SIAM J. COMPUT.
Vol. 4, No. 4, December 1975

COMPLEXITY RESULTS FOR MULTIPROCESSOR SCHEDULING UNDER RESOURCE CONSTRAINTS*

M. R. GAREY AND D. S. JOHNSON†

Abstract. We examine the computational complexity of scheduling problems associated with a certain abstract model of a multiprocessing system. The essential elements of the model are a finite number of identical processors, a finite set of tasks to be executed, a partial order constraining the sequence in which tasks may be executed, a finite set of limited resources, and, for each task, the time required for its execution and the amount of each resource which it requires. We focus on the complexity of algorithms for determining a schedule which satisfies the partial order and resource usage constraints and which completes all required processing before a given fixed deadline. For certain special cases, it is possible to give such a scheduling algorithm which runs in low order polynomial time. However, the main results of this paper imply that almost all cases of this scheduling problem, even with only one resource, are NP-complete and hence are as difficult as the notorious traveling salesman problem.

Key words. complexity of algorithms, NP-complete problems, scheduling theory

1. Introduction. In recent years, there has been considerable interest in scheduling problems associated with a certain abstract model of a multiprocessing system (see [6] for a survey). Much effort has been directed toward obtaining efficient algorithms for scheduling a given set of tasks to achieve the least possible finishing time. In a number of important special cases [7], [4], [9], [1], such efficient optimization algorithms have been obtained. In contrast, Ullman [10] has shown that certain other cases belong to the class of NP-complete problems, which is tantamount to proving that they are computationally intractable. In this paper, we consider the augmented multiprocessing model of [5], which allows for the possibility that certain tasks may require the use of various limited resources during their execution, and examine how such constraints affect the computational complexity of the associated scheduling problems.

We first describe the augmented multiprocessing model. Its three main components are *processors*, *resources* and *tasks*. The processors are all identical, each able to execute at most one task at a time, and there are at most a finite number of them. The set $\mathscr{R} = \{R_1, R_2, \cdots, R_r\}$ of resources is also finite and, for each resource R_j, there is a bound B_j which gives the total amount of that resource available at any given time. The tasks form a third finite set $\mathscr{T} = \{T_1, T_2, \cdots, T_m\}$, the elements of which are to be executed by the processors subject to a number of constraints.

First, associated with each task T_i is a positive task time τ_i which is the time required for execution of T_i. If a processor begins executing T_i at time t, it will complete the execution of T_i at time $t + \tau_i$, and until then can execute no other tasks.

Second, $\mathscr{T}$ is partially ordered by a binary relation $\prec$ which must be respected in the execution of $\mathscr{T}$ as follows: if $T_i \prec T_j$, then the execution of T_i must be completed before the execution of T_j can begin. It is frequently convenient to describe the partial order by a directed graph G with node set $\mathscr{T}$ and a directed edge $\langle T_i, T_j \rangle$ if and only if $T_i \prec T_j$.

* Received by the editors July 30, 1974.

† Bell Laboratories, Murray Hill, New Jersey 07974.

Finally, for each resource R_j and task T_i, there is a nonnegative resource requirement $R_j(T_i) \leqq B_j$ which is the amount of resource R_j required by T_i at all times during its execution. The execution of tasks is constrained by the requirement that no subset of tasks can be executed simultaneously if the sum of their requirements for any resource R_j exceeds B_j, the total amount of that resource available.

For the purposes of this paper, we shall assume that all τ_i, B_j and $R_j(T_i)$ are integers. This entails no loss of generality for the type of results we present, since it is equivalent to permitting arbitrary rational values.

A particular *input* for the general scheduling problem consists of the following information: a number n of processors; a set $\mathscr{R} = \{R_1, R_2, \cdots, R_r\}$ of resources, together with a bound B_j for each R_j; a set $\mathscr{T} = \{T_1, T_2, \cdots, T_m\}$ of tasks, together with a partial order $\prec$ on $\mathscr{T}$ and, for each T_i, a corresponding task time τ_i and resource requirements $R_1(T_i)$ through $R_r(T_i)$; a positive integer *deadline* D.

A function $f: \mathscr{T} \to \{0, 1, 2, \cdots, D-1\}$ is a *valid schedule* for such an input if it obeys the following four conditions:

(i) for each $T_i \in \mathscr{T}$, $f(T_i) + \tau_i \leqq D$;

(ii) for each $T_i, T_j \in \mathscr{T}$, if $T_i \prec T_j$, then $f(T_i) + \tau_i \leqq f(T_j)$;

(iii) for each integer t, $0 \leqq t < D$, the set

$$E_f(t) = \{T_i \in \mathscr{T} : f(T_i) \leqq t < f(T_i) + \tau_i\}$$

satisfies $|E_f(t)| \leqq n$, and

(iv) for each integer t, $0 \leqq t < D$, and each j, $1 \leqq j \leqq r$,

$$\sum_{T_i \in E_f(t)} R_j(T_i) \leqq B_j.$$

The set $E_f(t)$ defined in (iii) is called the set of tasks being executed at time t under schedule f.

Some additional terminology will be useful to describe certain special properties which may be satisfied by inputs. The partial order $\prec$ is said to be a *forest* if whenever $T_i \prec T_k$ and $T_j \prec T_k$, we have either $T_i \prec T_j$ or $T_j \prec T_i$. An input will be said to have the *saturated processor* property if $\sum_{i=1}^{m} \tau_i = n \cdot D$. Any valid schedule f for an input with this property must have $|E_f(t)| = n$ for each integer t, $0 \leqq t < D$, i.e., no processor can be idle until the deadline D is reached. An input has a *saturated resource* R_j if $\sum_{i=1}^{m} \tau_i R_j(T_i) = D \cdot B_j$. Any valid schedule f for an input with resource R_j saturated must satisfy $\sum_{T_i \in E_f(t)} R_j(T_i) = B_j$ for each integer t, $0 \leqq t < D$, i.e., the full amount of resource R_j must be in continued use until the deadline D is reached. We shall be dealing mainly with inputs for which all $\tau_i = 1$, in which case the saturated processor property can be written $|\mathscr{T}| = n \cdot D$, and resource R_j will be saturated if $\sum_{T \in \mathscr{T}} R_j(T) = D \cdot B_j$.

In this paper we shall be concerned with algorithms which, given any input I, are capable of answering the question, "Does there exist a valid schedule for I?". Notice that we phrase the question in such a way that the answer is always "yes" or "no". Such phrasing is convenient to use in proving computational complexity results and will not seriously limit the applicability of the results. The related problems of actually generating a valid schedule for a given input or (treating

the deadline D as a variable) generating a valid schedule which achieves the minimum possible deadline D have computational complexity within a polynomial of that for the simple existence problem.

In particular, we shall be concerned with algorithms which apply only to certain restricted classes of possible inputs. It will be convenient to denote such a *subproblem* by MS$[\varphi_1, \varphi_2, \cdots, \varphi_k]$, which represents the subproblem having input domain restricted to only those inputs satisfying all of the constraints φ_1 through φ_k. For instance, MS$[n \leqq 2$, each $\tau_i = 1]$ has input domain consisting only of those inputs with at most two processors and all task times equal to one. Occasionally we may include a vacuous restriction for emphasis. The subproblem MS$[n \leqq 2$, each $\tau_i = 1, r$ arbitrary$]$ is identical with the above but may be specified this way when compared to MS$[n \leqq 2$, each $\tau_i = 1, r = k]$ for some fixed k.

For a variety of subproblems, we will either give an algorithm for the subproblem which operates in time bounded by a polynomial in the length of the input or prove that the subproblem belongs to the class of "NP-complete" problems. There is a significant difference between these two possibilities, since, as a result of work by Cook [2] and Karp [8], it is widely believed that both possibilities cannot hold for the same problem. This belief is motivated by the knowledge that either all or none of the NP-complete problems can be solved with polynomial-time algorithms, along with the fact that this class contains a number of extensively studied problems, such as set covering and the "traveling salesman" problem, for which no such algorithm has ever been found. The reader who is unfamiliar with the terminology related to NP-complete problems, or who would like his belief in their intractability reinforced, is referred to [2] and [8] for a complete discussion. (Note that the term "NP-complete" is synonymous with "polynomial complete".) For the purposes of this paper, we shall use the following informal definitions.

A *problem* $P = (\mathscr{D}, \pi)$ consists of an *input domain* $\mathscr{D}$ and a *property* π. An algorithm for $P = (\mathscr{D}, \pi)$ is an algorithm which for any input $I \in \mathscr{D}$ determines whether or not property π holds for I (briefly, whether or not $\pi(I)$ holds). Observe that we have described the general scheduling problem and its subproblems in this format, all having the same property π = "there exists a valid schedule for input I".

For any two problems $P = (\mathscr{D}, \pi)$ and $P' = (\mathscr{D}', \pi')$, we say that P is *polynomially reducible* to P' if there exists a total function $g: \mathscr{D} \to \mathscr{D}'$ such that (i) g can be performed in polynomial time by a deterministic Turing machine, and (ii) for any $I \in \mathscr{D}$, $\pi(I)$ holds if and only if $\pi'(g(I))$ holds. If P is polynomially reducible to P', we write $P \propto P'$.

The class of problems NP consists of all problems which can be solved with a polynomial-time-bounded *non*deterministic Turing machine. A problem P is said to be NP-*complete* if both $P \in$ NP and, for every $P' \in$ NP, $P' \propto P$. An immediate consequence of the transitivity of the reducibility relation $\propto$ is that, in order to prove $P \in$ NP is NP-complete, one need only exhibit a known NP-complete problem P' such that $P' \propto P$. This is the method that will be used in our proofs, where we leave to the reader the trivial observation that $P \in$ NP and the straightforward verification that the mapping g can be executed in polynomial time.

We now briefly outline the contents of the paper. In § 2 we examine subproblems with $n = 2$ processors and show that MS[$n = 2$, $r = 1$, $\prec$ a forest, each $\tau_i = 1$] is NP-complete, while a polynomial time algorithm can be given for MS[$n = 2$, r arbitrary, $\prec$ empty, each $\tau_i = 1$]. In § 3, we examine subproblems with $n \geqq 3$ and derive, by a series of reductions, the NP-completeness of MS[$n = 3$, $r = 1$, $\prec$ empty, each $\tau_i = 1$]. In the final section, we discuss a number of different senses in which these results are "best possible" and state some additional results and open problems.

2. Two-processor subproblems. In this section we discuss special cases of the general scheduling problem for which the number of processors is restricted to $n = 2$. Quite a bit is known about such subproblems when the set of resources is required to be empty. For example, the following two subproblems are NP-complete [10]:

$$\text{MS}[n = 2, r = 0, \prec \text{ empty}, \tau_i \text{ arbitrary integers}];$$

$$\text{MS}[n = 2, r = 0, \prec \text{ arbitrary, each } \tau_i \in \{1, 2\}].$$

On the other hand, [4], [9] and [1] give polynomial time algorithms for MS[$n = 2$, $r = 0$, $\prec$ arbitrary, each $\tau_i = 1$]. We shall focus on subproblems for which $r > 0$.

We first observe that a polynomial-time scheduling algorithm can be given for the case MS[$n = 2$, r arbitrary, $\prec$ empty, each $\tau_i = 1$]. Given any input for this subproblem, one can construct an m-node graph G, having each node labeled by a distinct task, with an edge joining T_i to T_j if and only if

$$R_k(T_i) + R_k(T_j) \leqq B_k$$

for all k, $1 \leqq k \leqq r$. Thus tasks T_i and T_j can be executed at the same time if and only if there is an edge joining the corresponding nodes. One then applies the maximal matching algorithm of Edmonds [3] to obtain a maximal cardinality set E of edges from G such that no two edges share a common endpoint. It is not difficult to see that a valid schedule exists for this input if and only if $D \geqq m - |E|$. Since we can construct G in time proportional to $r \cdot m^2 \cdot \log(\max_j B_j)$ and since Edmonds' matching algorithm requires only time polynomial in the number m of graph nodes, the entire algorithm just described runs in polynomial time.

We now show, however, that if we widen the class of inputs slightly by permitting the partial order $\prec$ to be nonempty but no more complex than a forest, then the resulting subproblem is NP-complete, even if restricted to $r = 1$. This problem will be shown to be NP-complete by reducing the following NP-complete problem to it.

NODE COVER [8].

Input: Graph $G = (N, A)$, positive integer k.

Property: There exists a node cover of size k for G, i.e., a subset S of the set N of nodes, with $|S| = k$, such that S contains at least one endpoint from every edge in A.

THEOREM 2.1. NODE COVER $\propto$ MS[$n = 2, r = 1, \prec$ a forest, each $\tau_i = 1$].

Proof. Given a graph $G = (N, A)$ and a positive integer k, we construct a corresponding scheduling input I as follows: Let $N = \{N_1, N_2, \cdots, N_p\}$ and

$A = \{A_1, A_2, \cdots, A_q\}$, where $p = |N|$ and $q = |A|$. The input I is specified by

$$n = 2;$$

$$\mathcal{T} = \{T_i : 1 \leqq i \leqq p + 2q\} \cup \{V_i : 1 \leqq i \leqq p\} \cup \{E_i, \bar{E}_i : 1 \leqq i \leqq q\};$$

all task times equal 1; partial order is defined by

$$T_i \prec T_{i+1}, \qquad 1 \leqq i < p + 2q;$$

$$V_i \prec E_l \quad \text{whenever } A_l = \{N_i, N_j\} \in A \text{ and } j > i;$$

$$V_i \prec \bar{E}_l \quad \text{whenever } A_l = \{N_i, N_j\} \in A \text{ and } j < i;$$

single resource R_1 with bound $B_1 = 2q$;

$$R_1(T_i) = q, \qquad 1 \leqq i \leqq k \quad \text{or} \quad q + k < i \leqq p + q;$$

$$R_1(T_{k+i}) = R_1(T_{p+q+i}) = q + i, \qquad 1 \leqq i \leqq q;$$

$$R_1(V_i) = q, \qquad 1 \leqq i \leqq p;$$

$$R_1(E_i) = R_1(\bar{E}_i) = q - i, \qquad 1 \leqq i \leqq q;$$

deadline $D = p + 2q$.

It is easy to verify that input I is in the required domain. Furthermore, observe that it satisfies the saturated processor property and that R_1 is a saturated resource.

Intuitively, the V_i tasks represent the nodes of G, and the E_l and $\bar{E}_l$ tasks represent the arcs, with the pair E_l and $\bar{E}_l$ both representing arc A_l, one for each endpoint. The T_i tasks form a chain designed to be a backbone for the entire schedule, since in order to achieve the deadline these must be executed in sequence, with each task T_i executed starting at time $i - 1$. Thus during each time unit there will be a T_i task and one other task. Because of the saturated resource property, the resource requirement of each task T_i puts strict limits on the possibilities for the concurrently executed task. Thus a pattern is imposed on any valid schedule. The first k time units must all be filled with node tasks, and the next q time units must be filled with arc tasks, one from each pair $\{E_l, \bar{E}_l\}$. Because of the precedence constraints, neither E_l nor $\bar{E}_l$ can occur in this section unless a node task representing one of the endpoints of arc A_l is present in the first section. Thus we can have a valid schedule if and only if the k node tasks in the first section form the desired node cover.

With this intuition, we now prove that G has a node cover S with $|S| = k$ if and only if there exists a valid schedule for I.

First, suppose $S = \{s_1, s_2, \cdots, s_k\}$ is a node cover for G, and let $U = N - S = \{u_1, u_2, \cdots, u_{p-k}\}$. Then consider the function f, defined below:

$$f(T_i) = i - 1, \qquad 1 \leqq i \leqq p + 2q;$$

$$f(V_j) = i - 1 \quad \text{if } N_j = s_i \in S;$$

$$f(V_j) = q + k + i - 1 \quad \text{if } N_j = u_i \in U;$$

$$f(E_l) = k + l - 1 \quad \text{and} \quad f(\bar{E}_l) = p + q + l - 1$$

$$\text{if } A_l = \{N_i, N_j\}, i < j, \text{ and } N_i = s_h \in S;$$

$$f(E_l) = p + q + l - 1 \quad \text{and} \quad f(\bar{E}_l) = k + l - 1$$
$$\text{if } A_l = \{N_i, N_j\},\ i < j, \text{ and } N_i = u_h \in U.$$

One can easily check that f is a valid schedule for the input I.

Conversely, suppose we are given a valid schedule f for I. Observe first that, due to the partial order constraints and deadline $D = p + 2q$, we must have $f(T_i) = i - 1$ for $1 \leqq i \leqq p + 2q$. Thus the tasks of this type form a backbone for the schedule, and, due to the saturated resource R_1, each of the remaining tasks can be executed only at certain times. To be specific, we must have

$$\{f(V_j): 1 \leqq j \leqq p\} = \{t: 0 \leqq t \leqq k\} \cup \{t: q + k \leqq t < q + p\} \tag{2.1a}$$

and, for $1 \leqq l \leqq q$,

$$\{f(E_l), f(\bar{E}_l)\} = \{k + l - 1, p + q + l - 1\}. \tag{2.1b}$$

We claim that $S = \{N_i: f(V_i) < k\}$ forms a node cover of size k for G. By (2.1a) and the saturated processor property, we immediately have $|S| = k$. Consider any edge $A_l = \{N_i, N_j\} \in A$, where $i < j$. By (2.1b), either $f(E_l) = k + l - 1$ or $f(\bar{E}_l) = k + l - 1$. In the first case, since $V_i \prec E_l$ and $k + l - 1 < q + k$, we must have $f(V_i) < k$ and hence $N_i \in S$. In the second case, since $V_j \prec \bar{E}_l$, we similarly obtain $N_j \in S$. Thus in either case, we find that at least one endpoint of A_l belongs to S. Since A_l was an arbitrary edge from A, S must be a node cover.

Hence G has a node cover of size k if and only if the associated scheduling input I has a valid schedule, and the theorem follows. □

3. Three-processor subproblems. In this section we turn to special cases of the general scheduling problem for which the number n of processors is permitted to be larger than 2. Again, subproblems for which the set of resources is required to be empty have been studied previously. In [7] a polynomial-time algorithm is given for MS[n arbitrary, $r = 0$, $\prec$ a forest, each $\tau_i = 1$]. In contrast, Ullman [10] has shown that MS[n arbitrary, $r = 0$, $\prec$ arbitrary, each $\tau_i = 1$] is NP-complete. However, it is not known whether there is any fixed k for which MS[$n \leqq k$, $r = 0$, $\prec$ arbitrary, each $\tau_i = 1$] is NP-complete.

The principal result of this section is that if we substitute a single resource for the arbitrary partial order in the last problem above, requiring $\prec$ to be empty, it then is NP-complete for k as small as 3. Specifically, we show that MS[$n = 3$, $r = 1$, $\prec$ empty, each $\tau_i = 1$] is NP-complete.

We arrive at this result via a series of lemmas about scheduling problems obeying the further restriction that all inputs have the saturated processor property and all their resources saturated. For convenience, we use the abbreviation P[k, j] for MS[$n = k$, $r = j$, $\prec$ empty, each $\tau_i = 1$, saturated processors, all resources saturated]. Since the domain for P[3, 1] is a subset of the domain for MS[$n = 3$, $r = 1$, $\prec$ empty, each $\tau_i = 1$], it will suffice to show that P[3, 1] is NP-complete.

Our first lemma is as follows.

LEMMA 3.1. *For all integers $k \geqq 1$, $j \geqq 2$, P[k, j] $\propto$ P[k, 1].*

Proof. We actually show P[k, j] $\propto$ P[$k, j - 1$]. The result then follows by induction from the transitivity of $\propto$. Let I be an input for P[k, j], with tasks T_i,

$1 \leqq i \leqq m$, resources R_l, $1 \leqq l \leqq j$, bounds B_l, $1 \leqq l \leqq j$, and deadline D. (We omit reference to those items in the input which must be the same for all inputs to the problem under consideration, such as $\prec$ empty, each $\tau_i = 1$, and $n = k$.) The corresponding input I' for P$[k, j-1]$ will have

$$\mathscr{T}' = \{T_i' : 1 \leqq i \leqq m\};$$

$$\mathscr{R}' = \{R_l' : 1 \leqq l \leqq j-1\};$$

$$B_l' = B_l, \qquad 2 \leqq l \leqq j-1;$$

$$B_1' = B_1 + B_j(kB_1 + 1);$$

$$R_l'(T_i') = R_l(T_i), \qquad 2 \leqq l \leqq j-1, \quad 1 \leqq i \leqq m;$$

$$R_1'(T_i') = R_1(T_i) + (k \cdot B_1 + 1) \cdot R_j(T_i), \qquad 1 \leqq i \leqq m;$$

and deadline $D' = D$.

Intuitively, this merely encodes resources R_1 and R_j into the single resource R_1'. The multiplier $(kB_1 + 1)$ on R_j used in this encoding is chosen to be large enough so that, for any k or fewer tasks which might be executed simultaneously, their total usage of resource R_1' uniquely determines the corresponding usage of both R_1 and R_j. This causes any valid schedule for I' to be, in effect, a simulation of a valid schedule for I.

Observe first that since I has the saturated processor property, $|\mathscr{T}'| = |\mathscr{T}|$ and $D' = D$, it follows that I' also has the saturated processor property. Similarly, we can see that R_2' through R_{j-1}' are saturated in I' because the corresponding resources are saturated in I. Finally observe that for R_1'

$$\sum_{i=1}^{m} R_1'(T_i') = \sum_{i=1}^{m} (R_1(T_i) + (k \cdot B_1 + 1) \cdot R_j(T_i))$$

$$= D \cdot B_1 + (k \cdot B_1 + 1) \cdot D \cdot B_j = D' \cdot B_1'$$

since R_1 and R_j are saturated in I. Thus R_1' is saturated in I' and I' is in the input domain for P$[k, j-1]$.

To complete the proof, we must show that I' has a valid schedule if and only if I does. Suppose f is such a schedule for I. Define f' by $f'(T_i') = f(T_i)$. Clearly f' has the proper domain and range and obeys all requirements of the definition of valid schedule, except possibly condition (iv) for R_1', since f is a valid schedule for I. Since R_1 and R_j are saturated for I, we have for all t, $0 \leqq t \leqq D-1$,

$$\sum_{T_i' \in E_{f'}(t)} R_1'(T_i') = \sum_{T_i \in E_f(t)} (R_1(T_i) + (k \cdot B_1 + 1) \cdot R_j(T_i))$$

$$= B_1 + (k \cdot B_1 + 1) \cdot B_j = B_1',$$

where we recall that $E_f(t) = \{T \in \mathscr{T} : f(T) = t\}$. Thus (iv) is satisfied for R_1' also and f' is a valid schedule for I'.

Conversely, suppose f' is a valid schedule for I' and define f by $f(T_i) = f'(T_i')$. The only way f could fail to be a valid schedule for I would be for resource constraint (iv) to be violated for R_1 or R_j. We show this to be impossible.

Since R'_1 is saturated for I' and f' is a valid schedule, we know that for all t, $0 \leqq t \leqq D - 1$,

$$\sum_{T_i \in E_{f'}(t)} (R_1(T_i) + (k \cdot B_1 + 1) \cdot R_j(T_i)) = B_1 + (k \cdot B_1 + 1) \cdot B_j.$$

If for any t we had $\sum_{T_i \in E_{f'}(t)} R_j(T_i) > B_j$, then since B_j and all the $R_j(T_i)$ are integers, we would have

$$\begin{aligned} \sum_{T_i \in E_{f'}(t)} R_1(T_i) &\leqq B_1 + (k \cdot B_1 + 1) \cdot B_j - (k \cdot B_1 + 1) \cdot (B_j + 1) \\ &\leqq B_1 - k \cdot B_1 - 1 < 0, \end{aligned}$$

which is impossible since all resource requirements are nonnegative. Similarly, if $\sum_{T_i \in E_{f'}(t)} R_j(T_i) < B_j$, we would have

$$\begin{aligned} \sum_{T_i \in E_{f'}(t)} R_1(T_i) &\geqq B_1 + (k \cdot B_1 + 1) \cdot B_j - (k \cdot B_1 + 1) \cdot (B_j - 1) \\ &\geqq B_1 + k \cdot B_1 + 1 > k \cdot B_1, \end{aligned}$$

which is impossible since each $R_1(T_i) \leqq B_1$ and $|E_{f'}(t)| = k$. Thus we must have $\sum_{T_i \in E_{f'}(t)} R_j(T_i) = B_j$ for all t, $0 \leqq t \leqq D - 1$, which in turn implies that $\sum_{T_i \in E_{f'}(t)} R_1(T_i) = B_1$ for all required t. That is, neither resource constraint can be violated, and f' is a valid schedule.

Thus I' has a valid schedule if and only if I does, and the desired reduction has been demonstrated. □

LEMMA 3.2. *For any integer* $k \geqq 1$, $\mathrm{P}[k, 1] \propto \mathrm{P}[k + 1, 1]$.

Proof. Suppose I is an input for $\mathrm{P}[k, 1]$ with tasks T_i, $1 \leqq i \leqq m$, resource R_1 with bound B_1, and deadline D. The corresponding input I' for $\mathrm{P}[k + 1, 1]$ will have

$$\begin{aligned} &\mathscr{T}' = \{T'_i : 1 \leqq i \leqq m\} \cup \{S_i : 1 \leqq i \leqq D\}; \\ &\mathscr{R}' = \{R'_1\}; \qquad B'_1 = 3 \cdot B_1; \\ &R'_1(T'_i) = R_1(T_i), \qquad 1 \leqq i \leqq m; \\ &R'_1(S_i) = 2 \cdot B_1, \qquad 1 \leqq i \leqq D; \end{aligned}$$

and deadline $D' = D$.

The basic idea here is that the resource requirements of the S_i-tasks insure that no two of them can be executed simultaneously. Since there are D such tasks, any valid schedule must execute exactly one of them during each time unit, which has the effect of using up the extra processor.

The reader may verify easily that I' has the saturated processor property and saturated resource R'_1 and hence is in the input domain for $\mathrm{P}[k + 1, 1]$. It is also easy to see that if f is a valid schedule for I, then f' defined by

$$\begin{aligned} &f'(T'_i) = f(T_i), \qquad 1 \leqq i \leqq m; \\ &f'(S_i) = i - 1, \qquad 1 \leqq i \leqq D, \end{aligned}$$

is a valid schedule for I'.

Conversely, suppose that f' is any valid schedule for I' and define $f:\mathscr{T} \to \{0, 1, \cdots, D-1\}$ by $f(T_i) = f'(T_i')$. We shall show that f is a valid schedule for I.

Clearly f has the proper domain and range and obeys properties (i) and (ii) of the definition of valid schedule. For (iii) and (iv), we first observe that each $E_{f'}(t)$, $0 \leqq t \leqq D-1$, contains exactly one S_i task since $R_1'(S_i) > \frac{1}{2}B_1'$ and there are D such tasks. Thus

$$|E_f(t)| = |E_{f'}(t) \cap \{T_i' : 1 \leqq i \leqq m\}| = k + 1 - 1 = k,$$

and property (iii) is satisfied. Furthermore, since R_1' is saturated in I', we have for each t, $0 \leqq t \leqq D-1$,

$$\sum_{T_i \in E_f(t)} R_1(T_i) = B_1' - 2B_1 = B_1,$$

so f also satisfies (iv) and hence is a valid schedule for I.

Thus I' has a valid schedule if and only if I does and the desired reduction has been demonstrated. □

LEMMA 3.3. *For each integer* $k \geqq 2$, $\mathrm{P}[2k, 1] \propto \mathrm{P}[k+1, 1]$.

Proof. We actually show that $\mathrm{P}[2k, 1] \propto \mathrm{P}[k+1, 2]$, and the desired result follows from that by an application of Lemma 3.1. Let I be an input for $\mathrm{P}[2k, 1]$ with tasks T_i, $1 \leqq i \leqq m$, resource R_1 with bound B_1, and deadline D. Our construction of the corresponding input I' for $\mathrm{P}[k+1, 2]$ is somewhat more complicated than the preceding constructions. Define

$$\mathscr{A} = \left\{ A \subseteq \{1, 2, \cdots, m\} : |A| = k \text{ and } \sum_{j \in A} R_1(T_j) \leqq B_1 \right\},$$

with $M = |\mathscr{A}|$. (Observe that $M \leqq m^k$.) In addition, let

$$T = \{T_i' : 1 \leqq i \leqq m\};$$
$$F = \{F_{i,j} : 1 \leqq i \leqq M - D, 1 \leqq j \leqq k-1\};$$
$$U = \{U_A : A \in \mathscr{A}\}; \qquad V = \{V_A : A \in \mathscr{A}\}.$$

The input I' for $\mathrm{P}[k+1, 2]$ corresponding to I has

$$\mathscr{T}' = T \cup F \cup U \cup V;$$
$$\mathscr{R}' = \{R_1', R_2'\};$$
$$B_1' = B_1; \qquad B_2' = 2k;$$
$$R_1'(T_i') = R_1(T_i) \quad \text{and} \quad R_2'(T_i') = 1 \qquad \text{for } T_i' \in T;$$
$$R_1'(F_{i,j}) = R_2'(F_{i,j}) = 0 \qquad \text{for } F_{i,j} \in F;$$
$$R_1'(U_A) = \sum_{j \in A} R_1(T_j) \quad \text{and} \quad R_1'(V_A) = B_1 - R_1'(U_A) \qquad \text{for } A \in \mathscr{A};$$
$$R_2'(U_A) = R_2'(V_A) = k \qquad \text{for } A \in \mathscr{A};$$

and deadline $D' = M + D$.

Before embarking on the formal proof, a few intuitive comments concerning the correspondence between valid schedules for I and I' may be helpful. We want single time units in the $2k$ processor case to correspond to *pairs* of time units in the $k+1$ processor case, with each set of $2k$ concurrently executed tasks in the one case partitioned into two sets of k concurrently executed tasks in the other. We need the $(k+1)$st processor and the U_A and V_A tasks to insure that this happens in the right way. The resource requirements for U_A and V_A are designed so that essentially only one of two things can happen: either U_A and V_A are executed together, with "filler" tasks from F occupying the remaining $k-1$ processors, or else U_A and V_A are executed at different times, each together with a set of k T_i' tasks, such that the two corresponding sets of T_i tasks could all be executed concurrently in the $2k$ processor case.

We first observe that $m = 2kD$ since I has the saturated processor property, which implies

$$|\mathscr{T}'| = 2kD + (M - D)(k - 1) + 2M = (M + D)(k + 1).$$

Thus I' has the saturated processor property. (The tasks in F are essentially "filler" to make this the case.) The reader may verify similarly that resources R_1' and R_2' are saturated, showing that I' indeed belongs to the input domain for $P[k+1, 2]$.

Suppose now that f is a valid schedule for I. Since I has the saturated processor property, we have $|E_f(t)| = 2k$ for each t, $0 \leqq t \leqq D - 1$. Define

$$X(t) = \{i \in E_f(t) : |\{1, 2, \cdots, i\} \cap E_f(t)| \leqq k\}$$

and

$$Y(t) = E_f(t) - X(t)$$

for $0 \leqq t \leqq D - 1$. Then $|X(t)| = |Y(t)| = k$, $X(t) \cup Y(t) = E_f(t)$, and both $X(t)$ and $Y(t)$ belong to $\mathscr{A}$. Let $S = \mathscr{A} - \{X(t) : 0 \leqq t \leqq D - 1\}$ and label the elements of S as $A_1, A_2, \cdots, A_{M-D}$. We now are prepared to define the schedule $f' : \mathscr{T} \to \{0, 1, 2, \cdots, M + D - 1\}$. For each t, $0 \leqq t \leqq D - 1$, set

$$f'(V_{X(t)}) = f'(T_i') = 2t \quad \text{for } i \in X(t);$$

$$f'(U_{X(t)}) = f'(T_i') = 2t + 1 \quad \text{for } i \in Y(t).$$

To complete the definition of f', set

$$f'(U_{A_i}) = f'(V_{A_i}) = f'(F_{i,j}) = 2D - 1 + i \quad \text{for } 1 \leqq j \leqq k - 1,$$

for each $A_i \in S$, $1 \leqq i \leqq M - D$. It is not difficult to check that f' satisfies all the properties required for a valid schedule.

Now suppose f' is a valid schedule for I'. Define $W = U \cup V$, and for each integer $i \geqq 0$ define

$$H_i(f') = \{t : 0 \leqq t \leqq M + D - 1 \text{ and } |E_{f'}(t) \cap W| = i\}.$$

Thus $H_i(f')$ is the set of integer times at which exactly i tasks from W are being executed under f'. Since $B_2' = 2k$ and each $w \in W$ has $R_2'(w) = k$, we know that $H_i(f')$ must be empty for $i > 2$. Furthermore, since each task not in W requires at most 1 unit of R_2' and R_2' is saturated in I', we must also have $H_0(f') = \varphi$.

Thus $|H_1(f')| + |H_2(f')| = M + D$, and, since $|W| = 2M$, it follows that $|H_1(f')| = 2D$ and $|H_2(f')| = M - D$.

Also, for any $t \in H_2(f')$, we know that $E_{f'}(t) \cap T = \varphi$ because the two tasks from W already use up the full amount of R'_2. Thus each T'_i satisfies $f'(T'_i) \in H_1(f')$ and, by the saturated processor property, each $F_{i,j}$ must satisfy $f'(F_{i,j}) \in H_2(f')$.

Define

$$\text{MATCH}(f') = \{t \in H_2(f') : \exists\, A \in \mathscr{A} \text{ such that } \{U_A, V_A\} \subseteq E_{f'}(t)\}.$$

Without loss of generality, we may assume that f' satisfies

$$|\text{MATCH}(f')| \geqq |\text{MATCH}(f'')|$$

for every valid schedule f'' for I'. We claim that this implies that $\text{MATCH}(f') = H_2(f')$. For, suppose there were a $t_0 \in H_2(f')$ such that $t_0 \notin \text{MATCH}(f')$. Let W_1 and W_2 be members of W belonging to $E_{f'}(t_0)$. Let W_3 be such that $\{W_2, W_3\} = \{U_A, V_A\}$ for some $A \in \mathscr{A}$, and let $t' = f'(W_3)$. Notice that this and the saturated resource property imply that

$$R'_1(W_3) = B_1 - R'_1(W_2) = R'_1(W_1).$$

Then the function f'', which is identical to f' except that $f''(W_3) = t_0$ and $f''(W_1) = t'$, is a valid schedule for I' with

$$|\text{MATCH}(f'')| > |\text{MATCH}(f')|,$$

contradicting the choice of f'. Therefore we may assume $\text{MATCH}(f') = H_2(f')$.

Since $\text{MATCH}(f') = H_2(f')$, it immediately follows that for every $A \in \mathscr{A}$, $f'(U_A) \in H_1(f')$ if and only if $f'(V_A) \in H_1(f')$. Accordingly, let

$$S = \{A \in \mathscr{A} : f'(U_A) \in H_1(f')\},$$

and sequence the members of S as $S_1, S_2, \cdots, S_D$. We now can define a valid schedule f for I by

$$E_f(t-1) = \{T_i \in \mathscr{T} : f'(T'_i) \in \{f'(U_{S_t}), f'(V_{S_t})\}\}$$

for each t, $1 \leqq t \leqq D$.

The function f is a total function from $\mathscr{T}$ to $\{0, 1, 2, \cdots, D-1\}$ since $f'(T'_i) \in H_1(f')$ for all $T'_i \in T$. Thus f trivially satisfies properties (i) and (ii) required of a valid schedule. Moreover, $|E_f(t)| = 2k$ for $0 \leqq t \leqq D-1$ since, for each $t \in H_1(f')$, $|E_{f'}(t) \cap T| = k$. Thus f also obeys property (iii). Finally, observe that for each t, $0 \leqq t \leqq D-1$,

$$\begin{aligned}\sum_{T_i \in E_f(t)} R_1(T_i) &= (B'_1 - R'_1(U_{S_{t+1}})) + (B'_1 - R'_1(V_{S_{t+1}})) \\ &= 2B'_1 - B'_1 = B'_1 = B_1,\end{aligned}$$

so property (iv) also is satisfied, and f is indeed a valid schedule for I.

Thus I' has a valid schedule if and only if I has a valid schedule, and the required reduction has been demonstrated. □

The final lemma of this section will show that P[5, 8] is NP-complete, enabling us to apply the preceding lemmas to obtain our main result. First, let us define the NP-complete three-dimensional matching problem.

THREE-DIMENSIONAL MATCHING (3DM) [8].

Input: Finite sets T and $S \subseteq T \times T \times T$.

Property: S contains a complete matching, i.e., a subset $S' \subseteq S$ with $|S'| = |T|$ such that for any two members $\langle x_1, y_1, z_1 \rangle$ and $\langle x_2, y_2, z_2 \rangle$ of S', $x_1 \neq x_2$, $y_1 \neq y_2$, and $z_1 \neq z_2$.

LEMMA 3.4. 3DM $\propto$ P[5, 8].

Proof. Let T and $S \subseteq T \times T \times T$ be given. We may assume without loss of generality that $T = \{1, 2, \cdots, N\}$ where $N = |T|$. For integers i, k, $1 \leqq i \leqq N$, $1 \leqq k \leqq 3$, let $m_k(i)$ be the number of ordered triplets in S having their kth component equal to i. We may assume that each $m_k(i) \geqq 1$, since if any $m_k(i) = 0$, we would know immediately that S contains no complete matching. The corresponding input I to P[5, 8] is the following:

$$n = 5;$$

$$\mathscr{T} = \{S[i, j, k] : \langle i, j, k \rangle \in S\} \cup \{X_0[i], Y_0[i], Z_0(i); 1 \leqq i \leqq N\}$$
$$\cup \{X[i; l] : 1 \leqq i \leqq N, 1 \leqq l < m_1(i)\} \cup \{Y[i; l] : 1 \leqq i \leqq N, 1 \leqq l < m_2(i)\}$$
$$\cup \{Z[i; l] : 1 \leqq i \leqq N, 1 \leqq l < m_3(i)\} \cup \{F_i : 1 \leqq i \leqq N\}$$
$$\cup \{G_i : 1 \leqq i \leqq |S| - N\};$$

$$\mathscr{R} = \{R_j : 1 \leqq j \leqq 8\},$$

with resource bounds and task resource requirements as given in Table 1, and with deadline $D = |S|$.

Though the input may appear rather complicated, the basic idea behind it is quite simple. The resource requirements and bounds insure that during each time unit of any valid schedule, the five tasks being executed will consist of one S-type task, one X_0- or X-type task, one Y_0- or Y-type task, one Z_0- or Z-type task, and one F- or G-type task. The S-type tasks each represent an ordered triple from S. The N F-type tasks select the triples which form the matching, i.e., those $\langle i, j, k \rangle$ for which $S[i, j, k]$ is executed concurrently with some F-type task. Finally, the X_0-type tasks, Y_0-type tasks and Z_0-type tasks are used to insure that the N selected S-type tasks actually represent a matching. This is done by specifying the resource requirements so that, since all resources are saturated, the only tasks which can be executed simultaneously with F_l and $S[i, j, k]$ are $X_0[i]$, $Y_0[j]$, and $Z_0[k]$. Since there is exactly one $X_0[i]$, $Y_0[i]$, and $Z_0[i]$ for each i, $1 \leqq i \leqq N$, the saturated processor property will guarantee that in any valid schedule the selected $S[i, j, k]$ tasks represent disjoint triples and thus give the required matching. We now give the formal proof.

Observe first that

$$|\mathscr{T}| = |S| + \sum_{k=1}^{3} \sum_{i=1}^{N} m_k(i) + N + (|S| - N)$$
$$= |S| + 3 \cdot |S| + |S| = 5 \cdot |S| = n \cdot D,$$

so that I has the saturated processor property. The reader may verify similarly that all eight resources are saturated, so I belongs to the input domain for P[5, 8].

TABLE 1
Resource requirements and bounds for Lemma 3.4

Task W	$R_1(W)$	$R_2(W)$	$R_3(W)$	$R_4(W)$	$R_5(W)$	$R_6(W)$	$R_7(W)$	$R_8(W)$
$S[i, j, k]$	$N - i$	$N - j$	$N - k$	1	0	0	0	0
$X_0[i]$	i	0	0	0	1	0	0	0
$X[i; l]$	i	0	0	0	1	0	0	1
$Y_0[j]$	0	j	0	0	0	1	0	0
$Y[j; l]$	0	j	0	0	0	1	0	1
$Z_0[k]$	0	0	k	0	0	0	1	0
$Z[k; l]$	0	0	k	0	0	0	1	1
F_i	0	0	0	0	0	0	0	3
G_i	0	0	0	0	0	0	0	0
Bound	N	N	N	1	1	1	1	3

Suppose S contains a complete matching S'. We must construct a valid schedule f for I. Order the elements of S as $S_1, S_2, \cdots, S_{|S|}$ so that $S' = \{S_l: 1 \leqq l \leqq N\}$. For each $S_l \in S$, let the components of S_l be denoted by $\langle i(l), j(l), k(l)\rangle$. We specify f by giving the sets $E_f(t)$, $0 \leqq t < D = |S|$. For $1 \leqq t \leqq N$, let

$$E_f(t-1) = \{S[i(t), j(t), k(t)], X_0[i(t)], Y_0[j(t)], Z_0[k(t)], F_t\}.$$

For each t, $N + 1 \leqq t \leqq |S|$, define

$$L_1(t) = |\{t': N + 1 \leqq t' \leqq t \text{ such that } i(t') = i(t)\}|,$$

$$L_2(t) = |\{t': N + 1 \leqq t' \leqq t \text{ such that } j(t') = j(t)\}|,$$

$$L_3(t) = |\{t': N + 1 \leqq t' \leqq t \text{ such that } k(t') = k(t)\}|.$$

Then for each t, $N + 1 \leqq t \leqq |S|$, let

$$E_f(t-1) = \{S[i(t), j(t), k(t)], X[i(t); L_1(t)], Y[j(t); L_2(t)], Z[k(t); L_3(t)], G_{t-N}\}.$$

We leave to the reader the straightforward verification that f is a valid schedule for I.

Conversely, suppose we have a valid schedule f for I. Since there are $|S|$ tasks of the form $S[i, j, k]$, the constraints on resource R_4 and the fact that $D = |S|$ imply that there is exactly one $S[i, j, k]$ in each $E_f(t)$, $0 \leqq t \leqq D - 1$. Similarly, if we let

$$X = \{X_0[i], X[i; l]: 1 \leqq i \leqq N, 1 \leqq l < m_1(i)\},$$

$$Y = \{Y_0[i], Y[i; l]: 1 \leqq i \leqq N, 1 \leqq l < m_2(i)\},$$

$$Z = \{Z_0[i], Z[i, l]: 1 \leqq i \leqq N, 1 \leqq l < m_3(i)\},$$

then, due to resources R_5, R_6, and R_7, each $E_f(t)$ contains exactly one element of X, one element of Y, and one element of Z. Since I has the saturated processor property, we can conclude that each $E_f(t)$ also must contain exactly one element of

$$\{F_i: 1 \leqq i \leqq N\} \cup \{G_i: 1 \leqq i \leqq |S| - N\}.$$

Let us define

$$H = \{t:0 \leqq t < D\} \text{ such that } E_f(t) \cap \{F_i:1 \leqq i \leqq N\} \neq \varphi\}.$$

Our previous arguments imply that $|H| = N$. Because of resource R_8, we know that for each $t \in H$, $E_f(t)$ contains no tasks of the form $X[i; l]$, $Y[i; l]$ or $Z[i; l]$, and hence $E_f(t)$ has the form

$$\{S[i, j, k], X_0[i'], Y_0[j'], Z_0[k'], F_l\}.$$

Furthermore, by saturated resources R_1, R_2 and R_3, it must be the case that $i = i'$, $j = j'$ and $k = k'$. It follows that

$$S' = \{\langle i, j, k\rangle \in S:f(S[i, j, k]) \in H\}$$

is a complete matching for S.

Thus I has a valid schedule if and only if S has a complete matching, and the desired reduction has been demonstrated. □

We now have our main result.

THEOREM 3.5. MS[$n = 3$, $r = 1$, $\prec$ *empty, each* $\tau_i = 1$] *is* NP*-complete.*

Proof. By starting with Lemma 3.4 and proceeding via an application of Lemma 3.1, an application of Lemma 3.2, and two applications of Lemma 3.3, we obtain

$$3\text{DM} \propto \text{P}[5, 8] \propto \text{P}[5, 1] \propto \text{P}[6, 1] \propto \text{P}[4, 1] \propto \text{P}[3, 1].$$

Since 3DM is NP-complete, we conclude that P[3, 1] is NP-complete. The theorem follows since the input domain for P[3, 1] is included in the input domain for MS[$n = 3$, $r = 1$, $\prec$ empty, each $\tau_i = 1$]. □

4. Concluding remarks. Though the two main scheduling problems which we have proved to be NP-complete may seem rather specialized, the results immediately imply that many other scheduling problems are NP-complete. In particular, for arbitrary integers $k \geqq 2$ and $j \geqq 1$, each of the following problems, and any problem whose input domain contains the input domain for such a problem, is NP-complete:

1. MS[$n = k, r = j, \prec$ a forest, each $\tau_i = 1$];
2. MS[$n = k + 1, r = j, \prec$ empty, each $\tau_i = 1$].

Furthermore, our results are best possible in the sense that further natural restrictions on the input domains lead to problems which can be solved with polynomial time algorithms. That is, if the input domains for problems 1 and 2 above are restricted further by choosing $k < 2$, $j < 1$, or (in problem 1) requiring $\prec$ empty, then the resulting problems can be solved in polynomial time by algorithms which have been mentioned previously.

Our results have thus determined the "boundary" between NP-completeness and polynomial time solvability with respect to the number of processors, number of resources, and type of partial order. Implicit in our proofs is another restriction on input domains for which we can determine a fairly narrow "frontier".

The NP-completeness of *some* problems hinges quite strongly on the fact that one can use the expressive power of binary notation, that is, one can write an integer of magnitude n using only $\log_2 n$ symbols. For instance, MS[$n = k, r = 0$,

$\prec$ empty, τ_i arbitrary] is NP-complete for any $k \geqq 2$; but for every polynomial p and integer $k \geqq 1$, MS[$n = k$, $r = 0$, $\prec$ empty, each $\tau_i \leqq p(|\mathscr{T}|)$] can be solved in polynomial time (although the degree of the polynomial depends on k and p, and can be quite large).

In contrast, our proofs in each case can be used to construct a polynomial p such that the problem remains NP-complete *even if* we include the restriction that each $R_j(T_i) \leqq p(|\mathscr{T}|)$. The other side of the frontier is provided by the fact that for any *finite* set S, and integers j, $k \geqq 0$, MS[$n = j$, $r = k$, $\prec$ empty, each $\tau_i = 1$, each $R_j(T_i) \in S$] can be solved in polynomial time (although the degree of the polynomial again can be quite large, depending on S, j, and k). Further information is provided by the fact that MS[$n = 3$, r arbitrary, $\prec$ empty, each $\tau_i = 1$, each $R_j(T_i) \in \{0, 1\}$] *is* NP-complete. It is not yet known whether there exists an integer $k \geqq 0$ and finite set S such that MS[n arbitrary, $r = k$, $\prec$ empty, each $\tau_i = 1$, each $R_j(T_i) \in S$] is NP-complete.

REFERENCES

[1] E. G. COFFMAN AND R. L. GRAHAM, *Optimal scheduling for two-processor systems*, Acta Informat., 1 (1972), pp. 200–213.

[2] S. A. COOK, *The complexity of theorem proving procedures*, 3rd Ann. ACM Symp. on Theory of Computing, 1971, pp. 151–158.

[3] J. EDMONDS, *Paths, trees, and flowers*, Canad. J. Math., 17 (1965), pp. 449–467.

[4] M. FUJII, T. KASAMI AND K. NINOMIYA, *Optimal sequencing of two equivalent processors*, SIAM J. Appl. Math., 17 (1969), pp. 784–789.

[5] M. R. GAREY AND R. L. GRAHAM, *Bounds on scheduling with limited resources*, 4th Symp. on Operating System Principles, 1973, pp. 104–111.

[6] R. L. GRAHAM, *Bounds on multiprocessing anomalies and related packing problems*, AFIPS Conf. Proc., 40 (1972), pp. 205–217.

[7] T. C. HU, *Parallel scheduling and assembly line problems*, Operations Res., 9 (1961), pp. 841–848.

[8] R. M. KARP, *Reducibility among combinatorial problems*, Complexity of Computer Computations, R. E. Miller and J. W. Thatcher, eds., Plenum Press, New York, 1972, pp. 85–104.

[9] Y. MURAOKA, *Parallelism exposure and exploitation in programs*, Ph.D. thesis, Univ. of Illinois, Urbana, 1971.

[10] J. D. ULLMAN, *Polynomial complete scheduling problems*, 4th Symp. on Operating System Principles, 1973, pp. 96–101.