OPERATIONS RESEARCH
Vol. 26, No. 1, January-February 1978

0030-364X/78/2601-0036 $01.25
© 1978 Operations Research Society of America

Flowshop and Jobshop Schedules: Complexity and Approximation

TEOFILO GONZALEZ and SARTAJ SAHNI

University of Minnesota, Minneapolis, Minnesota

(Received March 1976; accepted, February 1977)

We show that finding minimum finish time preemptive and non-preemptive schedules for flow shops and job shops is NP-complete. Bounds on the performance of various heuristics to generate reasonably good schedules are also obtained.

FLOW SHOPS and job shops are ordered sets of $m \geqq 1$ processors (or machines), $\langle P_1, \cdots, P_m \rangle$. In the case of a *flow shop* the processing time required by task j of job i is denoted by $t_{j,i}$, $1 \leqq j \leqq m$. For any job i, task j is to be performed on processor P_j. In the case of a *job shop* the processing time required by task j of job i will be denoted $t_{k,i,j}$. Task j is to be performed on processor k. In this case, no restriction is placed on the number of tasks a job may have. In the case of both flow shops and job shops, for any job i the processing of task j, $j \geqq 2$ can begin only after task $j-1$ has been completed. The *finish time*, FT(S), of a schedule S is the time at which all tasks of all jobs have been completed (it is assumed that the schedule starts at time zero). If $f_i(S)$ denotes the time at which all tasks of job i have been completed in schedule S, then clearly $\mathrm{FT}(S) = \max_i \{f_i(S)\}$. An *optimal finish time* (OFT) schedule is one that has the least finish time among all feasible schedules. For any schedule S, the mean flow time, MFT(S), is defined to be the quantity $\sum f_i(S)/n$. An *optimal mean flow time* (OMFT) schedule is one that has the least mean flow time among all feasible schedules. Finally, for both of the models under consideration, we shall distinguish between preemptive and non-preemptive schedules. The reader is referred to [2] and [3] for more details on the scheduling models being considered. Gonzalez and Sahni [6] study a related model, the *open shop*.

Several strategies for obtaining OFT and OMFT schedules for flow shops and job shops have been advanced (see, for example, [2] and [3]). Despite all the research effort devoted to these problems, there are no efficient (polynomial time) algorithms known.

In [1] and [4] it is shown that these problems are NP-complete when one is restricted to non-preemptive schedules. In this paper we extend the

NP-completeness results of [1] and [4] for OFT non-preemptive schedules to the case of preemptive schedules. A more restricted version of the OFT non-preemptive flow-shop problem is also shown to be NP-complete. The best known algorithms for all known NP-complete problems have a worst case computing time larger than any polynomial in the problem size. Furthermore, any NP-complete problem has a polynomial time algorithm iff all the others also have polynomial time algorithms [9–11]. Hence, by showing certain flow-shop and job-shop problems to be NP-complete, we are in some sense establishing that it will indeed be very difficult to find efficient algorithms for them. In fact, empirical evidence and intuition suggest that NP-complete problems are not solvable in polynomial time.

Since it is unlikely that efficient algorithms for the problems of Section 1 exist, an alternate approach, obtaining approximately optimal schedules, is examined in Section 2. In this section we obtain bounds comparing optimal and arbitrary busy schedules (to be defined later) for flow shops and job shops. We also present heuristics that result in schedules with a mean finish time and finish time better than the worst busy schedules. Finally, a comparison is made between the finish times of preemptive and non-preemptive schedules.

In obtaining the NP-completeness results of the next section, we shall make use of the following NP-complete problems.

Partition. A multiset $S=\{a_1, \cdots, a_n\}$ is said to have a partition if there exists a subset, u, of the indices 1 through n such that $\sum_{i\in u} a_i = (\sum_1^n a_i)/2$. The partition problem [9] is to determine whether an arbitrary multiset S has a partition. The $\{a_i\}$ may be assumed to be integers.

3-Partition. Given a positive integer B and a multiset of integers $A=\{a_1, \cdots, a_m\}$ with $m=3n$, $\sum_{i=1}^m a_i = nB$ and $B/4<a_i<B/2$ for $1\leqq i\leqq m$, does there exist a partition of A into 3 element sets $\{A_1, \cdots, A_n\}$ such that $\sum_{a\in A_i} a=B$ for $i=1, \cdots, n$? This problem is shown NP-complete in [4], even if the problem size is measured by $\sum a_i$. The difference between NP-completeness with respect to binary encoding and the stronger NP-completeness with respect to unary encoding is important. The partition problem can be solved in time $0(n\sum a_i)$. For 3-partition there exists an algorithm of complexity polynomial in n and $\sum a_i$ if and only if there is a polynomial time algorithm for every NP-complete problem.

Since NP-complete problems are normally stated as language recognition problems, we restate the OFT problem as one.

FOFT: Given an m-processor, n-job flow-shop problem with task times $t_{j,i}$, $1\leqq j\leqq m$ and $1\leqq i\leqq n$ and a number τ, does it have a schedule with finish time $\leqq\tau$?

When it is necessary to distinguish between preemptive and non-preemptive schedules, we shall prefix FOFT with the type of schedule being considered.

JOFT: This is the same as FOFT except it refers to a job shop.

We shall also make use of the operator "α," as in $\mathcal{P}_1 \alpha \mathcal{P}_2$ to mean problem $\mathcal{P}_1$ polynomially reduces to problem $\mathcal{P}_2$. Informally, this will mean that if $\mathcal{P}_2$ can be solved in polynomial time, then so can $\mathcal{P}_1$. By $P = \text{NP}$ we shall denote the question: Is the class of non-deterministic polynomial time languages the same as the class of deterministic polynomial time languages? For more details regarding our notation on NP-complete problems, the reader is referred to [9–11].

1. COMPLEXITY OF PREEMPTIVE AND NON-PREEMPTIVE SCHEDULING

Flow Shop

OFT non-preemptive schedules for the two-processor ($m = 2$) flow shop can be obtained in $O(n \log n)$ time using Johnson's algorithm (see [8 or 3, p. 83]). For the case $m = 2$ one can easily show that an OFT preemptive schedule has the same finish time as an OFT non-preemptive schedule. Hence, Johnson's algorithm also gives an OFT preemptive schedule. A linear time algorithm is presented in [6], which guarantees OFT preemptive and non-preemptive schedules for the two-processor open shop. It is interesting to note that by eliminating the task ordering, a more efficient algorithm is obtained. In this section we shall show that finding OFT preemptive and non-preemptive flow shop schedules for $m > 2$ is NP-complete. This is true even when the jobs are restricted to have at most two nonzero tasks each. This then gives us the simplest case of the flow-shop problem that is NP-complete and for which no polynomial algorithm is known. (Note that when jobs have only one task each, OFT schedules may be trivially obtained.) When the complexity of the algorithm is measured in terms of the sum of the lengths of the tasks, the flow shop problem remains NP-complete, as shown in [4]. This result is extended for preemptive schedules. The extension, however, requires the use of a job with three tasks. In [6] it is shown that for $m > 2$ OFT preemptive schedules for the open shop can be obtained in polynomial time. For non-preemptive schedules, the problem remains NP-complete.

THEOREM 1. *FOFT with $m = 3$ and no job having more than two nonzero tasks is NP-complete.*

Proof. The proof is presented as two lemmas. Lemma 1 shows that Partition α Preemptive FOFT. Lemma 2 shows that Preemptive FOFT is recognizable in nondeterministic polynomial time. The same proof also shows that non-preemptive FOFT is also NP-complete.

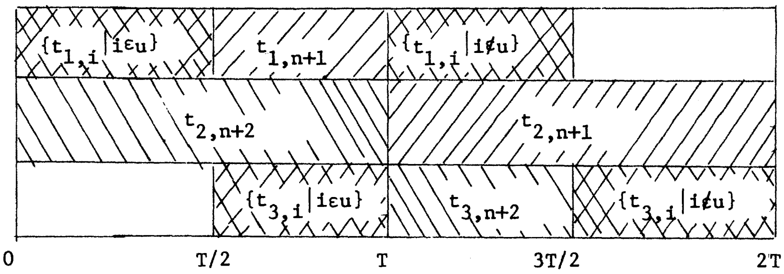
LEMMA 1. *Partition α preemptive FOFT with $m = 3$ and at most two nonzero tasks per job.*

Proof. From the partition problem $S=\{a_1, \cdots, a_n\}$ construct the following preemptive flow-shop problem, FS, with $n+2$ jobs, $m=3$ machines and at most 2 nonzero tasks per job: $t_{1,i}=a_i$; $t_{2,i}=0$; $t_{3,i}=a_i$, $1 \leqq i \leqq n$ $t_{1,n+1}=T/2$; $t_{2,n+1}=T$; $t_{3,n+1}=0$, $t_{1,n+2}=0$; $t_{2,n+2}=T$; $t_{3,n+2}=T/2$, where $T=\sum_1^n a_i$ and $\tau=2T$.

We now show that the above flow-shop problem has a preemptive schedule with finish time $\leqq 2T$ iff S has a partition.

(a) If S has a partition u, then there is a non-preemptive schedule with finish time $2T$. One such schedule is shown in Figure 1.

(b) If S has no partition, then all preemptive schedules for FS must have a finish time $>2T$. This can be shown by contradiction. Assume that there is a preemptive schedule for FS with finish time $\leqq 2T$. We make the



$\{t_{1,i} | i \varepsilon u\}$
$t_{1,n+1}$
$\{t_{1,i} | i \notin u\}$
$t_{2,n+2}$
$t_{2,n+1}$
$\{t_{3,i} | i \varepsilon u\}$
$t_{3,n+2}$
$\{t_{3,i} | i \notin u\}$
0
T/2
T
3T/2
2T

Figure 1

following observations regarding this schedule: (i) task $t_{1,n+1}$ must finish by time T (as $t_{2,n+1}=T$ and cannot start until $t_{1,n+1}$ finishes); (ii) task $t_{3,n+2}$ cannot start before T units of time have elapsed as $t_{2,n+2}=T$.

Observation (i) implies that only $T/2$ of the first T time units are free on machine one. Let V be the set of indices of tasks completed on machine 1 by time T (excluding task $t_{1,n+1}$). Then $\sum_{i \in V} t_{1,i} < T/2$ as S has no partition. Hence $\sum_{i \notin V} t_{3,i} > T/2$. The processing of jobs not included in V cannot commence on machine 3 until after time T since their machine 1 processing is not completed until after T. This together with observation (ii) implies that the total amount of processing left for machine 3 at time T is $t_{3,n+2}+\sum_{i \notin V} t_{3,i} > T$. The schedule length must therefore be greater than $2T$.

The following result, which follows from Lemma 1, is also obtained in [1]. Their proof is similar.

COROLLARY 1. *Partition α non-preemptive FOFT with $m=3$ and at most two nonzero tasks per job.*

Proof. The construction of Lemma 1 yields a flow-shop problem that has a non-preemptive schedule with finish time τ iff the corresponding partition problem has a partition.

LEMMA 2. *Preemptive FOFT is in NP.*

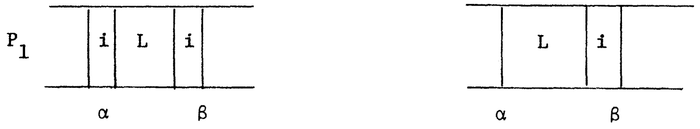
Proof. One may easily construct a non-deterministic Turing machine that guesses a preemptive schedule and verifies that it is of length $\leqq \tau$. In order for this Turing machine to be of polynomial complexity, we must show that every flow-shop problem has an optimal preemptive schedule with a polynomial number of preemptions. We show this by construction. Let R be an optimal preemptive schedule for an m-processor, n-job flow-shop problem. If on any processor j there is a job k such that between its preemption and next resumption on that processor no task $t_{j,i}$, $i \neq k$ is completed, then this preemption for job k can be eliminated without affecting the schedules for the other processors. That is, if the schedule for processor j has a subsequence[1] $(l_1 = k, s_1, f_1)$, (l_2, s_2, f_2), $\cdots$, $(l_{r-1}, s_{r-1}, f_{r-1})$, $(l_r = k, s_r, f_r) l_i \neq k$, $2 \leqq i < r$ and none of the tasks on j for jobs $l_2, l_3, \cdots l_{r-1}$ finishes in this interval, then the schedule may be modified to $(l_1 = k, s_1, f_1 + \Delta)$, $(l_2, s_2 + \Delta, f_2 + \Delta)$, $\cdots$, $(l_{r-1}, s_{r-1} + \Delta, f_r)$ where $\Delta = f_r - s_r > 0$. Repeating this preemption elimination process, one will eventually obtain a preemptive schedule with the same finish time as R and having at most n^2 preemptions per processor. Hence, there is an optimal preemptive schedule with at most $n^2 m$ preemptions.

Next we show that preemptive flow-shop schedules need not have preemptions on the first and last processors. This proof will be of use in Lemma 4.

LEMMA 3. *Any preemptive schedule S for a flow shop with m processors can be transformed to a schedule S' in which there are no preemptions on processors* 1 *and m and $FT(S') = FT(S)$.*

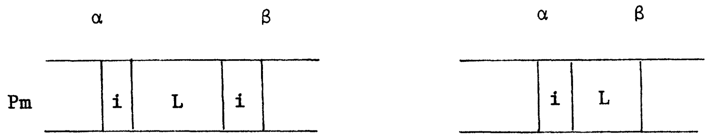
Proof. We show that S can be transformed into a schedule S' that is no longer than S. We begin by defining two transformations that when applied to S will not increase the schedule length.

The first transformation, which we shall term type 1, is applied to processor 1 and is illustrated below. In this case task i was preempted by tasks in set L. A new schedule is constructed by starting tasks in L in the same order at time α and then task i. The finish time of any task in L and task i is never increased, and thus no precedence constraints for their corresponding tasks on $P_2, \cdots, P_m$ can be violated.



[1] The triple (l_i, s_i, f_i) means task l_i is processed on processor j from s_i to f_i.

The second transformation (type 2) is applied on the last processor and is illustrated below. Task i has been preempted by the tasks in set L. We now modify the schedule so as to finish task i first and then all tasks in set L in the same order as before. As the start time of all these tasks is never decreased, none of the precedence constraints for their corresponding tasks on $P_1, \cdots, P_{m-1}$ is violated.



It follows from the definitions of the two transformations that after an exhaustive application of them, we obtain the schedule S' and $\mathrm{FT}(S') = \mathrm{FT}(S)$.

Our next result shows that preemptive FOFT is NP-complete even when the problem size is measured as the sum of the lengths of the tasks. Note that this proof requires jobs with 3 tasks.

THEOREM 2.[2] *Preemptive FOFT with $m=3$ and the problem size being measured as the sum of the length of the tasks is NP-complete.*

Proof. The proof is presented in two parts. Lemma 4 shows the reduction 3-partition α preemptive FOFT. Lemma 2 shows that FOFT is recognizable in nondeterminstic polynomial time.

LEMMA 4. *3-Partition α Preemptive FOFT with $m=3$.*

Proof. From the 3-partition problem $C=(a_1, \cdots a_s, B)$ and $s=3t$, construct the following preemptive flow shop problem FS, with $s+t+2$ jobs and $m=3$ machines: $t_{1,i}=t_{3,i}=a_i$; $t_{2,i}=0$ for $1 \leqq i \leqq s$; $t_{1,s+1}=0$; $t_{2,s+1}=2B$; $t_{3,s+1}=B$; $t_{1,s+i+1}=t_{3,s+i+1}=B$; $t_{2,s+i+1}=2B$ for $1 \leqq i \leqq t-2$; $t_{1,s+t}=B$; $t_{2,s+t}=2B$; $t_{3,s+t}=0$; $t_{1,s+t+1}=t_{2,s+t+1}=0$; $t_{3,s+t+1}=B$; $t_{1,s+t+2}=B$; $t_{2,s+t+2}=t_{3,s+t+2}=0$; where $\sum_{i=1}^{s} a_i = tB$ and $\tau = 2tB$.

We show that the above flow-shop problem has a preemptive schedule with finish time $\leqq 2tB$ iff C has a 3-partition.

(a) If 3-partition has a solution, then there is a non-preemptive schedule with finish time $\leqq 2tB$. One such schedule is shown in Figure 2. The L_i's represent disjoint sets such that $|L_i|=3$, $\bigcup_{i=1}^{t} L_i = \{1, \cdots, s\}$ and $\sum_{j \in L_i} a_j = B$ for $i=1, \cdots, t$.

(b) If C has no 3-partition, then all preemptive schedules for FS must have a finish time $>2tB$.

Assume there is a flow-shop schedule for FS with finish time $\leqq 2tB$. Then in this schedule exactly B units of the processing required for jobs

[2] M. R. Garey and D. Johnson have independently obtained a proof of this theorem.

$1, \cdots, s$ must be scheduled to complete on P_1, by time $2B$. To see this, observe that if more than B units are scheduled, then none of the P_2 tasks of jobs $s+2, \cdots, s+t$ can begin until after time $2B$. This implies the schedule length must be greater than $2tB$. On the other hand, if fewer than B units of the jobs $1, \cdots, s$ are scheduled to complete on P_1 before $2B$, then there would be some idle time on P_3 before $2B$. This would result in a schedule of length greater than $2tB$. If one of these jobs (i.e., jobs from $1, \cdots, s$ scheduled on P_1 before $2B$) finishes at time $2B$, then its corresponding P_3 task cannot commence until $2B$. This would again result in idle time on P_3. From Lemma 3 we know that no preemptions on P_1 are needed. Hence the task that finishes at $2B$ on P_1 must be one of length B and this task starts at B.

Before time $2B$ on P_2 the only job available to schedule is $t_{2,s+1}$; hence we should schedule it at that time. On P_3 we should schedule task $t_{3,s+t+1}$ and the corresponding tasks from $1, \cdots, s$ that were scheduled on P_1, as

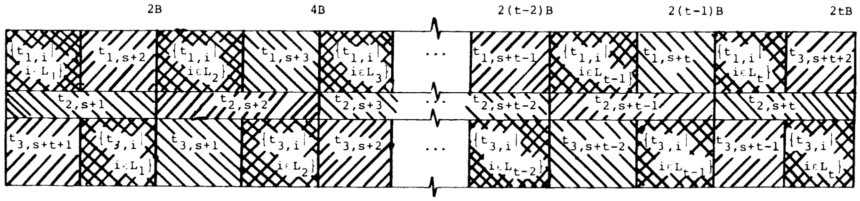


Figure 2

those are the only candidates and we must not introduce idle time on the processors.

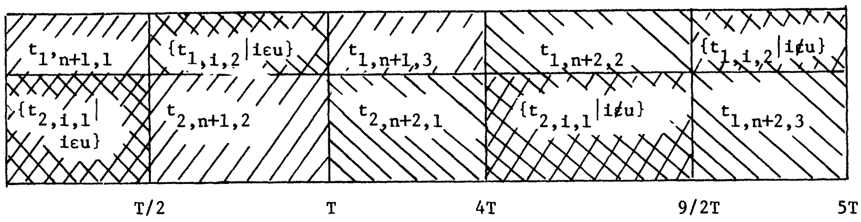
We now need to choose the task of time B for P_1. If we schedule job $t_{1,s+t+2}$, then after time $2B$ we will have idle time and a schedule of length $>2tB$. From the symmetry of the problem and by using the same arguments of the first part of the proof, we conclude that $t_{2,s+t}$ should be scheduled from $2(t-1)B$ to time $2tB$. Then if we schedule $t_{1,s+t}$ before $2B$, we would have idle time after $2B$ on P_2. So the only jobs with time B on P_1 are identical, and we may choose any one from jobs $s+2$ to $s+t-1$.

We have now reduced the problem and need only consider the part of the schedule from time $2B$ to $2tB$. Note from Figure 2 that this is the original problem but with a smaller number of jobs.

Now we continue to apply this transformation, which is clearly the only possibility, until we cannot find a partition equal to B for the remaining jobs $1, \cdots, s$. (Note that this will always happen, as there is no 3-partition.) In this case we would have $>B$ or $<B$ units to schedule on P_1, and we would introduce idle time on P_2 or P_3; thus our schedule would have length $>2tB$.

Job Shop

The preceding results trivially imply that a severely restricted form of the job-shop problem for $m>2$ is NP-complete. For the job shop with $m=2$, however, no polynomial time algorithm is known. In [3, p. 105] an $O(n \log n)$ algorithm to obtain OFT non-preemptive schedules when $m=2$ and the jobs are restricted to have at most two nonzero tasks is presented. For this case OFT preemptive schedules may be similarly obtained. For the non-preemptive case, it is known [1, 2] that when $m=2$ and the job mix consists of $n-1$ jobs with one nonzero task each and an additional job with three nonzero tasks, then the problem is NP-complete. We extend this result to the case of preemptive schedules. We show that finding OFT preemptive schedules when $m=2$ is NP-complete even when the job mix contains only two jobs with three nonzero tasks. The proof is



$t_{1,n+1,1}$
$\{t_{1,i,2} | i\epsilon u\}$
$t_{1,n+1,3}$
$t_{1,n+2,2}$
$\{t_{1,i,2} | i\notin u\}$
$\{t_{2,i,1} | i\epsilon u\}$
$t_{2,n+1,2}$
$t_{2,n+2,1}$
$\{t_{2,i,1} | i\notin u\}$
$t_{1,n+2,3}$
T/2
T
4T
9/2T
5T

Figure 3

presented in Lemmas 5 and 6. In [4] it is shown that even if the complexity is measured in terms of the sum of the lengths of the tasks, the two-processor, non-preemptive job-shop problem is NP-complete. This result is extended to include preemptive schedules. However, this extension requires a job with $n/3$ tasks.

LEMMA 5. *Partition α preemptive JOFT.*

Proof. From the partition problem $S=\{a_1, \cdots, a_n\}$ construct the following job-shop problem JS, with $n+2$ jobs, $m=2$ processors and all jobs having 2 nonzero tasks except for two jobs with 3 nonzero tasks: $t_{2,i,1}=t_{1,i,2}=a_i$ for $1 \leqq i \leqq n$; $t_{1,n+1,1}=t_{2,n+1,2}=T/2$; $t_{1,n+1,3}=3T$; $t_{2,n+2,1}=3T$; $t_{1,n+2,2}=t_{2,n+2,3}=T/2$; where $T=\sum_1^n a_i$ and $\tau=5T$.

We now show that the above job-shop problem JS has a preemptive schedule with finish time $\leqq 5T$ iff S has a partition.

(a) If S has a partition u, then there is a schedule with finish time $5T$. One such schedule is shown in Figure 3.

(b) If S has no partition, then all schedules for JS must have a finish time $>5T$.

This is shown by contradiction. Assume that there is a schedule for

JS with finish time $\leqq 5T$. Observe (i) on processor 2, task $t_{2,n+1,2}$ must be completed before time $2T$ and $t_{2,n+2,1}$ before $4T$ (therefore, before $4T$ only $T/2$ units are free for jobs $1, \cdots, n$), and (ii) on processor 1, $t_{1,n+1,3}$ cannot start until T and $t_{1,n+2,2}$ cannot start until $3T$. Hence, processor 1 is committed to processing $t_{1,n+1,3}$ and $t_{1,n+2,2}$ after time T. Since these two tasks together require $7T/2$ units, at most $T/2$ units are free after time T for jobs $1, \cdots, n$. This in turn implies that there are at least $T/2$ units of free time on processor 1 before time T. Let $\bar{U}$ be the set of indices of jobs being processed in the time interval $[0, T]$ on processor 1. Let $U=\{i|i\in\bar{U}$ and $i\leqq n\}$. Since the schedule has a finish time of $5T$, there can be no idle time on either of the processors. Hence, $\sum_{i\in U} t_{1,i,2}\geqq T/2$. But the only jobs that can start on processor 1 are those for which $t_{2,i,1}$ has completed. Hence, $\sum_{i\in U} t_{2,i,1}=\sum_{i\in U} t_{1,i,2}\geqq T/2$ must be completed by T on processor 2. By (i) this processor has at most $T/2$ units free before $4T$ and hence before T. Consequently, $\sum_{i\in U} t_{2,i,1}=T/2$. This contradicts the assumption that S has no partition.

Corollary 2. *Partition α nonpreemptive JOFT.*

Proof. Same as above. A simpler proof of this corollary may be found in [1, 2].

Lemma 6. *JOFT is in NP.*

Proof. Similar to that for Lemma 2. Note that the bound on the number of preemptions will now be n^2l, where l is the maximum number of non-zero tasks for any job.

The NP-completeness result of Lemmas 5 and 6 may be strengthened by increasing the number of tasks per job. When this is done, it can be shown that the problem of obtaining OFT preemptive and non-preemptive finish time schedules for the job shop is NP-complete even when the sum of task lengths is used as the complexity measure. This result is obtained by showing, in Lemma 7, 3-partition α preemptive JOFT with $m=2$. Lemma 6 provides the remainder of the proof.

Lemma 7. *3-Partition α preemptive JOFT with $m=2$.*

Proof. From the 3-partition problem $C=(a_1, \cdots, a_s, B)$ and $s=3t$, construct the following preemptive job-shop problem JS, with $s+1$ jobs and $m=2$ machines: $t_{1,i,1}=t_{2,i,2}=a_i$ for $1\leqq i\leqq s$; $t_{(l \bmod 2)+1,s+1,l}=B$ for $l=1, \cdots, 2t$, where $\sum_{i=1}^{s} a_i=tB$ and $\tau=2tB$.

We now show that the above job-shop problem has a preemptive schedule with finish time $\leqq 2tB$ iff C has a 3-partition.

(a) If 3-partition has a solution, then there is a non-preemptive schedule with finish time $\leqq 2tB$. One such schedule is given in Figure 4. L_i's represent disjoint sets such that $|L_j|=3$, $\bigcup_{i=1}^{t} L_i=\{1, \cdots, s\}$ and $\sum_{j\in L_i} a_j=B$ for $j=1, \cdots t$.

(b) If C has no 3-partition, then all preemptive schedules for JS must have a finish time $>2tB$.

This can be shown by contradiction. Assume there is a schedule with time $\leqq 2tB$. Job $s+1$ has to be scheduled as in Figure 4 if we want a schedule of length $2tB$. Now since C has no 3-partition, the first time we attempt to schedule a set of jobs with sum$\neq B$ we will introduce idle time on processor 1 or 2. Hence the schedule must be of length $>2tB$.

Note that this proof requires a job with $0(s/3)$ tasks, where $s+1$ is the total number of jobs in the shop. We could easily extend this result to the case in which all jobs will have at most $0(s^{1/l})$ tasks for some constant l. This extension is obtained by proving that $m^{1/l}$-partition is NP-complete. (The $m^{1/l}$-partition problem is to determine whether the multiset $A=$

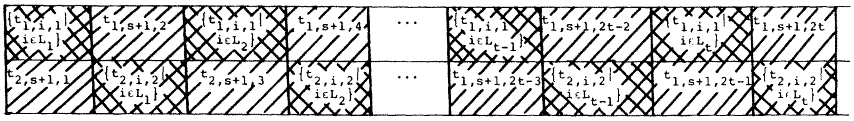

$\{t_{1,i,1} | i\varepsilon L_1\}$
$t_{1,s+1,2}$
$\{t_{1,i,1} | i\varepsilon L_2\}$
$t_{1,s+1,4}$
$\cdots$
$\{t_{1,i,1} | i\varepsilon L_{t-1}\}$
$t_{1,s+1,2t-2}$
$\{t_{1,i,1} | i\varepsilon L_t\}$
$t_{1,s+1,2t}$
$t_{2,s+1,1}$
$\{t_{2,i,2} | i\varepsilon L_1\}$
$t_{2,s+1,3}$
$\{t_{2,i,2} | i\varepsilon L_2\}$
$\cdots$
$t_{1,s+1,2t-3}$
$\{t_{2,i,2} | i\varepsilon L_{t-1}\}$
$t_{1,s+1,2t-1}$
$\{t_{2,i,2} | i\varepsilon L_t\}$

Figure 4

$\{a_1, \cdots, a_m\}$ has a partition $\{A_1, \cdots, A_n\}$ such that each A_{ij} has $m^{1/l}$ elements and all A_i have the same sum.) This will result with jobs with at most $0(s^{1/l})$ tasks for the job-shop problem JS with $s+1$ jobs. We should note that the maximum number of tasks per job is not a constant as was the case in Lemma 5.

2. APPROXIMATE SOLUTIONS

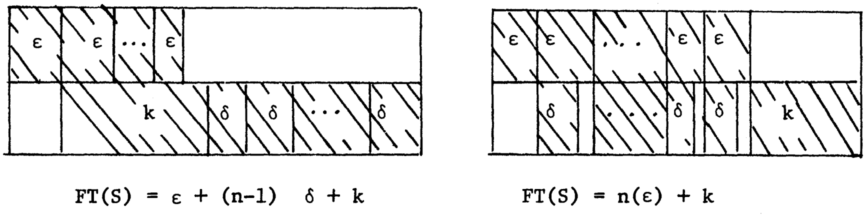
Since the problem of finding OFT and OMFT schedules for flow shops and job shops is NP-complete (see [4] for NP-completeness of OMFT), we turn our attention to obtaining schedules whose performance approximates that of optimal schedules. To begin with, we derive a bound for the ratio of worst and best schedules for the two performance measures being considered. We then present approximation algorithms that generate schedules with a worse case bound smaller than this. In examining "worst" schedules, we restrict ourselves to busy schedules. A *busy schedule* is a schedule in which at all times from start to finish at least one processor is processing a task. For a given set of jobs and a schedule S we denote by $\mathrm{FT}(S)$ the finish time of S and by $\mathrm{MFT}(S)$ the mean flow time of S.

LEMMA 8. *Let S^* be an OMFT schedule for an m-processor, n-job flow (or job) shop problem. Let S be any busy schedule for this problem. Then $MFT(S)/MFT(S^*)\leqq n$.*

Proof. For each job i define L_i to be the sum of its task times. Let $T=\sum_1^n L_i$. For any busy schedule S we know that $f_i(S)\leqq T$. Hence $\mathrm{MFT}(S)\leqq T$. For S^* we know that $f_i(S^*)\geqq L_i$ and so $\mathrm{MFT}(S^*)\geqq \sum L_i/n=T/n$. Hence, $\mathrm{MFT}(S)/\mathrm{MFT}(S^*)\leqq n$.

Since rather crude approximations were used to obtain this bound, it is surprising that there are OFT schedules S such that $\mathrm{MFT}(S)/\mathrm{MFT}(S^*)=n$. In fact, this bound may be achieved by OFT schedules generated by Johnson's algorithm on 2 processors [3]. The next example gives an instance of a job mix for which this happens.

Example 1. Consider the 2-processor, n-job flow-shop problem with task times $t_{1,i}=\epsilon$, $1\leqq i\leqq n$, $t_{2,1}=k$ and $t_{2,i}=\delta$, $2\leqq i\leqq n$. $\delta<\epsilon\ll k/n^2$. One



(a) OFT schedule from Johnson's algorithm (b) OMFT schedule

Figure 5

may easily verify that Johnson's algorithm generates the permutation schedule $S=(1,\cdots,n)$. The mean flow time of this schedule is

$$\begin{aligned}\mathrm{MFT}(S)&=((k+\epsilon)+(k+\epsilon+\delta)+\cdots+(k+\epsilon+(n-1)\delta))/n\\&=(k+\epsilon)+(n-1)\delta/2.\end{aligned}$$

An OMFT schedule, S^*, for this job set is obtained by using the permutation 2, 3, $\cdots$, n, 1. The mean flow time of this schedule (Figure 5) is:

$$\begin{aligned}\mathrm{MFT}(S^*)&=((\epsilon+\delta)+(2\epsilon+\delta)+\cdots+((n-1)\epsilon+\delta)+(n\epsilon+k))/n\\&=(n+1)\epsilon/2+\delta+k/n.\end{aligned}$$

Hence $\mathrm{MFT}(S)/\mathrm{MFT}(S^*)=[n(k+\epsilon)+n(n-1)\delta/2]/[n(n+1)\epsilon/2+n\delta+k]$. As ϵ approaches zero this bound becomes $\lim_{\epsilon\to 0}\mathrm{MFT}(S)/\mathrm{MFT}(S^*)=nk/k=n$. Note that this example also shows that the bound of n holds for job-shop schedules and also when preemptive schedules are allowed.

A simple heuristic that results in schedules with an *mft* that in the worst case is closer to the optimal than the bound of Lemma 8 (it is assumed that $m<n$) is obtained by processing jobs in order of nondecreasing

L_i (L_i = sum of task times for job i). This heuristic will be referred to as SPT (see [3], p. 76).

LEMMA 9. *Let S^* be an OMFT schedule for an m-processor, n-job flow (or job) shop problem. Let S be a SPT schedule for this problem. Then $MFT(S)/MFT(S^*) \leqq m$.*

Proof. Let L_i be the sum of the task times for job i. Without loss of generality we assume that $L_1 \leqq L_2 \leqq \cdots \leqq L_n$. Let $f_i(S)$ be the finish time of job i using the SPT schedule S. Then $f_i(S) \leqq \sum_1^i L_j$ and so $\mathrm{MFT}(S) = \sum_1^n f_i(S)/n \leqq (\sum_{i=1}^n \sum_{j=1}^i L_j)/n$. Let $(i_1, i_2, \cdots, i_n)$ be the order in which jobs finish in the OMFT schedule S^*. For S^* we have $f_{i_k}(S^*) \geqq \sum_{j=1}^k L_{i_j}/m \geqq \sum_{j=1}^k L_j/m$ and so $\mathrm{MFT}(S^*) \geqq (1/n) \sum_{k=1}^n \sum_{j=1}^k L_j/m$. Consequently, $\mathrm{MFT}(S)/\mathrm{MFT}(S^*) \leqq m$.

Example 2. Consider the 2-processor, $2n$ job-shop problem with task times $t_{1,i} = k$, $t_{2,i} = \epsilon_1$ for $1 \leqq i \leqq n$ and $t_{1,i} = \epsilon_2$, $t_{2,i} = k$ for $n+1 \leqq i \leqq 2n$, where $\epsilon_1 < \epsilon_2 \ll k/n^2$. One may easily verify that the SPT algorithm generates the permutation schedule $S = (1, \cdots, 2n)$. The mean flow time of this schedule is

$$\begin{aligned}\mathrm{MFT}(S) &= ((k+\epsilon_1) + (2k+\epsilon_1) + \cdots + (nk+\epsilon_1) \\ &\quad + ((n+1)k+\epsilon_2) + \cdots + (2nk+\epsilon_2))/n \\ &= (1/n) \textstyle\sum_{i=1}^{2n} ik + \epsilon_1 + \epsilon_2.\end{aligned}$$

An OMFT schedule, S^*, for this job set is obtained by using the permutation $(n+1, 1, n+2, 2, \cdots, n+i, i, \cdots, 2n, n)$. The mean flow time of this schedule (Figure 6) is

$$\begin{aligned}\mathrm{MFT}(S^*) &= ((k+\epsilon_2) + (k+\epsilon_2+\epsilon_1) + \cdots + (nk+n\epsilon_2) + (nk+n\epsilon_2+\epsilon_1))/n \\ &= (2 \textstyle\sum_{i=1}^n ik + 2 \sum_{i=1}^n i\epsilon_2 + n\epsilon_1)/n.\end{aligned}$$

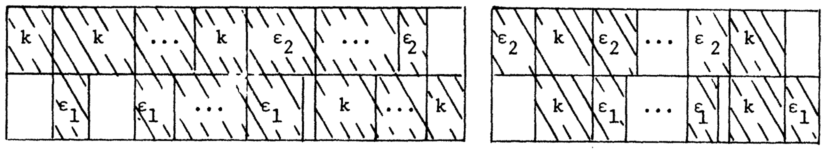
Hence $\mathrm{MFT}(S)/\mathrm{MFT}(S^*) = (\sum_{i=1}^{2n} ik + n\epsilon_1 + n\epsilon_2)/(2\sum_{i=1}^n ik + 2\sum_{i=1}^n i\epsilon_2 + n\epsilon_1)$. As ϵ_1 and ϵ_2 approach zero this bound becomes $\lim_{\epsilon_1 \to 0 \epsilon_2 \to 0} \mathrm{MFT}(S)/\mathrm{MFT}(S^*) = (2n)(2n+1)/(2n(n+1)) \simeq 2$. This may be extended to any arbitrary m by using the following job set: $t_{1,i} = k$, $t_{2,i} = \epsilon_1, \cdots, t_{m,i} = \epsilon_1$ for $1 \leqq i \leqq n$, $t_{1,i} = \epsilon_2$, $t_{2,i} = k, \cdots, t_{m,i} = \epsilon_2$ for $n+1 \leqq i \leqq 2n, \cdots, t_{1,i} = \epsilon_m$, $t_{2,i} = \epsilon_m, \cdots, t_{m,i} = k (m-1)n+1 \leqq i \leqq mn$, where $\epsilon_i < \epsilon_{i+1}$ for $1 \leqq i < m$ and $\epsilon_m \ll k/n^2$. This example shows that the bound of Lemma 9 is the best possible.

Let us now turn our attention to the finish time properties of busy schedules.

LEMMA 10. *Let S^* be an OFT schedule for an $m > 2$ processor n-job flow (or job) shop problem. Let S be any busy schedule for this problem. Then $FT(S)/FT(S^*) \leqq m$.*

Proof. Let T be the sum of task times for all n jobs. Then clearly, for any busy schedule S, $\mathrm{FT}(S) \leqq T$. Also, for any schedule S^*, we trivially have $\mathrm{FT}(S^*) \geqq T/m$. Hence $\mathrm{FT}(S)/\mathrm{FT}(S^*) \leqq m$.

Once again, as in the case of Lemma 8, the proof technique would seem to indicate that any "reasonable" heuristic would result in schedules with a worst case bound less than m. This unfortunately is not the case. We define by LPT the heuristic: schedule jobs in order of nonincreasing L_i. For LPT schedules the bound of Lemma 10 is tight. Note that this heuristic is similar to the one used by Graham [7] to schedule identical processors and by Gonzalez, Ibarra, and Sahni [5] to schedule uniform processors. In both these earlier applications of the heuristic, LPT schedules had a worst case finish time at most a "small" constant times the optimal finish time. This is no longer the case for flow-shop and job-shop schedules.



$\mathrm{MFT}(S) = \sum_{i=1}^{2n} ik + n\epsilon_1 + n\epsilon_2$

a) SPT Schedule

$\mathrm{MFT}(S^*) = 2\sum_{i=1}^{n} ik + 2\sum_{i=1}^{n} i\epsilon_2 + n\epsilon_1$

b) OMFT Schedule

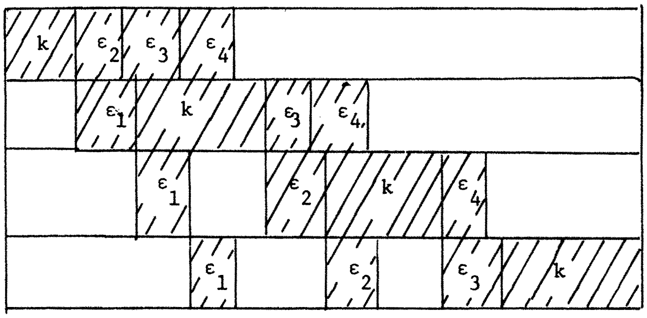
Figure 6

Example 3. Consider the m-processor, m-job flow shop with task times $t_{i,i}=k$, $1 \leqq i \leqq m$ and $t_{j,i}=\epsilon_i$, $1 \leqq i \leqq m$, $1 \leqq j \leqq m$ and $i \neq j$. $\epsilon_i > \epsilon_{i+1}$, $1 \leqq i < m$, and $\epsilon_1 \ll k$. $L_i = k+(m-1)\epsilon_i$ and so $L_i > L_{i+1}$, $1 \leqq i < m$. The LPT schedule, S, is the permutation schedule obtained by processing jobs in the order $(1, \cdots, n)$ on all processors. The finish time of S [Figure 7(a)] is $\mathrm{FT}(S) = mk + \sum_{i=1}^{m-1} \epsilon_i$. An optimal schedule, S^*, is shown in Figure 7(b). The finish time of S^* is $\mathrm{FT}(S^*) = \sum_{i=2}^{m} \epsilon_i + k + (m-1)\epsilon_1$. Hence, $\mathrm{FT}(S)/\mathrm{FT}(S^*) \cong m$ for $\epsilon_i \ll k$.

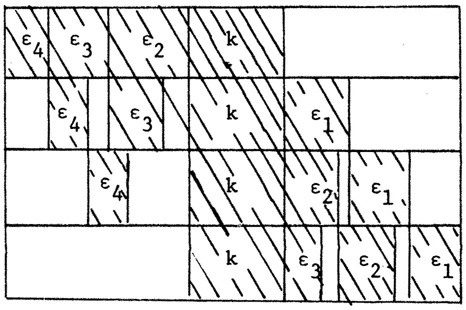
The worst case bound of m for busy schedules in a flow shop can be reduced to $\lceil m/2 \rceil$ by using the following heuristic H: Divide the m processors into $\lceil m/2 \rceil$ groups, each group containing at most two processors. The processors in group i are the $(2i-1)$st and $2i$th ones. Johnson's $O(n \log n)$ algorithm is used to obtain optimal finish time schedules for each of these $\lceil m/2 \rceil$ groups of machines. These $\lceil m/2 \rceil$ optimal schedules are then concatenated to obtain a schedule for the original m-processor flow shop. Since an optimal two-processor flow-shop schedule can be obtained in time $O(n \log n)$, the total time needed by algorithm H is $O(mn \log n)$.

LEMMA 11. *Let S be a schedule generated by algorithm H and let S^* be an OFT schedule. Then $FT(S)/FT(S^*) \leqq \lceil m/2 \rceil$.*

Proof. Let the length of each of the schedules $R(i)$, $1 \leqq i \leqq \lceil m/2 \rceil$ obtained in algorithm H be denoted by $f(R(i))$. Since each such schedule is optimal for its processor pair, it follows that $\mathrm{FT}(S^*) \geqq \max \{f(R(i))\}$.



(a) LPT permutation schedule



(b) Optimal schedule

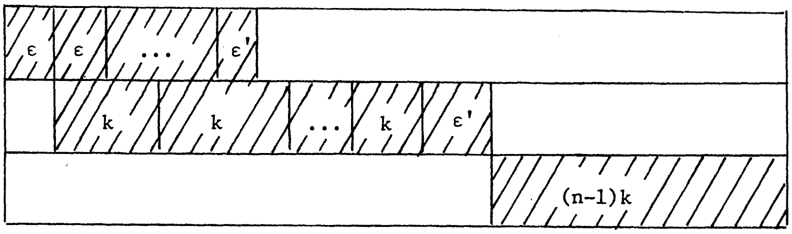
Figure 7. Schedules for Example 3 when $m=4$.

In the worst case the schedule S generated by algorithm H will have a finish time $\mathrm{FT}(S) \leqq \sum f(R(i)) \leqq \lceil m/2 \rceil \max \{f(R(i))\}$. Consequently, $\mathrm{FT}(S)/\mathrm{FT}(S^*) \leqq \lceil m/2 \rceil$.

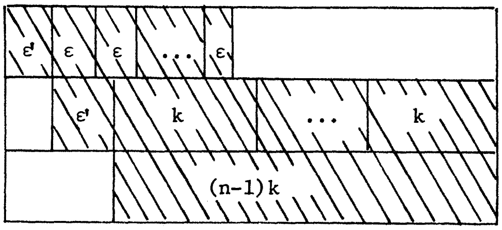
Example 4. Using algorithm H on the n-job flow-shop problem with $m=3$ and task times: $t_{1,i}=\epsilon$, $t_{2,i}=k$, $t_{3,i}=0$, $1 \leqq i < n$ and $t_{1,n}=\epsilon'$, $t_{2,n}=\epsilon'$, $t_{3,n}=(n-1)k$ $\epsilon < \epsilon' < k$ yields $R(1)=R(2)=1, \cdots, n$. This gives the schedule, S, of Figure 8(a). It has a finish time of $\epsilon+\epsilon'+2k(n-1)$. Figure 8(b) shows an optimal schedule S^*. $\mathrm{FT}(S^*)=2\epsilon'+k(n-1)$ and $\mathrm{FT}(S)/\mathrm{FT}(S^*) \simeq 2 = \lceil 3/2 \rceil$.

We close this section with a comparison of OFT preemptive and non-preemptive schedules. If S_p^* and S_n^* are OFT preemptive and non-preemptive schedules, respectively, then from the proof of Lemma 11 it follows that $\mathrm{FT}(S_n^*)/\mathrm{FT}(S_p^*) \leqq \lceil m/2 \rceil$. The next example shows that this is a tight bound when $m=3$ and when $m=4$.

Example 5. Consider the 3-processor flow shop with 3 jobs and task times: $t_{1,i}=k$, $t_{2,i}=\epsilon$, $t_{3,i}=k$, $i=1,2$, and $t_{1,3}=0$, $t_{2,3}=3k$, $t_{3,3}=0$. The



(a) Schedule S from algorithm H. FT(S) = ε + ε' + 2(n-1)k



(b) Optimal schedule S*. FT(S*) = 2ε' + (n-1)k

Figure 8

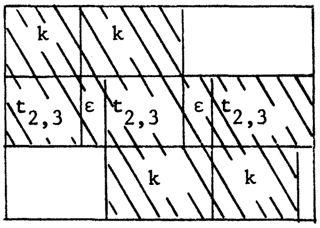
OFT preemptive schedule S_p^* has $\mathrm{FT}(S_p^*)=3k+2\epsilon$, while the OFT non-preemptive schedule S_n^* has $\mathrm{FT}(S_n^*)=5k+\epsilon$, $\mathrm{FT}(S_n^*)/\mathrm{FT}(S_p^*) \simeq 5/3$ for $\epsilon \ll k$ (see Figure 9).

Generalizing this to $n-1$ jobs with $t_{1,i}=k$, $t_{2,i}=\epsilon$, $t_{3,i}=k$, $1 \leqq i \leqq n-1$ and 1 job with $t_{1,n}=t_{3,n}=0$ and $t_{2,n}=nk$, we get $\mathrm{FT}(S_p^*)/\mathrm{FT}(S_n^*) \simeq 2-1/n$, which approximates $\lceil m/2 \rceil$ for large n.

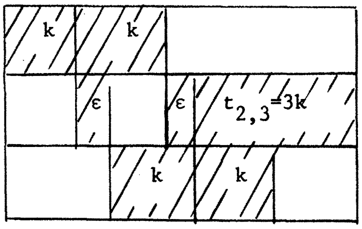
For the case of a job shop, we conclude from the proof of Lemma 10 that $\mathrm{FT}(S_n^*)/\mathrm{FT}(S_p^*) \leqq m$. The next example shows that this is a tight bound for $m=2$.

Example 6. Consider the two-processor job-shop problem with two jobs and task times: $t_{1,1,2i-1}=k$, $1 \leqq i \leqq r$; $t_{2,1,2i}=\epsilon$, $1 \leqq i < r$ and $t_{2,2,1}=rk$. An

$S_n{}^*$ and $S_p{}^*$ schedule for this set of tasks is given in Figure 10. We obtain $\mathrm{FT}(S_n{}^*)/\mathrm{FT}(S_p{}^*) = ((2r-1)k+(r-1)\epsilon)/(rk+(r-1)\epsilon)$. Hence, $\lim_{\epsilon\to 0} \mathrm{FT}(S_n{}^*)/\mathrm{FT}(S_p{}^*) = 2-1/r$, which approaches 2 as $r\to\infty$.

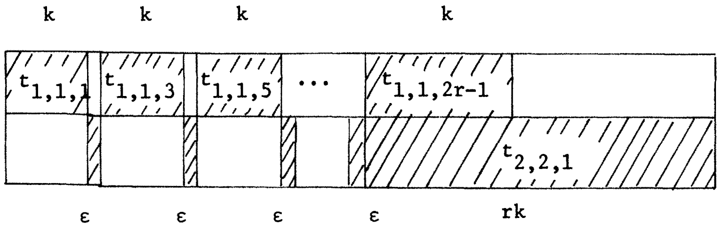


(a) $f(S_p^*) = 3k + 2\varepsilon$

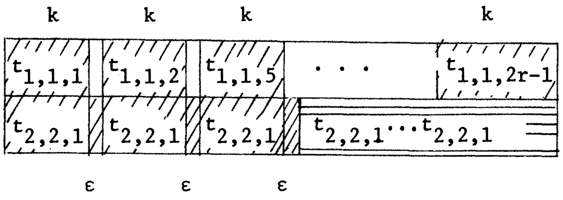


(b) S_n^*

Figure 9



(a) $\mathrm{FT}(S_n^*) = (2r-1)k + (r-1)\varepsilon$



(b) $\mathrm{FT}(S_p^*) = rk + (r-1)\varepsilon$

Figure 10

4. CONCLUSIONS

We have investigated the properties of OFT preemptive and non-preemptive schedules for flow shops and job shops. Finding such schedules is

NP-complete. Bounds for relative performance of various heuristics to obtain schedules with either a good finish time or good *mft* have been derived. These bounds are tight when the number of processors is small (i.e., $3 \leqq m \leqq 4$ for the flow shop).

ACKNOWLEDGMENTS

This research was supported in part by NSF grants DCR 74-10081, MCS 76-21024 and University of Minnesota Graduate School Research Grant no. 468-0100-4909-08.

REFERENCES

1. P. Brucker, J. K. Lenstra, and A. H. G. Rinnooy Kan, "Complexity of Machine Scheduling Problems," Mathematisch Centrum, Amsterdam, Technical Report BW 43/75, April 1975.
2. E. G. Coffman, Jr., *Computer and Job Shop Scheduling Theory*, John Wiley & Sons, New York, 1976.
3. R. W. Conway, W. L. Maxwell, and L. W. Miller, *Theory of Scheduling*, Addison-Wesley, Reading, Mass., 1967.
4. M. R. Garey, D. S. Johnson, and R. Sethi, "Complexity of Flow Shop and Job Shop Scheduling," *Math. Opns. Res.* **1,** 117–129 (1976).
5. T. Gonzalez, O. H. Ibarra, and S. Sahni, "Bounds for LPT Schedules on Uniform Processors," *SIAM J. Computing,* **5,** 155–166 (1977).
6. T. Gonzalez and S. Sahni, "Open Shop Scheduling to Minimize Finish Time," *J. Assoc. Comput. Machinery,* **23,** 665–679 (1976).
7. R. L. Graham, "Bounds on Multiprocessing Timing Anomalies," *SIAM J. Appl. Math.* **17,** 416–429 (1969).
8. J. R. Jackson, "An Extension of Johnson's Results on Job Lot Scheduling," *Naval Res. Logist. Quart.* **3,** 201–203 (1956).
9. R. M. Karp, "Reducibility among Combinatorial Problems," in *Complexity of Computer Computation*, pp. 85–104, R. E. Miller and J. W. Thatcher (Eds.), Plenum Press, New York, 1972.
10. R. M. Karp, "On the Computational Complexity of Combinatorial Problems," *Networks* **5,** 45–68 (1975).
11. S. Sahni, "Computationally Related Problems," *SIAM J. Computing* **3,** 262–279 (1974).