On *J*-maximal and *J*-minimal Flow-Shop Schedules

FRANCIS Y. CHIN AND LONG-LIEH TSAI

University of Alberta, Edmonton, Alberta, Canada

ABSTRACT. Scheduling problems are considered for a common kind of flow shop where the execution time for certain tasks in each job is always longer or shorter than that for the other tasks NP-completeness is shown for some cases, simple optimal algorithms are found for the others, and bounds are given for the worst cases.

KEY WORDS AND PHRASES job scheduling, flow-shop schedules, minimum finishing time, NP-complete, optional schedules, heuristic schedules

CR CATEGORIES 4.32, 5.25, 5.30

1. *Introduction*

A *flow shop* [4] consists of $m \geq 1$ processors $\{P_1, \ldots, P_m\}$ and $n \geq 1$ jobs $\{J_1, \ldots, J_n\}$. Each processor P_j performs a different task, and each job J_i has a chain of m tasks. $T_{j,i}$ is the jth task of J_i; its execution time is denoted by $t_{j,i}$. $T_{j,i}$ has to be processed on P_j and can only be executed after $T_{j-1,i}$ is finished.

A *schedule* for a flow shop is defined as the sequence of tasks to be executed by each processor. If we allow a task to be partitioned and done in several time intervals, the schedule is called *preemptive*. A *nonpreemptive* schedule is one in which a processor cannot be interrupted once it has begun execution of a task. The finish time of a schedule, that is, the latest completion time of the individual processors, is usually denoted by f and is also the time by which all the tasks are completed. An *optimal finish time* (OFT) schedule is one which has the shortest finish time, denoted by f^*, among all schedules. In this paper we are interested in nonpreemptive OFT scheduling problems.

Flow-shop problems have long existed, and, despite the great effort devoted to them [3, 4, 6], these problems, even those of a small practical size, still remain difficult. An $O(n \log n)$ OFT algorithm for $m = 2$ was first proposed by Johnson [7], but for $m \geq 3$ the problem has been shown to be NP-complete [5]. In this paper a restricted case of this problem, called the J-maximal (J-minimal) flow-shop problem, is studied.[1]

A *J-maximal flow shop* is a particular kind of flow shop in which the Jth task in each job has the longest execution time of all tasks in that job. The J-maximal (J-

[1] In [11], the authors define an *ordered flow shop* in a more restricted manner than our definition They assume additional relationships among the execution times of the jobs

This research was supported in part by Natural Sciences and Engineering Research Council of Canada Grant NSERC A4319.

Authors' present addresses· F Y Chin, Department of Computer Science, University of Alberta, Edmonton, Alberta, Canada T6G 2H1, L.-L. Tsai, Burroughs Corporation, 16701 West Bernardo Drive, San Diego, CA 92127

minimal) flow shop occurs often in real job-shop situations; an example of such a shop is an assembly line, where the workers or work stations are the processors, and it is usually true that a given task may take longer on one machine than on another. Similarly, we have the *J-minimal flow shop* in which the Jth task in each job has the shortest execution time. Nevertheless, even with such a restriction on the flow shop, the OFT problem remains very difficult.

Section 2 shows that both the J-maximal and J-minimal flow-shop problems are NP-complete. In Section 3 the problem is further restricted by assuming that all the other tasks in a job except the J-maximal or J-minimal one have the same execution time. We have found efficient algorithms for some of these cases, while others remain intrinsically difficult. Section 4 studies the ratio of the values of the worst possible and best possible solutions for the problems. Tight bounds for the worst case ratio are derived for the 1-minimal flow shop; an $O(\log m)$ lower bound and an $O(\sqrt{m})$ upper bound for the worst case ratio are obtained for the 1-maximal case. Particular attention is paid to the special case when $m = 3$.

2. *NP-Complete Cases*

In this section the J-maximal and J-minimal flow-shop problems are shown to be NP-complete. The J-maximal and J-minimal flow-shop problems can be defined exactly as follows.

Definition. A J-maximal (J-minimal) flow shop is one which has the longest (shortest) execution time on the Jth task; that is, $t_{j,i} \leq (\geq)\ t_{J,i}$ for $1 \leq i \leq n$ and $1 \leq j \leq m$.

In proving the NP-completeness results in this section we make use of the following known NP-complete problem.

Partition. A multiset $S = \{a_1, \ldots, a_n\}$ is said to have a partition if there exists a subset u of the indices $\{1, \ldots, n\}$ such that $\sum_{i\in u} a_i = (\sum_{i=1}^{n} a_i)/2$. The partition problem [8] is that of determining for an arbitrary multiset S whether or not it has a partition. The a_i may be assumed to be integers.

The OFT problem can be treated as a language recognition problem [8, 12] and restated as

FOFT. Given a flow shop of m processors and n jobs with task times $t_{j,i}$, $1 \leq j \leq m$ and $1 \leq i \leq n$, and a number τ, does the shop have a schedule with finish time $f \leq \tau$?

The customary interpretation of a processing time $t_{j,i}$ being zero is that job J_i has to spend an infinitesimally small amount of time on processor P_j. Under this interpretation, part (2) of the proof of Lemma 1 below still implies NP-completeness for the 2-maximal, 1- and 3-minimal FOFT. On the other hand, there exists an $O(n \log n)$ algorithm for the 2-minimal FOFT [2]; more specifically, the flow-shop problem FS1 used in part (1) of the proof of Lemma 1 is an ordered flow shop in the sense of [11] and hence solvable in polynomial time. However, it turns out that the 1- and 3-maximal FOFT remain NP-complete even under this interpretation of $t_{j,i}$ being zero [1].

In the following, we assume that if the processing time $t_{j,i}$ is zero, J_i does not have to visit P_j at all. As a result, the following theorem and lemma present a straightforward reduction, which is a slight variation on constructions given in [6, 9].

THEOREM 1. *The J-maximal FOFT for $m \geq 3$ is NP-complete.*

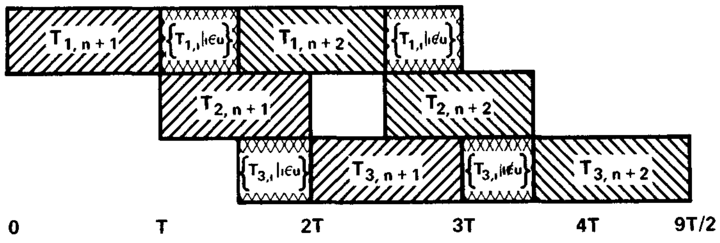

FIG 1. Example FS1

PROOF. Obviously FOFT can be recognized in polynomial time by a nondeterministic Turing machine [12]; therefore, so can the J-maximal FOFT. The Turing machine guesses the optimal permutation on each of the processors and tests whether the finish time f is less than or equal to τ. The remaining part of the proof is presented in the following lemma. It is sufficient to consider just the case $m = 3$.

LEMMA 1. *If the J-maximal FOFT with $m = 3$ is polynomially solvable, then so is PARTITION.*

PROOF. J-maximal flow-shop problems are constructed from the partition problem $S = \{a_1, \ldots, a_n\}$ so that for each problem there exists a schedule with a finish time less than or equal to the assigned time τ if and only if S has a partition. We have to consider three different cases in which the resulting flow-shop problems are 1-, 2-, and 3-maximal FOFTs.

(1) 1- and 3-maximal FOFTs. Construct the flow-shop problem with $n + 2$ jobs and $m = 3$ processors, as follows:

*FS*1:

$$\begin{array}{lll} J_i: & t_{1,i} = t_{3,i} = a_i, \quad t_{2,i} = 0 & \text{for} \quad 1 \le i \le n; \\ J_{n+1}: & t_{1,n+1} = t_{2,n+1} = t_{3,n+1} = T; & \\ J_{n+2}: & t_{1,n+2} = t_{2,n+2} = t_{3,n+2} = T; & \end{array}$$

where

$$T = \sum_{i=1}^{n} a_i \qquad \text{and} \qquad \tau = \frac{9T}{2}.$$

Since $t_{1,j} = t_{3,i} \ge t_{2,i}$ for $1 \le i \le n + 2$, FS1 is clearly a 1- or 3-maximal flow shop. The optimal schedule is as shown in Figure 1. It can be shown easily that S has a partition if and only if there is a nonpreemptive schedule with finish time $9T/2$. A similar proof already appears in [9].

(2) 2-maximal FOFT. The flow-shop problem which serves our purpose involves $n + 1$ jobs and is presented below.

*FS*2:

$$\begin{array}{lll} J_i: & t_{1,i} = t_{3,i} = 0, \quad t_{2,i} = a_i & \text{for} \quad 1 \le i \le n; \\ J_{n+1}: & t_{1,n+1} = t_{2,n+1} = t_{3,n+1} = \dfrac{T}{2}; & \end{array}$$

with

$$T = \sum_{i=1}^{n} a_i \qquad \text{and} \qquad \tau = \frac{3T}{2}.$$

FS2 is obviously a 2-maximal flow shop. If S has a partition, the optimal schedule is shown as in Figure 2. It is also easy to show that the finish time of the schedule

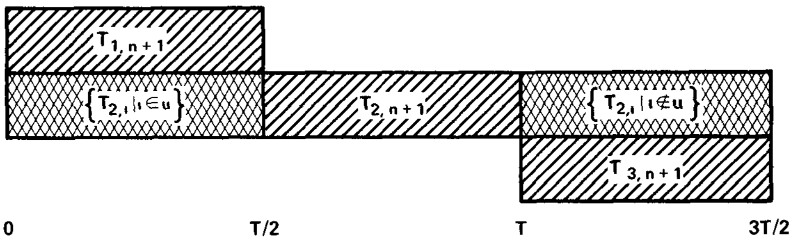

FIG. 2 Example FS2

must exceed $3T/2$ if S does not have a partition, and the proof is similar to those for the 1- and 3-maximal flow-shop problems. □

THEOREM 2. *The J-minimal FOFT for $m \geq 3$ is NP-complete.*

PROOF. The argument is very similar to the proof presented in Theorem 1, except that FS1 is used for proving the 2-minimal FOFT and FS2 for proving the 1- and 3-minimal FOFTs. □

3. *Optimal Cases*

We have shown that the J-maximal and J-minimal flow-shop problems are NP-complete for $m \geq 3$. In this section we study a very restricted class of J-maximal flow-shop problems, where the execution times of all the tasks in each job except the J-maximal or J-minimal one are the same. We use notations L, S, and E to stand for the tasks with the longest, shortest, and equal execution time. (L, E, E, E, E) stands for a 1-maximal flow shop with $m = 5$. OFT algorithms are found for the $(L, E, \ldots, E)$, $(S, E, \ldots, E)$, $(E, \ldots, E, L)$, and $(E, \ldots, E, S)$ cases. The more general "mixed" case, that is, $(—, E, \ldots, E)$ and $(E, \ldots, E, —)$, where — is either L or S, is also solved.

We first show that a very simple algorithm will guarantee the OFT for the case $(E, \ldots, E)$. Let us assume that k is the index of the largest $(E, \ldots, E)$ job, that is, $t_{j,k} \geq t_{j,i}$ for all i and j. The following lemma gives a lower bound for the finish time of the $(E, \ldots, E)$ flow-shop problem.

LEMMA 2. *Consider the flow shop $(E, \ldots, E)$ with m processors. If $t_{1,i}$ is the execution time for every task in the ith job and n is the number of jobs, then*

$$f^* \geq \sum_{i=1}^{n} t_{1,i} + (m-1)t_{1,k}.$$

PROOF. For any schedule let

$S = \{1, \ldots, n\}$, the set of all the indices of the jobs;
S_1 = the set of indices of those jobs that precede J_k on P_1;
S_2 = the set of indices of those jobs that follow J_k on P_m;
$S_3 = S - S_1 \cup S_2 \cup \{k\}$, that is, the set of indices of those jobs that do not precede J_k on P_1 or follow J_k on P_m.

Let $h \in S_3$. It is clear that J_h must overtake J_k somewhere in between P_1 and P_m, and thus f^* is at least

$$\sum_{i \in S_1} t_{1,i} + mt_{1,k} + \sum_{i \in S_2} t_{1,i} + \sum_{i \in S_3} t_{1,i} = \sum_{i=1}^{n} t_{1,i} + (m-1)t_{1,k}. \qquad \square$$

A *permutation schedule* [4] is simply a schedule with the same job order on all processors, that is, a schedule that is completely characterized by a single permutation

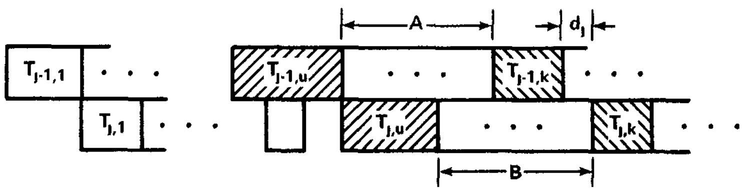

FIGURE 3

of the job indices 1, ..., n. Permutation schedules have the feature that a processor cannot be idle if there exists a task awaiting execution on it. Usually permutation schedules do not give optimal finish times for the flow-shop problem with $m > 3$ [4]. It turns out, however, that permutation schedules are sufficient to give an optimal finish time for our problem.

LEMMA 3. *Consider any permutation schedule. If $t_{j,i} \leq t_{j-1,i}$ for all i and $T_{j,k}$ starts d_j time units after the completion of $T_{j-1,k}$, with $d_j > 0$, then there exists a J_u such that $T_{j,u}$ starts immediately after the completion of $T_{j-1,u}$ and $t_{j,u} \geq d_j + t_{j-1,k}$.*

PROOF. Let J_u be the last job executed before J_k which has its jth task started immediately after its $(j - 1)$st task. As indicated in Figure 3, A and B are the total duration between J_u and J_k for P_{j-1} and P_j, respectively. Thus we have

$$t_{j,u} + B = A + t_{j-1,k} + d_j.$$

P_j will not be idle in B since u is the last job without a time gap between tasks. Since $t_{j,i} \leq t_{j-1,i}$, $A \geq B$. Thus we have

$$t_{j,u} \geq t_{j-1,k} + d_j. \qquad \square$$

COROLLARY. *Consider any permutation schedule for the $(E, \ldots, E)$ flow-shop problem. If J_k is the largest job, then $T_{j,k}$ is always executed immediately after $T_{j-1,k}$ for $1 < j \leq m$.*

LEMMA 4. *Consider any permutation schedule for the $(E, \ldots, E)$ flow-shop problem. P_j is always busy after the execution of $T_{j,k}$ for $1 \leq j \leq m$.*

The proof for this lemma is trivial and is omitted here.

THEOREM 3. *All permutation schedules give the optimal finish time for the $(E, \ldots, E)$ flow-shop problem.*

PROOF. The corollary to Lemma 3 shows that the difference d_j between the completion time of $T_{j-1,k}$ and the starting time of $T_{j,k}$ is zero. Lemma 4 states that P_j is busy after the execution of $T_{j,k}$. Since P_m is busy all the time after the execution of $T_{m,k}$, we can easily conclude that the finish time of the permutation schedule is

$$f = \sum_{i=1}^{n} t_{1,i} + (m - 1)t_{1,k},$$

which is the lower bound on f^* given in Lemma 2. Thus we can conclude that this is an optimal schedule. $\square$

Instead of considering optimal schedules for cases $(S, E, \ldots, E)$, $(L, E, \ldots, E)$, $(E, \ldots, E, L)$, and $(E, \ldots, E, S)$, we study the more general cases $(—, E, \ldots, E)$ and $(E, \ldots, E, —)$. We show that optimal schedules can be found easily as long as all the tasks except the first or the last one in each job have the same execution time. The following lemma shows that we can restrict our attention to permutation schedules only.

LEMMA 5. *Given any schedule S on m processors for the (—, E, ..., E) flow-shop problem, we can always construct a permutation schedule no longer than S.*

PROOF. Given the schedule S, we first rearrange the tasks on P_1 according to the sequence on P_2 (it is well known that this will not increase the completion time $C_{2,i}$ of any $T_{2,i}$) and left-justify the schedule. The schedule on P_2 has now been fixed; it consists of subsets of tasks separated by idle time periods. We next rearrange the tasks on $P_3, \ldots, P_m$ according to the sequence on P_2. If the idle periods on P_2 are sufficiently large, each of the subsets will generate a permutation schedule as characterized in Theorem 3. Otherwise these schedules will interfere in an obvious way, but there will still exist a J_x such that the permutation schedule is completed at time $C_{2,x} + (m-2)t_{2,x} + \sum_{i \in S} t_{2,i}$, where S indicates all the jobs following J_x on P_2. As in the proof of Lemma 2, it can easily be argued that this expression is a lower bound on the completion time of every schedule, given the sequence on P_2. □

Johnson [7] found the OFT algorithm for the two-processor flow-shop problem. An optimal permutation schedule can be constructed by having J_u executed before J_v if $\min(t_{1,u}, t_{2,v}) \leq \min(t_{2,u}, t_{1,v})$. The following theorem shows that by applying Johnson's condition on the first two processors, the optimal sequence for the (—, $E, \ldots, E$) flow-shop schedule will be achieved.

THEOREM 4. *For the (—, E, ..., E) flow-shop problem, the permutation schedule which has the same sequence as the optimal sequence for the first two processors is the optimal finish-time schedule for the problem.*

PROOF. Let S be an optimal finish-time schedule for the problem. From Lemma 5 we can assume, without loss of generality, that all tasks on each processor are executed in the same order.

As pointed out in McMahon's thesis [10], the relative order (J_u, J_v) of two adjacent jobs in a permutation flow-shop schedule is optimal if Johnson's condition holds for all processor pairs, that is, if $\min\{t_{i,u}, t_{j,v}\} \leq \min\{t_{j,u}, t_{i,v}\}$ for $1 \leq i < j \leq m$.

Since $t_{i,v} = t_{j,v}$ for $2 \leq i \leq j \leq m$, it is sufficient to have Johnson's condition hold for the case $i = 1, j = 2$. The condition then follows immediately for all other machine pairs. □

COROLLARY. *The optimal permutation schedule for the (—, E, ..., E) flow-shop problem can be found in $O(n \log n)$ time, where n is the number of jobs.*

LEMMA 6. *Given any schedule S on m processors for (E, ..., E, —), we can always construct a permutation schedule no longer than S.*

PROOF. The proof is similar to that of Lemma 5. □

THEOREM 5. *The optimal permutation schedule for the (E, ..., E, —) flow-shop problem, which has the same optimal job sequence as the last two processors, can be found in $O(n \log n)$ time, where n is the number of jobs.*

PROOF. By Lemma 6, the proof is similar to that of Theorem 4 and its corollary.

In fact, the following theorem states a more general result than Theorems 4 and 5.

THEOREM 6. *The optimal permutation schedule for the ($E_1, \ldots, E_1, E_2, \ldots, E_2$) flow-shop problem can be found in $O(n \log n)$ time, where n is the number of jobs.*

PROOF. Assume that the first r tasks of each job have the same execution time while the other task execution times have a second value. Using similar arguments [10] as before, it can be shown that the optimal permutation schedule has the same optimal job sequence as the two-processor problem on P_r and P_{r+1}. By applying Johnson's algorithm on P_r and P_{r+1}, the optimal job sequence can be found in $O(n \log n)$ time. What remains to be proved in this theorem is that any schedule for this flow shop can be rearranged into a permutation schedule no longer than the original schedule. This can be done by rearranging all tasks on $P_1, \ldots, P_r, P_{r+2}, \ldots, P_n$ according to the sequence on P_{r+1}. By Lemma 6, the completion time $C_{r+1,i}$ of any $T_{r+1,i}$ will not be increased, and the remaining part of the proof is similar to Lemma 5. □

4. *Bounds on Arbitrary Permutation Schedules*

In Section 2 we showed NP-completeness for the J-maximal and J-minimal flow-shop problems. Only in very special situations, however, can efficient optimal algorithms be found for these problems. In this section, therefore, we investigate the bounds on the ratio of the values of the worst possible and the best possible situations. In what follows, permutation schedules are assumed in all cases.

It is a well-known fact that a permutation schedule for the general flow-shop problem can have a bound for f/f^* as bad as m [6]. However, we have derived better bounds on f/f^* for the 1-maximal and 1-minimal problems. The J-maximal and J-minimal problems with J not equal to 1 can be solved similarly. Since tight bounds can be derived for the 1-minimal flow-shop problems, we consider these first.

Before we proceed, we need to introduce a process called *right-shift*. In this process, the tasks on $P_2, \ldots, P_m$ are shifted to later initiation times, the shifts being the minimum necessary in order to eliminate the interior idle times in all the processors.

LEMMA 7. *After the tasks in a* 1*-minimal flow shop are right-shifted,* P_2 *always starts no later than* $\max_{1 \le i \le n}\{t_{1,i}\}$ *time units after the beginning of* P_1.

PROOF. We can make use of the result in Lemma 3 to prove this lemma. Consider $T_{j-1,k}$ in Lemma 3 as the second task in this lemma and $T_{j,k}$ as the first; in addition, the last job becomes the first and the first becomes the last. Then it is obvious that $d_j + t_{j-1,k}$ becomes the difference between the start times of P_1 and P_2. It follows directly from Lemma 3 that the time difference is no more than $\max_{1 \le i \le n}\{t_{1,i}\}$. □

We shall make use of the previous lemma to prove the following theorems.

THEOREM 7. *For the* 1*-minimal flow shop with* $m = 2$, $f/f^* \le 3/2$, *and this bound is the best possible.*

PROOF. Let δ_j be the start time of P_{j+1} minus the start time of P_j. From Lemma 7, $\delta_1 \le \max_{1 \le i \le n}\{t_{1,i}\}$. Since $f^* \ge \sum_{i=1}^{n} t_{2,i}$ and $f^* \ge 2 \max_{1 \le i \le n}\{t_{1,i}\}$, we have

$$\frac{f}{f^*} = \frac{\sum_{i=1}^{n} t_{2,i} + \delta_1}{f^*} \le \frac{3}{2}.$$

The worst case is illustrated in the following example.

Example. Let the schedule have two jobs, J_1 and J_2:

$$J_1: \quad t_{1,1} = t_{2,1} = 1;$$
$$J_2: \quad t_{1,2} = \epsilon, \quad t_{2,2} = 1.$$

The worst and optimal schedules are shown in Figure 4a and b, respectively, and the ratio $f^*/f = 3/(2 + \epsilon) \rightarrow 3/2$ as $\epsilon \rightarrow 0$. □

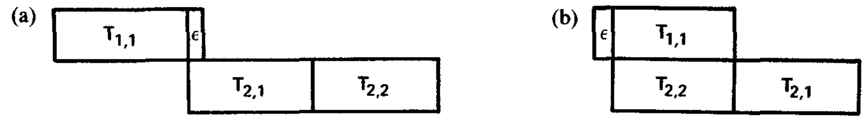

FIG. 4. 1-minimal flow shop for $m = 2, f/f^* \to 3/2$. (a) A bad schedule. (b) An optimal schedule

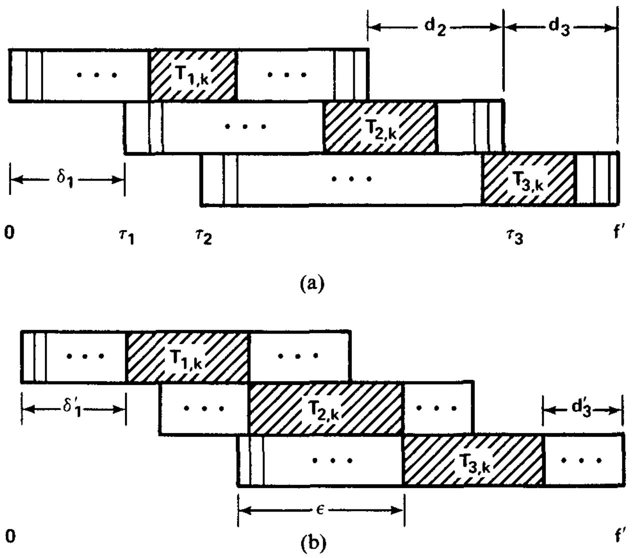

FIG 5 1-minimal flow shop for $m = 3, f/f^* \leq 2$.

LEMMA 8. *For the* 1-*minimal flow shop with* $m = 3, f/f^* \leq 2$.

PROOF. Consider the schedule in Figure 5a; let f' be the finish time of this schedule after it is right-shifted. Obviously $f' \geq f$, $f' = \delta_1 + \sum_{i=1}^{n} t_{2,i} + d_3$ (refer to Figure 5a), and $f^* \geq \sum_{i=1}^{n} t_{2,i}$. It is sufficient if we can show that $f^* \geq \delta_1 + d_3$. Let $t_{1,k} = \max_{1 \leq i \leq n}\{t_{1,i}\}$, and from Lemma 7 we have $\delta_1 \leq t_{1,k}$. Now let us consider the following cases.

(1) $T_{3,k}$ does not start immediately after $T_{2,k}$ (Figure 5a). Obviously P_3 will be busy at least for $t_{2,k}$ time units between τ_2 and τ_3, where τ_2 is the starting time for P_3 and τ_3 the finishing time of P_2. Since $t_{2,k} \geq t_{1,k} \geq \delta_1$ (by Lemma 7), $f^* \geq \sum_{i=1}^{n} t_{3,i} \geq t_{2,k} + d_3 \geq \delta_1 + d_3$.

(2) $T_{3,k}$ starts immediately after $T_{2,k}$ and thus $T_{3,k}$ must not start after τ_3. If $T_{3,k}$ finishes before τ_3, we have $\sum_{i=1}^{n} t_{3,i} \geq t_{3,k} + d_3 \geq \delta_1 + d_3$, and obviously $f^* \geq \delta_1 + d_3$. So we can assume $T_{3,k}$ starts before τ_3 and finishes after τ_3. In this case, if we further assume that $T_{1,k}$ starts after τ_1, the total execution time of the tasks executed before $T_{3,k}$ (i.e., between τ_2 and τ_3) must be longer than δ_1 from the 1-minimal property of the jobs, and so we have $f^* \geq \sum_{i=1}^{n} t_{3,i} \geq \delta_1 + d_3$.

Now consider the case when $T_{1,k}$ starts before τ_1. Since $t_{1,k}$ is the largest task on P_1, as a consequence $T_{2,k}$ must start immediately after $T_{1,k}$. Thus we have (Figure 5b) $f^* \geq t_{1,k} + t_{2,k} + t_{3,k}$; and because of the property of 1-minimality of the jobs, $\epsilon \geq \delta'_1$ (refer to Figure 5b); and as $f^* \geq \epsilon + t_{3,k} + d'_3 > \delta'_1 + d'_3$, we have

$$\left(\frac{f}{f^*}\right) \leq \left(\frac{f'}{f^*}\right) \leq 1 + (\delta'_1 + d'_3)\left(\frac{1}{f^*}\right) \leq 2. \qquad \square$$

THEOREM 8. *For the* 1-*minimal flow shop with* $m \geq 3$, $f/f^* \leq m - 1$, *and this bound is the best possible.*

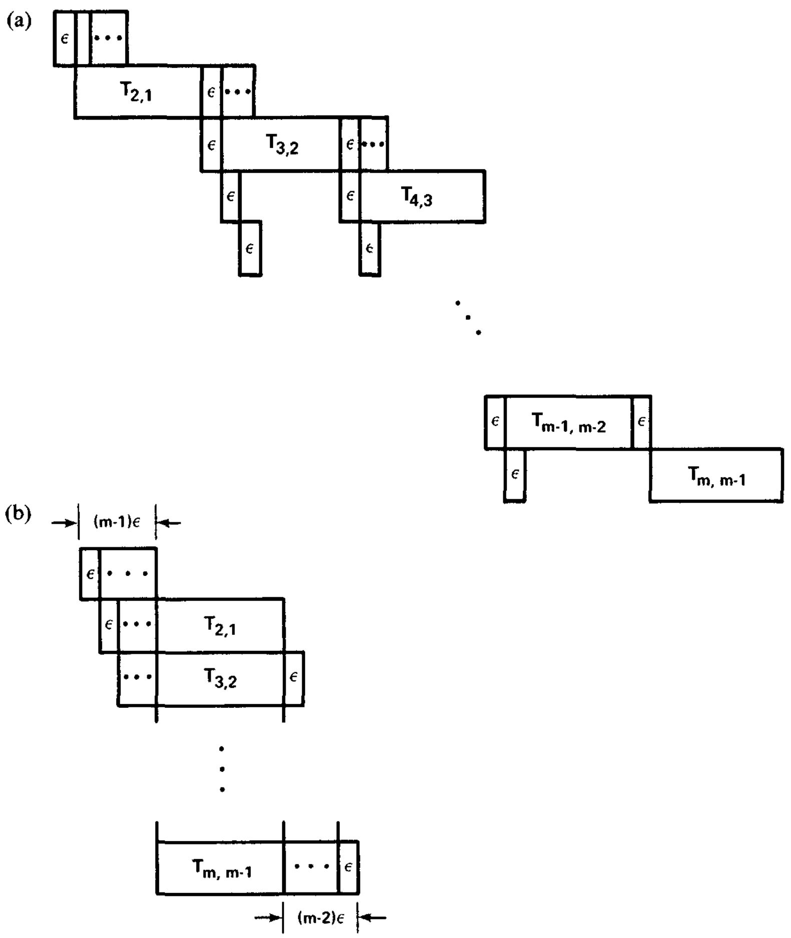

FIG 6 1-minimal flow shop for $m \geq 3$, $f/f^* \to m - 1$. (a) A bad schedule (b) An optimal schedule.

PROOF. Let f_i be the finish time of P_i; we then have $f = f_3 + (f - f_3)$. By Lemma 8, $f_3 \leq 2f^*$, and so $(f - f_3) \leq \sum_{i=1}^{n} \sum_{j=4}^{m} t_{j,i} \leq (m - 3)f^*$. Thus we have $f/f^* \leq m - 1$.

The bound is seen to be the best possible by considering the following example with $m - 1$ jobs.

Example. Let $t_{j,i} = \epsilon$ for all $j \neq i + 1$ and $t_{i+1,i} = 1$, where $1 \leq i \leq m - 1$. The schedule in Figure 6a gives $f = (m - 1)(1 + \epsilon)$, and the one in Figure 6b gives $f^* = 1 + (2m - 3)\epsilon$. Thus $f/f^* \to m - 1$ as $\epsilon \to 0$. □

For the 1-maximal flow-shop problem, the bound on f/f^* for an arbitrary permutation schedule is rather difficult to derive. We show in Lemma 9 that the bound is always less than $1/2 + \sqrt{m}$. However, the worst example we can construct has the bound $1/2(1 + 1/m)(1 + \log m)$ or $O(\log m)$.

LEMMA 9. *For the* 1-*maximal flow-shop problem*, $f/f^* \leq 1/2 + \sqrt{m}$.

PROOF. It is easy to see that for any permutation schedule, there always exists a chain of tasks starting from the first task initiated to the last task completed in the schedule such that there is no idle time or overlap between any two adjacent tasks in

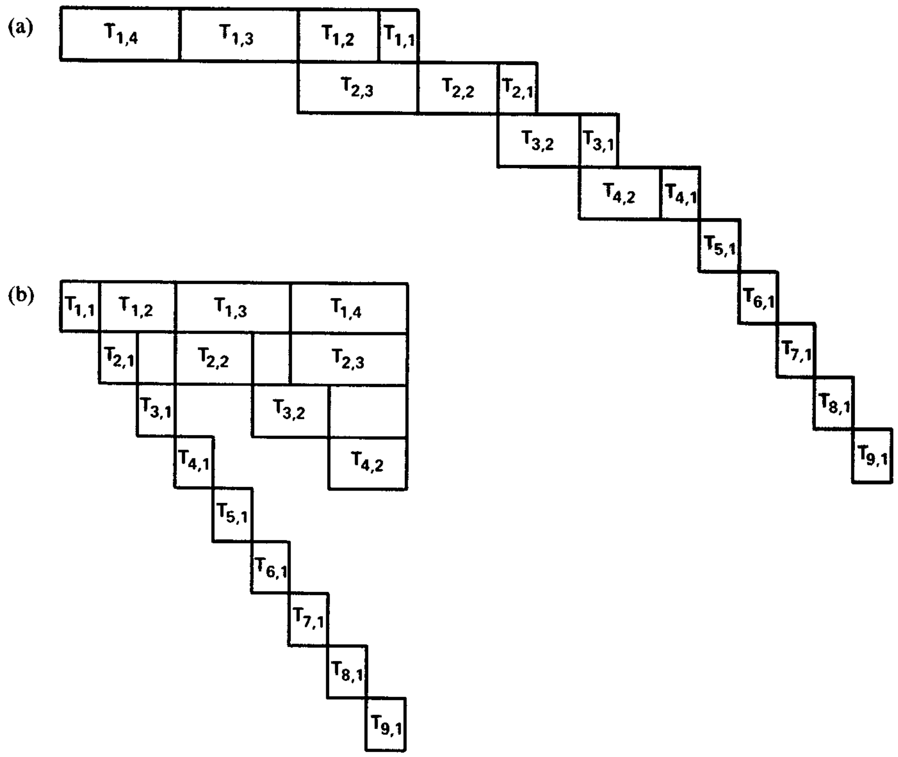
(a)
$T_{1,4}$ $T_{1,3}$ $T_{1,2}$ $T_{1,1}$
$T_{2,3}$ $T_{2,2}$ $T_{2,1}$
$T_{3,2}$ $T_{3,1}$
$T_{4,2}$ $T_{4,1}$
$T_{5,1}$ $T_{6,1}$ $T_{7,1}$ $T_{8,1}$ $T_{9,1}$
(b)
$T_{1,1}$ $T_{1,2}$ $T_{1,3}$ $T_{1,4}$
$T_{2,1}$ $T_{2,2}$ $T_{2,3}$
$T_{3,1}$ $T_{3,2}$
$T_{4,1}$ $T_{4,2}$
$T_{5,1}$ $T_{6,1}$ $T_{7,1}$ $T_{8,1}$ $T_{9,1}$

FIG. 7. 1-minimal flow shop for $m = 9, f/f^* = 21/9$ (a) A bad schedule, $f = 21$. (b) An optimal schedule, $f^* = 9$

the chain. For example, in the schedule given in Figure 7a, $T_{1,4}$, $T_{1,3}$, $T_{2,3}$, $T_{2,2}$, $T_{3,2}$, $T_{4,2}$, $T_{4,1}$, $T_{5,1}$, $T_{6,1}$, $T_{7,1}$, $T_{8,1}$, $T_{9,1}$ is such a chain.

Let L_i and k_i be the total execution time and the total number, respectively, of those tasks in the chain which belong to the ith job; for example, $L_2 = t_{2,3} + t_{2,2}$ and $k_2 = 2$. Since the flow shop is 1-maximal, the first task of the ith job takes at least $l_i = L_i/k_i$ time units. Obviously we have

$$f^* \geq \max\left\{\sum_{i=1}^{n} l_i, \max_{1\leq i\leq n}\{k_i l_i\}\right\} \quad \text{and} \quad f = \sum_{i=1}^{n} L_i = \sum_{i=1}^{n} k_i l_i,$$

and so we have

$$\frac{f}{f^*} \leq \frac{\sum_{i=1}^{n} k_i l_i}{\max\left\{\sum_{i=1}^{n} l_i, \max_{1\leq i\leq n}\{k_i l_i\}\right\}} = r, \tag{1}$$

and $\sum_{i=1}^{n} k_i = n + m - 1$ (chain length in number of tasks).

We want to show that under the above conditions, $r \leq (1 + \sqrt{4m - 3})/2$, by showing that $r \leq n$ and the equality $r = n$ only holds when $\sum_{j=1}^{n} l_j = k_i l_i$ for all i.

Let $k_x l_x = \max_i\{k_i l_i\}$. First we must observe the fact that no matter whether $\sum_{i=1}^{n} l_i$ or $k_x l_x$ is used for r in eq. (1), r is maximal only if all the k_i are greater than 1.

Let us assume that $k_x l_x > \sum_{i=1}^{n} l_i$. Then from eq. (1), we have

$$r = 1 + \frac{\sum_{i=1, i\neq x}^{n} k_i l_i}{k_x l_x},$$

and r can be increased by decreasing l_x as long as $k_x l_x > \sum_{i=1}^n l_i$. Thus we can conclude that r will be maximal only if $k_x l_x \le \sum_{i=1}^n l_i$, in which case

$$r = \frac{\sum_{i=1}^n k_i l_i}{\sum_{i=1}^n l_i} \le n,$$

and the maximal value of r is n and is achieved when $k_i l_i = \sum_{j=1}^n l_j$ for all i. So r will be maximal with respect to m when n is maximal with respect to m, given the condition

$$\sum_{i=1}^n k_i = n + m - 1$$

and

$$k_i l_i = \sum_{j=1}^n l_j \qquad \text{or} \qquad k_i = \frac{1}{l_i} \sum_{j=1}^n l_j \qquad \text{for all } i.$$

From the above equations, we have

$$n + m - 1 = \sum_{i=1}^n \frac{1}{l_i}\left(\sum_{j=1}^n l_j\right) = \left(\sum_{i=1}^n \frac{1}{l_i}\right)\left(\sum_{j=1}^n l_j\right).$$

Since it is easy to show that $(\sum_{i=1}^n (1/l_i))(\sum_{j=1}^n l_j) \ge n^2$, we have

$$n \ge 1 - m + n^2 \qquad \text{or} \qquad n \le \tfrac{1}{2}(1 + \sqrt{4m - 3}).$$

Thus we obtain the result that

$$r \le \tfrac{1}{2}(1 + \sqrt{4m - 3}) < \tfrac{1}{2} + \sqrt{m}.$$ □

Consider the following example of a 1-maximal flow shop with $m = 9$ where

$$\begin{aligned}
J_1&: & t_{j,1} &= 1 \quad \text{for } 1 \le j \le m;\\
J_2&: & t_{j,2} &= \begin{cases} 2 & \text{for } 1 \le j \le 4,\\ 0 & \text{otherwise;}\end{cases}\\
J_3&: & t_{j,3} &= \begin{cases} 3 & \text{for } 1 \le j \le 2,\\ 0 & \text{otherwise;}\end{cases}\\
J_4&: & t_{j,4} &= \begin{cases} 3 & \text{for } j = 1,\\ 0 & \text{otherwise.}\end{cases}
\end{aligned}$$

The worst and optimal schedules are given as in Figure 7, and they give the ratio $f/f^* = 21/9$. By considering just J_2, J_3, and J_4 in the example, we can derive the worst ratio 1.875 for $m = 4$, and similarly, J_3 and J_4 give 1.5 for $m = 2$. Basically, the worst example can be constructed recursively. Let $f^*(S)$ and $f(S)$ be the finishing times of the optimal and worst schedules for a set of jobs with the index set S. Consider our example for $m = 9$, $f^*(\{1, 2, 3, 4\}) = 9$, and $f^*(\{2, 3, 4\}) = 8$. Let $F(m)$ be the worst ratio of f/f^* for m processors. Since J_2, J_3, J_4 only take four processors, $f(\{2, 3, 4\}) = 8F(4)$. In the worst schedule, J_1 is the last job to be executed, as given in Figure 7a. It will take at least six more time units than $f(\{2, 3, 4\})$. Thus we have

$$F(9) = \frac{8F(4) + 6}{9} = \frac{21}{9},$$

and similarly,

$$F(4) = \frac{3F(2) + 3}{4} = 1.875,$$

$$F(2) = \frac{F(1) + 2}{2} = 1.5,$$

$$F(1) = 1.$$

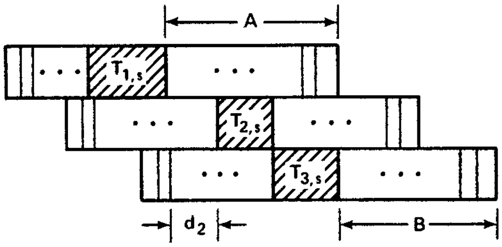

FIGURE 8

We conjecture that the worst example for different m is constructed recursively by a similar technique, and $F(m)$ will give the worst ratio of f/f^* for the 1-maximal flow-shop problem. Thus we can deduce from the above example a general formula for $F(m)$, namely,

$$F(m) = \max_{1 \le k \le m} \left[\frac{(m-1)F(k) + m - k + 1}{m} \right].$$

If, instead of finding the maximum k for $F(m)$, we choose $k = m/2$, then we have the following inequality for $F(m)$:

$$F(m) \ge \left(1 - \frac{1}{m}\right) F\left(\frac{m}{2}\right) + \frac{1}{2} + \frac{1}{m}.$$

Assume that $m = 2^i$; we have (see the appendix)

$$F(m) \ge \frac{1}{2}\left(1 + \frac{1}{m}\right)(1 + \log m).$$

We can conclude that for the 1-maximal flow shop, the worst ratio of f/f^* is $\Omega(\log m)$.

We have demonstrated that an upper bound on the ratio f/f^* for the J-maximal flow-shop problem is rather difficult to derive. In the following we try to concentrate on the approximate solution for $m = 3$.

LEMMA 10. *For $m = 3$, any schedule for the 1-maximal flow shop has the property that $f - f^* \le t_{3,s} + t_{2,u}$ for some J_s and J_u.*

PROOF. Consider the generalized schedule given in Figure 8. Let J_s be the last job such that $T_{3,s}$ executes immediately after $T_{2,s}$ and let d_2 be the time duration between the finish time of $T_{1,s}$ and the start time of $T_{2,s}$.

Obviously P_1 and P_3 are not idle during the periods A and B. Since $f^* \ge \sum_{i=1}^{n} t_{1,i}$, we have

$$f - f^* \le B + t_{3,s} + t_{2,s} + d_2 - A.$$

As the flow shop is 1-maximal, from Lemma 3 there exists a J_u such that

$$t_{2,u} \ge t_{1,s} + d_2 \ge t_{2,s} + d_2. \tag{2}$$

From the 1-maximal property, $A \ge B$; thus we have

$$f - f^* \le t_{3,s} + t_{2,u}. \qquad \square$$

THEOREM 9. *For $m = 3$, any schedule for the 1- and 3-maximal flow-shop problems gives $f/f^* < 5/3$, and the bound is the best possible.*

PROOF. The proofs for the 1- and 3-maximal flow-shop problems are so similar

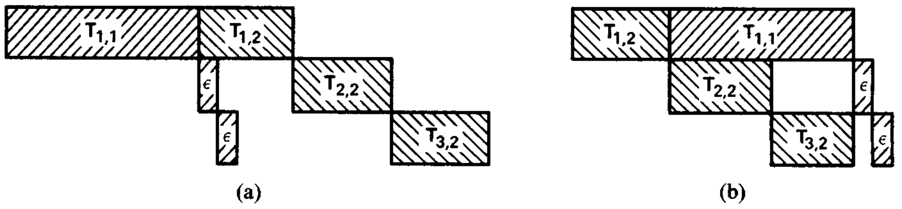

FIG. 9. 1-maximal flow shop for $m = 3, f/f^* \to 5/3$ (a) A bad schedule (b) An optimal schedule.

that only the 1-maximal flow-shop problem is considered. From eq. (2), since $t_{2,u} \geq t_{1,s}$ (footnote 2), it is easy to argue that $f^* \geq t_{1,u} + t_{2,u} + t_{3,s}$ (footnote 2), and thus from Lemma 10 we have

$$\frac{f}{f^*} \leq 1 + \frac{t_{3,s} + t_{2,u}}{t_{1,u} + t_{2,u} + t_{3,s}}.$$

As $t_{1,u} \geq t_{2,u}$ and $t_{1,u} \geq t_{3,s}$, it can be shown easily that

$$\frac{f}{f^*} \leq 1 + \frac{2}{3} = \frac{5}{3}.$$

The following example, with the schedules given in Figure 9, shows that this bound is the best possible.

Example. Let the flow shop have two jobs. They are

$$\begin{aligned} J_1: &\quad t_{1,1} = 2, \quad t_{2,1} = t_{3,1} = \epsilon; \\ J_2: &\quad t_{1,2} = t_{2,2} = t_{3,2} = 1. \end{aligned}$$

The heuristic and optimal schedules are illustrated in Figure 9a and b, respectively, and the ratio $f/f^* \to 5/3$ is achieved. □

THEOREM 10. *For $m = 3$, any schedule for the 2-maximal flow-shop problem gives $f/f^* \leq 2$, and this bound is the best possible.*

PROOF. Consider any schedule, and after the tasks are right-shifted (see Figure 5a), let f' be the finish time of this modified schedule. Obviously $f' \geq f$. Referring to Figure 5a, Lemma 7 shows that there exists a $t_{1,v}$ such that $t_{1,v} \geq \delta_1$, and Lemma 3 shows that there also exists a $t_{3,u}$ such that $t_{3,u} \geq d_3$. Then

$$f - f^* \leq f' - f^* \leq \delta_1 + d_3 < t_{1,v} + t_{3,u}. \tag{3}$$

Clearly, if $u = v$, then $f^* \geq t_{1,v} + t_{3,u}$. However, if $u \neq v$, we still have

$$f^* \geq t_{2,v} + t_{2,u} \geq t_{1,v} + t_{3,u},$$

so we have

$$\frac{f}{f^*} \leq 2.$$

The example which can achieve this bound as closely as possible is the following.

Example. Let the flow shop have two jobs. They are

$$\begin{aligned} J_1: &\quad t_{1,1} = t_{2,1} = 1, \quad t_{3,1} = \epsilon; \\ J_2: &\quad t_{1,2} = \epsilon, \quad t_{2,2} = t_{3,2} = 1; \end{aligned}$$

and the optimal and worst schedules are shown in Figure 10. □

[2] $t_{2,u} > t_{1,s}$ is only true when $u \neq s$; that is, $d_2 > 0$. However, if $u = s$ or $d_2 = 0, f^* \geq t_{1,u} + t_{2,u} + t_{3,s}$ is still true.

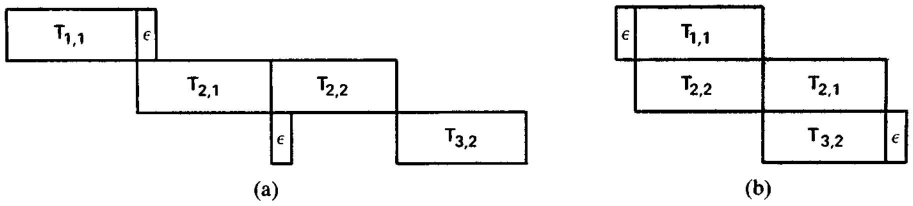

FIG. 10 2-maximal flow shop for $m = 3, f/f^* \rightarrow 2$. (a) A bad schedule (b) An optimal schedule

5. *Conclusion*

Much effort has been devoted to flow-shop problems because of their importance, but the problem remains intrinsically difficult for $m > 2$. Since the sizes of the tasks are usually known beforehand, it is worthwhile to study those problems with known task sizes. Similarly, assumptions about the lengths of execution time are often made in flow-shop problems [4, 11] such as I/O-bound systems, but they are usually too restricted.

In this paper we have considered a practical flow-shop model (J-maximal and J-minimal) in which certain tasks are larger or smaller than the others. We have shown that J-maximal and J-minimal flow-shop problems for $m > 2$ are still NP-complete if it is assumed that J_i can skip P_j if $t_{j,i} = 0$. Efficient optimal algortithms can be found only for some particular cases, such as $J = 1$ (or m) and all the other tasks in the same job are equal. Bounds for the worst ratio f/f^* have been derived. We have obtained a tight bound of $m - 1$ for the J-minimal flow shop with $m \geq 3$ and $\frac{3}{2}$ for $m = 2$. As for the J-maximal flow-shop problem, we can show only that the bound is always less than $O(\sqrt{m})$ and greater than $O(\log m)$ for a 1-maximal flow shop with $m > 3$, and we can show tight bounds for $m = 3$. The tight bound for J-maximal flow-shop problems is still unsolved, but we can conjecture that the bound $F(m)$ is given recursively by

$$F(m) = \max_{1 \leq k \leq m} \left\{ \left(1 - \frac{1}{m}\right) F(k) - \frac{k}{m} \right\} + 1 + \frac{1}{m},$$

with $F(1) = 1$.

Appendix

LEMMA

$$F(2^i) \geq \frac{(1 + 2^{-i})(i + 1)}{2}.$$

PROOF. The proof is by induction on i. For $i = 0$, $F(1) = 1$, and for $i = 1$, $F(2) = 1.5$. Assume the lemma is true for $i = k$. Then

$$\begin{aligned} F(2^{k+1}) &= (1 - 2^{-k-1})F(2^k) + \tfrac{1}{2} + 2^{-k-1} \\ &\geq \tfrac{1}{2}(1 - 2^{-k-1})(1 + 2^{-k})(k + 1) + \tfrac{1}{2} + 2^{-k-1} \\ &= \tfrac{1}{2}(1 + 2^{-k-1} - 2^{-2k-1})(k + 1) + \tfrac{1}{2}(1 + 2^{-k-1}) + 2^{-k-2} \\ &= \tfrac{1}{2}(1 + 2^{-k-1})(k + 2) - \tfrac{1}{2}(k + 1)2^{-2k-1} + 2^{-k-2} \\ &= \tfrac{1}{2}(1 + 2^{-k-1})(k + 2) + 2^{-k-2}(1 - (k + 1)2^{-k}). \end{aligned}$$

As $(k + 1)2^{-k}$ is always ≤ 1, we have proved the lemma. □

ACKNOWLEDGMENTS. The authors are very much indebted to Anne Brindle for her careful reading of, and helpful comments on this paper, and to an anonymous referee for noting two serious problems with the original manuscript and offering valuable suggestions in improving this paper.

REFERENCES

1. ACHUGBUE, J.O., AND CHIN, F.Y. Complexity and solutions of some three-stage flow-shop scheduling problems. Tech. Rep., Univ. of Alberta, Edmonton, Alberta, Canada, 1980.
2. BURNS, F., AND ROOKER, J. Three-stage flow shops with recessive second stage. *Oper. Res 26*, 1 (Jan-Feb 1978), 207-208.
3. COFFMAN, E.G. JR., ED. *Computer and Job/Shop Scheduling Theory.* Wiley, New York, 1976.
4. CONWAY, R.W., MAXWELL, W.L., AND MILLER, L.W. *Theory of Scheduling.* Addison-Wesley, Reading, Mass., 1967.
5. GAREY, M.R., JOHNSON, D.S., AND SETHI, R. The complexity of flowshop and jobshop scheduling. *Math. Oper Res. 1* (1976), 117-129.
6. GONZALEZ, T., AND SAHNI, S. On flowshop and jobshop schedules. *Oper Res. 26* (1978), 36-52.
7. JOHNSON, S.M Optimal two- and three-stage production schedules with setup times included. *Nav. Res. Log. Q. 1* (1954), 61-68.
8 KARP, R.M. Reducibility among combinatorial problems. In *Complexity of Computer Computations,* R.E Miller and J W. Thatcher, Eds., Plenum Press, New York, 1972, pp. 85-103
9. LENSTRA, J.K., RINNOOY KAN, A H.G., AND BRUCKER, P. Complexity of machine scheduling problems. *Ann. Discrete Math. 1* (1977), 343-362.
10. MCMAHON, G.B A study of algorithms for industrial scheduling problems. Ph D. Dissertation, Univ. of New South Wales, Kensington, New South Wales, Australia, 1971.
11 SMITH, M.L., PANWALKAR, S.S., AND DUDEK, R.A Flow sequencing with ordered processing time matrics. *Manage. Sci. 21* (1975), 544-549.
12. ULLMAN, J.D. Complexity of scheduling problems. In *Computer & Job/Shop Scheduling Theory,* E.G Coffman, Jr., Ed., Wiley, 1976, pp. 139-164.

RECEIVED OCTOBER 1977, REVISED APRIL 1980; ACCEPTED MAY 1980