Scheduling Two-Point Stochastic Jobs to Minimize the Makespan on Two Parallel Machines

Sem Borst [1 4] *John Bruno* [2 5] *E. G. Coffman, Jr.* [1] *Steven Phillips* [3]

ABSTRACT

Simple optimal policies are known for the problem of scheduling jobs to minimize expected makespan on two parallel machines when the job running-time distribution has a monotone hazard rate. But no such policy appears to be known in general. We investigate the general problem by adopting two-point running-time distributions, the simplest discrete distributions not having monotone hazard rates. We derive a policy that gives an explicit, compact solution to this problem and prove its optimality. We also comment briefly on first-order extensions of the model, but each of these seems to be markedly more difficult to analyze.

[1] Bell Labs, Lucent Technologies, Murray Hill, NJ 07974

[2] Department of Computer Science, University of California, Santa Barbara, CA 93106

[3] AT&T Research, Murray Hill, NJ 07974

[4] The author was supported in part by a fellowship from the Netherlands Organization for Scientific Research (NWO)

[5] The author was supported in part by a University of California MICRO grant and the Xerox Corporation.

Scheduling Two-Point Stochastic Jobs to Minimize the Makespan on Two Parallel Machines

Sem Borst [1 4] *John Bruno* [2 5] *E. G. Coffman, Jr.* [1] *Steven Phillips* [3]

ABSTRACT

1. Introduction

Scheduling stochastic jobs on parallel machines to minimize expected makespan (latest finishing time) is a problem at the heart of stochastic scheduling theory. In the version of interest here, scheduling is preemptive and job running times are n independent samples from a given distribution G. Optimal policies have been known for some time when G has a monotone hazard-rate function; according to these policies, at every time t, unfinished jobs are scheduled in non-decreasing order of their hazard rates at time t [9]. However, nothing appears to be known about general distributions G. In particular, there is no concrete measure of the difficulty of the general problem, as is the case for example with the class of stochastic optimization problems studied in [7].

To broaden the understanding of the general problem, this paper studies what is probably the simplest non-trivial case where the running-time distribution does not have a monotone hazard-rate function: There are two machines and G is a two-point distribution on the integers 1 and $k+1$, with $p = \Pr\{X = 1\}$ arbitrary, where X is a generic job running time. The expected-makespan minimization problem is discrete in that scheduling decisions are limited to the integer times $0, 1, \ldots$. All that a policy knows about an unfinished job is the time the job has already run. Thus, if a job has already received at least 1 unit of running time, the job's remaining running time is also known.

After the preliminaries of the next section, we present in Section 3 the desired algorithm and a proof of its optimality. Section 4 concludes the paper with a few comments on open problems. The remainder of this section remarks briefly on related literature.

[1] Bell Labs, Lucent Technologies, Murray Hill, NJ 07974
[2] Department of Computer Science, University of California, Santa Barbara, CA 93106
[3] AT&T Research, Murray Hill, NJ 07974
[4] The author was supported in part by a fellowship from the Netherlands Organization for Scientific Research (NWO)
[5] The author was supported in part by a University of California MICRO grant and the Xerox Corporation.

This paper is a sequel to the research in [4] where the same problem is studied except that the makespan objective function is replaced by the sum of finishing times (flow time). The analysis here is quite different and leads to stronger results. The approach to the somewhat more difficult problem in [4] entails asymptotic methods. We also refer to [4] for a brief discussion of potential applications.

A variant to the general problem studied here assumes machines of different speeds; see e.g. [3]. In a variant of the probability model, jobs are taken as independent samples of exponential distributions with rate parameters that may vary from job to job [1, 5, 11]. For additional references, see [10, 4].

2. Preliminaries

A *state* of the system is given by a $(k+1)$-tuple of non-negative integers $\mathbf{n} = (n_0, \ldots, n_k)$ where n_i denotes the number of unfinished jobs that have accrued i units of running time. Given a state $\mathbf{n}$, the number of unfinished jobs is denoted by $\#\mathbf{n} := \sum_{i=0}^{k} n_i$.

A *policy* is a mapping π from states $\mathbf{n}$ into assignments of machines to unfinished jobs. Thus, if $\#\mathbf{n} \geq 2$ then $\pi(\mathbf{n}) = (i, j)$ means that the machines are assigned to jobs which have received i and j units of running time without finishing, respectively. For an assignment to be feasible we require: if $i = j$ then $n_i \geq 2$, otherwise $n_i \geq 1$ and $n_j \geq 1$. If $\#\mathbf{n} = 1$ then π assigns one machine to the lone unfinished job and the other machine remains idle.

A *schedule* for a collection of jobs with known running times is represented by a sequence of job/duration pairs for each machine. The job sequences begin at time 0 on both machines and satisfy the following constraints: The total running time of a job on the two machines is equal to the job's running-time requirement, and at no time is any job scheduled to run on both machines at the same time. The *makespan* of a schedule is its latest job finishing time.

Let $M_\pi(\mathbf{n})$ denote the (random) makespan of the schedule determined by policy π beginning in state $\mathbf{n}$. A policy π^* is defined to be *optimal* if $\mathrm{E}M_{\pi^*}(\mathbf{n}) = \min_\pi \mathrm{E}M_\pi(\mathbf{n})$ for all $\mathbf{n}$.

A greedy algorithm for our scheduling problem assigns machines to jobs in increasing order of accrued running time. The expected makespan under this least-accrued-time (LAT) policy is easily computed as follows. Let ℓ denote the number of long jobs (i.e., jobs that require $k+1$ units of processing time) in the schedule. If $\ell \neq 1$, then LAT yields a makespan $\lceil \frac{n+k\ell}{2} \rceil$. If $\ell = 1$, LAT produces a makespan $\frac{n}{2} + k$ when n is even, but either $\frac{n+1}{2} + k$ or $\frac{n-1}{2} + k$ according as the last job is or is not long, respectively, when n is odd. Thus, if π is the LAT policy and n is even, we get the formula

$$\text{(2.1)}\qquad \mathrm{E}M_\pi(\mathbf{n}) = \sum_{\ell=0}^{n}\left(\frac{n}{2}+\left\lceil\frac{k\ell}{2}\right\rceil\right)\binom{n}{\ell}p^{n-\ell}(1-p)^{\ell}+\left(k-\left\lceil\frac{k}{2}\right\rceil\right)np^{n-1}(1-p),$$

and if n is odd, then

$$\begin{aligned}\mathrm{E}M_\pi(\mathbf{n}) &= \sum_{\ell=0}^{n}\left(\frac{n+1}{2}+\left\lceil\frac{k\ell-1}{2}\right\rceil\right)\binom{n}{\ell}p^{n-\ell}(1-p)^{\ell}\\ &+ \left(k-\left\lceil\frac{k+1}{2}\right\rceil\right)(n-1)p^{n-1}(1-p)+\left(k-\left\lceil\frac{k-1}{2}\right\rceil\right)p^{n-1}(1-p).\end{aligned}$$

Working out the sums and using the indicator function $\mathrm{I}_{\{\}}$, these combine into

$$\begin{aligned}\mathrm{E}M_\pi(\mathbf{n}) = \left\lceil\frac{n}{2}\right\rceil+\frac{n(1-p)k}{2} &+ \left(\frac{kn}{2}-(n-1)\mathrm{I}_{\{n\text{ odd}\}}\right)p^{n-1}(1-p)\\ &+ \left(\mathrm{I}_{\{k\text{ odd}\}}-\frac{1}{2}\right)\left[np^{n-1}(1-p)+\frac{1}{2}(1-(2p-1)^n)\right].\end{aligned}$$

Note that, whenever the sample does not contain exactly one long job, LAT yields a *minimal* makespan, *i.e.*, a makespan of length $\lceil\sigma/2\rceil$, where σ is the sum of the job running times in the schedule. Thus, LAT will be a near-optimal policy for a wide range of values of the parameters k and p and the initial number of jobs n. For example, fix k and p and consider large n. Then with high probability (viz., with probability $1-np^{n-1}(1-p)\sim 1,\ n\to\infty$), schedules will contain either no long jobs or at least two long jobs. When there is exactly one long job, LAT produces a schedule with that job running alone at the end of the schedule, so the makespan is clearly *not* minimal.

Consistent with these remarks, Section 3 will show that LAT is an optimal *turnpike* policy in the sense that, if enough unstarted jobs (in our case at least $k+1$ such jobs) remain in state $\mathbf{n}$, then the LAT decision is optimal in state $\mathbf{n}$. The difficulty in designing an optimal policy will center on states where the work remaining consists of relatively few (in fact at most k) unstarted jobs and the final k time units of exactly one long job that has already received its first time unit. Section 3 shows that, for p large enough, we will not want to schedule all remaining unstarted jobs (as does LAT) before getting to the remainder of the long job; this strategy might be too likely to produce a poor schedule that ends with a large part of the remainder of the one long job running on one machine while the other machine is idle.

In the purely combinatorial version of our scheduling problem, the running times are known in advance, and the makespan is deterministic. If $\mathbf{n}$ is a *deterministic* state (i.e., $n_0 = 0$), then $M(\mathbf{n})$ denotes the corresponding minimal makespan. The lemma below evaluates $M(\mathbf{n})$ by applying Theorem 2.1 in [2] which is due to T. C. Hu. It is also an easy extension of the results in [6] and [8]; we omit the details. Let $\|\mathbf{n}\| = \sum_{i=1}^{k}(k+1-i)n_i$ denote the sum of the deterministic job lengths in $\mathbf{n}$, and let $[\mathbf{n}] = \max\{k+1-i | 1 \le i \le k,\ n_i > 0\}$ denote a largest running time over all deterministic jobs in $\mathbf{n}$.

Lemma 2.1. *Let* $\mathbf{n}$ *be deterministic. Then* $M(\mathbf{n}) = \max\{\lceil \|\mathbf{n}\|/2 \rceil, [\mathbf{n}]\}$*, and this value is achieved by a policy that assigns the machines to the jobs with the longest remaining running times first.*

We conclude this section by stating the principle of optimality for our stochastic scheduling problem. This formula brings out the recursive structure of an optimal policy and can serve as the definition of such policies. Although the formula is not explicitly used in what follows, it was used for computations that suggested the properties of optimal policies.

Let ϵ_i denote a $(k+1)$-tuple $(\epsilon_0^i, \ldots, \epsilon_k^i)$ where $\epsilon_i^i = 1$ and $\epsilon_j^i = 0$, $j \ne i$. Let 0_{k+1} denote the $(k+1)$-tuple of all zeros. The makespan of optimal schedules is denoted by $M_*(\mathbf{n}) := \min_\pi \mathrm{E} M_\pi(\mathbf{n})$. Define q_i to be the conditional probability that a job which has not finished after receiving i units of running time, finishes after receiving $i+1$ units of running time. Note that $q_0 = p$, $q_k = 1$, $q_i = 0$ for all $i \ne 0, k$. Then a straightforward analysis gives

1. $M_*(0_{k+1}) = 0$ and $M_*(\epsilon_k) = 1$.

2. For $0 \le i < k$, $M_*(\epsilon_i) = 1 + (1-q_i)M_*(\epsilon_{i+1})$.

3. If $\#\mathbf{n} \ge 2$,

$$\begin{aligned} M_*(\mathbf{n}) \quad = \quad & 1 + \min_{i,j}\{q_i q_j M_*(\mathbf{n} - \epsilon_i - \epsilon_j) + \\ & (1-q_i)q_j M_*(\mathbf{n} - \epsilon_i - \epsilon_j + \epsilon_{i+1}) + \\ & q_i(1-q_j)M_*(\mathbf{n} - \epsilon_i - \epsilon_j + \epsilon_{j+1}) + \\ & (1-q_i)(1-q_j)M_*(\mathbf{n} - \epsilon_i - \epsilon_j + \epsilon_{i+1} + \epsilon_{j+1})\} \end{aligned}$$

   where the range of the minimization is taken over all i, j satisfying $0 \le i \le j \le k$ and either $i = j$ and $n_i \ge 2$ or $n_i \ge 1$ and $n_j \ge 1$.

3. Main result

This section presents the desired optimal algorithm along with a proof of its optimality. First we describe a situation where clairvoyance does not help obtain smaller makespans. Assume that $\mathbf{n}$ is a *stochastic* state (i.e., $n_0 > 0$) and let $X_1, \ldots, X_{n_0}$ denote the running-time random variables associated with the n_0 stochastic jobs. Define $M_\pi(\mathbf{n}; x_1, \ldots, x_{n_0})$ as the conditional makespan given that $X_1 = x_1, \ldots, X_{n_0} = x_{n_0}$, where each x_i is either 1 or $k+1$. Let $M(\mathbf{n}; x_1, \ldots, x_{n_0})$ denote the minimum makespan over all schedules where the running times of the stochastic jobs are known in advance and equal to $x_1, \ldots, x_{n_0}$, respectively. If $\mathbf{n}$ is deterministic, then the x_i are omitted, and the notation reduces to $M(\mathbf{n})$ as before. Clearly,

$$M(\mathbf{n}; x_1, \ldots, x_{n_0}) \leq M_\pi(\mathbf{n}; x_1, \ldots, x_{n_0}),$$

since π must make its scheduling decisions without knowledge of the x_i's.

A state $\mathbf{n}$ is called *predictable* if there exists a policy π such that $M_\pi(\mathbf{n}; x_1, \ldots, x_{n_0}) = M(\mathbf{n}; x_1, \ldots, x_{n_0})$ for all $x_1, \ldots, x_{n_0}$. We define the following policy for use in predictable states.

Policy $\boldsymbol{\pi_0}$: If in state $\mathbf{n}$ with

1. $n_0 > 0$: Assign one machine to a deterministic job with the largest remaining running time and the other machine to one of the stochastic jobs.

2. $n_0 = 0$: Assign the machines to jobs with the longest remaining running times.

■

Let $det(\mathbf{n}) = n_1 + \cdots + n_k$ denote the number of deterministic jobs in $\mathbf{n}$.

Lemma 3.1. *Let $\mathbf{n}$ be a state satisfying $det(\mathbf{n}) \geq 2$ and $\|\mathbf{n}\| \geq (det(\mathbf{n}) - 1)k + n_0$. Then $\mathbf{n}$ is a predictable state and $M_{\pi_0}(\mathbf{n}; x_1, \ldots, x_{n_0}) = M(\mathbf{n}; x_1, \ldots, x_{n_0})$ for all $x_1, \ldots, x_{n_0}$.*

Proof: It follows from the conditions of the lemma that $n_0 \leq k$. Accordingly, the lemma is easily seen to be true when $k = 1$.

Assume $k \geq 2$, and let $\mathbf{n}'$ denote a state that results after n_0 decision epochs of policy π_0. Since π_0 schedules deterministic jobs with the longest remaining running time, $det(\mathbf{n}') \geq det(\mathbf{n})$. It is also easy to check that π_0 maintains the *invariant* $\|\mathbf{n}\| \geq (det(\mathbf{n}) - 1)k + n_0$ for successive states, and consequently $\|\mathbf{n}'\| \geq (det(\mathbf{n}') - 1)k$.

Suppose $det(\mathbf{n}') > 2$. It follows from the invariant and Lemma 2.1 that $M(\mathbf{n}') = \lceil \|\mathbf{n}'\|/2 \rceil$. The resulting schedule under policy π_0 is minimal, and therefore

$$M_{\pi_0}(\mathbf{n}; x_1, \ldots, x_{n_0}) = M(\mathbf{n}; x_1, \ldots, x_{n_0})$$

for all outcomes $x_1, \ldots, x_{n_0}$ such that $det(\mathbf{n}') > 2$.

Suppose $det(\mathbf{n}') = 2$. In this case, all the stochastic jobs must have finished after one unit of running time. If the resulting schedule is minimal, then the lemma follows. If one of the machines has more than one unit of idle time, then the other machine must be running a deterministic job which was available in state $\mathbf{n}$ and assigned a machine at every decision epoch. Once again the lemma follows.

■

We are now ready to prove that the policy below is optimal if started in a state $(n, 0, \ldots, 0)$. Each step of the policy is labeled with the lemmas to be used in proving its optimality.

Policy π_*: If in state $\mathbf{n}$ with,

1. $n_0 \geq k + 1$: Assign both machines to stochastic jobs. (Lemmas 3.3 and 3.4)
2. $0 < n_0 \leq k$ and
   (a) $det(\mathbf{n}) \geq 2$: Use policy π_0 from this point onward. (Lemma 3.2 and Lemma 3.1)
   (b) $det(\mathbf{n}) = 1$: Let $\Delta = n_0 - \max\{1, \lceil \frac{n_0 - k + ||\mathbf{n}||)}{2} \rceil\}$. If $p > \frac{\Delta}{\Delta+1}$, then assign one machine to the deterministic job and the other machine to a stochastic job. Otherwise, assign both machines to the stochastic jobs. (Lemmas 3.5, 3.6, and 3.7)
   (c) $det(\mathbf{n}) = 0$: Assign both machines to stochastic jobs. (There is no choice.)
3. $n_0 = 0$: Assign the machines to jobs with the longest remaining running times. (Lemma 2.1)

■

Theorem 3.1. *Policy π_* is optimal beginning in state $\mathbf{n}$ where $n_i = 0$ for $i = 1, \ldots, k$.*

Proof: We need only give the proofs of Lemmas 3.3–3.7 below (and referenced above); the theorem will then follow at once from the definition of π_*.

Lemma 3.2. *Assume that we use policy π_* beginning in a state that has no deterministic jobs. Whenever step 2 of π_* applies to the current state $\mathbf{n}$, we have $||\mathbf{n}|| \geq (det(\mathbf{n}) - 1)k + n_0$.*

Proof: Straightforward induction on the number of times that step 2 in policy π_* has executed.

■

The next two lemmas apply to step 1 of policy π_*.

Lemma 3.3. *If $n_0 > 0$, then there exists an optimal policy which assigns at least one of the machines to a stochastic job in state $\mathbf{n}$.*

Proof: Let π be an optimal policy. Suppose π beginning in state $\mathbf{n}$ assigns both machines to the deterministic jobs J_1 and J_2. Let time t be the next time at which π assigns a machine to a stochastic job, say U. Let X denote the other job that is assigned a machine at time t by policy π. If X is a deterministic job, we can assume without loss of generality that X is not equal to J_1. Define a policy π' starting in state $\mathbf{n}$ that initially assigns machines to the deterministic job J_2 and the stochastic job U; π' then makes exactly the same assignments as π until time t. At time t, π' assigns one machine to J_1 and the other to X. At time $t+1$, the states reached by both policies are identical, and from this point onward π' mimics π. Clearly, $\mathrm{E}M_{\pi'}(\mathbf{n}) \leq \mathrm{E}M_{\pi}(\mathbf{n})$.

■

Lemma 3.4. *If $n_0 \geq k+1$, then there exists an optimal policy that assigns both machines to stochastic jobs in state $\mathbf{n}$.*

Proof: Let π be an optimal policy. We assume that the stochastic jobs are assigned in a particular order, namely, $U_1, \ldots, U_{n_0}$. Bearing in mind Lemma 3.3, we suppose that at time 0, policy π assigns one machine to deterministic job J and the other machine to stochastic job U_1 (Lemma 3.3). Let time t (a random variable) be the next time at which policy π assigns one machine to a stochastic job and the other machine is assigned to a job X which is *not* J. There has to be such an epoch since $n_0 \geq k+1$. Now introduce a policy π' that assigns the machines to stochastic jobs U_1 and U_2 at time 0. From time 1 until time $t-1$, when policy π assigns one machine to a stochastic job, say U_i, and the other machine to job J, policy π' assigns one machine to job J and the other machine to stochastic job U_{i+1}. At time t, policy π' assigns one machine to job J and the other machine to job X. At time $t+1$, the states obtained by both policies are identical, and from this point onward π' mimics π. Clearly, $\mathrm{E}M_{\pi'}(\mathbf{n}) \leq \mathrm{E}M_{\pi}(\mathbf{n})$.

■

A state $\mathbf{n}$ is called *critical* if $det(\mathbf{n}) = 1$ and $||\mathbf{n}|| \geq n_0 > 0$. Let $\mathbf{n}$ and $\mathbf{n}'$ be critical states. We say that $\mathbf{n}'$ is a *successor* of $\mathbf{n}$ if there exists an integer $i > 0$ such that $||\mathbf{n}|| = ||\mathbf{n}'|| + i$ and $n_0 = n_0' + i$. A policy π is said to *panic* in state $\mathbf{n}$ if $\mathbf{n}$ is critical and in state $\mathbf{n}$ π assigns a machine to the deterministic job.

The next three lemmas justify the assignments made by policy π_* in step 2(b).

Lemma 3.5. *Whenever step 2(b) in policy π_* applies, the corresponding state is critical.*

Proof: By Lemma 3.2 and the definition of a critical state.

■

Lemma 3.6. *Suppose policy π panics in critical state $\mathbf{n}$ and does not panic in $\mathbf{n}'$, a successor of $\mathbf{n}$. Then there exists a policy π' that does not panic in state $\mathbf{n}$ and $\mathrm{E}M_{\pi'}(\mathbf{n}) \leq \mathrm{E}M_{\pi}(\mathbf{n})$.*

Proof: Assume $\|\mathbf{n}\| = \|\mathbf{n}'\| + i$ where $i > 0$. Accordingly, there are at least $i+2$ stochastic jobs in state $\mathbf{n}$ (since π does not panic in state $\mathbf{n}'$). Under the condition that the first i stochastic jobs finish after receiving one unit of running time, policy π assigns one machine to the deterministic job J and assigns the other machine to stochastic jobs $U_1, \ldots, U_i$ at times 0 through $i-1$. At time i, policy π assigns the machines to U_{i+1} and U_{i+2}.

At time 0, policy π' assigns the machines to stochastic jobs U_1 and U_2. Under the condition that the first i stochastic jobs finish after receiving one unit of running time, policy π' assigns one machine to the deterministic job J at times 1 through i. The other machine is assigned to stochastic jobs $U_3, \ldots, U_{i+2}$ at times 1 through i, respectively. Clearly, under the condition that the first i stochastic jobs finish after receiving one unit of running time, policies π and π' achieve the same state at time $i+1$. From time $i+1$ onward, policy π' mimics policy π.

Assume that at least one of the first i stochastic jobs requires $k+1$ units of running time. The analysis can be divided into two cases: 1) U_1 does not finish after receiving one unit of running time and 2) U_j, with $1 < j \leq i$, is the first stochastic job not to finish after receiving one unit of running time.

1) Suppose that U_1 requires $k+1$ units of running time. Let t denote the time at which policy π assigns a machine to the stochastic job U_2, and let X denote the other job assigned a machine at time t. As before, policy π' assigns the machines to stochastic jobs U_1 and U_2 at time 0. ¿From time 1 to time t, policy π' mimics policy π. At time t, policy π' assigns machines to jobs depending upon X.

If $X = U_1$, then π' assigns machines to jobs U_1 and J at time t. It is not difficult to see that under these assumptions both policies achieve the same state at time $t+1$. ¿From time $t+1$ onward, policy π' mimics policy π.

If $X = U_3$, then π' assigns machines to jobs J and U_3 at time t. Again, it is not difficult to see that under these assumptions both policies achieve the same state at time $t+1$. ¿From time $t+1$ onward, policy π' mimics policy π.

If $X = J$, then policy π' assigns the machines to jobs J and U_1 at time t. Let d_J and d_{U_1} be the remaining running times of jobs J and U_1, respectively, at time $t+1$ under policy π. The remaining running times of J and U_1 under policy π' at time $t+1$ are $d_J + 1$ and $d_{U_1} - 1$. Since at each epoch between time 0 and t policy π assigns the machines to J and U_1, we have that $d_{U_1} > d_J$. ¿From time $t+1$ onward, policy π' mimics π with one exception. Since $d_{U_1} > d_J$, there must be some time after time t at which policy π assigns one machine to U_1 and the other machine to a job X not equal to J. At this time, policy π' assigns one machine to J and the other machine to X. It is not difficult to see that both policies achieve the same state at the following epoch, and make identical assignments thereafter.

2) Assume U_j, with $1 < j \leq i$, is the first stochastic job not to finish after receiving one unit of running time. Until time $j-1$, policy π' makes the assignments prescribed above under the condition that the stochastic jobs finish after receiving one unit of running time. At time $j-1$, policy π' assigns one machine to job J and the other to job U_j (at this time U_j is a deterministic job under policy π'). Let d_J and $d_{U_j} = k$ denote the remaining running times of jobs J and U_j, respectively, at time j under policy π. The remaining running times of J and U_j at time j under policy π' are $d_J + 1$ and $d_{U_j} - 1 = k - 1$, respectively. Policy π' mimics π from time j onward with one exception. Since $d_J < k$, there must be some time after $j-1$ at which policy π assigns one machine to U_j and the other machine to a job X not equal to J. At this time, policy π' assigns one machine to J and the other machine to X. It is not difficult to see that both policies achieve the same state at the next epoch.

■

Lemma 3.7. *Let $\mathbf{n}$ be a critical state, $n_0 > 0$, and $\Delta = n_0 - \max\{1, \lceil\frac{n_0-k+\|\mathbf{n}\|)}{2}\rceil\}$. Then an optimal policy must panic if $p > \frac{\Delta}{\Delta+1}$, must not panic if $p < \frac{\Delta}{\Delta+1}$, and may or may not panic if $p = \frac{\Delta}{\Delta+1}$.*

Proof: The proof is by induction on n_0. When $n_0 = 1$, we have $\Delta = 0$, and the lemma obviously holds. Consider some $n_0 > 1$, and assume that the lemma holds for all critical states with fewer stochastic jobs. Since $\mathbf{n}$ is a critical state, we have $\|\mathbf{n}\| \geq n_0$. We compare two policies π and π', where π panics whenever it encounters a critical state (including state $\mathbf{n}$), and where π' does not panic in state $\mathbf{n}$, but panics whenever it encounters any other critical state. By Lemmas 3.5 and 3.1, whenever either policy encounters a stochastic job that does not finish the resulting state is predictable. Both policies behave optimally when a predictable state is encountered.

To begin with, we calculate $\mathrm{E}M_\pi(\mathbf{n}) - \mathrm{E}M_{\pi'}(\mathbf{n})$. It is easy to see that for sample paths in which at least two of the stochastic jobs require $k+1$ running time, the makespans under both policies are identical (the schedules are minimal). If all the stochastic jobs finish after one unit of running time, then the contribution to the expected makespan under policy π is $p^{n_0}\|\mathbf{n}\|$. The corresponding contribution under policy π' is $p^{n_0}(\|\mathbf{n}\| + 1)$.

The situation is more complex when exactly one of the stochastic jobs requires $k + 1$ units of running time. Let $U_1, \ldots, U_{n_0}$ denote the stochastic jobs and J denote the deterministic job. Suppose that U_i, $i \geq 2$, is the stochastic job requiring $k + 1$ units of running time. At time i, under policy π the remaining running time of job J is $\|\mathbf{n}\| - i$, the number of stochastic jobs is $n_0 - i$, and the remaining running time of job U_i is k. At time $i - 1$, under policy π' the remaining running time of job J is $\|\mathbf{n}\| - (i - 2)$, the number of stochastic jobs is $n_0 - i$, and the remaining running time of job U_i is k. These states are predictable, and thus we can compare the expected contribution to the makespan from these sample paths. If enough stochastic jobs remain, the resulting makespan under policies π and π' will be minimal, and the contributions under both policies are equal. It is not too difficult to see that if $\|\mathbf{n}\| - i$ plus $n_0 - i$ is less than or equal to $k - 2$, then the makespan under policy π is not minimal, *i.e.*,

$$\|\mathbf{n}\| + n_0 - 2i \leq k - 2. \tag{3.1}$$

Since policy π' detects the long job (U_i) at time $i - 1$, the makespan of the schedule under π' is one less than the makespan of the schedule under policy π. If U_1 were the long job, both policies detect this at the same time, and the resulting makespans are equal.

Let Δ denote the number of sample paths for which the schedule under π is one longer than the schedule under π'. The number Δ is equal to the number of values of i satisfying $2 \leq i \leq n_0$ and (3.1). We get

$$\begin{aligned} \Delta &= n_0 + 1 - \max\{2, \lceil \|\mathbf{n}\| + n_0 - k + 2)/2 \rceil\} \\ &= n_0 - \max\{1, \lceil \|\mathbf{n}\| + n_0 - k)/2 \rceil\}. \end{aligned} \tag{3.2}$$

The probability of each of these sample paths is $p^{n_0 - 1}(1 - p)$, and thus the difference in the contribution to the makespan between π and π' is $p^{n_0-1}(1 - p)\Delta$. Since for all other sample paths the two policies achieve identical makespans, we get

$$\mathrm{E}M_\pi(\mathbf{n}) - \mathrm{E}M_{\pi'}(\mathbf{n}) = p^{n_0-1}(1 - p)\Delta - p^{n_0}. \tag{3.3}$$

Policy π is no worse than policy π' when $p \geq \Delta/(\Delta+1)$. Since $\Delta/(\Delta+1)$ is non-increasing as n_0 decreases, it follows from the induction hypothesis that π_0 is an optimal policy whenever $p \geq \Delta/(\Delta + 1)$ (some care has to be taken when $n_0 = 2$).

Policy π' is better than policy π when $p < \Delta/(\Delta + 1)$. It follows from Lemma 3.6 that panicing in state $\mathbf{n}$ could not have been optimal.

■

The proof of Theorem 3.1 is complete.

4. Concluding remarks

We conclude with some comments on open problems. As mentioned earlier, the problem considered here is probably the simplest non-trivial case where the running-time distribution does not have a monotone hazard-rate function. One obvious generalization would be the case where the number of machines is $m > 2$. We strongly conjecture that the optimal policy will then continue to have a turnpike property, namely, that if at least $k(m - 1) + 1$ unstarted jobs remain, then the LAT decision is optimal. Determining what to do in states with at most $k(m-1)$ unstarted jobs will get more complicated, since the state will not be predictable until $m - 1$ long jobs have been detected. Another natural generalization would be the case where the running times have a two-point distribution on the integers a, b with $1 < a < b$. We conjecture that even then a turnpike policy will continue to be optimal, but so far a proof has eluded us. A final extension which is worth mentioning, is the case where the running-times follow a p-point distribution, $p > 2$.

References

[1] J. L. Bruno, P. J. Downey, and G. N. Frederickson. Sequencing tasks with exponential service times to minimize the expected flow time or makespan. *Assoc. Comput. Mach.*, 28:100–113, 1981.

[2] E. G. Coffman, Jr., editor. *Computer and Job-Shop Scheduling Theory*. John Wiley and Sons, 1976.

[3] E. G. Coffman, Jr., L. Flatto, M. R. Garey, and R. R. Weber. Minimizing expected makespans on uniform processor systems. *Adv. Appl. Prob.*, 19:177–201, 1987.

[4] E. G. Coffman, Jr., M. Hofri, and G. Weiss. Scheduling stochastic jobs with a two-point distribution on two parallel machines. *Probability in the Engineering and Informational Sciences*, 3:89–116, 1989.

[5] K. D. Glazebrook. Scheduling tasks with exponential service times on parallel processors. *J. Appl. Prob.*, 16:658–689, 1979.

[6] R. McNaughton. Scheduling with deadlines and loss functions. *Management Science*, 1959.

[7] C. Papadimitriou. Games against nature. *Journal of Computer and System Sciences*, 31:288–301, 1985.

[8] M. H. Rothkopf. Scheduling independent tasks on parallel processors. *Management Science*, 1966.

[9] R. R. Weber. Scheduling jobs with stochastic processing requirements on parallel machines to minimize makespan or flow time. *J. Appl. Prob.*, 19:167–182, 1982.

[10] R. R. Weber, P. Varaiya, and J. Walrand. Scheduling jobs with stochastically ordered processing times on parallel machines to minimize expected flow time. *J. Appl. Prob.*, 23:841–847, 1986.

[11] G. Weiss and M. Pinedo. Scheduling tasks with exponential service times on non-identical processors to minimize various cost functions. *J. Appl. Prob.*, 17:187–202, 1980.