MATHEMATICS OF OPERATIONS RESEARCH
Vol. 7, No. 1, February 1982
Printed in U.S.A.

UNIT EXECUTION TIME SHOP PROBLEMS*†

TEOFILO GONZALEZ‡

The Pennsylvania State University

The problem of preemptively (and nonpreemptively) scheduling a set of n independent jobs on an m machine open shop, flow shop or job shop is studied. It is shown that the problem of constructing optimal mean finishing time preemptive and nonpreemptive schedules is NP-hard. These problems are not only NP-hard in the strong sense, but remain NP-hard even when all nonzero tasks have identical execution time requirements. These results will also apply to the case when the problem is to construct an optimal finish time preemptive and nonpreemptive schedule for a flow shop or a job shop. We also discuss the problem of constructing no-wait schedules for these problems.

I. Introduction. There are $n \geqslant 1$ independent jobs $(J_1, \ldots, J_n)$ to be processed by an m machine/processor $(P_1, \ldots, P_m)$ shop. Each job, J_i, consists of l_i tasks. The jth task of job J_i is referred to by $\tau_{i,j}$ and it is to be processed by machine $P_{q_{i,j}}$ for $t_{i,j}$ time units. For open shop and flow shop problems, all jobs consist of m tasks; and all tasks have $q_{i,j} = j$. The order in which tasks are executed in a flow shop and a job shop is important, i.e., the $j+1$st task cannot be executed before the jth task terminates. From the above discussion, it is simple to verify that a flow shop is a special case of a job shop; and an open shop is a flow shop in which the order of execution of tasks is not important. Detailed descriptions of these models as well as applications appear in [C], [CMM], [GS1], and [G1].

The following application will serve as an illustration for our problem. The processors are professors and the jobs are students. $t_{i,j}$ is the amount of time student J_i must meet with professor P_j. A schedule will consist of a sequence of assignments of students to professors in such a way that:

(i) no student meets with more than one professor at a time,
(ii) no professor meets with more than one student at a time, and
(iii) student J_i meets with professor P_j for exactly $t_{i,j}$ time units.

From this example it should be clear that one would be interested in algorithms for which n and m are parameters. A preemptive schedule is desired when there is no penalty for an interruption, whereas a nonpreemptive one will be used in the opposite case. In a no-wait schedule, once a student starts his meetings he must terminate them without interruptions. The reader will find it simple to see the effects of the different optimization criteria.

Let $f_i(S)$ represent the finishing time for job J_i in schedule S. The finish time, $ft(S)$, of a schedule S is $\max\{f_i(S)\}$. An *optimal finish time* (OFT) schedule is one with the least finish time among all feasible schedules. The *mean finishing time*, $mft(S)$, of a schedule S is $\sum f_i(S)/n$. An *optimal mean finishing time* (OMFT) schedule is one with the least mean finishing time among all feasible schedules. In this paper we restrict our

*Received June 4, 1979; revised August 25, 1980.
AMS 1980 subject classification. Primary 90B35. Secondary 68C25.
IAOR 1973 subject classification. Main: Scheduling. Cross reference: Computational analysis.
OR/MS Index 1978 subject classification. Primary 584 Production/scheduling, job shop.
Key words. Open shop, flow shop, job shop, preemptive and nonpreemptive schedules, NP-complete problems, mean finishing time, finish time.
†Supported in part by the National Science Foundation under Grant MCS-7721092.
‡Now at The University of Texas at Dallas.

0364-765X/82/0701/0057$01.25

attention to preemptive, nonpreemptive, and no-wait schedules. In a nonpreemptive schedule, once a task begins execution on some machine, it must continue executing without interruption until the task has been completed. Preemptive schedules allow the interruption of the execution of a task, i.e., a task does not have to be executed continuously. A schedule with no wait is one in which once a job starts execution, it must continue executing without interruption until the job has been terminated. It should be clear that the set of all preemptive schedules includes all nonpreemptive and no-wait schedules. Also, the set of all nonpreemptive schedules includes all no-wait schedules. The reverse is obviously not true.

In this paper we study the problem of constructing OFT and OMFT preemptive, nonpreemptive and no-wait schedules for open shop, flow shop, and job shop problems in which all nonzero tasks have identical execution time requirements. We show that it is unlikely that there exists an efficient algorithm to solve any of these problems, i.e., the problems belong to the class of problems known as NP-complete. The interesting fact is that if there exists an efficient algorithm to solve any of these problems, then one would have an efficient algorithm to solve the more general cases as well as many other well-known problems.

The operator, 'α', will be used as in $P_1 \alpha P_2$ to mean that problem P_1 polynomially reduces to problem P_2. A problem P_1 is *NP-hard* iff satisfiability αP_1. Problem P_1 is said to be *NP-complete* iff it is NP-hard and $P_1 \in NP$. A reader interested in more details about NP-complete problems is referred to [K1], [K2], and [C]. A problem P_1 is *NP-complete in the strong sense* iff it is NP-complete even when the input is presented in unary. In the context of scheduling theory, this means that the sum of all the input parameters to the problem is bounded by a polynomial on n, which is usually the number of jobs or machines. The reader is referred to [GJ] for more details. All strongly NP-complete problems are also NP-complete, but the reverse might not be true. For example, scheduling independent jobs on two machines so as to minimize the finish time is NP-hard, but the problem is not strongly NP-hard unless $P = NP$.

Gonzalez and Sahni [GS1] present a linear time algorithm to construct OFT preemptive and nonpreemptive schedules for a two machine open shop. The problem of obtaining an OFT nonpreemptive schedule is NP-hard when there are more than two machines in the open shop [GS1], and in general it is strongly NP-hard [L]. Efficient algorithms exists when the problem is to construct OFT preemptive schedules [GS1] [G1]. These algorithms will also construct OFT nonpreemptive schedules when all the nonzero tasks have identical execution time requirements. In §II it is shown that the problem of constructing OMFT preemptive and nonpreemptive schedules for open shops is NP-hard even when the execution time requirements for all nonzero tasks is identical.

Johnson [J] showed that OFT preemptive and nonpreemptive schedules for a two machine flow shop can be constructed efficiently. However, when there are more than two machines, the problem of constructing OFT preemptive and nonpreemptive schedules is NP-hard in the strong sense [GJSe], [GS2], [LRB]. When there are two machines in the flow shop, the problem of constructing an OMFT nonpreemptive schedule is strongly NP-hard [GJSe]. This result cannot be extended to preemptive schedules or to the case when all nonzero tasks have equal execution times. In §III it is shown that the problem of constructing OFT and OMFT preemptive and nonpreemptive schedules for flow shops is NP-hard even when the execution time of all nonzero tasks is identical.

Job shop scheduling problems are harder than flow shop problems. The problem of constructing OFT preemptive and nonpreemptive schedules for a two machine job shop is strongly NP-hard [GJSe], [GS2], [LRB]. Lenstra and Rinnooy Kan [LR] have shown that the problem of constructing OFT nonpreemptive schedules for a three machine job shop in which all tasks have equal execution times is NP-hard. OMFT

nonpreemptive scheduling for two machine job shops is NP-hard [GJSe]. In §III it is shown that the problem of constructing OFT and OMFT preemptive and nonpreemptive schedules for job shops is NP-hard even when the execution time of all tasks is identical.

Our results also apply to the problem of constructing OFT and OMFT no-wait schedules. These results are presented in §IV.

In order to prove our NP-complete results, we make use of the following problem which is shown to be NP-complete in Appendix I. This problem is closely related to a problem shown to be NP-complete in [GJSt].

(3, 4d)-graph coloration. Given an undirected graph, $G = (N, E)$, in which all nodes are of degree exactly 4, do there exist three disjoint sets of nodes (S_1, S_2, S_3) such that $\bigcup_{i=1}^{3} S_i = N$; and if $\{i, j\} \in E$, then node i and j are in different sets? ∎

Since NP-complete problems are stated as language recognition problems, we restate the OFT and OMFT problems mentioned above as follows:

LOMFT: Given an m processor, n job open shop with task times $t_{i,j}$, $1 \leqslant j \leqslant m$ and $1 \leqslant i \leqslant n$ and a number d, is there a schedule with $mft \leqslant d$?

FOMFT: Same as LOMFT but it refers to a flow shop problem.

JOMFT: Same as LOMFT but it refers to a job shop problem.

LOFT, FOFT and JOFT: Same as LOMFT, FOMFT and JOMFT but the problem is to determine whether there is a schedule with $ft \leqslant d$.

We should point out that if we show that these decision problems are NP-complete, then their corresponding optimization problems are NP-hard.

Sometimes it is necessary to distinguish between preemptive, nonpreemptive and no-wait schedules, so we just prefix LOMFT, . . . , JOFT with the type of schedule being considered.

II. NP-hard open shop problems (OMFT). Gonzalez and Sahni [GS1] present efficient algorithms to construct OFT preemptive and nonpreemptive schedules for two-machine, open shop problems. When there are more than two machines in the shop, the problem of obtaining an OFT nonpreemptive schedule is NP-hard [GS1]; and for an arbitrary number of machines, it is NP-hard in the strong sense [L]. Efficient algorithms have been presented in [GS1] and [G1] to construct OFT preemptive schedules. These algorithms will also construct OFT nonpreemptive schedules for the case when all nonzero tasks are of equal length.

In this section it is shown that preemptive and nonpreemptive LOMFT are NP-complete in the strong sense. These problems remain NP-complete even when the length of all nonzero tasks is identical.

In our proof we make use of the (3, 4d)-graph coloration problem which is shown to be NP-complete in Appendix I.

THEOREM 1. *Preemptive LOMFT is NP-complete even when the length of all nonzero tasks is identical.*

PROOF. The proof is in two lemmas. Lemma 1 shows that (3, 4d)-graph coloration α preemptive LOMFT. Lemma 4 shows that preemptive LOMFT is recognizable in nondeterministic polynomial time. ∎

COROLLARY. *Nonpreemptive LOMFT is NP-complete even when the length of all nonzero tasks is identical.*

PROOF. Similar to Lemmas 1 and 4. ∎

LEMMA 1. *(3, 4d)-graph coloration α preemptive LOMFT in which the length of all nonzero tasks is identical.*

PROOF. Given any graph, $G = (N, E)$, which is an input to the (3, 4d)-graph coloration problem, let us construct the following open shop problem, OS, with

$n' = 95n + 5r$ jobs and $m' = 25n + 5r$ processors, where $r = |E|$ and $n = |N|$. Assume without loss of generality that $N = \{v_1, v_2, \ldots, v_n\}$ and $E = \{e_1, e_2, \ldots, e_r\}$.

The set of processors is partitioned into five levels. The set of processors in level l is partitioned into edge processors $({}_EP_{l,1}, {}_EP_{l,2}, \ldots, {}_EP_{l,r})$ and node processors $({}_NP_{l,1,q}, {}_NP_{l,2,q}, \ldots, {}_NP_{l,n,q})$ for all $q \in \{1, \ldots, 5\}$. The set of jobs is partitioned into node-edge jobs, node jobs and edge jobs. The edge jobs are introduced to simplify the accounting of the mean flow time. The node jobs will guarantee that a certain subset of node-edge jobs will be executed in the same time interval; and if two subsets of node-edge jobs execute in the same time interval, then the subset of vertices they represent in G have no edge in common. We may assume without loss of generality that all nonzero tasks have unit execution time requirements. The nonzero tasks for each job as well as the number of jobs of each type will now be specified.

(i) *node-edge jobs.* For each vertex $v_i \in N$, there are five jobs in level l $(1 \leqslant l \leqslant 5)$. These jobs will be referred to by ${}_{NE}J_i^{p,l}$ for $1 \leqslant p \leqslant 5$. Let e_{i_1}, e_{i_2}, e_{i_3} and e_{i_4} be the edges incident upon vertex v_i. Job ${}_{NE}J_i^{p,l}$ will have nonzero tasks to be executed by the edge processors ${}_EP_{l,i_1}$, ${}_EP_{l,i_2}$, ${}_EP_{l,i_3}$, ${}_EP_{l,i_4}$ and by the node processor ${}_NP_{l,i,p}$.

(ii) *node jobs.* For each vertex $v_i \in N$, there are 14 jobs in each level l $(1 \leqslant l \leqslant 5)$. These jobs will be referred to by ${}_NJ_i^{p,l}$ for $p \in \{1, \ldots, 14\}$. Job ${}_NJ_i^{p,l}$ will have a nonzero task to be executed by each of the node processors in level l, i.e., by processors ${}_NP_{l,i,1}, \ldots, {}_NP_{l,i,5}$.

(iii) *edge jobs.* These jobs have nonzero tasks in all five levels. For each edge $e_j \in E$, there are five jobs which will be referred to by ${}_EJ_j^p$ for $p \in \{1, \ldots, 5\}$. ${}_EJ_j^p$ has a nonzero task to be executed by the jth edge processor in each level, i.e., by processors ${}_EP_{l,j}$ for $1 \leqslant l \leqslant 5$.

The value for d is 10. We now show that G is 3-colorable iff the above open shop problem has a preemptive schedule with $mft(S) \leqslant 10$.

(a) If G is three colorable, then OS has a schedule, S, with $mft(S) \leqslant 10$.

We prove this part by showing that there exists a schedule, S, for open shop OS with $mft(S) \leqslant 10$ when G is three colorable. Since G is three colorable, we can partition the set of nodes N into sets S_1, S_2 and S_3 in such a way that if v_i and $v_j \in S_k$ then $\{v_i, v_j\} \notin E$.

First of all, the set of jobs is partitioned into sets A_1, A_2, and A_3 using sets S_1, S_2 and S_3. Then it is shown that each of these sets uses the m' processors in the shop for five time units. This together with the results in [GS1], shows the existence of a schedule for open shop OS with $mft(S) \leqslant 10$.

First, let us define the following sets of jobs which will be used to define sets A_1, A_2 and A_3. For $1 \leqslant k \leqslant 3$

$$ {}_{NE}T_k = \{ {}_{NE}J_i^{p,l} \mid 1 \leqslant l \leqslant 5,\ 1 \leqslant p \leqslant 5 \text{ and } v_i \in S_k \} $$

and

$$ {}_ET_k = \{ {}_EJ_j^p \mid e_j \text{ is not incident upon a vertex in } S_k \text{ and } 1 \leqslant p \leqslant 5 \}. $$

$$ {}_NT_1 = \{ {}_NJ_i^{p,l} \mid 1 \leqslant l \leqslant 5,\ 1 \leqslant p \leqslant 4 \text{ and } v_i \in S_1 \} $$
$$ \cup \{ {}_NJ_i^{p,l} \mid 1 \leqslant l \leqslant 5,\ 5 \leqslant p \leqslant 9 \text{ and } v_i \in S_2 \cup S_3 \}. $$

$$ {}_NT_2 = \{ {}_NJ_i^{p,l} \mid 1 \leqslant l \leqslant 5,\ 5 \leqslant p \leqslant 9 \text{ and } v_i \in S_1 \} $$
$$ \cup \{ {}_NJ_i^{p,l} \mid 1 \leqslant l \leqslant 5,\ 1 \leqslant p \leqslant 4 \text{ and } v_i \in S_2 \} $$
$$ \cup \{ {}_NJ_i^{p,l} \mid 1 \leqslant l \leqslant 5,\ 10 \leqslant p \leqslant 14 \text{ and } v_i \in S_3 \}. $$

$$ {}_NT_3 = \{ {}_NJ_i^{p,l} \mid 1 \leqslant l \leqslant 5,\ 10 \leqslant p \leqslant 14 \text{ and } v_i \in S_1 \cup S_2 \} $$
$$ \cup \{ {}_NJ_i^{p,l} \mid 1 \leqslant l \leqslant 5,\ 1 \leqslant p \leqslant 4 \text{ and } v_i \in S_3 \}. $$

The sets $A_k (1 \leqslant k \leqslant 3)$ are as follows:

$$A_k = {}_{NE}T_k \cup {}_{E}T_k \cup {}_{N}T_k.$$

It is simple to verify that the sets ${}_{NE}T_k$, ${}_{E}T_k$ and ${}_{N}T_k$ partitions the set of node-edge jobs, edge jobs, and node jobs. Since these sets are included once in the sets A_k, it then follows that (A_1, A_2, A_3) is a partition of the jobs in the open shop OS.

We now show that the jobs in each set A_k utilize all processors for five time units. The proof is just for set A_1 (the proof for A_2 and A_3 are similar and will be omitted). Let us consider one processor at a time. There are two cases depending on the type of processor.

Case 1. edge processor ${}_{E}P_{l,j}$.

Clearly the only jobs using edge processors are the node-edge and the edge jobs. There are two subcases depending on whether edge e_j is incident upon a vertex in S_1 or not.

Subcase 1.1. e_j is incident upon $v_i \in S_1$.

Processor ${}_{E}P_{l,j}$ is used by the node-edge jobs ${}_{NE}J_i^{p,l}$ for $1 \leqslant p \leqslant 5$ which are included in set ${}_{NE}T_1 \subseteq A_1$. The vertex adjacent to v_i through e_j is certainly not in S_1. Since e_j is incident upon $v_i \in S_1$, it follows from the definition of ${}_{E}T_1$ that ${}_{E}J_j^p \notin {}_{E}T_1$. Hence, ${}_{E}P_{l,j}$ is used for five time units.

Subcase 1.2. e_j is not incident upon a vertex $v_i \in S_1$.

Since e_j is not incident upon a vertex $v_i \in S_1$, it follows that no job in ${}_{NE}T_1$ uses processor ${}_{E}P_{l,j}$. However, the jobs ${}_{E}J_j^p$ for $1 \leqslant p \leqslant 5$ are included in ${}_{E}T_1$. So, processor ${}_{E}P_{l,j}$ is used for five time units.

Case 2. node processor ${}_{N}P_{l,i,q}$.

The jobs using this type of processors are the node-edge jobs and the node jobs. There are two cases depending on whether $v_i \in S_1$ or not.

Subcase 2.1. $v_i \in S_1$.

The only node-edge job using processor ${}_{N}P_{l,i,q}$ is ${}_{NE}J_i^{q,l}$ and it is included in ${}_{NE}T_1 \subseteq A_1$. Since $v_i \in S_1$, ${}_{N}J_i^{z,l}$ for $1 \leqslant z \leqslant 4$ belong to ${}_{N}T_1$ and ${}_{N}J_i^{z,l}$ for $5 \leqslant z \leqslant 14$ does not belong to ${}_{N}T_1$. Hence, ${}_{N}P_{l,i,q}$ is used for five time units.

Subcase 2.2. $v_i \notin S_1$.

For this case, it can be easily shown that no job in ${}_{NE}T_1$ uses processor ${}_{N}P_{l,i,q}$. Since $v_i \notin S_1$, then ${}_{N}J_i^{z,l}$ for $5 \leqslant z \leqslant 9$ belong to ${}_{N}T_1$ and ${}_{N}J_i^{z,l}$ for $z = 1, \ldots, 4, 10, \ldots, 14$ does not belong to ${}_{N}T_1$. So, it must be that ${}_{N}P_{l,i,q}$ is used for five time units.

Hence, each processor is used for five time units when considering the jobs in set A_k. Since each job has five nonzero tasks, it then follows that $|A_k| = m'$.

Using the results in [GS1], we have that there is a schedule for the set of jobs in A_k with a makespan of five time units. Schedule S is obtained by concatenating the schedules for A_1, A_2 and A_3. Clearly $mft(S) = 10$. This concludes the proof for part (a).

(b) If open shop OS has a preemptive schedule S with $mft(S) \leqslant 10$, then G is three colorable.

Let S be preemptive schedule with $mft(S) \leqslant 10$ for open shop problem OS. Since open shop OS and schedule S satisfy the conditions of Lemma 2, it must be that in schedule S exactly m' jobs finish at time 5, 10 and 15. Let C_k be the set of jobs which terminate at time $5*k (1 \leqslant k \leqslant 3)$ in S. Since there are $3m'$ jobs in open shop OS and all jobs have execution time requirements of five units, it must be that $|C_k|$[1] $= m'$ and all jobs in C_k execute continuously from time $5*(k-1)$ to time $5*k$ in schedule S. Hence, the only jobs executing from time $5*(k-1)$ to time $5*k$ in S are those jobs in set C_k.

[1] $|C_k|$ denotes the number of elements in set C_k.

Clearly, the conditions of Lemma 3 are satisfied, so it must be that the set of node-edge jobs ${}_{NE}J_i^{\cdot,\cdot}$ belong to the same set C_k. This, together with the definition of the node-edge jobs, implies that if ${}_{NE}J_y^{p,l}$ and ${}_{NE}J_z^{p,l}$ belong to C_k, then $\{v_y, v_z\} \notin E$. Since S is a schedule for OS, we have that $C_k' = \{v_z \mid {}_{NE}J_z^{p,1} \in C_k\}$ for $k = 1,2,3$ is a partition of the nodes in G with the property that if v_y and $v_z \in C_k'$ then $\{v_y, v_z\} \notin E$. Clearly, C_k' gives a three coloration for G.

This completes the proof of Lemma 1. ∎

Let S be a preemptive or nonpreemptive schedule for open shop OS. Let f_i for $1 \leqslant i \leqslant 3m'$ be the finish time for job i in schedule S. Assume without loss of generality that $f_i \leqslant f_{i+1}$ for $1 \leqslant i < 3m'$.

LEMMA 2. *Let S and f_i be as defined above. $mft(S) \leqslant 10$ iff $f_{im'} = 5i$ for $1 \leqslant i \leqslant 3$.*

PROOF. The proof for the if part of the lemma is obvious. Before proving the only if part, we prove the following:

(1) for $1 \leqslant i \leqslant 2$, if $f_{im'} = 5i$ then $f_{im'+1} \geqslant 5(i+1)$;

(2) for $1 \leqslant i \leqslant 3$, $\sum_{j=(i-1)m'+1}^{im'} f_j \geqslant 5im'$;

(3) $\sum_{j=1}^{3m'} f_j \geqslant 30m'$.

The proof for (3) follows from (2), and the one for (1) is simple so it will be omitted. We now prove (2):

PROOF FOR (2). We prove this part for any i. If $f_{(i-1)m'+1} \geqslant 5i$, then (2) is obviously true. So, let us assume $f_{(i-1)m'+1} < 5i$. At this point, let's consider the schedule with jobs $im'+1, im'+2, \ldots, 3m'$ deleted. Since $f_{(i-1)m'+1} < 5i$, it must be that from time $f_{(i-1)m'+1}$ to time $5i$ there can be at most $m'-1$ jobs executing (note that jobs $im+1, \ldots, 3m'$ have been deleted) and some machine must be idle from time $f_{(i-1)m'+1}$ to time $5i$. Since all jobs have execution time requirements of five units, it must be that $f_{im'} > 5i$. Let k be such that $f_{(i-1)m'+1} \leqslant \ldots \leqslant f_k < 5i$ and $f_{k+1} \geqslant 5i$. Clearly, such a k must exist. From time f_l $((i-1)m'+1 \leqslant l \leqslant k)$ to time $5i$ there can be at most $m' - (l - ((i-1)m'+1) + 1)$ jobs executing and the total idle time before time $5i$ is $I \geqslant \sum_{j=(i-1)m'+1}^{k} 5i - f_j$. Since there are im' jobs $(im'+1, \ldots, 3m'$ were deleted) and all jobs have execution time requirements of five time units, it must be that after time $5i$ there are I units of time being used by jobs $k+1, \ldots, im'$. So, $\sum_{j=k+1}^{im'} f_j - 5i = I$. Hence, $\sum_{j=k+1}^{im'} f_j - 5i = I \geqslant \sum_{j=(i-1)m'+1}^{k} 5i - f_j$ and $\sum_{j=(i-1)m'+1}^{im'} f_j \geqslant 5im'$.

This completes the proof for (2). ∎

In order to complete the proof of the lemma it is required to show that equality in (3) holds true only if $f_{im'} = 5i$ for $1 \leqslant i \leqslant 3$.

It should be clear that $f_1, f_2, \ldots, f_{m'} \geqslant 5$. If $f_{m'} > 5$ then $\sum_{j=1}^{m'} f_j > 5m'$ and hence $\sum_{j=1}^{3m'} f_j > 30m'$ (note that (3) was obtained from (2)). So assume $f_{m'} = 5$. From (1) it follows that $f_{m+1} \geqslant 10$ and consequently $f_{2m'} \geqslant 10$. If $f_{2m'} > 10$ then $\sum_{j=m'+1}^{2m'} f_j > 10m'$ and $\sum_{j=1}^{3m'} f_j > 30m'$ ((3) was obtained from (2)). So, assume $f_{2m'} = 10$. Using similar arguments it can be shown that $f_{3m'} = 15$. Therefore, $mft(S) \leqslant 10$ iff $f_{im'} = 5im'$ for $1 \leqslant i \leqslant 3$.

This completes the proof of the lemma. ∎

The notation used in Lemma 3 is the same as the one used in Lemma 1 part (b).

LEMMA 3. *If ${}_{NE}J_i^{p,l} \in C_k$, $|C_k| = m'$ and in schedule S the only jobs executing from time $5(k-1)$ to time $5k$ are jobs in C_k, then $\{{}_{NE}J_i^{p,l} \mid 1 \leqslant p \leqslant 5\} \subseteq C_k$.*

PROOF. The proof of this lemma is by contradiction. Assume ${}_{NE}J_i^{p,l} \in C_k$, $|C_k| = m'$ and in schedule S the only jobs executing from time $5(k-1)$ to time $5k$ are jobs in C_k but ${}_{NE}J_i^{q,l} \notin C_k$. The only jobs using processors ${}_{N}P_{l,i,z}$ for $1 \leqslant z \leqslant 5$ from time $5(k-1)$ to time $5k$ are some of the node-edge and node jobs in C_k (clearly, no edge job will use such a processor). Since ${}_{NE}J_i^{p,l} \in C_k$ and ${}_{NE}J_i^{q,l} \notin C_k$, then the node

processor ${}_NP_{l,i,p}$ is used for one time unit by the node-edge jobs in C_k, whereas the node processor ${}_NP_{l,i,q}$ is not used by the node-edge jobs in C_k. Since the node jobs use all processors ${}_NP_{l,i,z}$ for $1 \leqslant z \leqslant 5$, it follows that no more than four of these jobs can be in C_k as otherwise processor ${}_NP_{l,i,p}$ would be used for more than five time units and all jobs in C_k could not be scheduled from time $5(k-1)$ to time $5k$. But in this case processor ${}_NP_{l,i,q}$ will not execute five nonzero tasks; and since all jobs have five nonzero tasks, it must be that $|C_k| < m'$, which contradicts our earlier assumption. So, it must be that if ${}_{NE}J_i^{p,l} \in C_k$, then $\{{}_{NE}J_i^{p,l} | 1 \leqslant p \leqslant 5\} \subseteq C_k$.

This completes the proof of the lemma. ∎

LEMMA 4. *Preemptive LOMFT is recognizable in nondeterministic polynomial time.*

PROOF. It is simple to construct a nondeterministic Turing machine that guesses a preemptive schedule and verifies that its *mft* is less than or equal to d. The only problem that we could encounter is that there could be too many preemptions. But, using a similar argument to the one in [GS2], one can show that there is always an OMFT preemptive schedule with at most *rnm* preemptions for any instance I of LOMFT; thus for any d there need be no more than *rnm* preemptions. ∎

III. NP-hard flow shop and job shop problems. Johnson [J] showed that OFT preemptive and nonpreemptive schedules for a two machine flow shop can be constructed by efficient algorithms. When there are more than two machines, the problems of constructing preemptive and nonpreemptive schedules is NP-hard [GJSe], [GS2], [LRB]. The reductions used in proving these problems hard cannot be extended to the case when all nonzero tasks are identical. In this section we show that the problem of constructing OFT preemptive and nonpreemptive schedules is NP-hard even when all nonzero tasks are of equal length.

When there are two machines in the flow shop, the problem of constructing an OMFT nonpreemptive schedule is NP-hard [GJSe]. This reduction cannot be extended to the case when the objective is to construct an OMFT preemptive schedule or to the case when all nonzero tasks have equal execution time requirements. In this section it will be shown that the problem of constructing OMFT preemptive and nonpreemptive schedules for flow shops is NP-hard even when the execution times of all nonzero tasks is identical.

Job shop scheduling problems are harder than flow shop problems. The problems of constructing OFT preemptive and nonpreemptive schedules for a two machine job shop is NP-hard [GJSe], [GS2], [LRB]. For three machines, finding OFT nonpreemptive schedules is NP-hard even when all tasks are of equal length [L]. OMFT nonpreemptive scheduling problems for two machine job shops is NP-hard [GJSe]. In this section we show that the problems of constructing OFT and OMFT preemptive and nonpreemptive schedules for job shops is NP-hard even when all tasks are of equal length.

The reduction used in this section is similar to the one in the previous section. The (3, 4d)-graph coloration problem which is shown to be NP-complete in Appendix I will be used.

THEOREM 2. *Preemptive FOMFT is NP-complete even when all nonzero tasks are of equal length.*

PROOF. The proof is in two lemmas. Lemma 5 shows that (3, 4d)-graph coloration α preemptive FOMFT. Lemma 6 shows that preemptive FOMFT is recognizable in nondeterministic polynomial time. ∎

COROLLARY. *Nonpreemptive is FOMFT is NP-complete even when all nonzero tasks are of equal length.*

PROOF. Similar to Lemmas 5 and 6. ∎

LEMMA 5. *$(3, 4d)$-graph coloration α preemptive FOMFT in which all nonzero tasks have equal execution times.*

PROOF. The proof follows the same type of arguments as the one for Lemma 1; however, it is much more complex. An interested reader can see the proof in [G2]. ∎

LEMMA 6. *Preemptive FOMFT is recognizable in nondeterministic polynomial time.*

PROOF. This proof follows the same arguments as the ones used in Lemma 4. The bound for the number of preemptions is rn. ∎

THEOREM 3. *Preemptive FOFT is NP-complete even when all nonzero tasks have equal execution times.*

PROOF. The proof is similar to Theorem 2, but the reduction in Lemma 5 will ignore the final jobs. ∎

COROLLARY. *Nonpreemptive FOFT is NP-complete even when all nonzero tasks have equal execution times.*

PROOF. Similar to Theorem 3. ∎

Since all flow shop problem instances are also job shop problem instances, the same results for flow shop hold for job shops.

IV. No-wait schedules. In this section we study the problem of constructing no-wait schedules (once a job starts execution, it will remain executing until the job terminates) for open shops, flow shops and job shops. When all nonzero tasks have equal execution time requirements and all jobs use all the machines, the problem of constructing OFT and OMFT no-wait schedules for an open shop and a flow shop is trivial. The next case is when not all jobs use all the machines and all nonzero tasks have identical execution times. For this case, we show that the problem of constructing OFT and OMFT no-wait schedules for open shops, flow shops, and job shops is NP-hard.

For two machine open shop problems, Sahni and Cho [SC] have shown that the problem of constructing OFT no-wait schedules is NP-hard, even when all jobs require nonzero execution time requirements on both machines. In this section we show that the problem of constructing OFT and OMFT no-wait schedules is NP-hard even when all nonzero tasks are identical.

THEOREM 4. *No-wait LOMFT is NP-complete even when all nonzero tasks have equal length.*

PROOF. Similar to Theorem 1. ∎

THEOREM 5. *No-wait LOFT is NP-complete even when all nonzero tasks have equal length.*

PROOF. Similar to Theorem 1. ∎

For two machine flow shop problems, the problem of constructing OFT no-wait schedules when all tasks have nonzero execution times can be solved efficiently [GG]. Papadimitriou and Kanellakis [PK] have shown that the problem is NP-hard when there are four machines in the shop. If some tasks are allowed to skip execution on some machine, then the problem is NP-hard even when there are two machines in the shop [SC]. The problem of constructing an OMFT no-wait schedule for flow shops is NP-hard [LRB]. In this section we show that the problem of constructing OFT and OMFT no-wait schedules is NP-hard even when all nonzero tasks have identical execution time requirements.

THEOREM 6. *No-wait FOMFT is NP-complete even when all nonzero tasks have equal length.*

PROOF. Similar to Theorem 2. ∎

THEOREM 7. *No-wait FOFT is NP-complete even when all nonzero tasks have equal length.*

PROOF. Similar to Theorem 3. ∎

Sahni and Cho [SC] have shown that the problem of constructing OFT no-wait schedules for a two machine job shop is NP-hard even when all jobs have at least two tasks. The same results stated in Theorems 6 and 7 apply for job shops.

V. Discussion. We have shown that the problem of constructing OMFT preemptive and nonpreemptive schedules for an open shop, flow shop, and job shop is NP-hard. These results will also extend to the case when the problem is to construct an OFT preemptive and nonpreemptive schedule for a flow shop and job shop. In §IV we showed that the problem of constructing OFT and OMFT no-wait schedules for open shops, flow shops, and job shops is NP-hard. All of these problems remain hard even when all nonzero tasks have equal execution time requirements. The interesting fact is that if there exists an efficient algorithm to solve any of these restricted problems, then there exists an efficient algorithm to solve the more general problems as well as many other well-known problems.

Acknowledgement. The author is grateful to Professor Donald B. Johnson for his helpful comments and suggestions while preparing this manuscript.

Appendix I. In this section we show that the (3, 4d)-graph coloration problem is NP-complete. Garey, Johnson, and Stockmeyer [GJSt] showed that the 3-graph coloration problem in which all nodes are of degree at most 4 is NP-complete. We use such a problem to prove that the (3, 4d)-graph coloration problem is NP-complete. It would be of interest to prove the (3, 3d)-coloration problem to be NP-complete, but as noted in [GJSt], the 3-graph coloration problem can be solved efficiently when each node is of degree at most 3. The algorithm relies on the well-known results of Brooks [B] which implies that a connected graph with maximum degree 3 is 3-colorable iff it differs from K_4, the complete graph on four nodes, which is easy to determine.

THEOREM A.1. *The (3, 4d)-graph coloration problem is NP-complete.*

PROOF. It should be clear that this decision problem is in NP. We now show that 3-graph coloration with node degree at most 4α (3, 4d)-graph coloration.

Let $G = (N, E)$ be any graph with node degree at most 4. From this graph we construct $G'' = (N'', E'')$ in which all nodes are of degree four and such that G'' is 3-colorable iff G is 3-colorable.

First of all let us transform G to G' in such a way that each node in G' is of even degree $\leqslant 4$ and G' is 3-colorable iff G is 3-colorable. It can be easily shown that any graph G has an even number of odd degree vertices. If the graph has zero vertices of odd degree then $G' = G$. Otherwise, let $i_1, \ldots, i_k$ be the vertices of odd degree in G. Now, $G' = (N', E')$ where

$$N' = N \cup \{v'_1, v'_2, \ldots, v'_{k/2}\},$$

$$E' = E \cup \{\{v_{i_j}, v'_{\lceil j/2 \rceil}\} \mid j = 1, \ldots k\}.$$

It is simple to show that G' is 3-colorable iff G is 3-colorable.

We now construct G'' is of degree 4 and G'' is 3-colorable iff G is 3-colorable.

Initially, G'' is just a copy of G'. For each vertex i in G'' of degree 2 (vertices of zero degree can be eliminated), augment G'' with the subgraph in Figure 1. It should be

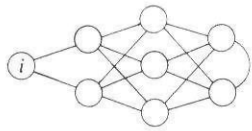

FIGURE 1. New edges for vertex i.

clear that G'' will now have all vertices of degree 4 and G'' is 3-colorable iff G is 3-colorable. This completes the proof of the theorem. ∎

References

[B] Brooks, R. L. (1941). On Coloring the Nodes of a Network. *Proc. Cambridge Philos. Soc.* **37** 194–197.

[C] Coffman, E. G., Jr. (1976). *Computer and Job Shop Scheduling Theory*. John Wiley and Sons, New York.

[CMM] Conway, R. W., Maxwell, W. L. and Miller, L. W. (1967). *Theory of Scheduling*. Addison-Wesley, Reading, Mass.

[J] Johnson, S. M. (1954). Optimal Two-and-Three-State Production Schedule. *Naval Res. Logist. Quart.* **1**.

[G1] Gonzalez, T. (1979). A Note on Open Shop Schedules. *IEEE Trans. Computers.* **C-28** 782–786.

[G2] ———. (May 1979). NP-hard Shop Problems. Technical Report CS-79-35, Department of Computer Science, The Pennsylvania State University.

[GJ] Garey, M.R. and Johnson, D. S. (1978). Strong NP-Completeness Results: Motivations, Examples, and Implicants. *J. Assoc. Comput. Mach.* **25** 499–508.

[GJSe] ———, Johnson, D. S. and Sethi, R. (1976). Complexity of Flow Shop and Job Shop Scheduling. *Math Oper. Res.* **1** 117–129.

[GJSt] ———, ——— and Stockmeyer, L. (1976). Some Simplified NP-Complete Graph Problems. *Theoret. Comput. Sci.* **1** 237–267.

[GG] Gilmore, P. and Gomory, R. (1964). Sequencing a One State-variable Machine: A Solvable Case of the Travelling Salesman Problem. *O. R.* **12** 655–679.

[GS1] Gonzalez, T. and Sahni, S. (1976). Open Shop Scheduling to Minimize Finish Time. *J. Assoc. Comput. Mach.* **23** 665–679.

[GS2] ——— and ———. Flowshop and Jobshop Schedules: Complexity and Approximation. *O. R.* **26** 36–52.

[K1] Karp, R. M. (1972). Reducibility among Combinational Problems. In *Complexity of Computer Computation*, pp. 85–104, R. E. Miller and J. W. Thatcher, eds. Plenum Press, New York.

[K2] ———. (1975). On the Computational Complexity of Combinational Problems. *Networks* **5** 45–68.

[L] Lenstra, J. K. Unpublished manuscript.

[LR] ——— and Rinnooy Kan, A. H. G. (1979). Computational Complexity of Discrete Optimization Problems. *Ann. Discrete Math.* **4** 121–140.

[LRB] ———, Rinnooy Kan, A. H. G. and Brucker, P. (1977). Computational Complexity of Machine Scheduling Problems. *Ann. Discrete Math.* **1** 343–362.

[PK] Papadimitriou, C. H. and Kanellakis, P. C. (October 1978). Flowshop Scheduling with Limited Temporary Storage. Proceeding of the 16th Annual Allenton Conference on Communication, Control and Computing, 214–223.

[SC] Sahni, S. and Cho, Y. (December 1977). Complexity of Scheduling Shops with No Wait in Process. Technical Report #77-20, Department of Computer Science, University of Minnesota.

PROGRAMS IN MATHEMATICAL SCIENCES, THE UNIVERSITY OF TEXAS AT DALLAS, RICHARDSON, TEXAS 75080