Preemptive Scheduling of Independent Jobs with Release and Due Times on Open, Flow and Job Shops

YOOKUN CHO
University of Minnesota, Minneapolis, Minnesota

SARTAJ SAHNI
University of Minnesota, Minneapolis, Minnesota

(Received November 1978; accepted October 1980)

We study the problem of obtaining feasible preemptive schedules for independent jobs. It is assumed that each job has associated with it a release and due time. No job can begin before its release time. All jobs must be completed by their respective due times. It is shown that determining the existence of feasible preemptive schedules for two processor flow and job shops is *NP*-hard in the strong sense even when all jobs have the same due time. A linear programming formulation for the open shop problem is obtained. Also, a fast polynomial time algorithm is obtained for a restricted class of open shop problems.

IN THIS PAPER we study the problem of preemptively scheduling n independent jobs, with release and due times, on m processor flow shops, job shops and open shops. We are concerned with determining the computational difficulty of deciding whether or not all the jobs can be scheduled to finish by their respective due times (of course, no job can be processed before its release time). This problem arises in many real life situations. For example, in a production shop we may have a list of "open" (i.e. as yet uncompleted) jobs. Each job has several remaining tasks and a delivery date. We would like to know if all the jobs on hand can be completed by the promised delivery dates.

We shall show that for the case of flow shops (and hence also job shops), determining whether or not all jobs can be completed by their due times is *NP*-hard. Hence, in all likelihood, there is no efficient algorithm for this problem. For the case of open shops, a linear programming formulation is obtained. A more efficient algorithm is obtained for the case when there are only two distinct release times and all jobs have the same due time.

A *shop* is an ordered set $\{P_1, P_2, \cdots, P_m\}$ of $m \geq 1$ processors (or machines). n jobs are to be scheduled on these processors. In the case of

Operations Research
Vol. 29, No. 3, May–June 1981
0030-364X/81/2903-0511 $01.25
© 1981 Operations Research Society of America

flow shops and *open shops*, each job has m tasks associated with it. The task time for task j of job i is denoted by $t_{i,j}$. Task j is to be processed on processor P_j, $1 \leq j \leq m$. In the case of an open shop, tasks may be processed in any order. In the case of a flow shop, task j of job i cannot start until task $j-1$ of that job has completed. In the case of a *job shop* there is a processor $p(i, j)$ associated with each task and job. The jth task of job i is to be processed on $P_{p(i,j)}$. Task j cannot start until task $j-1$ has completed. A job may have any number of tasks.

In addition to task times, job i also has associated with it a release time (or release date) $R(i)$ and a due time (or due date) $D(i)$. The processing of any task of job i cannot begin before time $R(i)$. All tasks of job i must finish by $D(i)$. A *feasible schedule* is an assignment of tasks to processors such that the processing of every job finishes by its due time and no job begins processing before its release time. In addition, no job can be simultaneously processed on more than one processor. In a nonpreemptive schedule, each task once started, continues to process until it finishes. In a preemptive schedule the processing of some tasks may be interrupted and later continued. In the following, we shall use the term schedule to mean feasible schedule.

We shall use R_i, $1 \leq i \leq r$ and D_i, $1 \leq i \leq d$ to respectively denote the distinct release and due times present in the sets $R(i)$, $1 \leq i \leq n$ and $D(i)$, $1 \leq i \leq n$. The problem of determining feasible schedules when $r = d = 1$ has been solved earlier. For the case of flow shops, Johnson (1954) has an $0(n \log n)$ algorithm that can be used when $m = 2$. His algorithm works for both preemptive and nonpreemptive schedules. When $m > 2$, it is known that determining the existence of feasible schedules (either preemptive or nonpreemptive) is *NP*-hard in the strong sense (see Garey et al. [1976], Gonzalez and Sahni [1978]). The term *NP*-hard in the strong sense is defined by Garey and Johnson (1978). Informally, a problem L is *NP*-hard in the strong sense if it remains *NP*-hard even when restricted to problem instances in which the magnitude of all numbers is bounded by a fixed polynomial in the length of the input. The distinction between an ordinary *NP*-hard problem and one which is *NP*-hard in the strong sense is important because some *NP*-hard problems can be solved by a pseudo-polynomial time algorithm (i.e. an algorithm whose complexity is polynomial in the length of the input and magnitude of the numbers involved). However, a problem which is *NP*-hard in the strong sense cannot be solved by a pseudo-polynomial time algorithm unless $P = NP$ (see Garey and Johnson [1978]). In the context of the scheduling problems discussed here, a problem which is *NP*-hard in the strong sense cannot be solved in time $p(n, l)$ where p is a fixed polynomial, n is the number of jobs and l is the length of an optimal schedule unless $P = NP$.

For the case of job shops, Gonzalez and Sahni (1978) have shown that obtaining preemptive or nonpreemptive feasible schedules is *NP*-hard in the strong sense for any fixed m, $m > 1$, $r = d = 1$. The preemptive scheduling problem for open shops can be solved in polynomial time when $r = d = 1$ (see Gonzalez and Sahni [1976]). The nonpreemptive scheduling problem can be solved in $0(n)$ time when $m = 2$, $r = 1$ and $d = 1$, and is *NP*-hard when $m > 2$ (see Gonzalez and Sahni [1976]). It is *NP*-hard in the strong-sense when $m = 2$ and many release times are allowed. It is also known that when $m = 2$, $r = 2$ and $d = 1$ then the nonpreemptive scheduling problem is *NP*-hard for flow and job shops (see Lenstra et al. [1977]).

In this paper we extend the results stated above. First, we show that the preemptive scheduling problem for flow shops (and hence also job shops) is *NP*-hard in the strong sense when $m = 2$, $d = 1$ and many release times exist. We also show that this problem remains *NP*-hard for flow shops when $m = 2$, $r = 2$ and $d = 1$.

Next, we turn our attention to open shops. It was shown by Graham et al. (1977) that the nonpreemptive scheduling problem for open shops is *NP*-hard when $m = 2$, $r = 2$ and $d = 1$. For the preemptive case, we obtain a linear programming formulation for the case $m \geq 2$, $r \geq 1$ and $d \geq 1$. This formulation is similar to that obtained by Lawler and Labetoulle (1978) for preemptive scheduling of unrelated processors. The solution to the linear program can be used in conjunction with the polynomial time preemptive scheduling algorithm of Gonzalez (1976), and Gonzalez and Sahni (1976) to obtain a preemptive schedule. In case there is no preemptive schedule for a given problem instance then the corresponding linear program is infeasible. Since in the worst case even the best linear programming algorithms are impractical (experience indicates that these algorithms are usually very practical), we study special cases of the open shop problem for which fast algorithms can be obtained. Feasible preemptive open shop schedules can be found in polynomial time when either $m = 2$, $r > 1$ and $d = 1$ or $m > 2$, $r = 2$ and $d = 1$. In Cho and Sahni (1978) we have developed the algorithm for the case $m = 2$, $r > 1$ and $d = 1$. Subsequently, Lawler et al. (unpublished) have obtained a simpler and more efficient (i.e., $0(n)$) algorithm. In this paper, we present our polynomial time algorithm for the case $m > 2$, $r = 2$ and $d = 1$.

Finally, we look at a class of restricted preemptive schedules for open shops. In this restriction one is not permitted to schedule a new task for any job j unless all previously scheduled tasks of this job have been completed. It is shown that obtaining feasible schedules satisfying this restriction is *NP*-hard even when $m = 3$, $r = d = 1$. This should be contrasted with the polynomial time algorithm of Gonzalez and Sahni (1976) for the case when this restriction is not imposed.

In order to show our problems *NP*-hard, we make use of the following known *NP*-hard problem (Karp [1972]):

PARTITION. *Given a multi set of n positive integers, a_i, $1 \leq i \leq n$ and $\sum_{i=1}^{n} a_i = 2T$ determine if there exists a subset I such that $\sum_{i \in I}^{n} a_i = T$.*

For the strong *NP*-hard hardness result the following problem which is known to be *NP*-hard in the strong sense (Garey et al.) is used:

3-PARTITION. *Given a positive integer B and a multiset A of positive integers $A = \{a_1, \cdots, a_p\}$ with $p = 3n$, $\sum_{i=1}^{p} a_i = nB$ and $B/4 < a_i < B/2$ for $1 \leq i \leq p$, 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 $i = 1, \cdots, n$?*

The reader unfamiliar with *NP*-hard problems is urged to read Karp to determine how one shows new problems to be *NP*-hard. A good discussion of the implications of a problem being *NP*-hard in the strong sense can be found in the paper by Garey and Johnson. Graham et al. contains a good survey of recent results in scheduling theory.

1. FLOW SHOPS AND JOB SHOPS

THEOREM 1. *For flow shops, the preemptive scheduling problem is NP-hard in the strong sense when $m = 2$, $d = 1$ and an arbitrary number of release times are permitted.*

Proof. Let $A = \{a_1, a_2, \cdots, a_p\}$, $p = 3n$ and B define an instance of the 3-partition problem. We may assume $\sum_{i=1}^{p} a_i = nB$. From this instance construct the following two processor $m + n + 2$ job flow shop instance:

$$t_{i,1} = 2a_i, \quad t_{i,2} = a_i, \quad 1 \leq i \leq m; \quad \text{release time is } R_1 = 0$$

$$t_{m+i,1} = B, \quad t_{m+i,2} = 2B, \quad \text{release time is } 3(i-1)B, \quad 1 \leq i \leq n$$

$$t_{m+n+1,1} = 0, \quad t_{m+n+1,2} = B, \quad \text{release time is } R_1 = 0$$

$$t_{m+n+2,1} = B, \quad t_{m+n+2,2} = 0, \quad \text{release time is } 3nB.$$

The common due time is $(3n + 1)B$. Note that since $\sum_{i=1}^{m+n+2} t_{i,1} = \sum_{i=1}^{m+n+2} t_{i,2} = (3n + 1)B$, there can be no idle time on either P_1 or P_2 in any feasible schedule for the above flow shop instance. We shall show that there is a preemptive schedule for the constructed flow shop instance if and only if (iff) there is a 3-partition for A. If there is a 3-partition then we may construct a preemptive schedule as in Figure 1.

Now, suppose there is a feasible schedule S. Job $m + n + 2$ must be scheduled in S as in Figure 1. Now suppose task 1 of job $m + n$ does not finish until $3(n - 1)B + B + x$, $x > 0$. Then task 2 cannot start until this time. Hence, in the interval $[3(n - 1)B + B + x, (3n + 1)B]$ only $B - x$

units will be free on P_2 to schedule other jobs. The free time on P_1 in this same interval is $2B - x$. No job with index j, $m < j \leq m + n$ can be started in this interval on P_1 as there is not enough free time on P_2 in this interval to process $t_{j,2} = 2B > B - x$. Therefore, only jobs with indices j, $j \leq m$ may be started in the free time in the interval $[3(n-1)B + B + x, 3(n+1)B]$ of P_1. The sum of their P_2 processing requirements is at least $(2B - x)/2$. No portion of this can be done before $3(n-1)B + B + x$. But there is only $B - x < B - x/2$ free time on P_2 after $3(n-1)B + B + x$. Hence, we must have $x = 0$ and only tasks with indices j, $j \leq m$ can be scheduled in the free time in the interval $[3(n-1)B + B, (3n+1)B]$. The preemptions of task 2 of job $m + n$ are easily eliminated (i.e., just slide the preempted pieces together by leftward shifts; this will move jobs scheduled in between preempted pieces of job $m + n$ to the right), and we can assume job $m + n$ is scheduled as in Figure 1. The free time on P_2 from $3nB$ to $(3n+1)B$ is reserved for the jobs scheduled on P_1 from $3(n-1)B + B$ to $3nB$.

By repeatedly using the above argument we see that there is no feasible

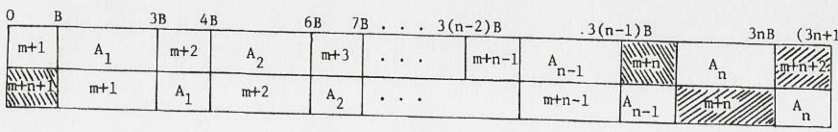

Figure 1

schedule for the $m + n + 2$ jobs unless there is one in which jobs j, $m < j \leq m + n$ and $j = m + n + 2$ are scheduled as in Figure 1. It is now clear that job $m + n + 1$ must also be scheduled as in Figure 1. Now consider the free slots left to right. The only way to fill up the slot on P_2 corresponding to A_1 is by scheduling a job set A_1 on P_1 in $[B, 3B]$ with $2\sum_{a \in A_1} a = 2B$ or $\sum_{a \in A_1} a = B$. This is also true for each of the slots $A_2, A_3, \cdots, A_n$. Hence, the existence of a feasible schedule S implies that there is a 3-partition of the multiset A.

Thus, there exists a feasible schedule for the $m + n + 2$ jobs constructed above iff there is a 3-partition of A.

We know that obtaining preemptive schedules for job shops is *NP*-hard in the strong sense when $m = 2$, $r = 1$ and $d = 1$ (see Gonzalez and Sahni [1978]). For the flow shop problem however, the problem is solvable in $0(n \log n)$ time when $m = 2$, $r = 1$ and $d = 1$. This leaves us with the question: Is the problem of obtaining preemptive schedules for flow shops *NP*-hard for any fixed r? This question is answered in the affirmative by the following theorem.

THEOREM 2. *Determining the existence of preemptive feasible schedules*

for a flow shop with $m = 2$ is NP-hard even when the problem instances are restricted to $r = 2$ and $d = 1$.

Proof. The proof of this theorem makes use of the partition problem. Let $\{a_1, a_2, \cdots, a_n\}$ be any instance of the partition problem.

Assume $\sum_{i=1}^{n} a_i = 2T$. Construct the following $n + 3$ job flow shop instance *FS*:

$$t_{i,1} = 2a_i, \quad t_{i,2} = a_i, \quad 1 \leq i \leq n$$
$$t_{n+1,1} = 0, \quad t_{n+1,2} = 2T$$
$$t_{n+2,1} = T, \quad t_{n+2,2} = 0$$
$$t_{n+3,1} = T, \quad t_{n+3,2} = 2T.$$

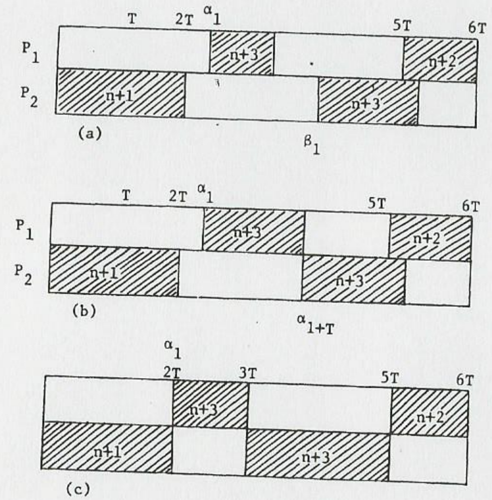

Figure 2

Jobs $1, 2, \cdots, n + 2$ have a release time $R_1 = 0$ while job $n + 3$ has a release time $R_2 = 2T$. The due time for all jobs is $D = 6T$.

Note that since $\sum_{i=1}^{n+3} t_{i,1} = \sum_{i=1}^{n+3} t_{i,2} = 6T$, neither P_1 nor P_2 can have any idle time in any feasible schedule for *FS*.

It is easy to see that if the a_i's have a partition then there exists a schedule for the $n + 3$ jobs (in fact there is a nonpreemptive schedule). We shall now show that if there exists a preemptive schedule then the a_i's have a partition.

Suppose that there exists a preemptive schedule S. It should be easy to see that all preemptions of jobs $n + 1$ and $n + 2$ may be removed from S and there exists a schedule S' in which job $n + 1$ is scheduled on P_2 from 0 to T while job $n + 2$ is scheduled on P_1 from $5T$ to $6T$. Further, all preemptions of job $n + 3$ may also be removed without affecting the feasibility of S'. Hence, if there exists a preemptive schedule for FS then there must exist one in which jobs $n + 1$, $n + 2$ and $n + 3$ are scheduled without preemption and as in Figure 2(a). At the risk of increasing the number of preemptions by 1, the situation of Figure 2(a) can be transformed into that of Figure 2(b). Let α_1 be the start time for job $n + 3$. Let Q be the set of jobs whose P_1 tasks have been completed in the interval $[0, \alpha_1]$ on P_1. Note that $Q \subset \{1, 2, \cdots, n\}$. Only the P_2 tasks of jobs in Q can be processed in the period $[2T, \alpha_1 + T\}$ on P_2. To avoid idle time on P_2 in this interval, we must have $\alpha_1/2 \geq \alpha_1 - T$ (note that $\sum_{j\in Q} t_{j,2} = ½ \sum_{j\in Q} t_{j,1}$ and that the length of the interval $[2T, \alpha_1 + T]$ is $\alpha_1 - T$). Also, we must have $\alpha_1 \geq 2T$ as job $n + 3$ has a release time of $2T$. Combining these two inequalities, we get $\alpha_1 = 2T$ and $\sum_{j\in Q} t_{j,1} = 2T$. Hence for a feasible schedule as in Figure 2(b) to exist the a_i's must have a partition. The schedule takes the form given in Figure 2(c). Hence, FS has a preemptive schedule iff the a_i's have a partition.

2. OPEN SHOPS

The problem of obtaining feasible preemptive schedules for open shops appears to be simpler than that for flow shops and job shops. An algorithm that can be exected to perform well is easily obtained by formulating the open shop scheduling problem as a linear programming problem. Let $a_1 < a_2 < \cdots < a_{p+1}$ be the ordered collection of all distinct values of R_i, $1 \leq i \leq r$ and D_i, $1 \leq i \leq d$. Let $R(j)$ and $D(j)$ respectively be the release and due times of job j. By $x_{i,j,k}$ we shall denote the amount of task j of job i that is to be processed in the interval $[a_k, a_{k+1}]$. Let $I_k = a_{k-1} - a_k$. Now consider the following linear program:

$$\begin{aligned}
&\textstyle\sum_{j=1}^{m} x_{i,j,k} \leq I_k, && 1 \leq i \leq n, \quad 1 \leq k \leq p\\
&\textstyle\sum_{i=1}^{n} x_{i,j,k} \leq I_k, && 1 \leq j \leq m, \quad 1 \leq k \leq p\\
&\textstyle\sum_{k=1}^{p} x_{i,j,k} = t_{i,j}, && 1 \leq j \leq m, \quad 1 \leq i \leq n \qquad (1)\\
&x_{i,j,k} \geq 0 \quad \text{if} \quad R(i) \leq a_k \quad \text{and} \quad D(i) \geq a_{k+1}\\
&x_{i,j,k} = 0 \quad \text{if} \quad R(i) > a_k \quad \text{or} \quad D(i) < a_k.
\end{aligned}$$

The first inequality requires that no job be scheduled for more than I_k time units in any interval. The second requires that the amount of processing assigned to any processor be no more than the interval length. The third equality requires that each job be finished. The constraints on $x_{i,j,k}$ ensure that no job is assigned to a processor either before its release time or after its due time.

LEMMA 1. *Let $t_{i,j}$, $1 \leq i \leq n$, $1 \leq i \leq m$ define an instance of the open shop problem with n jobs and m processors. Assume that all jobs are released at time 0. The minimum finish time, F, of any preemptive schedule for these n jobs is given by*

$$F = \max_{i,l}\{\textstyle\sum_{j=1}^{m} t_{i,j}, \sum_{k=1}^{n} t_{k,l}\}.$$

Gonzalez and Sahni (1976) present an $0(r(\min\{r, m^2\} + m \log n))$ algorithm to obtain a preemptive schedule with finish time F as defined in Lemma 1 (r is the number of nonzero tasks). Gonzalez has improved this algorithm to one with complexity $0(r + \min\{m^4, n^4, r^2\}$.

It is easy to see that for any feasible solution to (1), $\max_{i,l}\{\sum_{j=1}^{m} x_{i,j,k}, \sum_{s=1}^{n} x_{s,i,k}\} \leq I_k$. Hence, the scheduling assignments $x_{i,j,k}$ can be met in each of the intervals I_k. So, from a feasible solution to (1) a feasible schedule can be constructed using the algorithm of Gonzalez p times. Conversely, if a feasible schedule exists then (1) has a feasible solution.

A well known rule of thumb (Gass [1969]) is that the number of Simplex iterations needed to find a feasible solution to a linear program is "about" equal to the number of constraints. In this case there are $mn + mp + np$ constraints. So, "usually" $mn + mp + np$ iterations of the Simplex method are needed to find a feasible solution to (1). Note that in the worst case the number of iterations needed may be exponential in the number of equations.

Khachian (1979) has developed a polynomial time algorithm to solve linear programs. However, this algorithm is quite impractical and may be expected to out-perform the Simplex method only on those instances where neither algorithm remains feasible. We are thus motivated to search for a low order polynomial complexity algorithm for open shop scheduling.

3. OPEN SHOP PROBLEMS WITH $m > 2$, $r = 2$ AND $d = 1$

As remarked in the previous section, the *LP* formulation does not lead to a computationally feasible algorithm for open shops. In this section we consider the special case when $m > 2$, $r = 2$ and $d = 1$.

A polynomial time algorithm for this case may be obtained by transforming each instance into a network flow problem in which there is an upper and lower bound associated with each edge. Let u_i and l_i be respectively the upper and lower bounds on the flow through edge i. A flow is said to be a *feasible flow* in a network with upper and lower bounds iff the flow through each edge i is at least l_i and at most u_i. Each edge is a directed edge.

Let I be any instance of the open shop problem with $m > 2$, $r = 2$ and $d = 1$. We may assume $R_1 = 0$, $R_2 > 0$ and D (the due time) is greater than R. Let n_1 and n_2 respectively be the number of jobs released at R_1 and R_2. Let $t_{i,j}$, $1 \leq i \leq n_1$, $1 \leq j \leq m$ and $\tau_{i,j}$, $1 \leq i \leq n_2$, $1 \leq j \leq m$ respectively be the task times of the jobs released at R_1 and R_2.

Without loss of generality, we may assume that $\sum_{i=1}^{n_2} \tau_{ij} \le D - R_2$, $1 \le j \le m$ and $\sum_{j=1}^{m} \tau_{ij} \le D - R_2$, $1 \le i \le n_2$ (as otherwise by Lemma 1 there can be no feasible schedule). Define $T_j = \sum_{i=1}^{n_1} t_{i,j}$, $1 \le j \le m$ and $L_i = \sum_{j=1}^{m} t_{i,j}$, $1 \le i \le n_1$. The corresponding network will consist of $n_1 + m + 2$ vertices. Two of these are the source (s) and sink (t) vertices. The remaining $n_1 + m$ vertices are labeled J_i, $1 \le i \le n_1$ and P_j, $1 \le j \le m$. These are drawn in two columns (Figure 3). The edges and their upper and lower bounds are as below:

Edge	*Lower Bound*	*Upper Bound*
$\langle s, J_i \rangle$	$\max\{L_i - R_2, 0\}$	$\min\{L_i, D - R_2\}, 1 \le i \le n_1$
$\langle J_i, P_j \rangle$	0	$t_{i,j}, 1 \le i \le n_1, 1 \le j \le m$
$\langle P_j, t \rangle$	$\max\{T_j - R_2, 0\}$	$\min\{T_j, D - R_2 - \sum_{i=1}^{n_2} \tau_{i,j}\}, 1 \le j \le m.$

The interpretation of a feasible flow is that if the flow in the edge

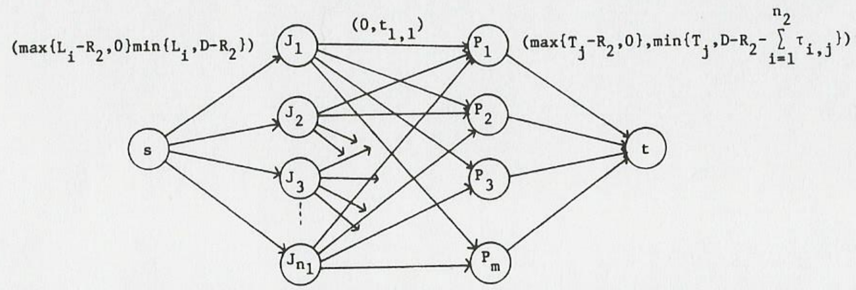

Figure 3. Flow network corresponding to open shop problem.

$\langle J_i, P_j \rangle$ is $f_{i,j}$ then $f_{i,j}$ units of task j of job i will be scheduled in $[R_2, D]$ and $t_{ij} - f_{ij}$ units will be scheduled in $[0, R_2]$. With this interpretation, it is not too difficult to see that there is a preemptive schedule for the $n_1 + n_2$ jobs iff there is a feasible flow for the network of Figure 3. If there is a feasible flow then the lower bound for edge $\langle s, J_i \rangle$ ensures that at most R_2 units of job i will have to be processed in $[0, R_2]$. The upper bound ensures that no more than $D - R_2$ units are deferred to $[R_2, D]$. The lower bound on $\langle P_j, t \rangle$ ensures that the amount to be scheduled on P_j in $[0, R_2]$ is at most R_2. The upper bound ensures that the amount of P_j processing deferred to $[R_2, D]$ is no more than the available time on P_j in this interval. The bounds on edge $\langle J_i, P_j \rangle$ ensure that the amount of task j of job i deferred to $[R_2, D]$ does not exceed the task length t_{ij}. So, if there is a feasible flow then Lemma 1 guarantees that the processing times assigned for the two intervals $[0, R_2]$ and $[R_2, D]$ can be scheduled. The algorithm of Gonzalez or Gonzalez and Sahni (1976) may be used for this purpose. On the other hand, if there is a feasible schedule then there is clearly a feasible flow for the network of Figure 3.

A feasible flow (if one exists) in a network with lower and upper bounds may be obtained using the construction of Even (1973). He shows how to transform a network N with lower bounds into another network $\bar{N}$ without lower bounds. From the maximum flow in $\bar{N}$ one can easily determine a feasible flow for N. If N has v vertices and e edges then $\bar{N}$ has $v + 2$ vertices and $e + 2v$ edges. A maximum flow in an E edge V vertex network with no lower bounds can be found in time $0(V^3)$ using the algorithm of Karzanov (1974). Since, for N, $v = n_1 + m + 2$, V for $\bar{N}$ is $n_1 + m + 4$. The time to determine a feasible flow (if any) in N is therefore $0((n_1 + m + 4)^3) = 0(n^3 + m^3)$ (note $n_1 < n$). The algorithm of Gonzalez takes $0(nm + \min\{n^4, m^4\})$ time to construct a schedule for each interval. So, the total time needed to obtain a schedule is $0(n^3 + m^3 + nm + \min\{n^4, m^4\})$. When $n \ge m$ this becomes $0(n^3 + m^4)$.

4. OPEN SHOP SCHEDULING WITH NO PASSING

A close examination of the algorithms of Gonzalez and Gonzalez and Sahni (1976) reveals that the preemptive schedules constructed by these algorithms (and hence by our algorithm of Section 3) have a property that may be undesirable in certain applications. It is quite possible that in a preemptive schedule constructed by the algorithms cited above (for $m > 2$) task j of job i gets preempted and before this task is resumed, another task of the same job may be processed. Thus it is possible to start processing a task for some job which has a started but unfinished other task. A schedule in which no jobs are scheduled in the manner just described is called a *schedule with no passing*. It is not too difficult to show that obtaining feasible schedules with no passing is NP-hard for every fixed m, $m > 2$, $r = 1$ and $d = 1$. To see this, let a_i, $1 \le i \le n$ be an instance of the partition problem. Let $T = (\sum a_i)/2$. Define the following open shop instance with $n + 3$ jobs:

$$m = 3, \quad R_1 = 0, \quad D = 6T$$
$$t_{1,1} = 3T, \quad t_{1,2} = 0, \quad t_{1,3} = 3T$$
$$t_{2,1} = T, \quad t_{2,2} = 4T, \quad t_{2,3} = T$$
$$t_{3,1} = 0, \quad t_{3,2} = 0, \quad t_{3,3} = 2T$$
$$t_{i,1} = a_i, \quad t_{i,2} = a_i, \quad t_{i,3} = 0, \quad 4 \le i \le n + 3.$$

Since the sum of the task times for each processor is $6T$ there can be no idle time on any processor in any feasible schedule. Figure 4 shows the only two ways to schedule jobs 1, 2 and 3 without passing and not exceeding the due time of $D = 6T$. If there is a partition of the a_i's then all jobs corresponding to the partition may be scheduled in $I1$ and the remainder in $I2$. Since at the start of $I3$ all tasks scheduled in $I1$ have completed, all jobs i, $4 \le i \le n + 3$ may be scheduled in $I3$ and a feasible schedule with no passing obtained.

If there is no partition then since there can be no idle time on any processor, there is at least one job for which the P_2 task is only partially processed by the end of the $I1$ interval on P_2. The P_1 task corresponding to this job cannot be scheduled on P_1 in $I3$ and so there will be idle time in this interval. So, there is no feasible schedule with no passing in this case.

5. SUMMARY

We have studied the problem of preemptively scheduling shops with due dates and release times. For the case of flow shops (and hence also job shops) we have shown that the preemptive scheduling problem is NP-hard in the strong sense when $m = 2$, $d = 1$ and an arbitrary number of release times are permitted. When the number of release times is restricted to be two, we have only been able to show the problem NP-hard. This leaves open the existence of a pseudo-polynomial time algorithm for the case where only a fixed number of distinct release times exist.

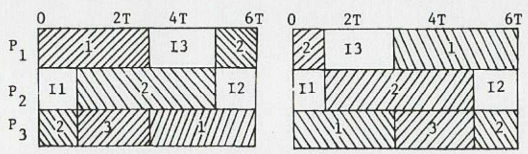

Figure 4. Two alternative schedules.

For the case of open shops, we have been unable to either show the scheduling problem NP-hard or obtain a polynomial time algorithm. An NP-hardness proof may be quite difficult to obtain. Two special cases of the open shop problem are, however, solvable in polynomial time. The discovery of additional interesting polynomially solvable classes of the open shop problem will be of interest.

The preemptive scheduling of open shops with no passing is NP-hard when $m > 2$, $r = 1$ and $d = 1$. This should be contrasted with the polynomial time algorithm of Gonzalez and Sahni (1976) when passing is allowed and $m \geq 2$, $r = 1$ and $d = 1$. When $m = 2$, $r = 1$ and $d = 1$ the algorithm of Gonzalez and Sahni (1976) generates feasible schedules with no passing.

The reader is referred to Graham et al. for the status of problems related to those considered in this paper.

ACKNOWLEDGMENT

This research was supported in part by NSF grants MCS76-21024 and MCS78-15455. We are grateful to an anonymous referee who suggested

a simplification to our original proof of Theorem 2 given in Cho and Sahni (1978).

REFERENCES

CHO, Y., AND S. SAHNI. 1978. Preemptive Scheduling of Independent Jobs with Release and Due Times on Open, Flow and Job Shops, Technical Report No. 78-5, University of Minnesota.

EVEN, S. 1973. *Algorithmic Combinatiorics,* Chapter 10. Macmillan, New York.

GAREY, M. R., D. S. JOHNSON AND R. SETHI. 1976. The Complexity of Flow Shop and Job Shop Scheduling. *Math. Opns. Res.* **22**, 117–129.

GAREY, M. R., AND D. S. JOHNSON. 1978. Strong *NP*-Completeness Results: Motivation, Examples and Implications. *J. Assoc. Comput. Mach.* **25**, 499–508.

GASS, S. 1969. *Linear Programming.* McGraw-Hill, New York.

GONZALEZ, T. 1976. A Note on Open Shop Preemptive Schedules, TR No. 214, Computer Science Dept., Pennsylvania State University.

GONZALEZ, T., AND S. SAHNI. 1976. Open Shop Scheduling to Minimize Finish Time. *J. Assoc. Comput. Mach.* **23**, 665–679.

GONZALEZ, T., AND S. SAHNI. 1978. Flow Shop and Job Shop Schedules: Complexity and Approximation. *Opns. Res.* **26**, 36–52.

GRAHAM, R. L., E. L. LAWLER, J. K. LENSTRA AND A. H. G. RINNOOY KAN. 1977. Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey, Stiching Mathematisch Centrum.

JOHNSON, S. M. 1954. Optimal Two-and-Three-Stage Production Schedules with Setup Times Included. *Naval Res. Logist. Quart.* **1**, 61–68.

KARP, R. M. 1972. Reducibility among Combinatorial Problems. In *Complexity of Computer Computations*, pp. 85–104, R. E. Miller and J. W. Thatcher (eds.). Plenum Press, New York.

KARZANOV, A. 1974. Determining the Maximal Flow in a Network by the Method of Preflows. *Sov. Math. Dokl.* **15**, 436–437.

LAWLER, E. L., AND J. LABETOULLE. 1978. On Preemptive Scheduling of Unrelated Parallel Processors by Linear Programming. *J. Assoc. Comput. Mach.* **25**, 612–619.

LAWLER, E. L., J. K. LENSTRA AND A. RINNOOY KAN. 1981. Minimizing Maximum Lateness in a Two-Machine Open Shop. *Math. Opns. Res.* **6**, 153–158.

LENSTRA, J. K., A. H. G. RINNOOY KAN AND P. BRUCKER. 1977. Computational Complexity of Machine Scheduling Problems. *Ann. Discrete Math.* **1**, 343–362.

KHACHIAN, L. 1979. A Polynomial Algorithm in Linear Programming. *Dokl. Akad. Nauk. SSSR* **224**, 1093–1096.