparently requires knowledge of the distibution means, so that presumably a historical record is necessary. To implement our scheduling rules does not require a historical record; depending on the rule, the information required is either the actual realizations a_i, b_i, or only which operation is more time-consuming, or even no information at all.

Pinedo (1982) derived some of the few results known on the expected makespan of m-machine flowshops, but unfortunately required highly restrictive assumptions on the processing times.

3 Johnson's schedule

In our model we may view Johnson's schedule as being generated in the following way. The processing times a_i and b_i of each successive job are independently drawn from $U[l, u]$. If $a_i \leq b_i$, then the job is inserted into the sequence S_1^* in accordance with the value of a_i. Otherwise, the job is inserted into the sequence S_2^* in accordance with the value of b_i. Then we assume that the sequence is converted into a schedule in the natural way: by requiring any idle machine to begin processing the next job as soon as it is available. We refer to such a schedule as a *busy schedule*. We assume this default implementation and so do not distinguish between sequences and schedules when it is clear from the context. When it is necessary to emphasize how the schedule is formed from a sequence, we write, for example, that the schedule $busy(S)$ is realized from sequence S.

Most of our effort is devoted to analyzing the first part of the schedule—that interval of time during which M_1 is working on jobs from S_1^*. Because the schedules we study are symmetric in a stochastic sense (or can be made so, as will be shown in Section 3.2), it will then be simple to determine corresponding results for the second half of the schedule and so for the entire schedule.

3.1 The first part of Johnson's schedule

Index the jobs according to their order of appearance in Johnson's sequence S^* so that $S_1^* = \{1, \ldots, N_1\}$ and $S_2^* = \{N_1 + 1, \ldots, n\}$, and let $N_2 = |S_2|$.

Note that N_1 is a random variable with binomial distribution with parameters n and $1/2$. In this section we abuse notation slightly to reduce clutter: We fix the value of N_1 to be n_1 and *all results of this section are conditioned on this.* Thus, for example, we write the expected processing times of the ith job in S_1^* as $(\mathrm{E}[a_i], \mathrm{E}[b_i])$ when in fact we mean $(\mathrm{E}[a_i |\, n_1], \mathrm{E}[b_i |\, n_1])$. Note that we also take the libertyof reindexing jobs according to the order of their appearance in the schedule under discussion.

Lemma 1. *For each job $i \in S_1^* = \{i : 1 \leq i \leq n_1\}$,*

$$\mathrm{E}[a_i] = u - \frac{n_1(n_1-1)\ldots(n_1-i+1)}{\left(n_1+\frac{1}{2}\right)\left(n_1-\frac{1}{2}\right)\ldots\left(n_1-i+\frac{3}{2}\right)}(u-l) \tag{1}$$

$$\mathrm{E}[b_i] = u - \frac{n_1(n_1-1)\ldots(n_1-i+1)}{2\left(n_1+\frac{1}{2}\right)\left(n_1-\frac{1}{2}\right)\ldots\left(n_1-i+\frac{3}{2}\right)}(u-l) \tag{2}$$

Proof. Since the jobs in S_1^* are sequenced in non-decreasing order of processing time on M_1, the distribution of the processing time on M_1 for the ith job in S_1^* is given by the ith order statistic of a set of n_1 random variables, each of which is of the form $\min\{x, y\}$, where x and y are independently sampled from $U[l, u]$. Thus, the conditional pdf of a_i is

$$f_{a_i}(\theta) = \frac{n_1!}{(i-1)!(n_1-i)!}\left[1 - F(\theta)\right]^{n_1-i} F(\theta)^{i-1} f(\theta),$$

where $f(\theta)$ is the pdf and $F(\theta)$ is the cdf of $\min\{x, y\}$. Since x and y are independently sampled from $U[l, u]$,

$$f(\theta) = 2\frac{u-\theta}{(u-l)^2} \text{ and } F(\theta) = 1 - \left(\frac{u-\theta}{u-l}\right)^2 \text{ for } l \leq \theta \leq u,$$

so that

$$f_{a_i}(\theta) = \frac{n_1!}{(i-1)!(n_1-i)!}\left(\frac{u-\theta}{u-l}\right)^{2(n_1-i)}\left[1 - \left(\frac{u-\theta}{u-l}\right)^2\right]^{i-1}\frac{2(u-\theta)}{(u-l)^2}.$$

Now we use $f_{a_i}(\theta)$ in the following.

$$\begin{aligned}
\mathrm{E}\left[\tfrac{u-a_i}{u-l}\right] &= \int_{\theta=l}^{u} \frac{u-\theta}{u-l} f_{a_i}(\theta)\, d\theta \\
&= i\binom{n_1}{i} \int_{\theta=l}^{u} \left(\frac{u-\theta}{u-l}\right)^{2(n_1-i+1)} \left[1-\left(\frac{u-\theta}{u-l}\right)^2\right]^{i-1} \left(\frac{2}{u-l}\right) d\theta
\end{aligned}$$

By substituting $-z = (u-\theta)/(u-l)$, where $0 \le -z \le 1$, we have

$$\mathrm{E}\left[\tfrac{u-a_i}{u-l}\right] = i\binom{n_1}{i} \int_0^1 \left(z^2\right)^{n_1-i+1} \left(1-z^2\right)^{i-1} 2\, dz. \tag{3}$$

Letting $t = z^2$ in (3) and appealing to the definition of the Beta Function we have

$$\begin{aligned}
\mathrm{E}\left[\tfrac{u-a_i}{u-l}\right] &= i\binom{n_1}{i} \frac{\Gamma(n_1-i+3/2)\,\Gamma(i)}{\Gamma(n_1+3/2)} \\
&= \frac{\Gamma(n_1+1)\,\Gamma(n_1-i+3/2)}{\Gamma(n_1-i+1)\,\Gamma(n_1+3/2)}.
\end{aligned}$$

The expected processing time a_i follows by invoking linearity of expectation.

The second claim holds since, given a_i, b_i is uniformly distributed between a_i and u. Thus $\mathrm{E}[b_i] = (u + \mathrm{E}[a_i])/2$. □

3.1.1 Idle time and makespan

Under a busy schedule, M_1 is never idle until it has completed processing all the jobs, and so the makespan in a 2-machine flowshop is entirely determined by the amount of idle time on M_2. Some relatively small amount of idle time on M_2 is unavoidable, however, since M_2 must wait for M_1 to complete the first job and so M_2 must be idle at least for the duration $a_1 \ge l$. More significantly, it is also possible that M_2 has idle time internal to its schedule; that is, M_2 might occasionally be idle in between the processing of successive jobs. Such idle time, which we call *internal* idle time, is a potentially significant delay to the completion of the set of jobs. Under Johnson's sequence, since each job in S_1^* has longer processing time on the second machine, one might expect little or no internal idle time on M_2 in this part of the schedule. In fact, we shall show that the probability of internal idle time on M_2 during processing of S_1^* tends

to zero as $n_1 \to \infty$. First, we need to show that the processing times of the first $\log n_1$ jobs on M_1 are very predictable. (Note: For notational convenience we will write $\log n_1$ instead of the more accurate $\lfloor \log n_1 \rfloor$ where the meaning is clear from the context.)

Lemma 2. *For* $1 \leq i \leq \log n_1$, $\lim_{n_1 \to \infty} \mathrm{Var}[a_i] = 0$.

Proof. We compute the variance of a_i as $\mathrm{Var}[a_i] = \mathrm{E}\big[a_i^2\big] - \mathrm{E}[a_i]^2$. From Lemma 1 we have an expression for $\mathrm{E}[a_i]$ and so derive an expression for $\mathrm{E}\big[a_i^2\big]$. First observe that

$$\mathrm{E}\left[\left(\tfrac{u-a_i}{u-l}\right)^2\right] = \frac{u^2 - 2u\mathrm{E}[a_i] + \mathrm{E}\big[a_i^2\big]}{(u-l)^2}$$

and therefore,

$$\mathrm{E}\big[a_i^2\big] = -u^2 + 2u\mathrm{E}[a_i] + \mathrm{E}\left[\left(\tfrac{u-a_i}{u-l}\right)^2\right](u-l)^2$$

The variance is now obtained from the following equation

$$\mathrm{Var}[a_i] = -u^2 + 2u\mathrm{E}[a_i] + \mathrm{E}\left[\left(\tfrac{u-a_i}{u-l}\right)^2\right](u-l)^2 - \mathrm{E}[a_i]^2 \tag{4}$$

where $\mathrm{E}\left[\left(\tfrac{u-a_i}{u-l}\right)^2\right]$ is given by

$$\mathrm{E}\left[\left(\tfrac{u-a_i}{u-l}\right)^2\right] = \frac{\Gamma(n_1+1)\,\Gamma(n_1-i+\frac{5}{2})}{\Gamma(n_1-i+1)\,\Gamma(n_1+\frac{5}{2})},$$

by algebra similar to the derivation in Lemma 1; and for $1 \leq i \leq \log n_1$,the latter term approaches 1 as $n_1 \to \infty$. Also, for $1 \leq i \leq \log n_1$, $\lim_{n_1 \to \infty} \mathrm{E}[a_i] = l$, so the claim holds. □

Let W_i be the workload, measured in work content, waiting for M_2 when job i completes processing on M_1. Then

$$W_i = \max_{k \leq i} \left\{ \sum_{j=1}^{k} a_j + \sum_{j=k}^{i} b_j \right\} - \sum_{j=1}^{i} a_j, \tag{5}$$

where the first term represents the time at which job i finishes at M_2 and the second term represents the time at which it finishes at M_1.

Also, W_i includes the workload in the buffer and the remaining processing time of the job currently being processed on M_2. Workload can be expressed recursively as

$$W_i = \max\{W_{i-1}, a_i\} + b_i - a_i, \text{ for } i \leq n_1. \tag{6}$$

Since $b_i \geq a_i$, W_i is an increasing function of i.

Let I_i be the internal idle time on M_2 prior to the completion of the ith job on M_2; then

$$W_i = b_1 + \sum_{j=2}^{i}(b_j - a_j) + I_i. \tag{7}$$

Now we show that the probability that there is internal idle time on M_2 during the first half of Johnson's schedule approaches zero as $n_1 \to \infty$. We establish this by first showing that the probability of internal idle time during the execution of the first $\log n_1$ jobs goes to zero as $n_1 \to \infty$. Then we observe that enough work is expected to have accumulated in the buffer by the time M_1 completes the $\log n_1$-th job that M_2 will never be idle during the remainder of S_1^*. Note that this latter result does not depend on how the remaining jobs are sequenced. Thus, for large n_1, the first half of any schedule that sequences the first $\log n_1$ jobs as Johnson's sequence is unlikely to have idle time on M_2.

Theorem 1. *The probability that the internal idle time, I_1^*, on M_2 in S_1^* is greater than zero approaches zero as $n_1 \to \infty$.*

Proof. Let $I_{1,1}^*$ be the internal idle time during the processing of the first $\log n_1$ jobs and let $I_{1,2}^*$ be the idle time during the processing of the remainder of S_1^*, so that $I_1^* = I_{1,1}^* + I_{1,2}^*$.

If $b_1 \geq a_{\log n_1}$ then there is no internal idle time on M_2 during the execution of the first $\log n_1$ jobs, so that $\Pr\big(I_{1,1}^* = 0\big) \geq \Pr(b_1 > a_{\log n_1})$. Moreover, as we now argue, $\lim_{n_1\to\infty} \Pr(b_1 > a_{\log n_1}) = 1$: By Lemma 2, $\mathrm{Var}[a_{\log n_1}] \to 0$ as $n_1 \to \infty$, while $\mathrm{E}[a_{\log n_1}] \to l$. Hence, using Chebyshev's Inequality, $a_{\log n_1} \xrightarrow{\mathrm{P}} l$ as $n_1 \to \infty$ (that is, it converges in probability).

We can now write $\Pr(b_1 > a_{\log n_1}) \geq \Pr(b_1 > l + \delta) - \Pr(a_{\log n_1} > l + \delta)$ for

all δ, and the right hand side can be made arbitrarily close to 1 by choosing δ appropriately and letting $n_1 \to \infty$. Thus as $n_1 \to \infty$, $\Pr\big(I^*_{1,1} = 0\big) \to 1$.

$W_{\log n_1}$ is the non-negative random variable representing the workload, as computed by (5), awaiting M_2 immediately after the $\log n_1$-th job completes processing on M_1. If $W_{\log n_1} > a_{\max}$ then M_2 cannot be idle during the rest of S^*_1 since M_1 will complete successive jobs before M_2 can empty the buffer. Thus

$$\Pr\big(I^*_{1,2} = 0\big) \geq \Pr(\, W_{\log n_1} > a_{\max}) \geq \Pr(\, W_{\log n_1} \geq u),$$

and the latter term approaches 1 as $n_1 \to \infty$, as we now argue.

$$\begin{aligned} \Pr(\, W_{\log n_1} \geq u) &= \Pr\Big(b_1 + \textstyle\sum_{j=2}^{\log n_1}(b_j - a_j) + I_{\log n_1} \geq u\Big) \\ &\geq \Pr\Big(b_1 + \textstyle\sum_{j=2}^{\log n_1}(b_j - a_j) \geq u\Big) \\ &\geq \Pr\Big(\textstyle\sum_{j=1}^{\log n_1}(b_j - a_j) \geq u\Big). \end{aligned}$$

For any k,

$$\sum_{j=1}^{k}(b_j - a_j) = \sum_{j=1}^{k}(u - a_j)Y_j,$$

where the Y_j are independent random variables, uniformly distributed between 0 and 1. The values of

$$(u - a_1),\ (u - a_2),\ \ldots, (u - a_k)$$

are an ordered sample from the uniform distribution on $(u - a_{k+1}, u - l)$. Each of these is stochastically larger than $Z_j \sim U(0, u - l)$. Hence $\sum_{j=1}^{k}(u - a_j)Y_j$ is stochastically larger than $\sum_{j=1}^{k} Z_j Y_j$, with $Z_j \sim U(0, u - l),\ Y_j \sim U(0, 1)$, all independent. Letting $X_j = Z_j Y_j$, we then have

$$\Pr(\, W_{\log n_1} \geq u) \geq \Pr\Big(\textstyle\sum_{j=1}^{\log n_1} X_j \geq u\Big) \to 1$$

as $n \to \infty$. Consequently $\lim_{n_1 \to \infty} \Pr\big(I^*_{1,2} = 0\big) = 1$.

Now

$$\Pr(I^*_1 = 0) \geq 1 - \Pr\big(I^*_{1,1} > 0\big) - \Pr\big(I^*_{1,2} > 0\big) = \Pr\big(I^*_{1,1} = 0\big) + \Pr\big(I^*_{1,2} = 0\big) - 1,$$

and the proof is completed by taking the limit as $n_1 \to \infty$. $\square$

The behavior predicted by this theorem seems robust. Simulation suggests that for n_1 as small as 20 there is negligible probability of idle time on M_2 during S_1^*.

Since Theorem 1 is the basis of all our subsequent results, it is worth remarking that it can be generalized to other processing time distributions. For example, let f be any processing time distribution and $k(n_1)$ any function that together satisfy the following weak conditions.

1. $a_{k(n_1)} - a_1 \xrightarrow{\mathrm{P}} 0$;
2. $\mathrm{E}[a_{n_1}]/k(n_1) \to 0$.

Then Theorem 1 holds by a similar argument applied to f with $a_{k(n_1)}$ replacing $a_{\log n_1}$.

Theorem 2. *The conditional expected makespan of S_1^* satisfies*

$$\mathrm{E}[\text{Makespan of } S_1^* | N_1 = n_1] = n_1 \frac{(l+2u)}{3} + O(1)$$

as $n_1 \to \infty$.

Proof. The conditional expected makespan is $\sum_{j=1}^{n_1} \mathrm{E}[b_j] + \mathrm{E}[a_1] + \mathrm{E}[I_1^*]$, but $\mathrm{E}[I_1^*] \to 0$ by the following argument. First note that in S_1^* the earliest M_2 can begin processing job i is when M_1 begins processing job $i+1$, and so M_2 is idle between processing jobs i and $i+1$ for at most $\max\{a_{i+1} - b_i, 0\}$. Thus, $I_1^* \le \sum_{i=1}^{n_1-1} \max\{a_{i+1} - b_i, 0\} \le \sum_{i=1}^{n_1-1} \max\{a_{i+1} - a_i, 0\} \le a_{n_1}$. It follows that $I_1^* \le a_{\max} \le u$, so that $\mathrm{E}[I_1^*] \le u\mathrm{Pr}(I_1^* > 0 \,|\, n_1)$. Now by Theorem 1 the latter term goes to zero as $n_1 \to \infty$. □

3.1.2 Workload, waiting times, and buffer level

We now derive bounds on the workload in the buffer and an expression for the expected waiting time of jobs in the buffer.

Theorem 3. *The conditional expected value of the maximum workload at M_2 during S_1^* occurs upon completion of the n_1-th job on M_1 and satisfies*

$$\lim_{n_1 \to \infty} \left\{ \mathrm{E}[\, W_{\max}] - \left(\frac{n_1(u-l)}{3} + \frac{u + 2n_1 l}{2n_1 + 1} \right) \right\} = 0$$

Proof. As $n_1 \to \infty$, $\mathrm{E}[I_i] \to 0$ and, by 7

$$\mathrm{E}[W_i] - \mathrm{E}[b_1] - \sum_{j=2}^{i} (\mathrm{E}[b_j] - \mathrm{E}[a_j]) \to 0.$$

Thus, as $n_1 \to \infty$

$$\mathrm{E}[W_{\max}] - \left(\frac{n_1(u-l)}{3} + u - \frac{n_1(u-l)}{n_1 + \frac{1}{2}} \right) \to 0,$$

since

$$\sum_{i=1}^{n_1} (\mathrm{E}[b_i] - \mathrm{E}[a_i]) + \mathrm{E}[a_1] = \frac{n_1(u-l)}{3} + u - \frac{n_1(u-l)}{n_1 + \frac{1}{2}}.$$

The last equation follows since, as may be proved by induction,

$$\sum_{i=1}^{n_1} \frac{n_1(n_1-1)\dots(n_1-i+1)}{\left(n_1+\frac{1}{2}\right)\left(n_1-\frac{1}{2}\right)\dots\left(n_1-i+\frac{3}{2}\right)} = \frac{2}{3} n_1.$$

□

Let $w_i i = W_i - b_i$ be the waiting time of job i in the buffer.

Theorem 4. *The expected average waiting time per job for jobs in S_1^* satisfies*

$$\lim_{n_1 \to \infty} \left\{ \sum_{i=1}^{n_1} \frac{\mathrm{E}[w_i]}{n_1} - (u-l) \left(\frac{3n_1 - 8}{15} \right) \right\} = 0.$$

Proof. Recall that $W_i = b_1 + \sum_{j=2}^{i} (b_j - a_j) + I_i$ where I_i is the internal idle time on M_2 prior to the completion of the ith job on M_2.

The waiting time of job i in the buffer, w_i, is $W_i - b_i$ and so for $1 \le i \le n_1$,

$$w_i = a_1 + \sum_{j=1}^{i} (b_j - a_j) + I_i - b_i.$$

The average waiting time for jobs in S_1^* is $\sum_{i=1}^{n_1} w_i / n_1$ and so its mean is given by

$$\begin{aligned}
\sum_{i=1}^{n_1} \mathrm{E}[w_i]/n_1 &= \frac{1}{n_1} \left(n_1 \mathrm{E}[a_1] + \sum_{j=1}^{n_1} \sum_{i=1}^{j} (\mathrm{E}[b_i] - \mathrm{E}[a_i]) - \sum_{i=1}^{n_1} \mathrm{E}[b_i] + \sum_{i=1}^{n_1} \mathrm{E}[I_i] \right) \\
&= \mathrm{E}[a_1] - \frac{1}{3}(l+2u) + \frac{1}{n_1} \sum_{j=1}^{n_1} (n_1 - j + 1) (\mathrm{E}[b_i] - \mathrm{E}[a_i]) \\
&\quad + \frac{1}{n_1} \sum_{i=1}^{n_1} \mathrm{E}[I_i] \\
&\to \mathrm{E}[a_1] - \frac{1}{3}(l+2u) + \frac{1}{2n_1} \frac{6n_1^2 + 4n_1}{15} (u-l),
\end{aligned}$$

where the last step holds since

$$\mathrm{E}[b_i] - \mathrm{E}[a_i] = \frac{n_1(n_1-1)\dots(n_1-i+1)}{2(n_1+\frac{1}{2})(n_1-\frac{1}{2})\dots(n_1-i+\frac{3}{2})}(u-l)$$

and, as may be proved by induction,

$$\sum_{j=1}^{n_1} \frac{n_1(n_1-1)\dots(n_1-j+1)^2}{(n_1+\frac{1}{2})(n_1-\frac{1}{2})\dots(n_1-j+\frac{3}{2})} = \frac{6n_1^2+4n_1}{15}.$$

Finally, since $\mathrm{E}[a_1] = u - (u-l)(2n_1)/(2n_1+1)$,

$$\sum_{i=1}^{n_1} w_i/n_1 = u - \frac{2n_1}{2n_1+1}(u-l) - \frac{1}{3}(l+2u) + \frac{3n_1+2}{15}(u-l)$$

and the result follows by simple algebra. □

In any case, the expected waiting time for jobs in S_1^* is no more than $(n_1+1)(u-l)/5$ for large n_1 and is greatest when $u \gg l$. When $u = l$ the waiting is zero, as one would expect.

Finally we derive the expected value of the number of the jobs in intermediate storage (the *buffer level*) for any time t during the partial schedule S_1^*. Let T be the completion time of the n_1-th job on M_2, and let w_T be the total waiting time of all jobs in S_1^*. Let $B(t)$ be the number of jobs in the buffer at time t.

Lemma 3. *The conditional average buffer level for S_1^* satisfies*

$$\lim_{n_1\to\infty} \left\{ \frac{1}{T}\int_{t=0}^{T} B(t)\,dt - \frac{(3n_1-8)(u-l)}{5(l+2u)} \right\} = 0. \tag{8}$$

Proof. Since $w_T = \sum_{j=1}^{n_1} w_j = \int_{t=0}^{T} B(t)\,dt$, the average buffer level for S_1^* is $w_T/T = (w_T/n_1)(n_1/T)$. By the law of large numbers, as $n_1 \to \infty$, $w_T/n_1 \to \sum_{i=1}^{n_1} w_i/n_1$, the average waiting time for jobs in S_1^*, and $T/n_1 \to (l+2u)/3$. □

3.2 The complete schedule

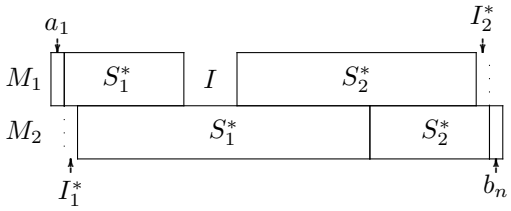
Much of the preceding analysis described the first part of the schedule S_1^* as $N_1 \to \infty$. We invoke these results to analyze complete schedules in which $n \to \infty$, which implies that $N_1, N_2 \to \infty$ almost surely. *Therefore most results for this section are doubly qualified: They hold for large n and almost surely.*

Figure 2: If S_1^* and S_2^* are processed separately, with the start times of the first jobs in each delayed without extending their makespans, then I_1^* is incurred at the beginning of S_1^* (on M_2), while I_2^* is incurred at the end of S_2^* (on M_1).

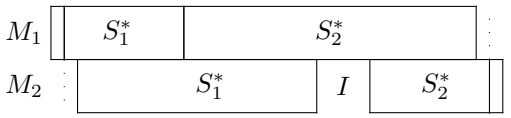
The remaining difficulty of analysis is that S_1^* and S_2^*, the first and second parts of Johnson's schedule, are not perfectly symmetric because idle time is more likely to occur in S_2^*. We circumvent this complication by analyzing an alternative implementation of Johnson's sequence that preserves properties like makespan and buffer size, but is more symmetrical (in a stochastic sense) and so easier to analyze.

Imagine the sequences S_1^* and S_2^* processed separately as in Figure 2, and let L_1 be the time from the completion of the last job of S_1^* on M_1 until its completion on M_2. Similarly, let L_2 be the time from the start of the first job of S_2^* on M_1 until its start on M_2. Consider the alternative implementation of Johnson's schedule suggested by Figure 3, in which the separate schedules S_1^* and S_2^* are joined without changing the positions of any jobs within a subschedule. This can be accomplished by modifying the busy rule to delay the starting time of the first job in S_1^* on M_2 until the first time that all jobs in S_1^* can be processed without internal idle time but also without extending the makespan beyond that of S_1^* under the busy implementation. Similarly, the first job of S_2^* on M_2 is delayed until the first time that all jobs in S_2^* can be processed without internal idle time but also without extending the makespan beyond that of S_1^* under the busy implementation. Therefore we refer to this alternative implementation as $delay(S^*)$ and to the busy implementation as $busy(S^*)$. Notice that the lengths of L_1 and L_2 are not affected by this new implementation.

Under $delay(S^*)$, the makespan of S_1^* is $a_1 + I_1^* + \sum_{i=1}^{n_1} b_i$ and that of S_2^* is $b_n + I_2^* + \sum_{i=1}^{n_2} a_i$ We have already shown that the probability of internal idle



Case 1: $L_1 > L_2$

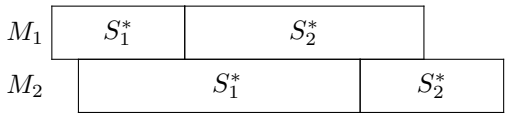


Case 2: $L_1 < L_2$

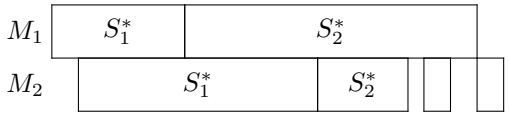
Figure 3: An alternative implementation of Johnson's schedule formed by joining the separate schedules for S_1^* and S_2^* with all jobs maintaining their relative positions within each subschedule. This implementation has the same makespan as the busy implementation, but tends to concentrate idle time interior to the schedule on a single machine.

time on M_2 within the first part of the schedule goes to zero as $n \to \infty$. By the same arguments with the order of machining reversed, the probability of internal idle time within the second part of the schedule goes to zero. Thus, for large n, the only internal idle time in Johnson's schedule is that due to the joining of S_1^* and S_2^*, which we call *intervening idle time*. As shown in Figure 3, if $L_1 > L_2$ the intervening idle time $I = L_1 - L_2$ occurs entirely on M_1; and if $L_1 < L_2$, then the intervening idle time $I = L_2 - L_1$ occurs entirely on M_2.

Note that we can imagine the busy implementation $busy(S^*)$ to be derived from $delay(S^*)$ by moving all the jobs in S_2^* to their earliest possible start times, while continuing to respect the sequence. As shown in Figure 4, if $L_1 > L_2$, then the intervening idle time disappears. On the other hand, if $L_1 < L_2$, then the intervening idle time becomes distributed throughout S_2^* on M_2.



Case 1: $L_1 > L_2$



Case 2: $L_1 < L_2$

Figure 4: The busy implementation of Johnson's schedule will have idle time only when $L_1 < L_2$.

3.2.1 Expected makespan

Since both the delayed and the busy implementations of Johnson's sequence result in schedules with the same makespan, the following results hold for both schedules.

Lemma 4. *In Johnson's schedule,* $\Pr(\textit{internal idle time on } M_2 > 0) = 1/2$.

Proof. This follows because the joint distribution of L_1 and L_2 is symmetric and because there is internal idle time on M_2 if and only if $L_2 > L_1$. □

Thus about half of the time, there will be some idle time on M_2 in Johnson's schedule and by Theorem 1, this idle time is more likely to occur in the second part of the schedule, when jobs from S_2^* are processed. Moreover, since the workload in the buffer is at its maximum after the last job of S_1^* is completed on M_1, this idle time will occur towards the end of the busy implementation.

Let $\mathcal{M}_n^J$ be the makespan of Johnson's schedule;

Theorem 5. *The expected makespan of Johnson's Schedule satisfies*

$$\mathrm{E}[\mathcal{M}_n^J] = \frac{n(l+u)}{2} + \frac{1}{2}\sqrt{\frac{n}{3\pi}}\,(u-l) + o(\sqrt{n}). \tag{9}$$

Proof. If $\sum_{i=1}^{n} a_i < \sum_{i=1}^{n} b_i$, $L_1 > L_2$ and the schedule will be as in Figure 4, case 1. The makespan then equals $a_1 + \sum_{i=1}^{n} b_i + I_1^*$. As we have seen, $I_1^* \to 0$ as $n \to \infty$. Similarly, if $\sum_{i=1}^{n} a_i > \sum_{i=n}^{n} b_i$, the asymptotic makespan is $\sum_{i=1}^{n} a_i + b_n$. Hence the asymptotic makespan is $\max\{b_n + \sum_{i=1}^{n} a_i, a_1 + \sum_{i=1}^{n} b_i\}$. Therefore

$$\begin{aligned} \frac{\mathcal{M}_n^J - n\left(\frac{l+u}{2}\right)}{\sqrt{\frac{n(u-l)^2}{12}}} &= \max\left\{ \frac{b_n + \sum_{i=1}^{n} a_i - n\left(\frac{l+u}{2}\right)}{\sqrt{\frac{n(u-l)^2}{12}}}, \frac{a_n + \sum_{i=1}^{n} b_i - n\left(\frac{l+u}{2}\right)}{\sqrt{\frac{n(u-l)^2}{12}}} \right\} \\ &\Rightarrow \max\{Z_1, Z_2\} \end{aligned} \tag{10}$$

by the central limit theorem and continuous mapping principle, where Z_1, Z_2 are iid normal random variables with mean 0 and variance 1. Also, straight forward calculation shows that $\mathrm{E}[\max\{Z_1, Z_2\}] = 1/\sqrt{\pi}$ and consequently,

$$\frac{\mathrm{E}\big[\mathcal{M}_n^J\big] - n(l+u)/2}{\sqrt{n/12}(u-l)} \to \frac{1}{\sqrt{\pi}}$$

which means that

$$\mathrm{E}\big[\mathcal{M}_n^J\big] = \frac{n(l+u)}{2} + \frac{1}{2}\sqrt{\frac{n}{3\pi}}\,(u-l) + o(\sqrt{n}).$$

□

Let $\mathcal{M}^{PJ}$ be the makespan of the Partial Johnson's schedule.

Theorem 6. *The expected makespan of Partial Johnson's Schedule satisfies*

$$\mathrm{E}\big[\mathcal{M}^{PJ}\big] = \frac{n(l+u)}{2} + \frac{1}{2}\sqrt{\frac{n}{3\pi}}\,(u-l) + o(\sqrt{n}). \tag{11}$$

Proof. First consider S_1^* separately. From Theorem 1, we know that

$$\lim_{n\to\infty} \Pr(W_{\log n_1} \geq u) = 1,$$

so that jobs processed after the $\log n_1$-th one can be scheduled in any order without incurring any internal idle time in S_1^*. By reversibility, the same is true for S_2^* when considered separately. The proof now follows by implementing both halves of the partial Johnson's schedule as described in $delay(S^*)$ and invoking the arguments of Theorem 5 □

It is interesting to note that while the Partial Johnson's schedule requires less effort to implement, it achieves the same asymptotic makespan as Johnson's schedule. Also, as we show in the next section, the workloads and buffer levels are of the same order in both schedules.

3.2.2 Buffer level

Theorem 7. *The probability that the maximum buffer level for* delay(S^*) *exceeds* $n(u-l)/6u$ *converges to one as* $n \to \infty$.

Proof. Write L_1^n to emphasize the dependence of L_1 on the number of jobs n. First note that

$$L_1^n \geq \sum_{i=1}^{n} (b_i - a_i)^+.$$

The random variables $(b_i - a_i)^+$ are iid with mean $(u-l)/6$ and variance $(u-l)^2/6$. By Chebyschev's inequality

$$P\left(\left|L_1^n - n\left(\frac{u-l}{3}\right)\right| \geq \frac{n(u-l)}{6}\right) \leq \frac{6}{n} \to 0.$$

Therefore,

$$P\left(\max\{L_1^n, L_2^n\} > n\left(\frac{u-l}{6}\right)\right) \geq P\left(L_1^n > n\left(\frac{u-l}{6}\right)\right) \to 1$$

as $n \to \infty$. Since the maximum job duration is u, it follows that the probability that the maximum number of jobs exceeds $n(u-l)/6u$ converges to 1 as $n \to \infty$. □

Now consider *busy*(S^*). The expected maximum buffer level in *busy*(S^*) is asymptotically equal to that in S_1^*. This is because if $L_1 > L_2$ the expected maximum buffer level is increased by $\Theta(\sqrt{n})$ whereas if $L_1 > L_2$ it is decreased by $\Theta(\sqrt{n})$. In either case, as $n \to \infty$, both the ratio of the maximum buffer level for *delay*(S^*) to that of S_1^*, and the ratio of the maximum buffer level of *busy*(S^*) to that of S_1^* approach 1.

For large n, we saw that the expected waiting times of jobs in *busy*(S^*) approach those in *delay*(S^*) but the expected value of the maximum number

of jobs in the buffer for the latter is $\Theta(\sqrt{n})$ larger than that of $busy(S^*)$. This is because when $L_1 < L_2$, $delay(S^*)$ does not take advantage of the idle time that occurs on M_2 and jobs from S_2^* that could have been processed earlier are left waiting in the buffer. From these observations, the best way to implement Johnson's schedule is to use $delay(S^*)$ when $L_1 > L_2$ and $busy(S^*)$ when $L_1 < L_2$. In this way waiting times are reduced without increasing the number of jobs in the buffer.

3.2.3 Expected behavior of Johnson's schedule

The preceding analysis enables us to describe in considerable detail the expected behavior of a balanced flowshop under Johnson's schedule. The behavior we describe here is immediately recognizable in simulations with 20–30 jobs.

Since we are describing expected behavior, we ascribe to the ith job in Johnson's sequence its expected processing times, $\mathrm{E}[a_i]$ and $\mathrm{E}[b_i]$. Now we set about deriving approximations to these processing times that are valid when the number of jobs is large. These approximations are not necessary for us to describe the expected behavior of the system; however, they have the advantages of being considerably simpler and easier to compute than the exact, discrete expressions. Furthermore, they are quite accurate even for 20–30 jobs. In addition, since they are continuous approximations, they allow us to draw continuous graphs, which illustrate the essentials of system behavior more clearly than do discrete graphs.

First observe that for $N_1 = n_1$ and $i < n_1$ the expected processing time of the ith job on M_1 can be written as follows.

$$\mathrm{E}[a_i] = u - (u - l)\frac{\Gamma(n_1 + 1)\,\Gamma(n_1 - i + 3/2)}{\Gamma(n_1 - i + 1)\,\Gamma(n_1 + 3/2)}.$$

Since the number of jobs n is large, the number of jobs n_1 in the first part of the schedule is likely to be large so we appeal to Stirling's formula to get the following approximation.

$$
\begin{aligned}
& \frac{\Gamma(n_1+1)\,\Gamma(n_1-i+3/2)}{\Gamma(n_1-i+1)\,\Gamma(n_1+3/2)} \\
\approx\ & \frac{(n_1+1)^{n_1+1/2}(n_1-i+3/2)^{n_1-i+1}}{(n_1-i+1)^{n_1-i+1/2}(n_1+3/2)^{n_1+1}} \\
=\ & \left(\frac{n_1+1}{n_1+3/2}\right)^{n_1+1}\left(\frac{n_1-i+3/2}{n_1-i+1}\right)^{n_1-i+1}\sqrt{\frac{n_1-i+1}{n_1+1}} \\
=\ & \left(1-\frac{1}{2n_1+3}\right)^{n_1+1}\left(1+\frac{1}{2(n_1-i+1)}\right)^{n-i+1}\sqrt{1-\frac{i}{n_1+1}} \\
\approx\ & \left(1-\frac{1}{2n_1}\right)^{n_1}\left(1+\frac{1}{2(n_1-i)}\right)^{n_1-i}\sqrt{1-i/n_1} \\
\approx\ & e^{-1/2}e^{1/2}\sqrt{1-i/n_1} \\
=\ & \sqrt{1-i/n_1}.
\end{aligned}
$$

Thus, for large n_1, we can write

$$\mathrm{E}[a_i] \approx u-(u-l)\sqrt{1-i/n_1};$$

and similarly,

$$\mathrm{E}[b_i] \approx u-(1/2)(u-l)\sqrt{1-i/n_1}.$$

Also, from equation 4, we can write

$$\mathrm{Var}[a_i] \approx (u-l)^2\left(\sqrt{1-i/n_1}-(1-i/n_1)\right)$$

and it can verified that for large n_1 the variances of the processing times of the first and last jobs of S_1 on M_1 go to zero.

For large numbers of jobs, J_1 and J_2 will be expected to contain approximately the same number of jobs, so that $N_1 \approx N_1 \approx n/2$. Thus we can write $\mathrm{E}[a_i] \approx u-(u-l)\sqrt{1-2i/n}$ for $i=1,\ldots,n/2$. Since n is constant, we scale the indices of the jobs by n and relabel them with their percentiles $\rho = i/n$, which give their relative positions in Johnson's sequence. This allows us to write $\mathrm{E}[a_\rho] \approx u-(u-l)\sqrt{1-2\rho}$ for $0 \le \rho \le 1/2$.

The symmetries of the expected processing times in Johnson's sequence allow us to write the expected processing times for an arbitrary job appearing ρ

percent of the way through Johnson's sequence.

$$\mathrm{E}[a_\rho] \approx \begin{cases} u - (u-l)\sqrt{1-2\rho} & \text{for } 0 \le \rho \le 1/2; \\ \mathrm{E}[b_{1-\rho}] & \text{for } 1/2 < \rho \le 1 \end{cases}$$

and

$$\mathrm{E}[b_\rho] \approx \begin{cases} u - (1/2)(u-l)\sqrt{1-2\rho} & \text{for } 0 \le \rho \le 1/2; \\ \mathrm{E}[a_{1-\rho}] & \text{for } 1/2 < \rho \le 1. \end{cases}$$

We write the total work on M_1 and on M_2 of the first ρ jobs as

$$A(\rho) = \int_0^\rho \mathrm{E}[a_\theta]\, d\theta; \text{ and } B(\rho) = \int_0^\rho \mathrm{E}[b_\theta]\, d\theta \text{ respectively.}$$

Since by our previous analysis there is little chance of idle time on M_2, we make the approximation that the makespan of the schedule is $A(1) + l$. Another way of interpreting this approximation is that after the first job we allow M_2 to begin work on job ρ immediately after M_2 has completed processing the preceding job, so that machines M_1 and M_2 might work on the same job simultaneously. This underestimates the duration of the schedule, but nevertheless enables us to describe general system behavior with considerable accuracy.

Figure 5 (a) shows, for any instant during the makespan, the expected length of the processing time of the current job on M_1 or M_2. The two curves inherit the symmetry of Johnson's sequence since all processing times are drawn from the same distribution. Observe that, before the first crossover point, M_1 is working on shorter jobs than is M_2 and so we expect jobs to accumulate in intermediate storage. Similarly, between the first and second crossover points M_1 is working on longer jobs than is M_2 and so we expect the number of jobs in intermediate storage to decrease. Therefore the number of jobs in intermediate storage should increase up until the first crossover point, where it will be at a maximum, dip in the middle, and then increase again to another maximum at the third crossover point. This is confirmed in Figure 5 (b).

Figure 5: (a) The processing time of the current job on each of M_1 and M_2, shown as a fraction of the largest possible processing time u. (b) The number of jobs in intermediate storage, shown as a fraction of n, the number of jobs in the batch. (Both illustrations were computed for processing times drawn from $U[0, 20]$)

4 Random Schedules

It is surprising that when the number of jobs is large, then the makespan of even a *random* schedule is nearly that of Johnson's schedule. In fact, we conjecture that this is true for more general m-machine permutation flowshops. This might have practical implications ("do not bother to schedule"). It also suggests a trap for the unwary algorithm designer: *any* algorithm will produce nearly optimum schedules when processing times are independent on the machines, unless the algorithm is downright perverse.

We also show that random schedules have the advantage of requiring intermediate storage of size only $\Theta(\sqrt{n})$ rather than the $\Theta(n)$ storage required in Johnson's schedule.

Under any sequence of jobs, M_1 is never idle as long as there are jobs to be processed. Consequently, the completion times of the jobs on M_1 form a renewal process (Ross, 1983). Since these completion times correspond to the arrival times of jobs on M_2, the latter can be modeled as a GI/G/1 queue and an analysis of the latter can be carried out to determine the expected buffer level and sojourn times of the jobs on M_2.

Since we are analyzing a balanced system, this GI/G/1 system has interarrival times distributed as the service times, so the system has traffic intensity $\rho = 1$, and will not reach stationary behavior. We are interested in the behavior of queue size (buffer level) and waiting times as functions of n, for large n. As is well known, the GI/G/1 system in heavy traffic, and in particular with $\rho = 1$, can be approximated by diffusion models. In what follows we are interested in the rate at which the queue grows and the expected time before it empties out.

Assume that the jobs are indexed in the order they are processed under a random schedule. Figure 6 shows how the workload on M_2 varies over time for the random schedule. The workload starts out at zero and jumps to b_1 at time a_1, which corresponds to the completion time of the first job on M_1. The second job arrives at time $(a_1 + a_2)$ and the workload at that point is given by $\max\{0, b_1 - a_2\} + b_2$.

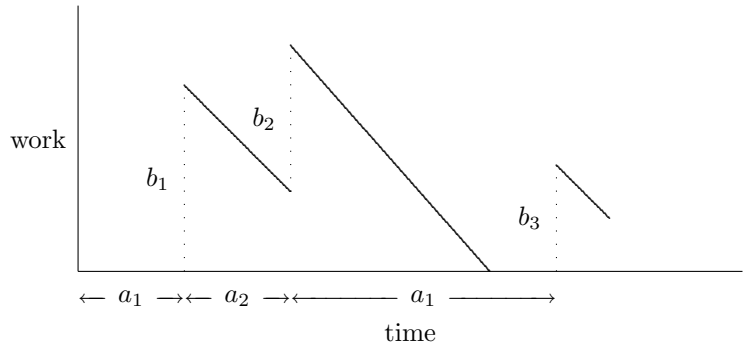


Figure 6: Workload in the buffer

4.1 Random Schedules

In this section we will examine the workload at M_2 under a random schedule. Such a schedule can be analyzed using the same methods as the GI/G/1 queue with traffic intensity equal to 1.

Let I_1 represent the internal idle time on M_2 during the processing of all jobs on M_1 with a random schedule. Let $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$ be the work times on M_1 and M_2, respectively, under a random schedule. Set $x_i = b_i - a_{i+1}$ for $i = 1, 2, \ldots, n-1$, and $S_k = x_1 + x_2 + \cdots + x_k$ for $1 \leq k \leq n-1$. The x_i's are iid with $\mathrm{E}[x_1] = 0$ and $\mathrm{Var}[x_1] = \sigma^2 = \mathrm{Var}[a_1 - b_1]$. Let W_i^R be the work awaiting M_2 just before M_1 completes job $i+1$. Then we can express W_i^R as

$$W_i^R = S_{i-1} - \min\{S_k : 0 \leq k \leq i-1\}; \tag{12}$$

furthermore W_i^R has the same distribution as $\max\{S_k : 0 \leq k \leq i-1\}$ (Karlin and Taylor, 1981). The makespan is given by

$$\mathcal{M}_n^R = \sum_{i=1}^{n} a_i + W_n^R + b_n. \tag{13}$$

It is well known that

$$\frac{W_n^R}{\sigma\sqrt{n}} \Rightarrow |Z|, \tag{14}$$

where Z is a standard normal random variable, $\sigma^2 = \mathrm{Var}[a_1 - b_1] = (u-l)^2/6$,

and $\Rightarrow$ signifies convergence in distribution. Therefore, we can conclude that

$$\frac{1}{\sigma\sqrt{n}}\left(\mathcal{M}_n^R - \sum_{i=1}^{n} a_i\right) \Rightarrow |Z| \tag{15}$$

as $n \to \infty$.

The expected maximum buffer level for the random algorithm is $O(\sqrt{n})$.

4.2 Johnson's Partition Schedule

In this section we will show that the difference in the makespan for Johnson's schedule and Johnson's Partition schedule converges to a finite random variable. Therefore as the makespans increase with n, the difference becomes negligible.

Number the jobs in the order processed by Johnson's Partition schedule; $a_i \leq b_i$ for $i = 1, 2, \ldots, N_1$, and $a_i > b_i$ for $i > N_1$. Using the same notation as in the previous section, we have $\mathrm{E}[x_i] = \mu_x > 0$ and $\mathrm{Var}[x_i] = \sigma_x^2 \in (0, \infty)$ for $1 \leq i \leq N_1$. In this case, since $\mathrm{E}[x_1] > 0$, the internal idle time on machine 2 during the processing of the first N_1 jobs converges in distribution to a random variable $\mathcal{I}$ as $N_1 \to \infty$, where

$$\mathrm{E}[\mathcal{I}] = \sum_{k=1}^{\infty} \frac{1}{k}\mathrm{E}[S_k : S_k \geq 0] \leq \frac{\sigma_x^2}{2\mu_x} \tag{16}$$

(problem #18, chapter 17, in Karlin and Taylor, 1981). (Assume that $\mathrm{E}\left[\mathcal{I}^2\right] < \infty$.) Now the makespan is

$$\mathcal{M}_n^{JP} = \max\left\{\sum_{i=1}^{n} a_i, \sum_{i=1}^{n} b_i\right\} + \max\left\{I_1^n, I_2^n\right\}, \tag{17}$$

where I_1^n, I_2^n both converge in distribution to $\mathcal{I}$. From (16) and using the fact that the I_j^n are increasing, $\mathrm{E}[\max\{I_1^n, I_2^n\}] \leq \mathrm{E}[(I_1^n + I_2^n)] \leq 2\mathrm{E}[\mathcal{I}] \leq \sigma_x^2/\mu_x$. In our case,

$$\mu_x = \frac{u-l}{3}, \qquad \sigma_x^2 = \frac{(u-l)^2}{9}, \tag{18}$$

and so

$$\lim_{n\to\infty} \mathrm{E}\left[\mathcal{M}_n^J - \mathcal{M}_n^{JP}\right] \leq \frac{u-l}{3}; \tag{19}$$

that is, in the limit there is not much difference between Johnson's schedule and Johnson's Partition schedule.

We can conclude that

$$\frac{\mathcal{W}_n^{JP}}{\sigma\sqrt{n}} \Rightarrow Z^+, \tag{20}$$

where Z is a standard normal random variable, $\sigma^2 = \mathrm{Var}[a_1 - b_1] = (u-l)^2/6$, and $Z^+ = \max\{0, Z\}$.

It is important to note that the internal idle time is most likely to occur during the early phase of S_1, so that for small n Johnson's partition schedule may have almost as much internal idle time as for large n.

The maximum buffer size for Johnson's Partition schedule is of the same order as for Johnson's schedule.

4.3 Comparison of Johnson's Partition and Random Schedules

How much better is Johnson's Partition schedule than a random schedule? For a fixed sequence of jobs, the difference in expected makespans is the difference in expected internal idle time on the second machine. For Johnson's Partition schedule this is

$$\mathrm{E}[\max\{I_1, I_2\}] \leq \mathrm{E}[(I_1 + I_2)] \leq \sigma_x^2/\mu \tag{21}$$

while for a random schedule,

$$\mathrm{E}[\mathcal{I}_n] = \sum_{k=1}^{n} \frac{1}{k} \mathrm{E}\left[S_k^+\right] \to \infty \tag{22}$$

as $n \to \infty$.

Comparing the makespans, we have

$$\mathcal{M}_n^J \leq \mathcal{M}_n^{PJ} \leq \mathcal{M}_n^{JP} \leq \mathcal{M}_n^R, \tag{23}$$

and

$$\frac{\mathcal{M}_n^R - \mathcal{M}_n^{JP}}{\sigma\sqrt{n}} \Rightarrow |Z|, \tag{24}$$

$$\mathcal{M}_n^{JP} - \mathcal{M}_n^{PJ} \Rightarrow I, \tag{25}$$

and

$$\mathcal{M}_n^{PJ} - \mathcal{M}_n^J \Rightarrow 0. \tag{26}$$

Sequence	Required information	Worst-case effort
Johnson's	values of a_j, b_j	$O(n \log n)$
Partial Johnson's	a_j, b_j of smallest jobs	$O(n \log \log n)$
Johnson's Partition	relative sizes of a_j, b_j	$O(n)$
Random	none	$O(1)$

Table 1: Comparison of schedules for 2-machine flowshops: As randomness increases, computational requirements decrease.

Sequence	Asymptotic makespan	Intermediate storage
Johnson's	$n\mu + \sigma\sqrt{n/\pi} + o(\sqrt{n})$	$\Theta(n)$
Partial Johnson's	$n\mu + \sigma\sqrt{n/\pi} + o(\sqrt{n})$	$\Theta(n)$
Johnson's Partition	$n\mu + \sigma\sqrt{n/\pi} + o(\sqrt{n})$	$\Theta(n)$
Random	$n\mu + 2\sigma\sqrt{n/\pi} + o(\sqrt{n})$	$\Theta(\sqrt{n})$

Table 2: Comparison of schedules for 2-machine flowshops: As randomness increases, makespan increases slightly but the requirement for intermediate storage decreases significantly. (μ is the mean of the distribution of processing times and σ is its standard deviation.)

In summary, the difference between the makespans under Johnson's algorithm and the partial Johnson's algorithm converges in distribution to zero. The difference between partial Johnson's and Johnson's partition algorithm converges to a finite random variable. Only when going to the random algorithm does the difference in makespan increase without bound; the makespan with the random algorithm will be of order $\sqrt{n}$ larger than the others.

5 Conclusions

Tables 1 and 2 show the trade-offs between makespan on the one hand, and simplicity, reduced data requirements, and reduced intermediate storage on the other hand. It is worth noting that intermediate storage is not significantly reduced until the sequence becomes completely random. Any heuristic that processes all the jobs of J_1 before those of J_2 requires $\Theta(n)$ intermediate storage. This is because all jobs in J_1 have shorter processing time on M_1, so that processing these jobs first leaves $\Theta(n)$ jobs waiting in the buffer when the last

job of J_1 is completed.

Our research suggests that, although Johnson's schedule guarantees minimal makespan, it might not be appropriate in practice when processing times are independent. If the flowshop is balanced, it might not be worthwhile to schedule because the expected makespan of a random schedule exceeds that of Johnson's schedule by $O(\sqrt{n})$. In fact, one might prefer *not* to schedule, since sequencing the jobs at random requires no estimates of the processing times, and, moreover, requires a factor of $O(\sqrt{n})$ less buffer capacity than does Johnson's schedule.

If the flowshop is unbalanced then it might not be worthwhile to schedule because the busier machine will likely remain busy throughout the schedule, independent of the sequence in which the jobs are processed. In fact it is fairly straightforward to extend our work to the case of unbalanced machines. Suppose that the expected processing time of each job is greater on the first (second) machine so that this machine is busier. Then as the number of jobs gets large, the expected makespan of Johnson's schedule approaches the value of the expected total processing time of the first (second) machine, plus the expected processing time of the smallest job on the second (first) machine. The expected makespan for a Partial Johnson's schedule in this case is the sum of the expected total processing time for the busier machine plus the expected processing time for a single job on the other machine. In the case of random schedules the system can be viewed as a queuing system with the busier machine defining the arrival rate and the other machine defining the service rate. The makespan can be approximated by the expected total processing time of the busier machine plus the expected sojourn time of a customer in the queuing system defined above.

It would be interesting to learn whether random schedules remain asymptotically optimal for general m-machine flowshops with unlimited intermediate storage. There are empirical reasons to believe so. For example, Dannenbring (1977) performed computational tests on heuristics to minimize makespan in m-machine flowshops. He remarked that

> The processing times for the problems were randomly generated in-

tegers uniformly distributed over the interval $(0, 99)$. Evidence exists that the uniform distribution provides the more difficult problems to solve.

In tests with "small" problems (3–6 jobs), a heuristic that produced random schedules ranked fifth out of eleven heuristics, while in tests with "large" problems (7, 10, 25, and 50 jobs), random schedules ranked second. We believe that the surprisingly good performance of random schedules might be an artifact of their (conjectured) asymptotic optimality.

If random schedules are indeed asymptotically optimal for general m-machine flowshops, then some performance studies of flowshop heuristics might be flawed. It is natural to generate processing times by sampling independently from well-behaved distributions, as in Dannenbring (1977); then, if only a single heuristic is tested and compared to lower bounds on the optimal makespan, that heuristic might look deceptively good (as long as it is not perversely biased).

Dudek, Panwalkar, and Smith (1992) suggested seven reasons for the lack of verifiable applications of flowshop scheduling. Our paper suggests some additional reasons, including that, even if the problem did exist as described, it might not present any difficulty; that is, if the load is imbalanced, then there tends to be a capacity problem and not a scheduling problem; and if the load is balanced, then one might not be able to do substantially better than schedule randomly.

Finally, it would be interesting to see how far these results can be extended when the processing times of an individual job are allowed to be correlated.

Acknowledgements

A. Ramudhin was supported in part by a grant from the Natural Sciences and Engineering Research Council of Canada (OPG0105858). J. Bartholdi was supported in part by a grant from the National Science Foundation (DDM-9101581) and by the Office of Naval Research (N00014-89-J-1571). J. Calvin

was supported in part by the National Science Foundation (DDM-9010770). J. Vande Vate was supported in part by the National Science Foundation (DDM-9101581). Gideon Weiss was supported in part by the National Science Foundation (DDM-8914863).

References

[1] Barany I. 1981. A vector-sum theorem and its application to improving flow shop guarantees, *Mathematics of Operations Research* **6**(3):445–452.

[2] Dannenbring D.G. 1977. The evaluation of flowshop sequencing heuristics, *Management Science* **23**(11):1174–1182.

[3] Dudek, R. A., S. S. Panwalkar, and M. L. Smith. 1992. The lessons of flowshop scheduling research, *Operations Research* **40**(1):7–13.

[4] Harrison M. 1985. *Brownian Motion and Stochastic Flow Systems*, John Wiley & Sons.

[5] Johnson S.M. 1954. Optimal two and three-stage production schedules with set-up times included, *Naval Research Logistics Quarterly* **1**(1):61–68.

[6] Ku P.S. and Niu S.C. 1986. On Johnson's two machine flow shop with random processing times, *Operations Research* **34**(1):130–36.

[7] Law A. and Kelton D. 1982. *Simulation Modeling and Analysis*, McGraw-Hill, NY.

[8] Papadimitriou C.H. and Kanellakis P.C. 1980. Flowshop scheduling with limited temporary storage, *J. Assoc. Comput. Mach.* **27**:533–549.

[9] Pinedo M. 1982. Minimizing the expected makespan in stochastic flow shops, *Operations Research* **30**(1):148–62.

[10] Ross S. 1983. *Stochastic Processes*, John Wiley & Sons.

[11] Karlin S. and Taylor H.M. 1981. *A Second Course in Stochastic processes*, Academic Press Inc.