Adv. Appl. Prob. **19**, 177–201 (1987)
Printed in N. Ireland

MINIMIZING EXPECTED MAKESPANS ON UNIFORM PROCESSOR SYSTEMS

E. G. COFFMAN, JR*
L. FLATTO* AND
M. R. GAREY,* *AT&T Bell Laboratories*
R. R. WEBER** *Queens' College, Cambridge*

Abstract

We study the problem of scheduling n given jobs on m uniform processors to minimize expected makespan (maximum finishing time). Job execution times are not known in advance, but are known to be exponentially distributed, with identical rate parameters depending solely on the executing processor. For $m = 2$ and 3, we show that there exist optimal scheduling rules of a certain threshold type, and we show how the required thresholds can be easily determined. We conjecture that similar threshold rules suffice for $m > 3$ but are unable to prove this. However, for $m > 3$ we do obtain a general bound on problem size that permits Bellman equations to be used to construct an optimal scheduling rule for any given set of m rate parameters, with the memory required to represent that scheduling rule being independent of the number of remaining jobs.

STOCHASTIC SCHEDULING; STOCHASTIC OPTIMIZATION ALGORITHMS

1. Introduction

We study the problem of scheduling given sets of jobs on $m \geqq 2$ processors $P_1, \cdots, P_m$, which differ only in the rates at which they operate. Job execution times are not known in advance, but on a processor with rate μ they are known to be independent samples from the exponential distribution with parameter μ; i.e. $P[\text{job-length} > x] = \exp(-\mu x)$, $x \geqq 0$. Scheduling is to be non-preemptive, i.e. once a job is assigned to a processor, it must be executed to completion. The rate of P_i is denoted by μ_i, $1 \leqq i \leqq m$, and we assume the ordering $\mu_1 \geqq \mu_2 \geqq \cdots \geqq \mu_m > 0$. We shall adopt the convenient normalization $\mu_1 = 1$.

Agrawala et al. [1] define the expected flow time (sum of finishing times) as the objective function and derive an optimal scheduling rule that minimizes this expected value. For the case of $m = 2$ processors this result was generalized by Lin and Kumar [2] and Walrand [3] to systems with arrivals.

Received 12 March 1985; revision received 19 November 1985.
* Postal address: AT&T Bell Laboratories, Murray Hill, NJ 07974, USA.
** Postal address: Queens' College, Cambridge CB3 9ET, UK.

In this paper we adopt the expected makespan (maximum finishing time) as the objective function. As we shall see, the problem of finding simple, non-enumerative optimization rules under this objective function is substantially more difficult. Indeed, our results will show that an algorithm with the simplicity of that in [1] is not possible for the makespan problem. This remark will be made more concrete after the definitions of Section 2.

In Section 3 we prove a number of results that characterize optimal algorithms. From these results very efficient optimization rules for $m = 2$ and 3 are derived in Section 4. In Section 5 it is proved that feasible, though less efficient algorithms can be found for general m. Section 6 concludes the paper with a discussion of open problems.

2. Definitions

Because of our exponential assumptions, remaining execution times are independent of elapsed execution times. It follows that scheduling decisions need to be made only at job completion times, and that a system state need only specify the number of waiting jobs and which processors are currently busy (i.e. assigned jobs). This is accomplished in our state notation (α, k), where $k \geqq 0$ denotes the number of waiting jobs and $\alpha = \alpha_1, \alpha_2, \cdots, \alpha_m$ is a bit vector with $\alpha_i = 1$, $1 \leqq i \leqq m$, if and only if P_i is currently busy. We let $|\alpha| = \sum \alpha_i$ denote the number of busy processors. We write $\mathbf{0}$ and $\mathbf{1}$ when referring to the vectors of m 0's and m 1's, respectively.

A scheduling policy is defined by specifying for each state (α, k), with $\alpha \neq \mathbf{1}$ and $k > 0$, which available processors, if any, are to have waiting jobs assigned to them. Implicitly, if an assignment role decides that no assignments are to be made when the system is in state (α, k), then the system executes or performs in (α, k) until the state changes as the result of one or more job completions. States (α, k) in which the system executes will be termed *stable* states.

We define $C(\alpha, k)$ as the minimum expected makespan assuming that the system is initially in state (α, k). This of course generalizes the initial condition $\alpha = \mathbf{0}$ that would normally apply in practice. The minimum expected makespan can be effectively defined by the Bellman equations that we now develop for $C(\alpha, k)$. Let I_α be the set of indices i for which $\alpha_i = 1$ and let J_α be the set of indices for which $\alpha_i = 0$. Define ϵ_i as the vector of all 0's except for a 1 in the ith position.

First, we have

$$C(\mathbf{0}, 0) = 0. \tag{2.1}$$

Next, the state $(\alpha, 0)$ with $\alpha \neq \mathbf{0}$ is obviously stable. Therefore, $C(\alpha, 0)$, $\alpha \neq \mathbf{0}$, can be written as the expected delay, $1/\sum_{I_\alpha} \mu_i$, to the first job completion plus

the minimum expected makespan in the resulting state. The probability that the first completion is on P_j, $j \in I_\alpha$, is simply $\mu_j / \sum_{I_\alpha} \mu_i$, and hence

$$C(\alpha, 0) = \frac{1}{\sum\limits_{i \in I_\alpha} \mu_i} + \sum_{i \in I_\alpha} \frac{\mu_i}{\sum\limits_{j \in I_\alpha} \mu_j} C(\alpha - \epsilon_i, 0), \quad \alpha \neq \mathbf{0},$$

or

$$C(\alpha, 0) = \frac{1}{\sum\limits_{I_\alpha} \mu_i} \left\{ 1 + \sum_{I_\alpha} \mu_i C(\alpha - \epsilon_i, 0) \right\}, \quad \alpha \neq \mathbf{0}. \tag{2.2}$$

Since a state $(\mathbf{1}, k)$ with no available processors must also be stable, (2.2) also applies in this case; i.e.

$$C(\mathbf{1}, k) = \frac{1}{\sum\limits_{1 \leqq i \leqq m} \mu_i} \left\{ 1 + \sum_{1 \leqq i \leqq m} \mu_i C(\mathbf{1} - \epsilon_i, k) \right\}, \quad k \geqq 0. \tag{2.3}$$

As a final boundary condition we observe that the state $(\mathbf{0}, k)$ is unstable for all $k \geqq 1$, and an assignment must be made to at least one processor. Thus,

$$C(\mathbf{0}, k) = \min_{1 \leqq i \leqq m} C(\epsilon_i, k-1), \quad k \geqq 1. \tag{2.4}$$

In the general case, $C(\alpha, k)$ can be expressed as the minimum over the two choices determined by whether or not (α, k) is taken as stable. Thus,

$$C(\alpha, k) = \min \left\{ \frac{1}{\sum\limits_{I_\alpha} \mu_i} \left(1 + \sum_{I_\alpha} \mu_i C(\alpha - \epsilon_i, k) \right), \min_{i \in J_\alpha} C(\alpha + \epsilon_i, k-1) \right\}, \tag{2.5}$$

$$k \geqq 1, \quad \alpha \neq \mathbf{0}, \mathbf{1}.$$

Note that evaluations of the recurrence represented by (2.1)–(2.5) must be performed in increasing lexicographic order of the pairs $(k, |\alpha|)$. For this reason we shall refer to the pair $(k, |\alpha|)$ as the *size* of state (α, k).

In terms of (2.5) state (α, k) is stable if $C(\alpha, k) = (1 + \sum_{I_\alpha} \mu_i C(\alpha - \epsilon_i, k)) / \sum_{I_\alpha} \mu_i$. It will be convenient to refer to (α, k) as *weakly* stable if it is stable and $C(\alpha, k) = C(\alpha + \epsilon_i, k-1)$ for some $i \in J_\alpha$; i.e. the decision to execute in state (α, k) is not uniquely optimal. The state (α, k) is *strongly* stable if it is stable, but not weakly stable.

Equations (2.1)–(2.5) will be referred to collectively as the Bellman equations. clearly, for any given initial state (α, k) the Bellman equations allow us to compute an optimal policy inductively from the optimal policies for all smaller states. The policy is representable as a transition function that

defines a stable successor state for (α, k) and each state smaller than (α, k). However, to represent such a transition function for scheduling a set of n jobs, it may be necessary to retain a list of (state, stable successor state) pairs whose length is on the order of $n2^m$, the number of possible states. As in [1] our objective is an optimal policy whose transition function can be represented in $O(2^m)$ space, independent of n and the initial state. For example, in [1] a policy of the following threshold type was found to be optimal: a waiting job is assigned to the fastest available processor, P_j, if and only if the number, k, of waiting jobs satisfies

$$k > \frac{\mu_1 + \cdots + \mu_{j-1}}{\mu_j} - (j-1).$$

According to this rule, if ever a processor P_i is allowed to remain idle while other processors are busy executing jobs, then no waiting job will be assigned to P_i throughout the remainder of the schedule.

Unfortunately, a threshold rule of this simplicity is not possible for our problem, except when $m = 2$. The above monotonicity property does not apply in general. In particular, for $m = 3$ we shall illustrate in Section 4 choices for μ_1, μ_2, μ_3 and k such that an optimal policy must not assign a waiting job to P_2 in state $(101, k)$, but it must do so in state $(100, k)$, which would follow $(101, k)$ with a job completion on P_3. Moreover, there are initial states such that $(101, k)$ is reached with positive probability.

However, in Section 5 we shall prove that there exists an integer, K, which is a function only of $\mu_1, \cdots, \mu_m$, such that the optimal decision in any state (α, k), $k > K$, is to assign a waiting job to *every* available processor in α. Thus, after a calculation using the Bellman equations, whose time complexity is bounded by a function only of $\mu_1, \cdots, \mu_m$, we can obtain an optimal transition function representable in $O(K2^m)$ space.

For the cases $m = 2$ and 3 we shall show that substantial improvements are possible. In particular, Section 4 shows that a more general policy of threshold type can be proved optimal for $m = 2$ and 3. With these policies, thresholds are identified with processor states rather than processors. In particular, for each processor state $\alpha \neq \mathbf{1}$ there will exist a threshold $t_\alpha \geqq 0$ such that a waiting job is assigned to a fastest available processor if and only if the number of waiting jobs exceeds t_α.

The following notation will be helpful in the remainder of the paper. We write $(\alpha, k) \subseteq (\beta, l)$ if and only if $\alpha_i \leqq \beta_i$, $1 \leqq i \leqq m$, and $|\alpha| + k = |\beta| + l$; i.e. we can go from (α, k) to (β, l) by assigning jobs to some of the idle processors. Accordingly, $(\alpha, k) \subseteq (\beta, l)$ implies that $C(\alpha, k) \leqq C(\beta, l)$. Moreover, for every (α, k) there exists a stable state (β, l) such that $(\alpha, k) \subseteq (\beta, l)$ and $C(\alpha, k) = C(\beta, l)$. Similarly, if $(\alpha, k) \subseteq (\gamma, j) \subseteq (\beta, l)$ and $C(\alpha, k) = C(\beta, l)$, then $C(\alpha, k) = C(\gamma, j)$.

We conclude this section with a useful property of the maximum of independent random variables.

Lemma 2.1. Let T and T' be the times required to finish the jobs executing on $P_1, \cdots, P_j$ at rates $\mu_1, \cdots, \mu_j$ and $\mu_1', \cdots, \mu_j'$, respectively. If $\mu_i \leqq \mu_i'$, $1 \leqq i \leqq j$, with inequality holding for at least one i, then $E(T) > E(T')$.

A direct proof of this result is easily supplied. However, as we shall see, it is a special case of (3.1) below.

3. Properties of an optimal algorithm

Several simplifying properties of optimal algorithms will be proved in this section for general $m \geqq 2$. In Section 4 we shall see that these properties establish rather easily the existence of simple threshold rules for $m = 2$ and 3. As our first objective, Theorem 3.1 will show that an optimal policy must always keep P_1 busy whenever jobs are waiting to be assigned.

It is convenient in the proof of Theorem 3.1 to make use of the intuitive observation, $C(\alpha, k+1) \geqq C(\alpha, k)$, $k \geqq 0$, i.e. for a given processor state, increasing the number can only increase the expected makespan under an optimal algorithm. After a preliminary result in Lemma 3.1, we shall in fact show that this inequality is strict and give a tight (positive) lower bound to the difference $C(\alpha, k+1) - C(\alpha, k)$.

Let us define the number

$$\Delta \equiv \Delta(\mu_2, \cdots, \mu_m) = 1 - \sum_{2 \leqq j \leqq m} \frac{1}{1+\mu_j} + \sum_{2 \leqq i < j \leqq m} \frac{1}{1+\mu_i+\mu_j} - \cdots \pm \frac{1}{1+\mu_2+\cdots+\mu_m}.$$

It is easy to verify that Δ may be represented more compactly as the integral

$$\Delta = (m-1)! \int_0^{\mu_2} \cdots \int_0^{\mu_m} \frac{dx_2 \cdots dx_m}{(1+x_2+\cdots+x_m)^m},$$

from which $\Delta > 0$ follows directly.

Lemma 3.1. Let $\alpha \neq \mathbf{1}$. Then for each $j \in J_\alpha$

(i) $C(\alpha + \epsilon_j, 0) - C(\alpha, 0) \geqq C(\mathbf{1}, 0) - C(\mathbf{1} - \epsilon_1, 0) = \Delta$,
(ii) $C(\alpha + \epsilon_j, 0) - C(\alpha, 0) \leqq C(\epsilon_j, 0) - C(\mathbf{0}, 0) = 1/\mu_j$.

Proof. For $\alpha \neq \mathbf{0}$ let T_i, $i \in I_\alpha$, denote the time required to finish the job currently assigned to P_i. Then

$$(3.1) \quad C(\alpha, 0) = \int_0^\infty P\left[\max_{I_\alpha} T_i > t\right] dt = \int_0^\infty \left[1 - \prod_{I_\alpha} (1 - \exp(-\mu_i t))\right] dt,$$

with a similar formula holding for $C(\alpha + \epsilon_j, 0)$. Hence, for $\alpha \neq \mathbf{0}$

$$C(\alpha + \epsilon_j, 0) - C(\alpha, 0) = \int_0^\infty \exp(-\mu_j t) \prod_{I_\alpha} (1 - \exp(-\mu_i t))\, dt. \tag{3.2}$$

We extend this formula to $\alpha = \mathbf{0}$ simply by defining $\prod_{I_0}(\cdot) = 1$. Using (3.2) we get (ii) from

$$C(\alpha + \epsilon_j, 0) - C(\alpha, 0) \leqq \int_0^\infty \exp(-\mu_j t)\, dt = \frac{1}{\mu_j}.$$

To derive (i) we bound the integrand in (3.2) as follows:

$$\begin{aligned} \exp(-\mu_j t) \prod_{I_\alpha} (1 - \exp(-\mu_i t)) &\geqq \exp(-\mu_j t) \prod_{i \neq j} (1 - \exp(-\mu_i t)) \\ &= \frac{1}{\exp(\mu_i t) - 1} \prod_{i=1}^{m} (1 - \exp(-\mu_i t)) \\ &\geqq \frac{1}{\exp(t) - 1} \prod_{i=1}^{m} (1 - \exp(-\mu_i t)). \end{aligned} \tag{3.3}$$

From (3.2) and (3.3) we conclude that

$$\begin{aligned} C(\alpha + \epsilon_j, 0) - C(\alpha, 0) &\geqq C(\mathbf{1}, 0) - C(\mathbf{1} - \epsilon_1, 0) \\ &= \int_0^\infty \exp(-t) \prod_{i=2}^{m} (1 - \exp(-\mu_i t))\, dt. \end{aligned}$$

Finally, the integral can be expanded to

$$\int_0^\infty \exp(-t) \prod_{i=2}^{m} (1 - \exp(-\mu_i t))\, dt = \int_0^\infty \Big[\exp(-t) - \sum_{i=2}^{m} \exp(-(1 + \mu_i)t) + \cdots \pm \exp(-(1 + \mu_2 + \cdots + \mu_m)t) \Big]\, dt,$$

from which $C(\mathbf{1}, 0) - C(\mathbf{1} - \epsilon_1, 0) = \Delta$ follows easily.

Lemma 3.2. For all α and k

$$C(\alpha, k+1) - C(\alpha, k) \geqq C(\mathbf{1} - \epsilon_1, 1) - C(\mathbf{1} - \epsilon_1, 0) = \Delta > 0.$$

Proof. It suffices to prove $C(\alpha, k+1) - C(\alpha, k) \geqq \Delta$, for then

$$\Delta \leqq C(\mathbf{1} - \epsilon_1, 1) - C(\mathbf{1} - \epsilon_1, 0) \leqq C(\mathbf{1}, 0) - C(\mathbf{1} - \epsilon_1, 0) = \Delta,$$

so that $\Delta = C(\mathbf{1} - \epsilon_1, 1) - C(\mathbf{1} - \epsilon_1, 0)$.

We proceed by induction on the size, $(k, |\alpha|)$, of state (α, k). The result holds for $k = 0$ and $\alpha = \mathbf{0}$, since by Lemma 3.2(i) $C(\mathbf{0}, 1) - C(\mathbf{0}, 0) = C(\epsilon_1, 0) - C(\mathbf{0}, 0) \geqq \Delta$.

Consider $k = 0$ and suppose the result holds for all states smaller than $(\alpha, 0)$.

We have

$$C(\alpha, 0) = \frac{1}{\sum_{I_\alpha} \mu_j}\left[1 + \sum_{I_\alpha} \mu_j C(\alpha - \epsilon_j, 0)\right]. \tag{3.4}$$

If $C(\alpha, 1) = C(\alpha + \epsilon_j, 0)$ for some $j \in J_\alpha$, then by Lemma 3.1 $C(\alpha, 1) - C(\alpha, 0) = C(\alpha + \epsilon_j, 0) - C(\alpha, 0) \geqq \Delta$, as desired. But if $(\alpha, 1)$ is stable and

$$C(\alpha, 1) = \frac{1}{\sum_{I_\alpha} \mu_j}\left[1 + \sum_{I_\alpha} \mu_j C(\alpha - \epsilon_j, 1)\right],$$

then by (3.4) and the inductive hypothesis

$$C(\alpha, 1) - C(\alpha, 0) = \frac{1}{\sum_{I_\alpha} \mu_j} \sum_{I_\alpha} \mu_j [C(\alpha - \epsilon_j, 1) - C(\alpha - \epsilon_j, 0)] \geqq \Delta,$$

again as desired.

Finally, suppose the result holds for all states smaller than (α, k) where $k \geqq 1$. We have

$$C(\alpha, k) = \min\left\{\frac{1}{\sum_{I_\alpha} \mu_j}\left[1 + \sum_{I_\alpha} \mu_j C(\alpha - \epsilon_j, k)\right], \min_{J_\alpha} C(\alpha + \epsilon_j, k - 1)\right\}. \tag{3.5}$$

By (3.5) and the inductive hypothesis, if $C(\alpha, k+1) = C(\alpha + \epsilon_j, k)$ for some $j \in J_\alpha$, then

$$C(\alpha, k+1) - C(\alpha, k) \geqq C(\alpha + \epsilon_j, k) - C(\alpha + \epsilon_j, k-1) \geqq \Delta,$$

whereas if

$$C(\alpha, k+1) = \frac{1}{\sum_{I_\alpha} \mu_j}\left[1 + \sum_{I_\alpha} \mu_j C(\alpha - \epsilon_j, k)\right],$$

then

$$C(\alpha, k+1) - C(\alpha, k) \geqq \frac{1}{\sum_{I_\alpha} \mu_j} \sum_{I_\alpha} \mu_j [C(\alpha - \epsilon_j, k+1) - C(\alpha - \epsilon_j, k)] \geqq \Delta.$$

Theorem 3.1. In (α, k) let $\alpha_1 = 0$ and $k \geqq 1$. Then (α, k) is unstable.

Corollary 3.1. From this result and Lemma 2.1 we conclude that P_1 should be assigned a waiting job whenever it is available. Hence, when $\alpha_1 = 0$, $C(\alpha, k) = C(\alpha + \epsilon_1, k-1)$. It also follows that, in the Bellman equations, (2.4) can be simplified to $C(\mathbf{0}, k) = C(\epsilon_1, k-1)$, $k \geqq 1$.

Proof of the theorem. The result holds for $\alpha = 0$ and $k = 1$ since $(\mathbf{0}, 1)$ is unstable. Thus, suppose the result holds for all states smaller than (α, k), and assume (α, k) is stable. Then $|\alpha| > 0$ and we conclude from the remark above that our inductive hypothesis can be written as

$$C(\alpha, k) = \frac{1}{\sum_{I_\alpha} \mu_i}\left[1 + \sum_{I_\alpha} \mu_i C(\alpha - \epsilon_i, k)\right]$$

(3.6)

$$= \frac{1}{\sum_{I_\alpha} \mu_i}\left[1 + \sum_{I_\alpha} \mu_i C(\alpha - \epsilon_i + \epsilon_1, k-1)\right].$$

Now let S_1 be the following scheduling policy with the initial state (α, k). In addition to executing the jobs initially on P_i, $i \in I_\alpha$, S_1 also executes a job on P_1, thus reducing the number of waiting jobs to $k-1$. No further assignments are made by S_1 until some P_i $(i \neq 1)$ finishes its job, at which point S_1 proceeds optimally; i.e. in any such resulting state S_1 is assumed to be an optimal policy. Let $C_1(\alpha, k)$ be the expected makespan under S_1. The probability that P_1 finishes before any P_i, $i \in I_\alpha$, is $1/(1 + \sum_{I_\alpha} \mu_i)$. Hence,

$$C_1(\alpha, k) = C_1(\alpha + \epsilon_1, k-1) = \frac{1}{\sum_{I_\alpha} \mu_i}\left[1 + \frac{1}{1 + \sum_{I_\alpha} \mu_i} \sum_{I_\alpha} \mu_i C(\alpha - \epsilon_i, k-1)\right.$$

(3.7)

$$\left. + \frac{\sum_{I_\alpha} \mu_i}{1 + \sum_{I_\alpha} \mu_i} \sum_{I_\alpha} \mu_i C(\alpha - \epsilon_i + \epsilon_1, k-1)\right].$$

From (3.6), (3.7) and Lemma 3.2 we obtain the contradiction $C_1(\alpha, k) < C(\alpha, k)$. Hence, (α, k) is not stable.

Another intuitive result that we shall now prove is that if waiting jobs are assigned in any state (α, k) by an optimal policy, then they must be assigned to fastest available processors. This property follows easily from the following result.

Theorem 3.2. Let α be such that $p, q \in J_\alpha$, $\mu_p \geqq \mu_q$. Then $C(\alpha + \epsilon_p, k) \leqq C(\alpha + \epsilon_q, k)$ for all k.

Proof. For $k = 0$ the theorem follows immediately from Lemma 2.1. Thus, let $k > 0$ and suppose the theorem holds for any state (lexicographically) smaller than (α, k). For $\beta \neq 0$ let

$$C'(\beta, l) = \frac{1}{\sum_{I_\beta} \mu_i}\left[1 + \sum_{I_\beta} \mu_i C(\beta - \epsilon_i, l)\right].$$

The Bellman equations give

$$C(\alpha+\epsilon_p, k) = \min\Big\{C'(\alpha+\epsilon_p, k),\ C(\alpha+\epsilon_p+\epsilon_q, k-1),\ \min_{J_\alpha-\{p,q\}} C(\alpha+\epsilon_p+\epsilon_i, k-1)\Big\},$$

$$C(\alpha+\epsilon_q, k) = \min\Big\{C'(\alpha+\epsilon_q, k),\ C(\alpha+\epsilon_p+\epsilon_q, k-1),\ \min_{J_\alpha-\{p,q\}} C(\alpha+\epsilon_q+\epsilon_i, k-1)\Big\}.$$

The theorem for (α, k) will therefore follow from $C'(\alpha+\epsilon_p, k) \leqq C'(\alpha+\epsilon_q, k)$.

From the formulas for $C'(\alpha+\epsilon_p, k)$ and $C'(\alpha+\epsilon_q, k)$ we get

$$\Big(\mu_p + \sum_{I_\alpha} \mu_i\Big)C'(\alpha+\epsilon_p, k) - \Big(\mu_q + \sum_{I_\alpha} \mu_i\Big)C'(\alpha+\epsilon_q, k)$$

$$= \sum_{I_\alpha} \mu_i[C(\alpha+\epsilon_p-\epsilon_i, k) - C(\alpha+\epsilon_q-\epsilon_i, k)] + (\mu_p-\mu_q)C(\alpha, k),$$

which we rewrite as

$$\Big(\mu_p + \sum_{I_\alpha} \mu_i\Big)[C'(\alpha+\epsilon_p, k) - C'(\alpha+\epsilon_q, k)]$$

$$= (\mu_p-\mu_q)[C(\alpha, k) - C'(\alpha+\epsilon_q, k)] + \sum_{I_\alpha} \mu_i[C(\alpha+\epsilon_p-\epsilon_i, k) - C(\alpha+\epsilon_q-\epsilon_i, k)]. \tag{3.8}$$

By Lemma 3.2

$$C(\alpha, k) \leqq C'(\alpha+\epsilon_q, k-1) \leqq C'(\alpha+\epsilon_q, k).$$

Hence the first term on the right of (3.8) is negative. The second term on the right of (3.8) is negative by assumption. We conclude from (3.8) that $C'(\alpha+\epsilon_p, k) \leqq C'(\alpha+\epsilon_q, k)$.

With Theorem 3.2 a further simplification of the Bellman equations is possible. In (2.5) the expression $\min_{i\in J_\alpha} C(\alpha+\epsilon_i, k-1)$ can be replaced by $C(\alpha+\epsilon_p, k-1)$, where p is the least index in J_α.

Next, we shall identify additional sets of states for which optimal decisions have the structure of a threshold rule, i.e. those states (α, k) for which there are thresholds $t_\alpha \geqq 0$ such that (α, k) is unstable if and only if $k > t_\alpha$. Note that if $\alpha_1 = 0$, then $t_\alpha = 0$ by Theorem 3.1. We now consider states (ϵ_1, k), $k \geqq 1$.

Theorem 3.3. Let $k \geqq 1$. If (ϵ_1, k) is stable, then $(\epsilon_1, k-1)$ is strongly stable.

Proof. Suppose (ϵ_1, k) is stable but $(\epsilon_1, k-1)$ is not strongly stable. Then $k \geqq 2$. By the Bellman equations, Corollary 3.1 and Theorem 3.2

$$C(\epsilon_1, k) = 1 + C(\mathbf{0}, k) = 1 + C(\epsilon_1, k-1) = 1 + C(\epsilon_1 + \epsilon_2, k-2). \tag{3.9}$$

Let S_1 be the following policy starting in state (ϵ_1, k). In addition to the job on P_1 suppose that S_1 executes a job on P_2, thus reducing the number of waiting jobs to $k-1$. When either P_1 or P_2 finishes, S_1 assigns a new job to the processor just finished and then proceeds optimally. If $C_1(\epsilon_1, k) = C_1(\epsilon_1 + \epsilon_2, k-1)$ denotes the expected makespan under S_1, then

$$C_1(\epsilon_1, k) = \frac{1}{1+\mu_2} + C(\epsilon_1 + \epsilon_2, k-2). \tag{3.10}$$

From (3.9) and (3.10) we get the contradiction, $C_1(\epsilon_1, k) < C(\epsilon_1, k)$. Hence, if (ϵ_1, k) is stable, then $(\epsilon_1, k-1)$ is strongly stable.

It follows immediately from Theorems 3.1 and 3.3 that an optimal threshold rule exists for $m = 2$. In particular, $t_{00} = t_{01} = 0$ by Theorem 3.1, and by Theorem 3.3 either $(10, k)$ is stable for all $k \geqq 0$, or there exists a threshold t_{10} such that in state $(10, k)$ a job is assigned to P_2 if and only if $k > t_{10}$. In the next section we shall express t_{10} as a simple function of θ_2, which is finite for all $\mu_2 > 0$.

For general m. Theorems 3.1 and 3.3 show that any state (α, k) for which $|\alpha| \leqq 1$ can be handled optimally by a threshold decision. We shall now extend this property to states (α, k) for which $|\alpha| = 2$. Then, by analogy with the above remarks for $m = 2$, we shall be able to derive an optimal threshold rule for $m = 3$. First, we need some preliminary results.

Let T_i^k denote the time required by P_i to execute k consecutive jobs. We shall write $T_i = T_i^{(1)}$. Introducing the vector notation $\boldsymbol{k} = (k_1, \cdots, k_m)$, we define $E(\boldsymbol{k}) = E(k_1, \cdots, k_m) = E(\max_{1 \leqq i \leqq m} \{T_i^{k_i}\})$. The following lemma gives the value of $E(\boldsymbol{k})$ for $1 \leqq m \leqq 3$ and certain values of $\boldsymbol{k}$ which will be useful later.

Lemma 3.3. Let $F_k(x) = 1/x(1+x)^k$. Then

(i) $E(k) = k$,
(ii) $E(k, 1) = k + F_k(\mu_2)$,
(iii) $E(k, 1, 1) = k + F_k(\mu_2) + F_k(\mu_3) - F_k(\mu_2 + \mu_3)$.

Proof. We have $T_1^k = X_1 + \cdots + X_k$ where the X_j's are independent and identically distributed with $P(X_j > t) = \exp(-t)$, $t \geqq 0$. Thus, (i) follows from

$$E(k) = E(T_1^k) = \sum_{j=1}^{k} E(X_j) = k.$$

For (ii) let $T = \max(T_1^k, T_2)$. Then

$$\begin{aligned} E(k, 1) &= \int_0^\infty P(T > t)\, dt \\ &= \int_0^\infty P(T_1^k > t)\, dt + \int_0^\infty P(T_2 > t)\, dt - \int_0^\infty P(T_1^k > t)P(T_2 > t)\, dt \\ &= E(T_1^k) + E(T_2) - \int_0^\infty \exp(-\mu_2 t) \sum_{i=0}^{k-1} \frac{t^i}{i!}\, dt \\ &= k + \frac{1}{\mu_2} - \sum_{i=0}^{k-1} \frac{1}{(1+\mu_2)^{i+1}} = k + F_k(\mu_2). \end{aligned}$$

By extending this reasoning to the calculation of $E(k, 1, 1) = E(\max(T_1^k, \max(T_2, T_3))$, we obtain (iii).

Lemma 3.4 below proves an important property of the function $E(\boldsymbol{k})$ which is then used in Lemma 3.5 to relate this function on $C(\alpha, k)$.

Lemma 3.4. For some fixed j suppose that two vectors $\boldsymbol{k}$ and $\boldsymbol{k}'$ of non-negative integers differ only in the first and jth elements, and that for these elements $k_1 = k + 1$, $k_1' = k$ and $k_j = 0$, $k_j' = 1$. Then if $E(\boldsymbol{k}) \leqq E(\boldsymbol{k}')$ when the remaining elements are chosen to be 0 (i.e. $k_i = k_i' = 0$, $i \neq 1, j$), then $E(\boldsymbol{k}) \leqq E(\boldsymbol{k}')$ must hold for all choices of $k_i = k_i' \geqq 0$, $i \neq 1, j$. Moreover, strict inequality in the first instance implies strict inequality in the second.

Proof. We shall take $j = 2$ and prove that if $E(k+1, 0, 0, \cdots, 0) \leqq E(k, 1, 0, \cdots, 0)$, then $E(k+1, 0, k_3, \cdots, k_m) \leqq E(k, 1, k_3, \cdots, k_m)$ for arbitrary non-negative integers $k_3, \cdots, k_m$, with strict inequality implying strict inequality. Examination of the proof will show that it applies *mutatis mutandis* for general $j > 1$.

Let $F_T(x) = P(T \leqq x)$ where $T = \max_{3 \leqq i \leqq m} \{T_i^{k_i}\}$. For $c \geqq 0$ define $E(k_1, k_2; c) = E(\max\{T_1^{k_1}, T_2^{k_2}, c\})$ and note that $E(k_1, k_2; 0) = E(k_1, k_2)$. We have

$$E(k+1, 0, k_3, \cdots, k_m) = \int_0^\infty E(k+1, 0; c)\, dF_T(c)$$

and

$$E(k, 1, k_3, \cdots, k_m) = \int_0^\infty E(k, 1; c)\, dF_T(c).$$

Thus, our result will follow from a proof that for any constant $c \geqq 0$, if $E(k+1, 0) \leqq E(k, 1)$, then $E(k+1, 0; c) \leqq E(k, 1; c)$ for all $k \geqq 0$, with strict inequality implying strict inequality.

To prove this define $X_c = \max(X, c)$, where X is a non-negative random

variable and $c \geqq 0$ is a constant. Then

$$E(X_c) = \int_0^\infty P(X_c > t)\, dt = \int_0^c dt + \int_c^\infty P(X > t)\, dt = E(X) + \int_0^c P(X \leqq t)\, dt.$$

Letting X first be T_1^{k+1} and then $\max(T_1^k, T_2)$ we obtain

$$E(k+1, 0; c) = k + 1 + \int_0^c P(T_1^{k+1} \leqq t)\, dt, \tag{3.11}$$

$$\begin{aligned} E(k, 1; c) &= k + \int_0^\infty \exp(-\mu_2 t) P(T_1^k \leqq t)\, dt \\ &\quad + \int_0^c P(T_1^k \leqq t)(1 - \exp(-\mu_2 t))\, dt \\ &= k + \int_0^c P(T_1^k \leqq t)\, dt + \int_c^\infty P(T_1^k \leqq t) \exp(-\mu_2 t)\, dt. \end{aligned} \tag{3.12}$$

Routine calculations similar to those in Lemma 3.3 yield

$$P(T_1^k \leqq x) = 1 - \sum_{j=0}^{k-1} \exp(-t) \frac{t^j}{j!} \tag{3.13}$$

and

$$\int_0^\infty \exp(-\mu_2 t) P(T_1^k \leqq t)\, dt = \frac{1}{\mu_2 (1 + \mu_2)^k}, \tag{3.14}$$

where the sum in (3.13) is interpreted to be 0 for $k = 0$. From (3.11)–(3.14) it follows that

$$\begin{gathered} E(k+1, 0) \leqq E(k, 1) \text{ if and only if} \\ 0 \leqq \mu_2 \leqq a \quad \text{where} \quad a(1+a)^k = 1, \qquad a > 0, \end{gathered} \tag{3.15}$$

and

$$\begin{gathered} E(k+1, 0; c) \leqq E(k, 1; c) \text{ if and only if} \\ \int_0^c [P(T_1^k \leqq t) - P(T_1^{k+1} \leqq t)]\, dt + \int_c^\infty P(T_1^k \leqq t) \exp(-\mu_2 t)\, dt \geqq 1. \end{gathered} \tag{3.16}$$

We observe here that $a = 1$ for $k = 0$ and $a < 1$ for $k \geqq 1$. We conclude from (3.13)–(3.16) that a proof of our result is equivalent to showing that $G(r, c) \geqq 1$ for $0 < r \leqq a$, $0 \leqq c < \infty$, where

$$G(r, c) = \int_0^c \frac{\exp(-t) t^k}{k!}\, dt + \int_c^\infty \left[1 - \sum_{j=0}^{k-1} \frac{\exp(-t) t^j}{j!}\right] \exp(-rt)\, dt.$$

For this purpose we note first that

$$G(r, 0) = \frac{1}{r(1+r)^k} \geqq 1, \qquad 0 < r \leqq a,$$
$$G(r, \infty) = \int_0^\infty \frac{\exp(-t)t^k}{k!}\, dt = 1. \tag{3.17}$$

In the following let a subscript c denote the derivative with respect to c. If $k = 0$, Then $G_c = \exp(-c) - \exp(-rc) \leqq 0$, so that $G(r, c) \geqq G(r, \infty) = 1$. If $k \geqq 1$, then

$$G_c = \exp(-rc) f(r, c), \quad f(r, c) \equiv \exp(-(1-r))\frac{c^k}{k!} + \exp(-c)\sum_{j=0}^{k-1}\frac{c^j}{j!} - 1$$
$$f_c = \exp(-(1-r)c)\frac{c^{k-1}}{(k-1)!} g(r, c), \quad g(r, c) \equiv 1 - (1-r)\frac{c}{k} - \exp(-rc). \tag{3.18}$$

For fixed $0 < r \leqq a < 1$, $g_c(r, c) = -(1-r)/k + r\exp(-rc)$ is a decreasing function of c with

$$g_c(r, 0) = \frac{(k+1)r - 1}{k}, \qquad g_c(r, \infty) = -\frac{1-r}{k} < 0.$$

Thus, if $r \leqq 1/(k+1)$ then $g(r, c) < 0$ for $0 < c < \infty$ and if $r > 1/(k+1)$ then there exists a function of r only, $c_0(r) > 0$, such that $g(r, c) > 0$ for $0 < c < c_0(r)$ and $g(r, c) < 0$ for $c > c_0(r)$. Integrating first f_c and then G_c with respect to c, we reach a similar conclusion for $G(r, c)$, $c_0(r)$ being replaced by a similar function $c_1(r) > 0$. From this fact and (3.17) it follows that $G(r, c) \geqq 1$ for $0 < r \leqq a$, $0 \leqq c < \infty$.

It remains to show that strict inequality implies strict inequality. First, $E(k+1, 0) < E(k, 1)$ means that $0 < \mu_2 < a$. Differentiating $G(r, c)$ with respect to r we find

$$G_r(r, c) = \int_c^\infty -t\left[1 - \sum_{j=0}^{k-1}\exp(-t)\frac{t^j}{j!}\right]\exp(-rt)\, dt < 0$$

so that for $0 < r < a$, $G(r, c) > G(a, c) \geqq 1$. We conclude that $E(k+1, 0; c) < E(k, 1; c)$.

It is worth remarking that Lemma 3.4 is not true in general; the reader will have little difficulty in finding other distributions of execution time for which the lemma is false. The next result establishes the proper connection between the functions $E(\boldsymbol{k})$ and $C(\alpha, k)$.

Lemma 3.5. Let $\alpha_1 = 1$. If (ϵ_1, k) is stable, then $C(\alpha, k) = E(k+1,$

$\alpha_2, \cdots, \alpha_m$). Furthermore, if (ϵ_1, k) is stable (strongly stable), then (α, k) is stable (strongly stable).

Proof. For $k = 0$ the result is trivial so let $k \geqq 1$ and suppose the result holds for all states smaller than (α, k). Consider first $2 \leqq |\alpha| \leqq m - 1$.

Suppose (ϵ_1, k) is stable. by Theorem 3.3, $(\epsilon_1, k-1)$ is also stable. Let S_1 be a policy that executes in (α, k) and is otherwise optimal. Let S_2 be a policy that in state (α, k) makes an assignment to P_j, $j \in J_\alpha$, and is optimal thereafter. Let $C_1(\alpha, k)$ and $C_2(\alpha, k)$ denote the respective expected makespans of S_1 and S_2 with the initial state (α, k).

Letting $\boldsymbol{k} = (k+1, \alpha_2, \cdots, \alpha_m)$ we have by the inductive hypothesis and Corollary 3.1

$$
\begin{aligned}
C_1(\alpha, k) &= \frac{1}{\sum_{I_\alpha} \mu_i}\left[1 + C(\alpha, k-1) + \sum_{I_\alpha - \{1\}} \mu_i C(\alpha - \epsilon_i, k)\right] \\
&= \frac{1}{\sum_{I_\alpha} \mu_i}\left[1 + \sum_{I_\alpha} \mu_i E(\boldsymbol{k} - \epsilon_1)\right] = E(\boldsymbol{k}).
\end{aligned} \tag{3.19}
$$

Again by the inductive hypothesis

$$
C_2(\alpha, k) = C(\alpha + \epsilon_j, k-1) = E(\boldsymbol{k} - \epsilon_1 + \epsilon_j). \tag{3.20}
$$

Since (ϵ_1, k) is stable, Theorem 3.3 gives

$$
C(\epsilon_1, k) = E(k+1, 0, \cdots, 0) \leqq E(k, 0, \cdots, 1, \cdots, 0) \tag{3.21}
$$

where the 1 in the last expression appears in position j. Moreover, (3.21) becomes an inequality if (ϵ_1, k) is strongly stable. We conclude from (3.21) and Lemma 3.4 that

$$
E(\boldsymbol{k}) \leqq E(\boldsymbol{k} - \epsilon_1 + \epsilon_j), \tag{3.22}
$$

inequality holding if (ϵ_1, k) is strongly stable. Clearly, (3.19), (3.20) and (3.22) jointly imply that $C(\alpha, k) = C_1(\alpha, k) = E(k+1, \alpha_2, \cdots, \alpha_m)$ and that (α, k) is stable (strongly stable) when (ϵ_1, k) is stable (strongly stable).

Finally, if $|\alpha| = m$, then we apply (3.19) to $C(\alpha, k) = C_1(\alpha, k)$ to obtain $C(\alpha, k) = E(\boldsymbol{k})$.

We are now poised for our next threshold result.

Theorem 3.4. Let $\alpha_1 = 1$ and $|\alpha| = 2$. For all $k \geqq 0$, if $(\alpha, k+1)$ is stable, then (α, k) is strongly stable.

Remark. We can prove this type of result for yet another special case: *if* $(\alpha, 1)$ *is unstable then so is* $(\alpha, 2)$. The proof appears to require considerable

effort; however, since the result is not needed in what follows, we omit the proof.

Proof. It is convenient to consider separately the two cases $\alpha_2 = 0, 1$. In both cases we suppose that (α, k) is not strongly stable, so that $k \geqq 1$, and show that $(\alpha, k+1)$ is unstable.

Case 1: ($\alpha_2 = 1$ and hence $\alpha_j = 0$, $j > 2$). Let S_1 be a policy that executes in $(\alpha, k+1)$ and is otherwise optimal. If the expected makespan under S_1 is denoted by $C_1(\cdot)$, then we can write

$$C_1(\alpha, k+1) = \frac{1}{1+\mu_2}[1 + C(\alpha, k) + \mu_2 C(\epsilon_1, k+1)]. \tag{3.23}$$

Since (α, k) is not strongly stable we have that $C(\alpha, k) = C(\alpha + \epsilon_3, k-1)$ by Theorem 3.2. Moreover, (ϵ_1, k) cannot be strongly stable by Lemma 3.5, and this in turn implies that $(\epsilon_1, k+1)$ is unstable by Theorem 3.3. Hence, $C(\epsilon_1, k+1) = C(\alpha, k)$ by Theorem 3.2. We conclude from these facts and (3.23) that

$$C_1(\alpha, k+1) = \frac{1}{1+\mu_2} + C(\alpha + \epsilon_3, k-1). \tag{3.24}$$

Now let S_2 be the following policy: in state $(\alpha, k+1)$ S_2 assigns a job to P_3 and executes in state $(\alpha + \epsilon_3, k)$. Upon the first completion, s_2 assigns a job to the processor that just finished a job and then executes again in state $(\alpha + \epsilon_3, k-1)$. Apart from the above prescriptions, S_2 makes optimal decisions. If $C_2(\cdot)$ denotes the expected makespan under S_2, then

$$C_2(\alpha, k+1) = C_2(\alpha + \epsilon_3, k) = \frac{1}{1+\mu_2+\mu_3} + C(\alpha + \epsilon_3, k-1). \tag{3.25}$$

Comparison of (3.24) and (3.25) yields $C_2(\alpha, k+1) < C_1(\alpha, k+1)$. Thus, $(\alpha, k+1)$ is unstable and the theorem follows.

Case 2: ($\alpha_2 = 0$ and hence $\alpha_j = 1$ for exactly one $j \neq 1, 2$). With S_1 and C_1 as in Case 1, we have

$$C_1(\alpha, k+1) = \frac{1}{1+\mu_j}[1 + C(\alpha, k) + \mu_j C(\epsilon_1, k+1)]. \tag{3.26}$$

Since (α, k) is not strongly stable, we have $C(\alpha, k) = C(\alpha + \epsilon_2, k-1)$. Also, (ϵ_1, k) is not strongly stable and this in turn implies that $(\epsilon_1, k+1)$ is unstable. Hence, $C(\epsilon_1, k+1) = C(\epsilon_1 + \epsilon_2, k)$ so that

$$C_1(\alpha, k+1) = \frac{1}{1+\mu_j}[1 + C(\alpha + \epsilon_2, k-1) + \mu_j C(\epsilon_1 + \epsilon_2, k)]. \tag{3.27}$$

Next, let S_2 be the following policy: in $(\alpha, k+1)$ S_2 assigns a job to P_2 and then executes in the stable state $(\alpha+\epsilon_2, k)$. In subsequent states S_2 performs optimally. If $C_2(\cdot)$ denotes the expected makespan under S_2, then we have

$$C_2(\alpha, k+1) = \frac{1}{1+\mu_2+\mu_j}[1 + C(\alpha+\epsilon_2, k-1) + \mu_2 C(\alpha, k) + \mu_j C(\epsilon_1+\epsilon_2, k)]. \tag{3.28}$$

By (3.27) and (3.28) $C_2(\alpha, k+1) < C_1(\alpha, k+1)$ is equivalent to

$$C(\alpha, k) < \frac{1}{1+\mu_j}[1 + C(\alpha+\epsilon_2, k-1) + \mu_j C(\epsilon_1+\epsilon_2, k)]. \tag{3.29}$$

To verify (3.29) let S_3 be the policy that executes in $(\alpha, k+1)$, assigns a job to P_2 after the first completion and proceeds optimally thereafter. The expected makespan, $C_3(\cdot)$, under S_3 is given by the right-hand side of (3.29). Thus, (3.29) follows from

$$C(\alpha, k) < C(\alpha, k+1) \leqq C_3(\alpha, k+1).$$

We conclude that $C_2(\alpha, k+1) < C_1(\alpha, k+1)$, i.e. $(\alpha, k+1)$ is unstable.

For $m=3$, $\alpha \neq \mathbf{1}$ implies $|\alpha| \leqq 2$. Thus, a threshold rule for $m=3$ follows immediately from Theorems 3.1, 3.3 and 3.4. The thresholds that have yet to be calculated are t_α for $\alpha = (100)$, (110) and (101). In Section 4 we shall verify that these thresholds are bounded for fixed $\mu_3 > 0$. Detailed calculations will also be discussed.

Although we conjecture that for all m every state is a threshold state, we have been unable to prove this; the methods used here appear quite inadequate for the general case. We shall return to this conjecture in the final section.

4. Optimal policies for $m = 2, 3$

We consider first the two-processor problem. From Theorem 3.1 we have $t_{00} = t_{01} = 0$, so for an optimal threshold policy it remains to find t_{10}. This is provided by the following result, where, for simplicity, the symbol r is used in place of μ_2.

Theorem 4.1. Let

$$t(r) = \frac{\log \frac{1}{r}}{\log (1+r)}. \tag{4.1}$$

Then the state $(10, k)$ is strongly stable for $k < t(r)$ and unstable for $k > t(r)$. If $t(r)$ is a positive integer then $(10, t(r))$ is weakly stable.

Remark. Clearly, the thresholds $t_{00} = t_{01} = 0$ and $t_{10} = |t(r)|$ define an optimal threshold rule. However, there are two optimal threshold rules when $t(r)$ is a positive integer; it is immaterial whether or not an assignment is made in state $(10, t(r))$, so we can choose $t_{10} = t(r)$ or $t(r) - 1$.

Proof. Let $k \geqq 1$. Let $(10, k)$ be stable and hence $C(10, k) = k + 1$. Since we are assuming $(10, k)$ is stable, we have

$$C(10, k) \leqq E(k, 1), \tag{4.2}$$

which by Lemma 3.3 is equivalent to

$$r(1 + r)^k \leqq 1. \tag{4.3}$$

Observe that (4.2) also holds when $k = 0$; thus, (4.3) holds whenever $(10, k)$ is stable. Since $r(1 + r)^k > 1$ for $k > t(r)$, we conclude that $(10, k)$ is unstable for $k > t(r)$.

Now $(10, 0)$ is strongly stable, so for the inductive hypothesis let $1 \leqq k \leqq t(r)$, and suppose that $(10, k - 1)$ is stable. By Lemmas 3.3 and 3.5

$$\begin{aligned} C(10, k - 1) &= k \\ C(11, k - 1) &= E(k, 1) = k + \frac{1}{r(1 + r)^k}. \end{aligned} \tag{4.4}$$

If $k < t(r)$ then we can conclude from (4.4) that

$$C(11, k - 1) > 1 + C(10, k - 1) \geqq C(10, k),$$

which means that $(10, k)$ is strongly stable. If $k = t(r) > 0$ then we conclude from (4.4) that

$$1 + C(10, t(r) - 1) = C(11, t(r) - 1),$$

which means that $(10, t(r))$ is weakly stable.

With t_{10} determined by Theorem 4.1, explicit forms for minimum expected makespans on two processors are easily found. Let $X_1, X_2, \cdots$ be a sequence of independent exponential random variables with rate parameter $1 + r$. Then the makespan for an initial state $(11, k)$, $k > t_{10}$, is obtained from the observation that after $k - t_{10}$ completions on P_1 and P_2, we reach the state $(11, t_{10})$ where P_2 must be executing its last job. Thus, the expected makespan is

$$C(11, k) = E[X_1 + \cdots + X_{k - t_{10}}] + E(t_{10} + 1, 1), \qquad k > t_{10}.$$

For $k \leqq t_{10}$ we easily have $C(11, k) = E(k + 1, 1)$. Thus,

$$C(11, k) = \begin{cases} E(k + 1, 1), & k \leqq t_{10} \\ E(t_{10} + 1, 1) + \dfrac{k - t_{10}}{1 + r}, & k > t_{10}. \end{cases}$$

A similar argument gives

$$C(10,k)=\begin{cases}k+1, & k\leqq t_{10}\\ \dfrac{k-t_{10}-1}{1+r}+E(t_{10}+1,1), & k>t_{10}.\end{cases} \tag{4.5}$$

In these expressions t_{10} and $E(j,1)$ are obtained from Theorem 4.1 and (4.1), respectively. Finally, $C(00,k)$ and $C(01,k)$ can be obtained from the relations $C(00,k+1)=C(10,k)$ and $C(01,k+1)=C(11,k)$.

We turn now to the case $m=3$, where r and s will be used in place of μ_2 and μ_3, respectively. Before discussing how the thresholds t_{100}, t_{101} and t_{110} can be found, we shall describe an anomaly which shows that the simpler threshold-type policy in [1] cannot be used for our problem when $m=3$. First, we need the following result.

Theorem 4.2. Let $(101,k)$ be strongly stable. According to any optimal policy P_3 must be executing its last job and P_2 cannot be used while P_3 is busy.

Proof. Suppose we start in $(101,k)$ and P_1 finishes first. By Corollary 3.1 the system transits to $(101,k-1)$, and by Theorem 3.4 this state must be strongly stable. Thus, the system must continue executing jobs in state $(101,k-1)$. Repeating this reasoning P_2 will not be used so long as P_3 is busy. When P_3 finishes the state will be either $(000,0)$ or $(100,l)$ for some $0\leqq l\leqq k$. In the latter case it follows from Theorems 3.2–3.4 that P_3 will not be assigned another job for the remainder of the schedule.

Now consider the following example. Let $a(1+a)=1$, $a>0$, and hence $a=(1+\sqrt{5})/2$. Let $r=s=a+\epsilon$, $\epsilon>0$. By Theorem 4.1 $(100,1)$ is unstable, so that

$$C(100,1)=C(110,0)=1+\frac{1}{r(1+r)},$$

$$\begin{aligned}C(101,1)&=\min\left[\frac{1}{1+r}+\frac{1}{1+r}C(101,0)+\frac{r}{1+r}C(110,0),\,C(111,0)\right]\\&=\min\left[\frac{1}{1+r}+C(110,0),\,C(111,0)\right]\\&=\min\left[1+\frac{1}{r},\,1+\frac{2}{r(1+r)}-\frac{1}{2r(1+2r)}\right].\end{aligned}$$

Using $a(1+a)=1$ the inequality

$$1+\frac{1}{a}<1+\frac{2}{a(1+a)}-\frac{1}{2a(1+2a)}$$

reduces to $a > \frac{3}{5}$. But $\frac{3}{5}(1+\frac{3}{5}) = \frac{24}{25} < 1$, so that indeed $a > \frac{3}{5}$. Thus, for $\epsilon > 0$ sufficiently small

$$1 + \frac{1}{r} < 1 + \frac{2}{r(1+r)} - \frac{1}{2r(1+r)}$$

and hence

$$C(101, 1) = \frac{1}{1+r} + C(110, 0) < C(111, 0).$$

Thus, in state $(101, 1)$ we must not use P_2 and in state $(100, 1)$, which follows $(101, 1)$ with a completion on P_3, we must use P_2 or P_3. To obtain the desired anomaly we can choose $r = s + \epsilon = a + 2\epsilon$, for it is easy to verify that for $\epsilon > 0$ sufficiently small we will reach the same conclusion, except that in state $(100, 1)$ we no longer have a choice between P_2 and P_3; we must use the faster processor P_2.

Now for k sufficiently large any optimal policy will lead with positive probability from $(000, k)$ to $(101, 1)$ so that the anomaly can occur. To verify this we note that for k large enough the initial stable state will be $(111, k-3)$. Thus by Theorem 4.1 state $(101, 1)$ will be produced by $k-4$ consecutive completions on P_1 followed by a completion on P_2.

We return now to the problem of calculating thresholds. It is readily verified that $t_{100} = t_{10}$ and therefore t_{100} can be calculated directly from Thorem 4.1. From a practical point of view the simplest approach to finding the thresholds t_{101} and t_{110} is by an evaluation of the Bellman equations. The values of $C(\alpha, k)$ are found in increasing lexicographic order of $(k, |\alpha|)$ until the first occurrences of the inequality

$$C(\alpha, k) = \min_{i \in J_\alpha} C(\alpha + \epsilon_i, k-1) < \frac{1}{\sum_{I_\alpha} \mu_i} \left[1 + \sum_{I_\alpha} \mu_i C(\alpha - \epsilon_i, k) \right]$$

have been encountered for each of $\alpha = (101), (110)$. At these times we record $t_\alpha = k - 1$.

The following result which is an easy consequence of Theorem 3.2, states that the evaluations may be terminated once the threshold t_{110} has been found.

Corollary 4.1. For all $k \geqq 0$, $C(110, k) \leqq C(101, k)$, and if $(101, k)$ is stable then so is $(110, k)$.

Next, for $m = 3$ we shall provide a bound on the number of states for which $C(\alpha, k)$ must be evaluated; as we shall see this bound is significantly tighter than the general bound proved in the next section.

Theorem 4.3. States $(100, k)$, $(101, k)$ and $(110, k)$ are unstable for

$$k > \left\lfloor \frac{\log \frac{1}{r}}{\log (1+r)} \right\rfloor + \left\lfloor \frac{\log \frac{1+r}{s}}{\log \left(1+\frac{s}{1+r}\right)} \right\rfloor + 1. \tag{4.6}$$

Proof. We shall prove the result for $(110, k)$. The analogous result for $(100, k)$ follows easily, and Corollary 4.1 takes care of the state $(101, k)$.

Let S_1 be a policy that executes $t_{10} + l + 2$ jobs optimally on P_1 and P_2 while it executes just one on P_3. Let Y denote the maximum finishing time between P_1 and P_2 and let Z be the finishing time on P_3. If $C_1(110, k) = C_1(111, k-1)$, $k = t_{10} + l + 1$, denotes the expected makespan of the schedule produced by S_1, then

$$\begin{aligned} C_1(110, k) &= E[\max(Y, Z)] \\ &= \int_0^\infty P(Y > t)\, dt + \int_0^\infty P(Z > t)\, dt - \int_0^\infty P(Y > t)P(Z > t)\, dt, \end{aligned}$$

whereupon substitution of $1 - P(Y > t) = P(Y \leqq t)$ gives

$$C_1(110, k) = \int_0^\infty P(Y > t)\, dt + \int_0^\infty P(Z > t)P(Y \leqq t)\, dt.$$

The first integral is simply $C(11, k-1)$, and the second can be bounded by the use of

$$P(Y \leqq t) \leqq P(X_1 + \cdots + X_l \leqq t),$$

where the X_i's are independent exponentials with parameter $1 + r$. Substitutions yield

$$\begin{aligned} C_1(110, k) &\leqq C(11, k-1) + \int_0^\infty \exp(-st) P(X_1 + \cdots + X_l \leqq t)\, dt \\ &\leqq C(11, k) + \int_0^\infty \exp(-st) \exp(-(1+r)t) \sum_{n=l}^{\infty} \frac{(1+r)^n t^n}{n!}\, dt \\ &= C(11, k-1) + \frac{1}{s}\left(\frac{1+r}{1+r+s}\right)^l. \end{aligned}$$

Suppose $(110, k)$ were stable. Then $C(110, k) = C(11, k) = C(11, k-1) + 1/(1+r)$, so that

$$C_1(110, k) \leqq C(110, k) - \frac{1}{1+r} + \frac{1}{s}\left(\frac{1+r}{1+r+s}\right)^l. \tag{4.7}$$

But if

$$l \geqq \frac{\log \frac{1+r}{s}}{\log\left(1+\frac{s}{1+r}\right)} + 1, \tag{4.8}$$

then

$$\frac{1}{s}\left(\frac{1+r}{1+r+s}\right)^{l} < \frac{1}{1+r}$$

and we conclude from (4.7) that $C_1(110, k) < C(110, k)$, a contradiction. It follows that $(110, k)$ is unstable when (4.8) holds.

An obvious upper bound to the number of states for which $C(\alpha, k)$ must be evaluated is given by $8(K+1)$, where K denotes the right-hand side of (4.6). Hence, the time required by the calculation is a function only of r and s. As an alternative to the Bellman equation approach t_{101} and t_{110} can be obtained from the roots of certain transcendental equations. (Note that this statement also applies to $t_{10} = \lfloor t(r) \rfloor$ where $t(r)$ is the solution of $r(1+r)^k = 1$; see (4.1).)

The combinatorial basis for the calculations is straightforward and corresponds in the $m = 2$ case to the calculations in (4.4). To compute t_{101} one finds the largest k such that $C(101, k) \leqq C(111, k-1)$. In the calculation of each expected makespan P_3 is assumed to be executing its last job. Thus, the problems reduce to two processor problems when the completion on P_3 occurs. We note also from Theorem 4.2 that in calculating $C(101, k)$ no assignments to P_2 need be considered while P_3 is busy. Thus, if the completion on P_3 is the $(i+1)$th completion event after state $(101, k)$ for $i \leqq k$, then the expected remaining makespan is simply $C(10, k-i)$, which is obtainable from (4.5). To complete this example, the probability of the above event is simply

$$\left(\frac{1}{1+s}\right)^{i} \frac{s}{1+s},$$

and hence

$$C(101, k) = \frac{1}{s} + \sum_{i=0}^{k} \frac{s}{1+s}\left(\frac{1}{1+s}\right)^{i} C(10, k-i).$$

A similar approach to calculating $C(111, k-1)$ must also take into account the fact that since $k \leqq t_{101}$ no more waiting jobs can be assigned to P_2 until P_3 finishes.

Similarly, for t_{110} one finds the largest k such that $C(110, k) \leqq C(111, k-1)$. Note that $C(110, k) = C(11, k)$ is immediately a two-processor problem, and that the calculation of $C(111, k-1)$ uses the fact that P_3 is executing its last

job. Also, of course, the calculation must reflect the fact that while P_3 is busy, waiting jobs can be assigned to P_2 as it becomes available only while more than t_{101} waiting jobs remain.

The detailed calculations that we have outlined are routine but lengthy, and the final results from which thresholds are found numerically are quite awkward. Thus, in the interests of conserving space, we omit them.

5. An optimal algorithm for general m

The burden of this section is a proof that for any $\alpha \neq \mathbf{1}$ the state (α, k) is unstable for sufficiently large k. This means that there exists a K such that for $k > K$ any state (α, k), $\alpha \neq \mathbf{1}$, is unstable and an optimal algorithm must assign waiting jobs to all processors in J_α to produce the stable state $(\mathbf{1}, k - |J_\alpha|)$. We begin by stating bounds on $C(\mathbf{0}, k)$.

Lemma 5.1. For all $k \geqq m$

$$\frac{k-m}{\mu_1 + \cdots + \mu_m} + \sum_{i=1}^{m} \frac{1}{\mu_1 + \cdots + \mu_i} \leqq C(\mathbf{0}, k) \leqq \frac{k-m}{\mu_1 + \cdots + \mu_m} + \sum_{i=1}^{m} \frac{1}{\mu_i + \cdots + \mu_m}.$$

Proof. The lower bound is simply the expected makespan of a preemptive algorithm that always employs the fastest available processors; clearly, no non-preemptive algorithm can do this well.

For the upper bound consider an algorithm that keeps all processors busy until state $(\mathbf{1}, 0)$ is reached after a time interval of expected length, $(k-m)/(\mu_1 + \cdots + \mu_m)$. At this point the remaining expected time to completion of the last job is $C(\mathbf{1}, 0)$. But $C(\mathbf{1}, 0)$ is clearly less than the expected makespan achieved by a preemptive algorithm that always employs the slowest available processors, i.e.

$$C(\mathbf{1}, 0) \leqq \sum_{i=1}^{m} \frac{1}{\mu_i + \cdots + \mu_m}.$$

The upper bound follows.

We now prove that it is optimal to use all m processors for sufficiently large k.

Theorem 5.1. There exists a $k \leqq \lfloor m^3/\mu_m^2 \rfloor + 1$ such that starting in state $(\mathbf{0}, k)$ any optimal algorithm will assign jobs to all processors and continue execution in state $(\mathbf{1}, k-m)$.

Proof. Let $k \geqq m$ and suppose that for all j, $m \leqq j \leqq k$, there is an optimal policy for the initial state $(0, j)$ which does not make assignments to all processors. This means that for each j we may choose a stable state (α, l),

$\alpha \neq \mathbf{1}$, such that $(\mathbf{0}, j) \subseteq (\alpha, l)$ and

$$C(\mathbf{0}, j) = C(\alpha, l) = \frac{1}{\sum_{I_\alpha} \mu_i} \left[1 + \sum_{I_\alpha} \mu_i C(\alpha - \epsilon_i, l) \right]. \tag{5.1}$$

Since $(\mathbf{0}, j-1) \subseteq (\alpha - \epsilon_i, l)$, we have $C(\mathbf{0}, j-1) \leqq C(\alpha - \epsilon_i, l)$ and we conclude from (5.1) that

$$C(\mathbf{0}, j) \geqq \frac{1}{\mu_1 + \cdots + \mu_m} + C(\mathbf{0}, j-1), \qquad m \leqq j \leqq k. \tag{5.2}$$

Reasoning as in Lemma 5.1 we obtain

$$C(\mathbf{0}, m-1) \geqq \frac{m-1}{\mu_1 + \cdots + \mu_{m-1}}. \tag{5.3}$$

With (5.3) as a basis the recurrence in (5.2) gives

$$C(\mathbf{0}, k) \geqq \frac{k}{\mu_1 + \cdots + \mu_{m-1}}. \tag{5.4}$$

This inequality and the right-hand inequality of Lemma 5.1 lead to

$$k \leqq \sum_{i=1}^{m} \frac{1}{\mu_i + \cdots + \mu_m} \frac{(\mu_1 + \cdots + \mu_m)^2}{\mu_m} \leqq \frac{m^3}{\mu_m^2}. \tag{5.5}$$

It follows that there exists a $k \leqq \lfloor m^3/\mu_m^2 \rfloor + 1$, such that any optimal policy starting from $(\mathbf{0}, k)$ assigns jobs to all processors.

We now prove that every state (α, k), $\alpha \neq \mathbf{1}$, is unstable for sufficiently large k.

Theorem 5.2. Let K be the least integer k such that in state $(\mathbf{0}, k)$ any optimal algorithm will assign jobs to all available processors and continue processing from state $(\mathbf{1}, k-m)$. Then state (α, k), $\alpha \neq \mathbf{1}$, is unstable for all k satisfying $k + |\alpha| > K$.

Proof. Whenever $k + |\alpha| \geqq K$ we have

$$\frac{k + |\alpha| - K}{\mu_1 + \cdots + \mu_m} + C(\mathbf{0}, K) \leqq C(\alpha, k) \leqq \frac{k + |\alpha| - K}{\mu_1 + \cdots + \mu_m} + C(\mathbf{1}, K - m). \tag{5.6}$$

The lower bound follows from the fact that $1/(\mu_1 + \cdots + \mu_m)$ is a lower bound on the expected time between successive job completions. The upper bound is just the expected makespan of an algorithm that keeps all processors busy until $k + |\alpha| - K$ job completions have occurred, and is optimal thereafter. By the definition of K we have $C(\mathbf{0}, K) = C(\mathbf{1}, K - m)$ and so the bounds in (5.6) are achieved with equality.

Now suppose that $k + |\alpha| > K$, $\alpha \neq \mathbf{1}$, and (α, k) is stable. Then

$$C(\alpha, k) = \frac{1}{\sum_{I_\alpha} \mu_i} \left[1 + \sum_{I_\alpha} \mu_i C(\alpha - \epsilon_i, k) \right]. \tag{5.7}$$

Applying (5.6) to each $C(\alpha - \epsilon_i, k)$ in (5.7) we conclude that

$$C(\alpha, k) > \frac{k + |\alpha| - K}{\mu_1 + \cdots + \mu_m},$$

which contradicts (5.6). Hence (α, k) is unstable for $k + |\alpha| > K$.

The implications of Theorems 5.1 and 5.2 are that if the Bellman equations are to be solved numerically then their solution need only continue until the number of waiting jobs K is such that an optimal action in state $(\mathbf{0}, K)$ is to assign jobs to all processors and continue processing in state $(\mathbf{1}, K - m)$.

We conjecture that this K has an alternative characterization as the least k for which the state $(11 \cdots 10, k)$ is unstable. (Recall that this was proved for $m = 2, 3$.)

6. Conclusions and open problems

We have shown that for $m = 2$ and 3 there exist simple threshold rules for minimizing expected makespans, and we have given simple methods for evaluating the corresponding thresholds. We have also conjectured that similar threshold rules, depending only on processor states, exist for all larger values of m, although this seems quite difficult to prove. Direct computation using the Bellman equations and the bound of Section 5 have verified the existence of such threshold rules for $m = 4, 5$ and all values of $\mu_1 = 1 \geqq \mu_2 \geqq \mu_3 \geqq \mu_4 \geqq \mu_5$ that are multiples of 0.1. But it is not clear how much these results should increase one's confidence in the general validity of the conjecture.

If our conjecture is true, it follows immediately that for any fixed $\{\mu_i\}$ the process of scheduling jobs on processors with those rates to minimize expected makespan is a relatively simple one, requiring only that one determine and store a single threshold for each processor state. However, it does not follow that the task of finding the correct thresholds is necessarily also simple. Indeed, the finite algorithm implied by the bound of the preceding section, for fixed μ_i, is potentially quite laborious for m much greater than 5. Notice that the algorithm applies independently of the validity of our conjecture, although the optimal scheduling rules it finds need not be of threshold type. Clearly, it would be useful to find alternative algorithms capable of extending the range within which optimal scheduling rules can be derived in practice.

In a more mathematical vein, it is apparent from the content of this paper that there is a great need for new mathematical techniques useful for simplifying the derivation of results about expected makespan scheduling. At present even quite natural and intuitive 'facts' require non-trivial proofs, and this is further complicated by the observation that such 'facts' do not always even turn out to be true. Thus it would be a valuable contribution simply to find a more elegant way to obtain the results we have presented. It is to be hoped that such methods will lead the way towards resolving the conjecture we have mentioned and towards analyzing extensions of our model, e.g. general distributions for job execution times, job-up times, preemptive scheduling with preemption costs, etc.

References

[1] AGRAWALA, A. K., COFFMAN, E. G. JR., GAREY, M. R. AND TRIPATHI, S. K. (1984) A stochastic optimization algorithm minimizing expected flow times on uniform processors. *IEEE Trans. on Computers,* **33,** 351–356.

[2] LIN, W. AND KUMAR, P. R. (1982) Optimal control of a queueing system with two heterogeneous servers. Preprint.

[3] WALRAND, J. (1983) A note on optimal control of a queueing system with two heterogeneous servers. Technical Report, EECS, University of California, Berkeley.