NP-hard Shop Problems[†]

Teofilo Gonzalez, May 1979
CS-79-35

Department of Computer Science
The Pennsylvania State University
University Park, Pennsylvania 16802

[†]Supported in part by National Science Foundation under Grant MCS-21092.

NP-hard Shop Problems[†]

by

Teofilo Gonzalez
Computer Science Department
The Pennsylvania State University
University Park, PA 16802

Abstract

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.

Key words: open shop, flow shop, job shop, preemptive and nonpreemptive schedules, NP-complete problems, mean finishing time, finish time.

[†]Supported in part by National Science Foundation under Grant MCS-21092.

I. Introduction

There are $n \geq 1$ 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 [G].

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 $\Sigma f_i(S)/n$. 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 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 there will exist an efficient algorithm to solve the more general cases and 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 α P_1. Problem P_1 is said to be NP-complete iff it is NP-hard and $P_1 \epsilon NP$. 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 exist when the problem is to construct OFT preemptive schedules [GS1] [G]. These algorithms will also construct OFT nonpreemptive schedules when all the nonzero tasks have identical execution time requirements (see appendix II). In section 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 problems 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 section 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 times 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 section III it is shown that the problem of constructing OFT and OMFT preemptive and nonpreemptive

schedules for job shops in NP-hard even when the execution time of all tasks is identical.

Our results also apply for the problem of constructing OFT and OMFT no wait schedules. These results are presented in section 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, does 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 $t_{i,j}$, $1 \leq j \leq m$ and $1 \leq i \leq n$ and a number d, is there a schedule with $mft \leq d$?

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 $ft \leq d$.

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 [G] to construct OFT preemptive schedules. These algorithms will also construct OFT nonpreemptive schedules for the case when all nonzero tasks are of equal length. This will be shown in appendix II since it will be of use in lemma 1.

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 $({}_{E}P_{\ell,1}, {}_{E}P_{\ell,2}, \cdots, {}_{E}P_{\ell,r})$ and node processors $({}_{N}P_{\ell,1,q}, {}_{N}P_{\ell,2,q}, \ldots, {}_{N}P_{\ell,n,q})$ for $q\varepsilon\{1, \ldots, 5\}$. 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 $v_i \varepsilon N$ there are five jobs in level ℓ $(1 \leq \ell \leq 5)$. These jobs will be referred to by ${}_{NE}J_i^{p,\ell}$ for $1 \leq p \leq 5$. Let e_{i_1}, e_{i_2}, e_{i_3} and e_{i_4} be the edges incident upon vertex v_i. Job ${}_{NE}J_i^{p,\ell}$ will have nonzero tasks to be executed by the edge processors ${}_{E}P_{\ell,i_1}$, ${}_{E}P_{\ell,i_2}$, ${}_{E}P_{\ell,i_3}$, ${}_{E}P_{\ell,i_4}$ and by the node processor ${}_{N}P_{\ell,i,p}$.

ii) node jobs: For each vertex $v_i \varepsilon N$ there are 14 jobs in each level ℓ $(1 \leq \ell \leq 5)$. These jobs will be referred to by ${}_{N}J_i^{p,\ell}$ for $p\varepsilon\{1, \ldots, 14\}$. Job ${}_{N}J_i^{p,\ell}$ will have a nonzero task to be executed by each of the node processors in level ℓ, i.e., by processors ${}_{N}P_{\ell,i,1}, \cdots, {}_{N}P_{\ell,i,5}$.

iii) edge jobs: These jobs have nonzero tasks in all 5 levels. For each edge $e_j \varepsilon E$, there are five jobs which will be referred to by ${}_{E}J_j^{p}$ for

$p\varepsilon\{1,\ldots, 5\}$. ${}_{E}J_{j}^{p}$ has a nonzero task to be executed by the jth edge processor in each level, i.e., by processors ${}_{E}P_{\ell,j}$ for $1 \leq \ell \leq 5$.

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

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

We prove this part by showing that there exists a schedule, S , for open shop OS with $mft(S) \leq 10$ 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 $v_j \varepsilon S_k$ 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 lemma A.1, shows the existance of a schedule S for open shop OS with $mft(S) \leq 10$.

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

$${}_{NE}T_k = \{{}_{NE}J_i^{p,\ell} \mid 1 \leq \ell \leq 5,\ 1 \leq p \leq 5 \text{ and } v_i \varepsilon S_k\}$$

and ${}_{E}T_k = \{{}_{E}J_j^{p} \mid e_j$ is not incident upon a vertex in S_k and $1 \leq p \leq 5\}$.

$${}_{N}T_1 = \{{}_{N}J_i^{p,\ell} \mid 1 \leq \ell \leq 5,\ 1 \leq p \leq 4 \text{ and } v_i \varepsilon S_1\} \cup \{{}_{N}J_i^{p,\ell} \mid 1 \leq \ell \leq 5,\ 5 \leq p \leq 9 \text{ and } v_i \varepsilon S_2 \cup S_3\}$$

$${}_{N}T_2 = \{{}_{N}J_i^{p,\ell} \mid 1 \leq \ell \leq 5,\ 5 \leq p \leq 9 \text{ and } v_i \varepsilon S_1\} \cup \{{}_{N}J_i^{p,\ell} \mid 1 \leq \ell \leq 5,\ 1 \leq p \leq 4 \text{ and } v_i \varepsilon S_2\} \cup \{{}_{N}J_i^{p,\ell} \mid 1 \leq \ell \leq 5,\ 10 \leq p \leq 14 \text{ and } v_i \varepsilon S_3\}$$

$$ {}_{N}T_3 = \{ {}_{N}J_i^{p,\ell} \mid 1 \le \ell \le 5,\ 10 \le p \le 14 \text{ and } v_i \in S_1 \cup S_2 \} \cup $$
$$ \{ {}_{N}J_i^{p,\ell} \mid 1 \le \ell \le 5,\ 1 \le p \le 4 \text{ and } v_i \in S_3 \} $$

The sets A_k $(1 \le k \le 3)$ 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 5 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.

<u>case 1</u>: edge processor ${}_{E}P_{\ell,j}$

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.

<u>subcase 1.1</u>: e_j is incident upon $v_i \in S_1$.

Processor ${}_{E}P_{\ell,j}$ is used by the node-edge jobs ${}_{NE}J_i^{p,\ell}$ for $1 \le p \le 5$ 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 ${}_{E}P_{\ell,j}$ is used for 5 time units.

subcase 1.2: e_j is not incident upon a vertex $v_i \varepsilon S_1$.

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

case 2: node processor $_{N}P_{\ell,i,q}$

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

subcase 2.1: $v_i \varepsilon S_1$

The only node-edge job using processor $_{N}P_{\ell,i,q}$ is $_{NE}J_i^{q,\ell}$ and it is included in $_{NE}T_1 \subseteq A_1$. Since $v_i \varepsilon S_1$, $_{N}J_i^{z,\ell}$ for $1 \leq z \leq 4$ belong to $_{N}T_1$ and $_{N}J_i^{z,\ell}$ for $5 \leq z \leq 14$ does not belong to $_{N}T_1$. Hence $_{N}P_{\ell,i,q}$ 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 $_{N}P_{\ell,i,q}$. Since $v_i \notin S_1$, then $_{N}J_i^{z,\ell}$ for $5 \leq z \leq 9$ belong to $_{N}T_1$ and $_{N}J_i^{z,\ell}$ for z=1,...,4,10,...,14 does not belong to $_{N}T_1$. So, it must be that $_{N}P_{\ell,i,q}$ is used for 5 time units.

Hence each processor is used for 5 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 lemma A.1 we have that there is a schedule for the set of jobs in A_k in 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 G is not three colorable then all schedules for open shop OS must have mft > 10.

This will be shown by contradiction. Assume G is not three colorable but there is a schedule, S, with mft(S) $\leq$ 10. 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 ($1 \leq k \leq 3$) 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 $|C_k|^* = m'$ 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.

Clearly, the conditions of lemma 3 are satisfied so it must be that the set of node-edge jobs ${}_{NE}J_i^{\cdot,\ell}$ belong to the same set C_k. This together with the definition of the node-edge jobs implies that if ${}_{NE}J_y^{p,\ell}$ and ${}_{NE}J_z^{p,\ell}$ belong to C_k then $\{v_y, v_z\} \notin E$. Since S is a schedule for OS with mft(S) $\leq$ 10 it then follows that G is three colorable. A contradiction. Hence there is no schedule, S, for OS with mft(S) $\leq$ 10 when G is not three colorable.

This completes the proof of lemma 1. □

---

* $|C_k|$ denotes the number of elements in set C_k

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

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

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

1) for $1 \leq i \leq 2$, if $f_{im'} = 5i$ then $f_{im'+1} \geq 5(i+1)$

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

3) $\sum_{j=1}^{3m'} f_j \geq 30m'$

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 $f_{(i-1)m'+1} \geq 5i$ then (2) is obviously true. So, let us assume $f_{(i-1)m'+1} < 5i$. At this point lets 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 5 units it must be that $f_{im'} > 5i$. Let k be such that $f_{(i-1)m'+1} \leq \ldots \leq f_k < 5i$ and $f_{k+1} \geq 5i$. Clearly such a k must exist. From time f_ℓ $((i-1)m'+1 \leq \ell \leq k)$

to time $5i$ there can be at most $m' - (\ell - ((i-1)m'+1) + 1)$ jobs executing and the total idle time before time $5i$ is $I \geq \sum_{j=(i-1)m'+1}^{k} 5i-f_j$. Since there are im' jobs $(im'+1,\ldots, 3m'$ were deleted) and all jobs have execution time requirements of 5 time units, it must be that after time $5i$ there are I units of time being using by jobs $k+1, \ldots, im'$. So, $\sum_{j=k+1}^{im'} f_j - 5i = I$. Hence, $\sum_{j=k+1}^{im'} f_j - 5i = I \geq \sum_{j=(i-1)m'+1}^{k} 5i-f_j$ and

$$\sum_{j=(i-1)m'+1}^{im'} f_j \geq 5im' .$$

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 $1 \leq i \leq 3$.

It should be clear that $f_1, f_2, \ldots, f_{m'} \geq 5$. 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 $f_{m+1} \geq 10$ and consequently $f_{2m'} \geq 10$. 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 $mft(S) \leq 10$ iff $f_{im'} = 5im'$ for $1 \leq i \leq 3$.

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

Proof: The proof of this lemma is by contradiction. Assume $_{NE}J_i^{p,\ell} \in C_k$, $|C_k| = m'$ and in schedule S the only jobs executing from time $5(k-1)$ to time $5k$ are jobs in C_k but $_{NE}J_i^{q,\ell} \notin C_k$. The only jobs using processors $_{N}P_{\ell,i,z}$ for $1 \leq z \leq 5$ 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 $_{NE}J_i^{p,\ell} \in C_k$ and $_{NE}J_i^{q,\ell} \notin C_k$, then the node processor $_{N}P_{\ell,i,p}$ is used for one time unit by the node-edge jobs in C_k whereas the node processor $_{N}P_{\ell,i,q}$ is not used by the node-edge jobs in C_k. Since the node jobs use all processors $_{N}P_{\ell,i,z}$ for $1 \leq z \leq 5$, it follows that no more than four of these jobs can be in C_k as otherwise processor $_{N}P_{\ell,i,p}$ 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 $_{N}P_{\ell,i,q}$ will not execute 5 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 $_{NE}J_i^{p,\ell} \in C_k$ then $\{_{NE}J_i^{p,\ell} \mid 1 \leq p \leq 5\} \subseteq C_k$.

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. □

Corallary: 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 non-zero tasks have equal execution times.

Proof: From the (3,4d)-graph coloration problem $G = (N,E)$ construct the following flow shop problem, FS , with $n' = (3r + w + 9)\, n$ jobs and $m' = r + n\,(2w + 7r + 11)$ processors, where $r = |E|$, $n = |N|$ and $w = 3rn + 6n + 4r + 8$.

The reduction is similar to the one in section II, however, it is more complex. The set of jobs is partitioned into node, dummy, position and final jobs. The set of node jobs are essentially as the node-edge jobs in the previous section. Two node jobs will execute in the same time interval iff the nodes they represent in G are not adjacent. In order to guarantee that this property will be met, it is required to introduce the set of position jobs. The set of dummy jobs will fill up

the empty gaps left on the processors so as to simplify the accounting of the mean finishing time as well as to identify three different regions for the node jobs to execute in. The final jobs will ensure that the position jobs will leave no more free space than the one required.

Node and dummy jobs are executed by the top, edge, edge-node and bottom processors. In order to guarantee that these jobs will execute in some specific time intervals, the position jobs will use the top and bottom processors. The position jobs will also use the high, medium and low processors. The final jobs will force all the other jobs to fall into correct place by making use of the front, top, medium bottom, low and end processors.

Before specifying all jobs in flow shop FS, let us identify the different processors to be used together with their names and order in the shop.

type of processor	name	range for the indices
front	$F^P_{i,j}$	$i\in N$, $j\in\{1,\ldots,w+3r+5\}$
high	$H^P_{i,j}$	$i\in N$, $j\in\{1,\ldots,3r+6\}$
top	T^P_i	$i\in N$
edge	E^P_j	$j\in E$
edge-node	$EN^P_{i,j}$	$i\in N$, $j\in E$
medium	$M^P_{i,j}$	$i\in N$, $j\in\{1,\ldots,r+2\}$
bottom	B^P_i	$i\in N$
low	$L^P_{i,j}$	$i\in N$, $j\in\{1,\ldots,2r+3\}$
end	$D^P_{i,j}$	$i\in N$, $j\in\{1,\ldots,w-3r-7\}$

The ordering of the processors in the shop is as follows:

$$F^P_{i,j} \lhd F^P_{k,\ell} \quad , \text{ for } i(w+3r+5) + j < k(w+3r+5)+\ell \ ;$$

$$F^P_{i,j} \lhd H^P_{1,1} \quad , \text{ for all } i \text{ and } j \ ;$$

$$H^P_{i,j} \lhd H^P_{k,\ell} \quad , \text{ for } i(3r+6) + j < k(3r+6)+\ell \ ;$$

$$H^P_{i,j} \lhd T^P_1 \quad , \text{ for all } i \text{ and } j \ ;$$

$$T^P_i \lhd T^P_j \quad , \text{ for } i < j \ ;$$

$$T^P_i \lhd E^P_1 \quad , \text{ for all } i \ ;$$

$$E^P_j \lhd EN^P_{i,j} \quad , \text{ for all } i \text{ and } j \ ;$$

$$EN^P_{i,j} \lhd EN^P_{i,k} \quad , \text{ for } j < k \text{ and all } i \ ;$$

$$E^P_i \lhd E^P_j \quad , \text{ for } i < j \ ;$$

$$EN^P_{i,j} \lhd M^P_{1,1} \quad , \text{ for all } i,j \ ;$$

$$M^P_{i,j} \lhd M^P_{k,\ell} \quad , \text{ for } i(r+2) + j < k(r+2)+\ell \ ;$$

$$M^P_{i,j} \lhd B^P_1 \quad , \text{ for all } i,j \ ;$$

$$B^P_i \lhd B^P_j \quad , \text{ for all } i < j \ ;$$

$$B^P_i \lhd L^P_{1,1} \quad , \text{ for all } i$$

$$L^P_{i,j} \lhd L^P_{k,\ell} \quad , \text{ for } i(2r+3) + j < k(2r+3)+\ell \ ;$$

$$L^P_{i,j} \lhd D^P_{1,1} \quad , \text{ for all } i,j \ ;$$

$$D^P_{i,j} \lhd D^P_{k,\ell} \quad , \text{ for } i(w-3r-7) + j < k(w-3r-7)+\ell$$

From the above partial order it is simple to verify that there is one and only one total ordering of the processors in the shop. In what follows we formally specify how each job is to be composed. Note that all nonzero tasks require unit execution time.

1.- node jobs: For each vertex $v_i \in N$ there is a node job $({}_N J_i)$. Let e_{j_1}, e_{j_2}, e_{j_3} and e_{j_4} be the edges incident upon vertex v_i. Node job ${}_N J_i$ has nonzero tasks to be executed by the following processors:

a) top processor ${}_T P_i$;

b) edge processors ${}_E P_{j_1}$, ${}_E P_{j_2}$, ${}_E P_{j_3}$ and ${}_E P_{j_4}$;

c) edge-node processors ${}_{EN} P_{i,j}$, $e_j \in E - \{e_{j_1}, e_{j_2}, e_{j_3}, e_{j_4}\}$;

and d) bottom processors ${}_B P_i$.

2.- dummy jobs: For each vertex $v_i \in N$ there are two dummy jobs $({}_D J_{i,1}$ and ${}_D J_{i,2})$. These jobs have nonzero tasks to be executed by the following processors:

a) top processor ${}_T P_i$;

b) edge-node processors ${}_{EN} P_{i,j}$ for $e_j \in E$;

and c) bottom processor ${}_B P_i$.

3.- position jobs: For each vertex $v_i \in N$ and $j \in \{1,\ldots,3r+6\}$ there is a position job denoted by ${}_P J_{i,j}$. Job ${}_P J_{i,j}$ has a nonzero task to be executed by the following processors:

a) for $j > 1$ job ${}_P J_{i,j}$ has a nonzero task to be executed by the set of high processors

$$\{{}_H P_{i,k} \mid k \in \{3r+7-j, \ldots, 3r+5\}\}$$

b) for $j \in \{1, r+3, 2r+5\}$ job ${}_P J_{i,j}$ has a nonzero task to be executed by the high processor ${}_H P_{i,3r+6}$.

c) for $j \in \{2, \ldots, r+2, r+4, \ldots, 2r+4, 2r+6, \ldots, 3r+6\}$ job ${}_P J_{i,j}$ has a nonzero task to be executed by the top processor ${}_T P_i$.

d) for $j \in \{1, \ldots, 2r+5\}$ job ${}_P J_{i,j}$ has a nonzero task to be executed by the set of middle processors $\{{}_M P_{i,k} \mid k \in \{1, \ldots, r+1\}\}$

e) for $j\varepsilon\{2r+6,\ldots, 3r+5\}$ job ${}_{P}J_{i,j}$ has a nonzero task to be executed by the set of middle processors $\{{}_{M}P_{i,k} \mid k\varepsilon\{1,\ldots, 3r+6-j\}\}$.

f) for $j\varepsilon\{r+2, 2r+4\}$ job ${}_{P}J_{i,j}$ has a nonzero task to be executed by the middle processor ${}_{M}P_{i,r+2}$.

g) for $j\varepsilon\{1,\ldots, r+1, r+3,\ldots, 2r+3\}$ job ${}_{P}J_{i,j}$ has a nonzero task to be executed by the bottom processor ${}_{B}P_{i}$.

h) for $j\varepsilon\{1,\ldots, 2r+3\}$ job ${}_{P}J_{i,j}$ has a nonzero task to be executed by the set of low processors $\{{}_{L}P_{i,k} \mid k\varepsilon\{1,\ldots, 2r+4-j\}\}$.

Figure 2 shows the position jobs ${}_{P}J_{i,(\cdot)}$.

4.- <u>final jobs</u>: For each vertex $v_i\varepsilon N$ and $j\varepsilon\{1,\ldots, w\}$ there is a final job denoted by ${}_{F}J_{i,j}$. Job ${}_{F}J_{i,j}$ has a nonzero task to be executed by the following processors:

a) for $j \geq 2$ job ${}_{F}J_{i,j}$ has a nonzero task to be executed by the set of front processors $\{{}_{F}P_{i,k} \mid k\varepsilon\{w+1-j,\ldots, w-1\}\}$

b) for $j\varepsilon\{1,\ldots, 3r+6\}$ job ${}_{F}J_{i,j}$ has a nonzero task to be executed by the set of front processors $\{{}_{F}P_{i,k} \mid k\varepsilon\{w,\ldots, w+3r+6-j\}\}$

c) for $j \geq 3r+7$ job ${}_{F}J_{i,j}$ has a nonzero task to be executed by the top processor ${}_{T}P_{i}$

d) for $j\varepsilon\{2r+5,\ldots, w-1\}$ job ${}_{F}J_{i,j}$ has a nonzero task to be executed by the set of middle processors $\{{}_{M}P_{i,k} \mid k\varepsilon\{\max\{3r+7-j,1\},\ldots, \min\{r+2, w-j\}\}\}$

e) for $j\varepsilon\{2r+4,\ldots, w-r-3\}$ job ${}_{F}J_{i,j}$ has a nonzero task to be executed by the bottom processor ${}_{B}P_{i}$.

f) for $j\varepsilon\{1,\ldots, w-r-4\}$ job ${}_{F}J_{i,j}$ has a nonzero task to be executed by the set of low processors $\{{}_{L}P_{i,k} \mid k\varepsilon\{\max\{2r+4-j,1\},\ldots, \min\{2r+3, w-r-3-j\}\}\}$

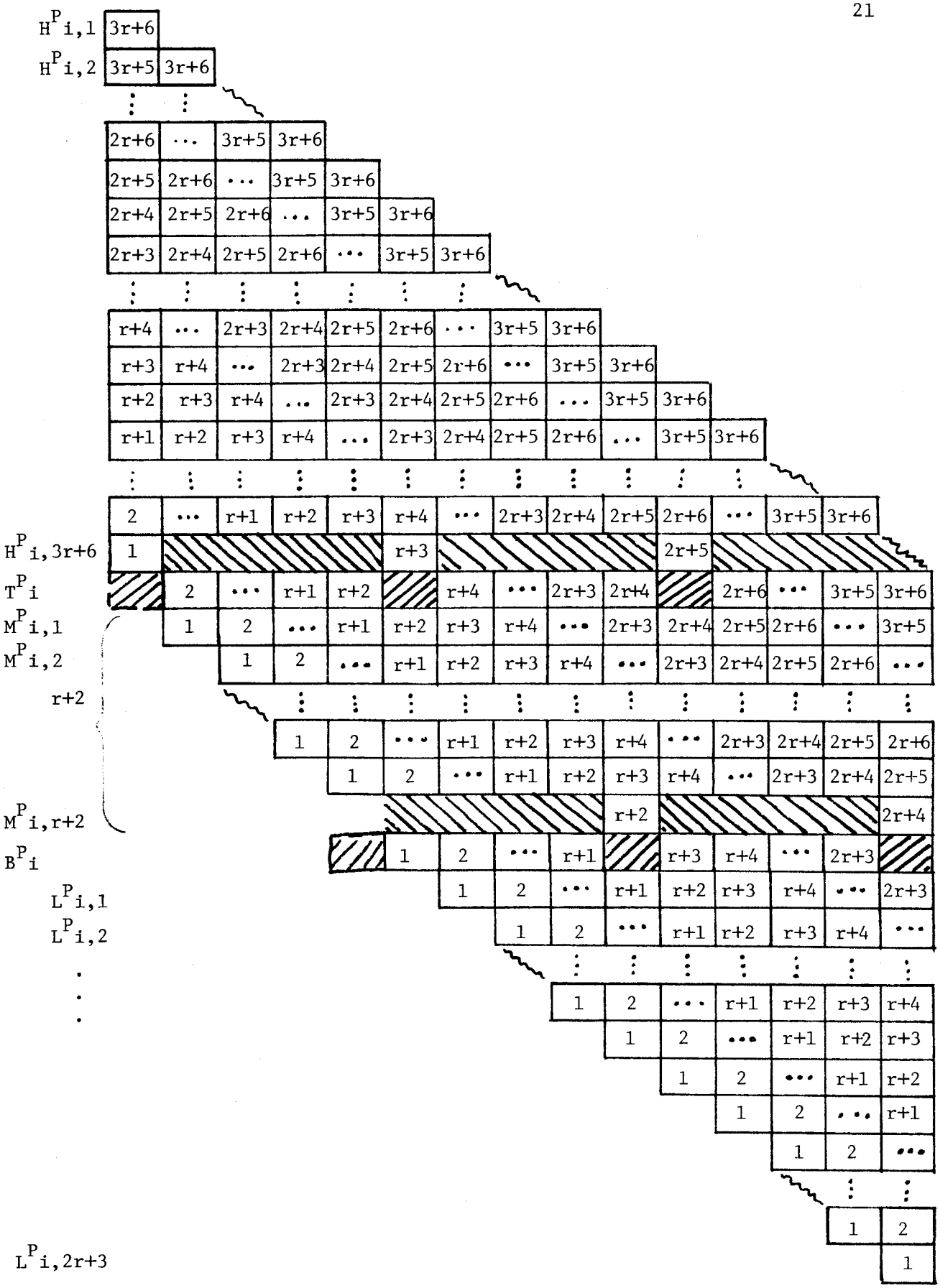

Figure 2: Position jobs $P^J_{i,(\cdot)}$.

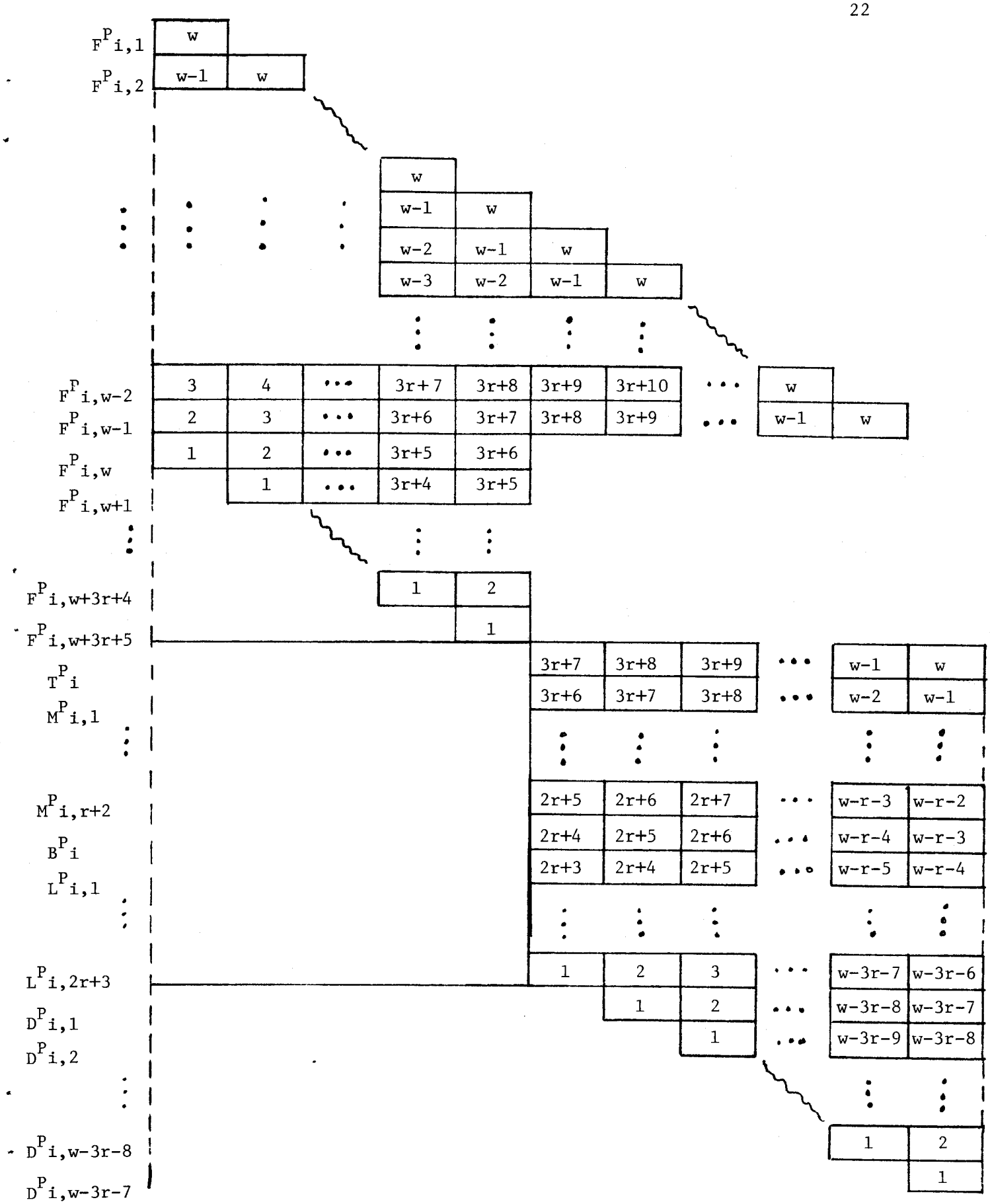

Figure 3: FINAL JOBS $F^J_{i,(\cdot)}$.

g) for $j\varepsilon\{1,\ldots,w-3r-7\}$ job ${}_{F}J_{i,j}$ has a nonzero task to be executed by the set of end processors $\{{}_{D}P_{i,k} | k\varepsilon\{1,\ldots,w-3r-6-j\}$

Figure 3 shows the final jobs ${}_{F}J_{i,(\cdot)}$.

The value for d is $w^2+(3r+8)(3r+6)$. We now state the following two claims which can be easily verified by inspection of figures 2 and 3.

Claim 1: All position jobs have $3r+6$ nonzero tasks.

Claim 2: All final jobs have w nonzero tasks.

We now show that the above flow shop problem has a preeemptive schedule, S, with $mft(S) \leq w^2+(3r+8)(3r+6)$ iff G is three colorable.

a) If G is three colorable then FS has a schedule with $mft(S) \leq w^2+(3r+8)(3r+6)$.

Let us now state a very useful claim which can be easily verified from figures 2 and 3.

Claim 3:

i) All final jobs can be scheduled to complete at time w.

ii) After scheduling all final jobs to complete at time w, all the high, edge, edge-node, top, middle, bottom and low processors are idle from time 0 to time $3r+6$.

iii) After scheduling the set of final jobs to complete at time w, all position jobs can be scheduled to complete at time $3r+6$.

iv) After scheduling all final and position jobs (as in (i) and (iii)), ${}_{T}P_{i}$ has idle time in the intervals $[1,2],[r+2;r+3]$, $[2r+4;2r+5]$; ${}_{B}P_{i}$ is idle in the time interval $[r+1;r+2]$, $[2r+3;2r+4],[3r+5;3r+6]$; and the set of edge and node-edge processors are idle from time 0 to time $3r+6$. □

Since G is three colorable it is possible to partition the set of nodes N into sets S_1, S_2 and S_3 in such a way that if node i and node j belong to S_k then $\{i,j\} \notin E$. Using these sets we partition the set of node and dummy jobs into sets A_1, A_2 and A_3 as follows:

$$A_1 = \{{}_N J_i | v_i \in S_1\} \cup \{{}_D J_{i,1} | v_i \in S_2 \cup S_3\}$$

$$A_2 = \{{}_N J_i | v_i \in S_2\} \cup \{{}_D J_{i,1} | v_i \in S_1\} \cup \{{}_D J_{i,2} | v_i \in S_3\}$$

and $$A_3 = \{{}_N J_i | v_i \in S_3\} \cup \{{}_D J_{i,2} | v_i \in S_1 \cup S_2\}$$

Claim 4: The set of jobs in A_k, $k = 1, 2, 3$ will use processors ${}_T P_i$, ${}_{EN} P_{i,j}$, ${}_E P_j$ and ${}_B P_i$ for no more than one time unit.

Proof: We just prove this for $k = 1$. The proofs for $k = 2$ and 3 are similar and will be omitted. Since all node jobs and dummy jobs have $r+2$ nonzero tasks and since (S_1, S_2, S_3) is a partition of N, it follows that the number of nonzero tasks for the set of jobs in A_1 is $n(r+2)$. From the definition of node and dummy jobs we have that the set of processors used by jobs in A_1, denoted by R_1 is

$$R_1 = \{{}_T P_i | v_i \in S_1\} \cup \{{}_E P_j | \text{edge } e_j \text{ is incident upon a vertex } v_i \text{ in } S_1\} \cup \{{}_{EN} P_{i,j} | e_j \text{ is not incident upon a vertex } v_i \text{ in } S_1\} \cup \{{}_B P_i | v_i \in S_1\} \cup \{{}_T P_i | v_i \in S_2 \cup S_3\} \cup \{{}_{EN} P_{i,j} | v_i \in S_2 \cup S_3 \text{ and } e_j \in E\} \cup \{{}_B P_i | v_i \in S_2 \cup S_3\}$$

Since $S_1 \cup S_2 \cup S_3 = N$, $S_i \cap S_j = \emptyset$ for all $i \neq j$ and no two nodes in S_1 are adjacent, we get

$$R_1 = \{{}_T P_i | v_i \in N\} \cup \{{}_E P_j | \text{edge } e_j \text{ is incident upon a vertex } v_i \text{ in } S_1\} \cup \{{}_{EN} P_{i,j} | (v_i \in S_1 \text{ and } e_j \text{ is not incident upon a vertex } v_i \text{ in } S_1) \text{ or } (v_i \in S_2 \cup S_3) \text{ and } e_j \in E)\} \cup \{{}_B P_i | v_i \in N\}$$

Since the number of processors in R_1 is the same as the number of nonzero tasks, it follows that processor ${}_T P_i$, ${}_{EN} P_{i,j}$, ${}_E P_j$ and ${}_B P_i$ are used for no more than one time unit. □

We now show that the set of jobs in A_k can be scheduled in the time interval $(k-1)(r+2)$ to $k(r+2)$. This will be shown only for $k=1$ (the cases $k=2$ and 3 are similar). The set A_1 consists of n jobs. Since all jobs have their first nonzero task to be executed by ${}_T P_i$; all jobs use different processor ${}_T P_i$ (claim 4); and the processors are free from time 1 to 2 (claim 3 iv) it then follows that the first nonzero task of all jobs in A_1 can be scheduled to complete at time 2. The tasks using processors ${}_{EN} P_{i,j}$ and ${}_E P_j$ can be scheduled from time 2 to $r+1$ because of claim 3 iv and claim 4. The last nonzero task for all jobs in A_1 is to be executed by processor ${}_B P_i$. Since ${}_B P_i$ is free from time $r+1$ to time $r+2$ (claim 3 iv) it follows that all the last tasks for each job can be scheduled to complete at time $r+2$. Hence, the set of jobs in A_1 can be scheduled to start at time 0 and terminate by time $r+2$.

The total contribution to the mft from the jobs in A_1, A_2 and A_3 is $\sum_{k=1}^{3} (k)(r+2)n = 6(r+2)n$. The contribution from the position jobs (claim 3 iii) is $(3r+6)^2 n$ and the contribution from the final jobs (claim 3 i) is $w^2 n$. Hence, $mft(S) = n(w^2 + 6(r+2) + (3r+6)^2)/n = w^2 + (3r+6)(3r+8)$. This completes the proof of part a. □

b) If G is not three colorable then all schedules for flow shop, FS, must have $mft(S) > w^2 + (3r+6)(3r+8)$.

This will be shown by contradiction. Assume there is a schedule S' with $mft(S') \leq w^2 + (3r+6)(3r+8)$.

Before proceeding to the proof we explain how this part is proven. First we will show that if S' exists then there is another schedule S" with no preemptions in which all tasks start and terminate at integer times and $mft(S") \leq mft(S')$. We then proceed to show that if $mft(S") \leq w^2 + (3r+6)(3r+8)$ then in S" no node job, dummy job or position job will be scheduled after time $3r+6$. Since all the position jobs must execute for $3r+6$ time units, it will then follow that these jobs are scheduled in S" as in figure 2. Now, if S" is to have $mft(S") \leq w^2 + (3r+6)(3r+8)$ all the node and dummy jobs terminate by time $3r+6$ and can only use the top processor ${}_{T}P_i$ during the time intervals $[1;2],[r+2;r+3],[2r+4;2r+5]$ and the bottom processors ${}_{B}P_i$ during the time intervals $[r+1;r+2],[2r+3;2r+4],[3r+5;3r+6]$. This implies that all the node jobs and dummy jobs must execute continuously in three disjoint regions which in turn implies that G is three colorable. We now state the proof in detail.

From lemma A.2 in appendix III it follows that if there exists a schedule S' with $mft(S') \leq w^2 + (3r+6)(3r+8)$ then there exists another schedule S" with no preemptions in which all tasks start and terminate at integer times and $mft(S") \leq mft(S')$.

Claim: If $mft(S") \leq w^2 + (3r+6)(3r+8)$ then in S" no node job, dummy job or position job can execute after time $3r+6$.

Proof: We show that if there is a node job, dummy job or position job with finish time $> 3r+6$ then $mft(S") > w^2 + (3r+6)(3r+8)$. Let v be the time at which such a job terminates. Since all node jobs and dummy jobs have execution time requirements of $r+2$ units it must be that their finishing times are at least $r+2$. Also the position jobs have execution time requirements of $3r+6$ units so their finishing

times are at least $3r+6$. Since the late job is a node, dummy or position job it must be that it executes from time $v-1$ to time v in one of the top, middle, bottom or low processors. Since all of these processors are used by the final jobs it then follows that some final jobs will not complete at time w when $v \leq w$.

case 1: $v \geq w + 1$

The total contribution to the mft by the jobs in the shop in as follows:

node and dummy jobs	$\geq$	$(3n-1)(r+2)$
position jobs	$\geq$	$(n(3r+6)-1)(3r+6)$
final jobs	$\geq$	nw^2
late job	$\geq$	$w+1$

Therefore $mft(S'') \geq ((3n-1)(r+2) + (n(3r+6)-1)(3r+6) + nw^2 + w + 1)/n$

Since $w = 3rn + 6n + 4r + 8$ it then follows that

$mft(S'') > w^2 + (3r+6)(3r+8)$

case 2: $v < w + 1$

For this case we have that at least $w - v + 1$ final jobs must complete at time $\geq w+1$ (see figure 2). The total contribution to the mft by the jobs in the shop is as follows:

node and dummy jobs	$\geq$	$(3n-1)(r+2)$
position jobs	$\geq$	$(n(3r+6)-1)(3r+6)$
final jobs	$\geq$	$(w-v+1)(w+1) + (v-1)(w) + (n-1)w^2$
late job	$=$	v

Therefore $mft(S'') \geq ((3n-1)(r+2) + (n(3r+6)-1)(3r+6) + (w-v+1)(w+1) + (v-1)(w) + v)/n$

Since $w = 3rn + 6n + 4r + 8$ it then follows that

$mft(S'') > w^2 + (3r+6)(3r+8)$.

This completes the proof of the claim. □

So, all node jobs, dummy jobs and position jobs should complete by time $3r+6$ in S''. Since all position jobs must execute for $3r+6$ time units, it follows that these jobs are scheduled in S'' as in figure 2. Hence, the top processors ${}_{T}P_i$ will only be free in the time intervals $[1;2][r+2;r+3][2r+4;2r+5]$ and the bottom processors ${}_{B}P_i$ during the time intervals $[r+1;r+2],[2r+3;2r+4],[3r+5;3r+6]$. Since ${}_{N}J_i$, ${}_{D}J_{i,1}$ and ${}_{D}J_{i,2}$ use both processors ${}_{T}P_i$ and ${}_{B}P_i$ and since these jobs must complete by time $3r+6$, it must be that these jobs execute continuouly in the time intervals $(k-1)(r+2)$ to $k(r+2)$ for $k = 1, 2$ and 3. But if ${}_{N}J_i$ & ${}_{N}J_j$ execute in the same time interval it implies that in G, $\{i,j\} \notin E$. Hence, if all node jobs and dummy jobs are scheduled to complete by time $3r+6$ it must be that the set of nodes in G can be partitioned into three disjoint sets such that if two nodes belong to the same set then they are not adjacent. This contradicts that G is not three colorable. So, all schedules must have $mft(S') > w^2 + (3r+6)(3r+8)$ when G is not three colorable.

This concludes the proof of the lemma. □

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 and 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 it trival. 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 in 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 in 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 an 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 theorem 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 in 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 section IV we showed that the problem of contructing 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 algorithms 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 comment and suggestions while preparing this manuscript.

REFERENCES

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

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

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

[J] Johnson, S. M., "Optimal Two-and-Three-State Production Schedule," Naval Research and Logistics Quarterly, 1, 1 (1954)

[G] Gonzalez, T., "A note on Open Shop Preemptive Schedules," Technical Report #214, Department of Computer Science, The Pennsylvania State University, December 1976. (To appear IEEE TC)

[GJ] Garey, M. R. and D. S. Johnson, "Strong NP-Completeness Results: Motivations, Examples, and Implicants," JACM 25, 3, 499-508(1978).

[GJSe] Garey, M. R.., D. S. Johnson, R. Sethi, "Complexity of Flow Shop and Job Shop Scheduling," Math. Opns. Res. 1, 117-129 (1976).

[GJSt] Garey, M. R., Johnson, D. S., and Stockmeyer, L., "Some Simplified NP-Complete Graph Problems," Theoret. Comptr. Sci. 1, 237-267(1976).

[GG] Gilmore, P. and R. Gomory, "Sequencing A One State-variable Machine: A Solvable Case of the Travelling Salesman Problem," Op. Res., 12 1964, p. 655-679.

[GS1] Gonzalez, T., and Sahni, S. "Open Shop Scheduling to Minimize Finish Time," JACM, 23, 665-679 (1976).

[GS2] Gonzalez, T., and Sahni, S. "Flowshop and Jobshop Schedules: Complexity and Approximation," Oper. Res. 26, 1, 36-52 (1978).

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

[K2] Karp, R. M., "On the Computational Complexity of Combinational Problems," Networks 5, 45-68 (1975).

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

[LR] Lenstra, J. K., A. H. G. Rinnooy Kan, "Computational complexity of discrete optimization problems," Ann. Discrete Math., (to appear).

[LRB] Lenstra, J. K., A. H. G. Rinnooy Kan and P. Brucker, "Computational Complexity of Machine Scheduling Problems," Ann. Discrete Math., 1, 1977, p. 343-362.

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

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

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 4 α (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 ≤ 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 4.

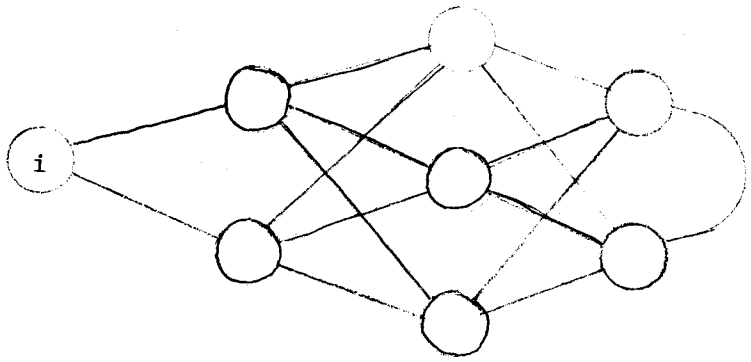

Figure 4 New edges for vertex i.

It should be clear that G'' will now have all vertices of degree four and G'' is 3-colorable iff G is 3-colorable. This completes the proof of the theorem.

□

APPENDIX II

Let T be an open shop problem in which all nonzero tasks have unit execution time. Let m be the number of machines and n the number of tasks. Let L_i be the number of nonzero tasks in job i and let M_j be the number of nonzero tasks to be executed by machine j.

Gonzalez and Sahni [GS1] have shown that for an open shop there is a preemptive schedule of length $\max_{i,j} \{L_i, M_j\}$. The way their algorithm proceeds will not produce preemptions when all the nonzero tasks are of equal length. Hence, an optimal finish time schedule can be obtained with no preemptions for this case. We will just state this result and refer the reader to [GS1].

Lemma A.1: An OFT nonpreemptive schedule for an open shop problem in which all nonzero tasks have equal length can be obtained by an efficient algorithm, and the length of the schedule in $\max_{i,j} \{L_i, M_j\}$.

APPENDIX III

Let T be a flow shop problem in which all nonzero tasks have unit execution time requirements. Let S be a preemptive schedule for T.

Lemma A.2: Let T and S be as defined above. S can be transformed to a schedule S' for T with the following properties:

1) S' has no preemptions

2) $ft(S') \leq ft(S)$

3) $mft(S') \leq mft(S)$

Proof: Let us assume that up to time α (integer) we have a schedule with no preemptions, and all tasks start and finish at integer times. We now show that a schedule up to time $\alpha+1$ can be constructed with no preemptions, in which all tasks start and finish at integer times. Let us now transform the schedule for machine j in such a way that from time α to time $\alpha+1$ either there is idle time or only one task executes. If no tasks have been assigned from α to $\alpha+1$ then the machine will be idle from time α to $\alpha+1$. Otherwise let $a_1,\ldots,a_k$ be the set of tasks that execute in this interval. Let $f(a_i)$ be the latest time at which task a_i executes on this machine. Assume without loss of generality that $f(a_i) < f(a_{i+1})$ for $1 \le i < k$. The schedule is now modified as shown in figure 5.

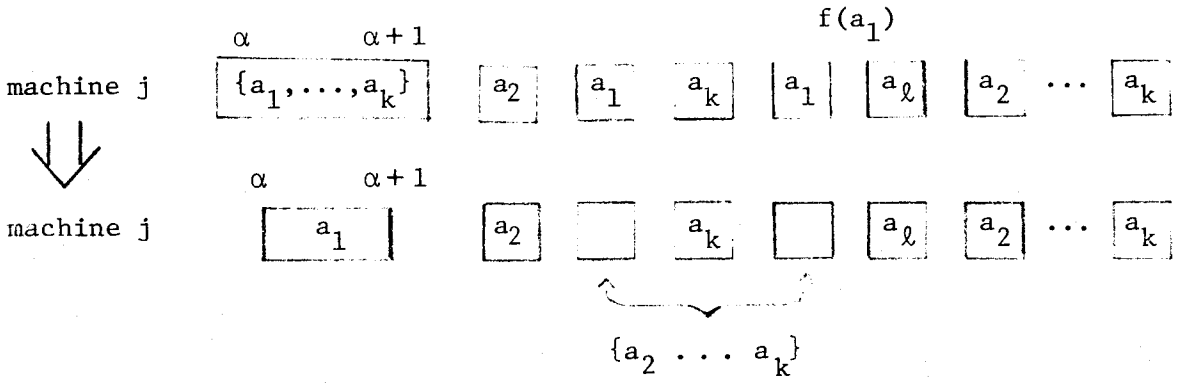

Figure 5: transformation

a_1 is scheduled from time α to $\alpha+1$ and the tasks $a_2 \ldots a_k$ which were scheduled in the interval $[\alpha,\alpha+1)$ are scheduled where a_1 was scheduled after time $\alpha+1$. Clearly the ft and mft will not increase. The only question that remains is if some precedence constraints are violated. Since $f(a_i)$ is not increased for any task then no precedence constraints are violated when considering tasks to be executed by machines $j+1,\ldots,m$. Precedence constraints on machines $1,\ldots,j-1$ are not violated since tasks $a_1 \ldots a_k$ were not scheduled on these machines after time α

(note that if a_ℓ is scheduled after time α on machine $j'' \leq j-1$ then it cannot be scheduled on machine j from time α to $\alpha+1$ since task a_ℓ would not complete before time $\alpha+1$ on machine j'')

This same transformation in applied to the remaining machines from time α to time $\alpha+1$. This completes the proof of the lemma. □

We should note that this same result also holds for job shop problems.