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

0030-364X/78/2601-0022 $01.25


# Complexity of Scheduling under Precedence Constraints

J. K. LENSTRA

*Mathematisch Centrum, Amsterdam, The Netherlands*

A. H. G. RINNOOY KAN

*Erasmus University, Rotterdam, The Netherlands*

(Received April 1976; accepted May 1977)

**Precedence constraints between jobs that have to be respected in every feasible schedule generally increase the computational complexity of a scheduling problem. Occasionally, their introduction may turn a problem that is solvable within polynomial time into an *NP*-complete one, for which a good algorithm is highly unlikely to exist. We illustrate the use of these concepts by extending some typical *NP*-completeness results and simplifying their correctness proofs for scheduling problems involving precedence constraints.**

---

THE PIONEERING WORK by Cook [7] and Karp [19, 20] has led to new techniques for examining the computational complexity of combinatorial problems. These techniques have been applied extensively in the area of machine scheduling, to locate the borderline between the class of problems for which a polynomial-bounded or "good" algorithm is available and the class of *NP*-complete problems, for which the existence of such an algorithm is very unlikely [3, 4, 10, 12, 14, 18, 23, 27, 29, 30, 32, 35].

In this paper we illustrate the use of these concepts by extending some typical *NP*-completeness results and simplifying their proofs for scheduling problems involving precedence constraints between jobs. We recall the basic concepts from complexity theory in Section 1 and delineate a specific class of scheduling problems in Section 2. Sections 3, 4, and 5 contain the complexity results. A survey of these results and some concluding remarks are presented in Section 6.

## 1. REDUCIBILITY AND *NP*-COMPLETENESS

Cook [7] and Karp [19] first explored the relation between the classes $\mathcal{P}$ and $\mathcal{NP}$ of (language recognition) problems solvable by deterministic and non-deterministic Turing machines, respectively, in a number of steps bounded by a polynomial in the length of the input. With respect to combinatorial optimization, we do not really require mathematically

rigorous definitions of these concepts. For our purposes we may safely identify $\mathcal{P}$ with the class of problems for which a polynomial-bounded or good algorithm exists, whereas all problems in $\mathcal{NP}$ can be solved by polynomial-depth backtrack search.

In this context, all problems are stated in terms of recognition problems that require a yes/no answer. In order to deal with the complexity of a combinatorial minimization problem, we transform it into the problem of determining the existence of a solution with value at most equal to $y$, for some threshold $y$.

It is clear that $\mathcal{P} \subset \mathcal{NP}$, and the question arises if this inclusion is a proper one or if, on the contrary, $\mathcal{P} = \mathcal{NP}$ [28]. Although this is still an open problem, the equality of $\mathcal{P}$ and $\mathcal{NP}$ is considered to be very unlikely, since $\mathcal{NP}$ contains many notorious combinatorial problems for which, in spite of an intensive research effort, no good algorithms have been found so far.

Further insight into the relation between $\mathcal{P}$ and $\mathcal{NP}$ is obtained by introducing the following concepts.

Problem $P'$ is *reducible* to problem $P$ (notation: $P' \propto P$) if for any instance of $P'$ an instance of $P$ can be constructed in polynomial-bounded time such that solving the instance of $P$ will solve the instance of $P'$ as well.

$P$ is *NP-complete* if $P \in \mathcal{NP}$ and $P' \propto P$ for every $P' \in \mathcal{NP}$.

Informally, the reducibility of $P'$ to $P$ implies that $P'$ can be considered as a special case of $P$; the $NP$-completeness of $P$ indicates that $P$ is, in a sense, the most difficult problem in $\mathcal{NP}$.

A polynomial-bounded algorithm for any $NP$-complete problem could be used to construct a good algorithm for every problem in $\mathcal{NP}$, and its existence would thus imply that $\mathcal{P} = \mathcal{NP}$. Many problems that are infamous for their computational intractability have been proved $NP$-complete, such as the general 0–1 programming problem, the traveling salesman problem, and the job-shop scheduling problem.

Actually, $NP$-completeness was first established with respect to the so-called SATISFIABILITY problem [7]. This problem can be formulated as follows.

SATISFIABILITY: *Given clauses* $C_1, \cdots, C_u$, *each of which is a disjunction of literals from the set* $X = \{x_1, \cdots, x_t, \bar{x}_1, \cdots, \bar{x}_t\}$, *is the conjunction of the clauses satisfiable, i.e., does there exist a subset* $S \subset X$ *such that* $S$ *does not contain a complementary pair of literals* $(x_i, \bar{x}_i)$, *and* $S \cap C_j \neq \phi$ *for* $j = 1, \cdots, u$?

Cook proved this result by specifying a polynomial-bounded "master reduction" that, given $P \in \mathcal{NP}$, constructs for any instance of $P$ an equivalent boolean expression in conjunctive normal form.

Subsequently, Karp [19] established $NP$-completeness for many problems $P \in \mathfrak{N}\mathcal{P}$ by specifying a reduction $P' \propto P$ for some $NP$-complete $P'$ (for every $P'' \in \mathfrak{N}\mathcal{P}$, $P'' \propto P'$ and $P' \propto P$ imply that $P'' \propto \mathcal{P}$ as well).

Karp's work has led to a large amount of research on the location of the borderline separating the "easy" problems (in $\mathcal{P}$) from the "hard" ($NP$-complete) ones. It turns out that a minor change in a problem parameter (notably—for some as yet mystical reason—an increase from two to three) often transforms an easy problem into a hard one. Not only does knowledge of the borderline lead to fresh insights as to what characteristics of a problem determine its complexity, but there are also important consequences with respect to the solution of these problems. Establishing $NP$-completeness of a problem may avoid a probably fruitless search for good algorithms and justifies the development of enumerative optimization methods [29, 32] or concentration on approximation algorithms [13, 21].

Machine scheduling problems seem especially attractive objects for this type of research since their structure is relatively simple and there exist standard problem parameters that have demonstrated their usefulness in previous research.

Before describing a class of scheduling problems, let us emphasize that membership of $\mathcal{P}$ versus $NP$-completeness yields only a very coarse measure of complexity. In particular, there are significant differences in complexity within the class of $NP$-complete problems.

One possible refinement of the complexity measure may be based on the way in which numerical problem data are encoded. To illustrate this point, let us consider the following two $NP$-complete problems.

KNAPSACK: *Given positive integers* $a_1, \cdots, a_t, b$, *does there exist a subset* $S \subset T = \{1, \cdots, t\}$ *such that* $\sum_{i \in S} a_i = b$?

3-PARTITION: *Given positive integers* $a_1, \cdots, a_{3t}, b$, *does there exist a partition* $(T_1, \cdots, T_t)$ *of* $T = \{1, \cdots, 3t\}$ *such that* $|T_j| = 3$ *and* $\sum_{i \in T_j} a_i = b$ *for* $j = 1, \cdots, t$?

For both problems, the length of the input is $O(t \log b)$ in the standard *binary* encoding, and $O(tb)$ if a *unary* encoding is allowed. 3-PARTITION has been proved $NP$-complete even with respect to a unary encoding [10]. KNAPSACK is $NP$-complete with respect to a binary encoding [19], but solution by dynamic programming requires $O(tb)$ steps and thus yields a polynomial-bounded algorithm with respect to a unary encoding. Similar situations exist for several scheduling problems. Such "pseudopolynomial" algorithms [26] need not necessarily be "good" in the practical sense of the word, but it may pay nonetheless to distinguish between complexity results with respect to unary and binary encodings (cf. [14]). Unary $NP$-completeness or binary membership of $\mathcal{P}$ would then be the strongest possible result, and it is quite feasible for a problem to be binary $NP$-

complete and to allow a unary polynomial-bounded solution. We shall not explore this distinction any further here; it is easily checked that the results presented in Theorems 1–8 are strong in the sense that they hold with respect to a unary encoding.

## 2. A CLASS OF SCHEDULING PROBLEMS

The scheduling problems we will discuss can be formulated as follows (cf. [6, 29, 32]). Each of $n$ *jobs* $J_i$, $i \in I$, has to be processed without preemption on one of $m$ identical *machines*, each of which can handle at most one job at a time. For each $i \in I$, non-negative integers $p_i$, $w_i$ and $d_i$ denote the *processing time*, *weight* and *due date* of $J_i$. Furthermore, a *precedence relation* $<$ between the jobs is given; $J_i < J_j$ implies that $J_j$ cannot start before $J_i$ has been completed. If the transitive reduction of the corresponding precedence graph $(I, \{(i,j) | J_i < J_j\})$ is an *arborescence*, i.e., if it has

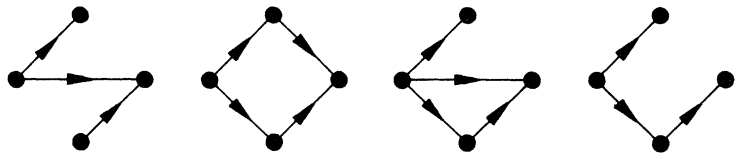


**Figure 1.** Examples of precedence graphs.

either indegree at most one for each vertex or outdegree at most one for each vertex, the relation $<$ is said to be of type *tree*; the general type will be denoted by *prec*. Figure 1 shows some examples; note that graph (d) is the transitive reduction of graph (c).

Given a feasible schedule, we can compute for each $J_i$ its *completion time* $C_i$ and its *tardiness* $T_i = \max\{0, C_i - d_i\}$. Define $U_i = 0$ if $C_i \leqq d_i$ and $U_i = 1$, otherwise. The schedule can be evaluated according to various criteria; for instance, we may seek to minimize $C_{\max} = \max_{i \in I}\{C_i\}$, $\sum w_i C_i = \sum_{i \in I} w_i C_i$, $\sum w_i T_i = \sum_{i \in I} w_i T_i$, or $\sum w_i U_i = \sum_{i \in I} w_i U_i$. If all weights are equal, the last three criteria will be denoted by $\sum C_i$, $\sum T_i$ and $\sum U_i$.

We can now characterize each of the scheduling problems within a four-parameter classification $\alpha|\beta|\gamma|\delta$ where $\alpha$ indicates the number of machines equal to a constant or to a variable $m$, $\beta$ indicates either the type of precedence constraints (*prec* or *tree*) or (by its absence) the absence of such constraints, $\gamma$ indicates either the restriction that $p_i \in \{\gamma\}$ for all $i \in I$ or the absence of such a restriction, and $\delta$ indicates the optimality criterion. All scheduling problems of this type can be solved by polynomial-depth backtrack search and thus belong to $\mathfrak{NP}$.

In the reductions to be presented, scheduling problems will be related to problems defined on an undirected graph $G = (V, E)$. Consequently,

our reductions involve *vertex jobs* $J_i$ or $J_i^{(h)}$, $i \in V$, and *edge jobs* $J_{\{i,j\}}$ or $J_{\{i,j\}}^{(h)}$, $\{i,j\} \in E$. As to precedence constraints, we will typically require an edge job to follow the corresponding vertex jobs, e.g. $J_i < J_{\{i,j\}}$, $i \in V$, $\{i,j\} \in E$.

## 3. SINGLE MACHINE: COMPLETION TIME CRITERIA

We shall first examine the computational complexity of $1|\beta|\gamma|C_{\max}$ and $1|\beta|\gamma|\sum w_iC_i$ problems.

The $1|prec||C_{\max}$ problem is clearly trivial because every feasible schedule is optimal. In fact, the general $1|prec||\max_{i \in I}\{f_i(C_i)\}$ problem can be solved for arbitrary non-decreasing cost functions $f_i$ by means of an $O(n^2)$ algorithm due to Lawler [24].

Smith [34] proved that the $1||\sum w_iC_i$ problem can be solved by ordering the jobs according to non-decreasing $p_i/w_i$ ratio. Horn [16] and Sidney [33] independently extended this result by developing $O(n \log n)$ algorithms for the $1|tree||\sum w_iC_i$ problem (see also [2, 8]). Knuth [22], Adolphson [1] and Lawler [27] have further extended these algorithms to deal with the more general type of *series parallel* precedence constraints.

Most likely, this is as far as we can get since both the $1|prec|0, 1|\sum C_i$ and the $1|prec|1|\sum w_iC_i$ problem are *NP*-complete. Within our classification, this is essentially the strongest possible result, because with respect to the $1|prec|1|\sum C_i$ problem every feasible schedule is optimal. In order to prove *NP*-completeness for the two above problems, it is sufficient to specify reductions from an *NP*-complete problem to both of them. We will start from the LINEAR ARRANGEMENT problem, which was shown to be *NP*-complete in [15]. Its use was suggested by Lawler; his reductions to slightly more restricted $1|prec||\sum w_iC_i$ problems are given in [27].

LINEAR ARRANGEMENT: *Given an undirected graph* $G=(V,E)$ *with* $V=\{1, \cdots, v\}$ *and an integer* $k$, *does there exist a permutation* $\pi$ *of* $V$ *such that* $\sum_{\{i,j\} \in E} |\pi(i)-\pi(j)| \leqq k$?

We define $e=|E|$, $u_i=|\{\{i,j\}|\{i,j\} \in E\}|$ (the *degree* of $i$) and assume that $e > \max_{i \in V}\{u_i\}$.

THEOREM 1. LINEAR ARRANGEMENT $\propto 1|prec|0, 1|\sum C_i$.

*Proof.* First, we will specify a $1|prec||\sum w_iC_i$ problem and a threshold value $y$, and prove that LINEAR ARRANGEMENT has a solution if and only if there exists a schedule with value $\sum w_iC_i \leqq y$. Subsequently, this $1|prec||\sum w_iC_i$ problem will be transformed into an equivalent $1|prec|0, 1|\sum C_i$ problem.

Given $G=(V,E)$ and $k$, we define the following $1|prec||\sum w_iC_i$ problem, where just for the moment we allow negative weights $w_i$: $n=v+e$; vertex jobs $J_i$, $i \in V$: $p_i=1$, $w_i=-u_i$; edge jobs $J_{\{i,j\}}$, $\{i,j\} \in E$: $p_{\{i,j\}}=0$, $w_{\{i,j\}}=2$;

$J_i < J_{\{i,j\}}$, $i \in V$, $\{i,j\} \in E$; $y = k$. Consider a permutation $\pi$ of $V$, indicating that $J_i$ is scheduled in position $\pi(i)$ among the vertex jobs. We may evidently assume that $J_{\{i,j\}}$ will be processed as soon as both $J_i$ and $J_j$ are completed. Thus we have that $C_i = \pi(i)$, $i \in V$; $C_{\{i,j\}} = \max\{\pi(i), \pi(j)\}$, $\{i,j\} \in E$. The value of this schedule is given by

$$\begin{aligned}\sum w_i C_i &= \sum_{i \in V} -u_i \pi(i) + \sum_{\{i,j\} \in E} 2 \max\{\pi(i), \pi(j)\} \\ &= \sum_{\{i,j\} \in E} (2 \max\{\pi(i), \pi(j)\} - \pi(i) - \pi(j)) \\ &= \sum_{\{i,j\} \in E} |\pi(i) - \pi(j)|.\end{aligned}$$

It follows that LINEAR ARRANGEMENT has a solution if and only if there is a schedule with value $\leqq y$.

We can easily get rid of the negative weights by redefining $w_i = e - u_i$, $i \in V$; $y = \frac{1}{2}v(v+1)e + k$. The threshold value $y$ is increased by $\frac{1}{2}v(v+1)e$, equal to the additional contribution of the vertex jobs to the value of any schedule.

We now transform this problem into one with unit weights by replacing each $J_x$, $x \in V \cup E$, by a chain of jobs $J_x^{(1)} < J_x^{(2)} < \cdots < J_x^{(w_x)}$ with $p_x^{(1)} = p_x$, $p_x^{(2)} = \cdots = p_x^{(w_x)} = 0$. By a straightforward interchange argument, we may assume the jobs in such a chain to be scheduled consecutively. Thus, the following equivalent $1|prec|0, 1|\sum C_i$ problem results: $n = ve$; vertex jobs $J_i^{(1)}, \cdots, J_i^{(e-u_i)}$, $i \in V$: $p_i^{(1)} = 1$, $p_i^{(2)} = \cdots = p_i^{(e-u_i)} = 0$; edge jobs $J_{\{i,j\}}^{(1)}$, $J_{\{i,j\}}^{(2)}$, $\{i,j\} \in E$: $p_{\{i,j\}}^{(1)} = p_{\{i,j\}}^{(2)} = 0$; $J_i^{(1)} < J_i^{(2)} < \cdots < J_i^{(e-u_i)} < J_{\{i,j\}}^{(1)} < J_{\{i,j\}}^{(2)}$, $i \in V$, $\{i,j\} \in E$; $y = \frac{1}{2}v(v+1)e + k$. Clearly, LINEAR ARRANGEMENT has a solution if and only if there exists a $1|prec|0, 1|\sum C_i$ schedule with value $\leqq y$, which proves the theorem.

THEOREM 2. LINEAR ARRANGEMENT $\propto 1|prec|1|\sum w_i C_i$.

*Proof.* Following the same line of proof as for Theorem 1, we define the following $1|prec||\sum w_i C_i$ problem: $n = v + e$; vertex jobs $J_i$, $i \in V$: $p_i = t$, $w_i = e - u_i$; edge jobs $J_{\{i,j\}}$, $\{i,j\} \in E$: $p_{\{i,j\}} = 1$, $w_{\{i,j\}} = 2$; $J_i < J_{\{i,j\}}$, $i \in V$, $\{i,j\} \in E$; $y = \frac{1}{2}v(v+1)et + t(k+1)$, where $t = (v+3)e^2$. Note that this is an inflated version of the $1|prec||\sum w_i C_i$ problem in the proof of Theorem 1. Suppose that $J_i$ is scheduled in position $\pi(i)$ among the vertex jobs. Again we may assume that the last vertex job preceding $J_{\{i,j\}}$ is either $J_i$ or $J_j$. Thus we have that

$$\begin{aligned}&t\pi(i) \leqq C_i \leqq t\pi(i) + e, \\ &t|\pi(i) - \pi(j)| \leqq |C_i - C_j| \leqq t|\pi(i) - \pi(j)| + e, \\ &\max\{C_i, C_j\} < C_{\{i,j\}} < \max\{C_i, C_j\} + e,\end{aligned}$$

and hence

$$t|\pi(i) - \pi(j)| < 2C_{\{i,j\}} - C_i - C_j < t|\pi(i) - \pi(j)| + 3e.$$

If $\sum_{\{i,j\}\in E} |\pi(i)-\pi(j)| \leqq k$, then we have

$$\begin{aligned}\sum w_iC_i &= e\sum_{i\in V} C_i + \sum_{\{i,j\}\in E} (2C_{\{i,j\}} - C_i - C_j)\\ &< e\sum_{i\in V} (t\pi(i)+e) + \sum_{\{i,j\}\in E} (t|\pi(i)-\pi(j)|+3e)\\ &= \tfrac{1}{2}v(v+1)et + (v+3)e^2 + t\sum_{\{i,j\}\in E} |\pi(i)-\pi(j)| \leqq y.\end{aligned}$$

If $\sum_{\{i,j\}\in E} |\pi(i)-\pi(j)| > k$, then we have

$$\begin{aligned}\sum w_iC_i &= e\sum_{i\in V} C_i + \sum_{\{i,j\}\in E} (2C_{\{i,j\}} - C_i - C_j)\\ &> e\sum_{i\in V} t\pi(i) + \sum_{\{i,j\}\in E} t|\pi(i)-\pi(j)|\\ &= \tfrac{1}{2}v(v+1)et + t\sum_{\{i,j\}\in E} |\pi(i)-\pi(j)| \geqq y.\end{aligned}$$

It follows that LINEAR ARRANGEMENT has a solution if and only if there is a schedule with value $\leqq y$.

We transform this problem into one with unit processing times by replacing each vertex job $J_i$ by a chain $J_i^{(1)} < \cdots < J_i^{(t-1)} < J_i^{(t)}$ with $w_i^{(1)} = \cdots = w_i^{(t-1)} = 0$, $w_i^{(t)} = e - u_i$.

The jobs in such a chain may again be assumed to be scheduled consecutively. Thus we have obtained the following equivalent $1|prec|1|\sum w_iC_i$ problem: $n = vt+e$; vertex jobs $J_i^{(1)}, \cdots, J_i^{(t)}$, $i\in V$: $w_i^{(1)} = \cdots = w_i^{(t-1)} = 0$, $w_i^{(t)} = e-u_i$; edge jobs $J_{\{i,j\}}$, $\{i,j\}\in E$: $w_{\{i,j\}} = 2$; $J_i^{(1)} < \cdots < J_i^{(t-1)} < J_i^{(t)} < J_{\{i,j\}}$, $i\in V$, $\{i,j\}\in E$; $y = \frac{1}{2}v(v+1)et + t(k+1)$. This completes the proof.

## 4. SINGLE MACHINE: DUE DATE CRITERIA

Let us now investigate the complexity of $1|\beta|\gamma|\sum w_iU_i$ and $1|\beta|\gamma|\sum w_iT_i$ problems. Even without precedence constraints, these problems can be quite difficult. *NP*-completeness for the $1||\sum w_iU_i$ and $1||\sum w_iT_i$ problems has been established in [19] and [30], respectively. In the case of unit weights, the $1||\sum U_i$ problem can be solved by an $O(n \log n)$ algorithm due to Moore and Hodgson [31], and the $1||\sum T_i$ problem by an $O(n^4 \sum_{i\in I} p_i)$ algorithm from Lawler [26]. This "pseudopolynomial" method is exponential with respect to the standard binary encoding of the problem data and can only be considered good in the formal sense if $p_i \in \{1, \cdots, P(n)\}$ for all $i\in I$ and a fixed polynomial $P$ in $n$. The complexity status of the general $1||\sum T_i$ problem remains a major open question. In the case of unit processing times, the $1|1|\sum w_iU_i$ problem is solvable by Moore's algorithm, as pointed out by Lawler [25], and the $1|1|\sum w_iT_i$ problem is easily seen to be a linear assignment problem and thereby solvable in $O(n^3)$ steps.

The introduction of general precedence constraints transforms both problems into *NP*-complete ones, even if $p_i = w_i = 1$ for all $i\in I$. The *NP*-completeness proofs for the $1|prec|1|\sum U_i$ and $1|prec|1|\sum T_i$ problems involve the CLIQUE problem, proved *NP*-complete in [7, 19].

CLIQUE: *Given an undirected graph $G=(V, E)$ and an integer $k$, does $G$ have a clique (i.e., a complete subgraph) on $k$ vertices?*

We assume that $V=\{1, \cdots, v\}$ and define $e=|E|$, $l=\frac{1}{2}k(k-1)$, $k'=v-k$, $l'=e-l$.

THEOREM 3. CLIQUE $\propto 1|prec|1|\sum U_i$.

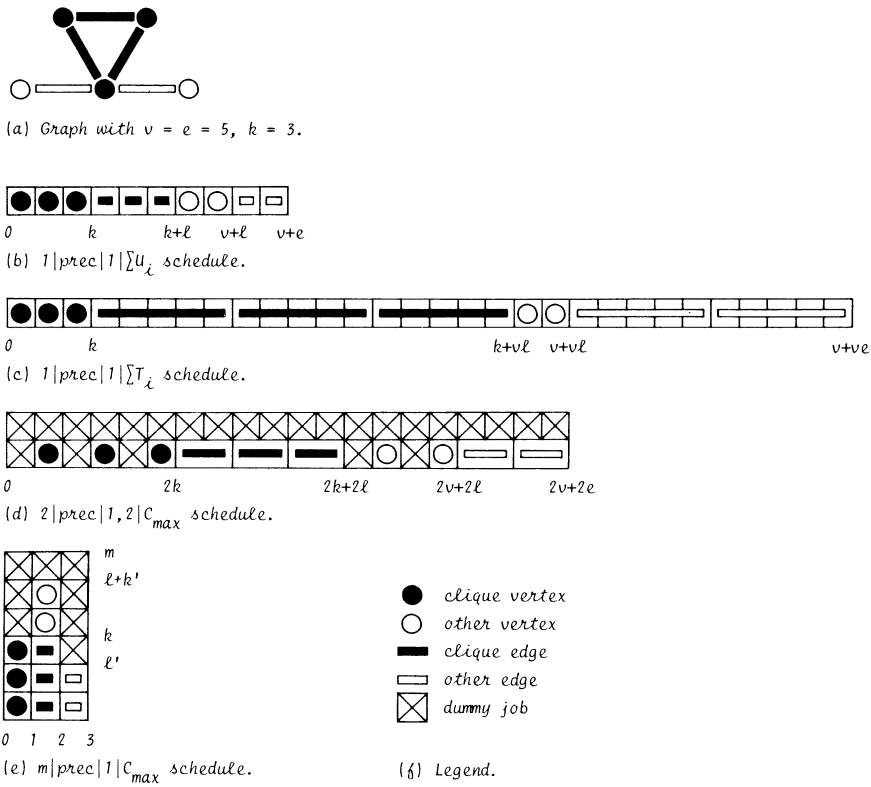


**Figure 2.** Illustration of reductions from CLIQUE.

*Proof.* This reduction is due to Garey and Johnson [12]. Given $G=(V, E)$ and $k$, we define the following $1|prec|1|\sum U_i$ problem: $n=v+e$; vertex jobs $J_i$, $i\in V$: $d_i=v+e$; edge jobs $J_{\{i,j\}}$, $\{i, j\}\in E$: $d_{\{i,j\}}=k+l$; $J_i<J_{\{i,j\}}$, $i\in V$, $\{i, j\}\in E$; $y=l'$.

If $G$ has a clique on a subset $K\subset V$ of $k$ vertices, we first schedule the $k$ vertex jobs $J_i$, $i\in K$, and are then free to schedule the $l$ corresponding edge jobs $J_{\{i,j\}}$, $i, j\in K$, before their due date. For any processing order of the remaining jobs, $l'$ edge jobs are late and we have a schedule with value $\sum U_i=l'=y$ (cf. Figure 2b). If $G$ has no clique on $k$ vertices, then at most $l-1$ edge jobs can be on time and we have for any schedule that $\sum U_i\geqq l'+1>y$.

THEOREM 4. CLIQUE $\propto 1|prec|1|\sum T_i$.

*Proof.* Ignoring the constraint on the processing times, we first define the following $1|prec||\sum T_i$ problem: $n=v+e$; vertex jobs $J_i$, $i\in V$: $p_i=1$, $d_i=v+lv$; edge jobs $J_{\{i,j\}}$, $\{i,j\}\in E$: $p_{\{i,j\}}=v$, $d_{\{i,j\}}=k+lv$; $J_i<J_{\{i,j\}}$, $i\in V$, $\{i,j\}\in E$; $y=l'k'+\frac{1}{2}l'(l'+1)v$.

Consider any processing order in which $J_h$ is the first late vertex job. If $J_h$ starts at $d_h=v+lv$, then a period of length $v+lv$ would have to be filled by edge jobs and between 1 and $v-1$ vertex jobs. This is impossible, and we conclude that $J_h$ starts after its due date and is therefore preceded directly by an edge job $J_{\{i,j\}}$. Interchanging $J_{\{i,j\}}$ and $J_h$ will maintain feasibility and decrease $\sum T_i$. Repeating such improvements, we eventually arrive at a schedule in which all vertex jobs are on time. Thus, we have for every feasible schedule that $\sum T_i \geqq \sum_{i=1}^{l'}(k'+iv)=y$. If CLIQUE has a solution, then this lower bound can be attained, as illustrated in Figure 2c. If CLIQUE has no solution, then at least $l'+1$ edge jobs are late and $\sum T_i \geqq 1+y>y$.

We transform this problem into one with unit processing times by replacing each edge job $J_{\{i,j\}}$ by a chain $J_{\{i,j\}}^{(1)}<\cdots<J_{\{i,j\}}^{(v-1)}<J_{\{i,j\}}^{(v)}$ with $d_{\{i,j\}}^{(1)}=\cdots=d_{\{i,j\}}^{v-1}=v+ev$, $d_{\{i,j\}}^{(v)}=k+lv$. By virtue of this choice of due dates, we may assume the jobs in such a chain to be scheduled consecutively. Thus, we have obtained an equivalent $1|prec|1|\sum T_i$ problem, which completes the proof.

We note that the complexity status of the $1|tree|1|\sum U_i$ and $1|tree|1|\sum T_i$ problems is unknown. However, if we allow nonequal *release dates* at which the jobs become available or *deadlines* at which the jobs have to be completed, the former problem has been proved *NP*-complete by reductions from SET PACKING [18] and NODE COVER respectively (see [19] for definitions of these problems).

## 5. MULTIPLE MACHINES

Finally, we will examine the complexity of $\alpha|\beta|\gamma|C_{\max}$ and $\alpha|\beta|\gamma|\sum w_iC_i$ problems for $\alpha=2$ and $\alpha=m$. The $2|||C_{\max}$ and $2|||\sum w_iC_i$ problems are easily seen to be *NP*-complete [3, 30]. The $m|||\sum C_i$ problem is solvable by an $O(n\log n)$ algorithm due to Conway, Maxwell and Miller [6].

If we allow precedence constraints and assume unit processing times, there are two important algorithmic results. First, the $2|prec|1|C_{\max}$ problem can be solved by an $O(n^2)$ algorithm due to Coffman and Graham [5]. Garey and Johnson have developed an $O(n^2)$ algorithm for the case in which each job has to meet a given deadline [12] and an $O(n^3)$ algorithm for the case in which both release dates and deadlines are specified [11]. Second, the $m|tree|1|C_{\max}$ problem can be solved by an $O(n)$ algorithm due to Hu [17]. Furthermore, the Coffman-Graham algorithm also pro-

duces an optimal $2|prec|1|\sum C_i$ schedule [9]; the optimal $m|tree|1|\sum C_i$ schedule, however, may be suboptimal with respect to the $C_{\max}$ criterion for $m \geqq 3$ [9].

In one of the first papers dealing with complexity of scheduling problems, Ullman [35] showed that these results are borderline cases in the sense that both the $2|prec|1, 2|C_{\max}$ and the $m|prec|1|C_{\max}$ problem are *NP*-complete. His proofs, however, are long and technically complicated, certainly compared to the reductions to be presented in Theorems 5 and 6. Theorems 7 and 8 extend *NP*-completeness to the $2|prec|1, 2|\sum C_i$ and $m|prec|1|\sum C_i$ problems. It has been brought to our attention that the latter result already appears in [4]. We shall again start with the CLIQUE problem from the previous section. It is one of the classical *NP*-complete problems and already appears as such in Cook's paper [7].

THEOREM 5. CLIQUE $\propto 2|prec|1, 2|C_{\max}$.

*Proof.* The basic idea behind the reduction is the following. We define vertex jobs $J_i$, $i \in V$, with $p_i = 1$, edge jobs $J_{\{i,j\}}$, $\{i, j\} \in E$, with $p_{\{i,j\}} = 2$, and the usual precedence constraints $J_i < J_{\{i,j\}}$, $i \in V$, $\{i, j\} \in E$. Furthermore, we introduce sets of dummy jobs such that in every feasible schedule with $C_{\max} \leqq y$, one machine can be assumed to process dummy jobs only, while on the other machine a pattern of idle time periods is created. This pattern can be filled properly if and only if CLIQUE has a solution. From left to right, it consists of $k$ separate unit periods for the clique vertex jobs, one period of length $2l$ for the clique edge jobs, $k'$ separate unit periods for the remaining vertex jobs, and one period of length $2l'$ for the remaining edge jobs.

More formally, given $G = (V, E)$ and $k$, we define the following $2|prec|1, 2|C_{\max}$ problem: $n = 4v + 3e$; vertex jobs $J_i$, $i \in V$: $p_i = 1$; edge jobs $J_{\{i,j\}}$, $\{i, j\} \in E$: $p_{\{i,j\}} = 2$; dummy jobs $J_{ht}$, $h \in D_t$, $t = 1, 2, \cdots, 2v+2e$, $D_t = \{1, 2\}$, $t = 1, 3, \cdots, 2k-1$ and $t = 2k+2l+1, 2k+2l+3, \cdots, 2v+2l-1$, $D_t = \{1\}$, otherwise; $p_{ht} = 1$; $J_i < J_{\{i,j\}}$, $i \in V$, $\{i, j\} \in E$; $J_{gt} < J_{h,t+1}$, $g \in D_t$, $h \in D_{t+1}$, $t = 1, 2, \cdots, 2v+2e-1$; $y = 2v+2e$. We may assume that the dummy jobs $J_{ht}$, $h \in D_t$, are completed at time $t$, since otherwise we clearly have $C_{\max} > 2v+2e = y$.

If $G$ has a clique on a subset $K \subset V$ of $k$ vertices, we complete the $k$ vertex jobs $J_i$, $i \in K$, at times $2, 4, \cdots, 2k$ and schedule the $l$ corresponding edge jobs $J_{\{i,j\}}$, $i, j \in K$, consecutively from $2k$ to $2k+2l$. The $k'$ remaining vertex jobs are completed at $2k+2l+2, 2k+2l+4, \cdots, 2v+2l$ and the $l'$ remaining edge jobs are scheduled consecutively from $2v+2l$ to $2v+2e = y$ (cf. Figure 2d). If $G$ has no clique on $k$ vertices, then at most $l-1$ edge jobs can be scheduled before time $2k+2l$ and we have for any schedule that $C_{\max} \geqq 2v+2e+1 > y$.

THEOREM 6. CLIQUE $\propto m|prec|1|C_{\max}$.

*Proof.* Following the same idea as for the previous reduction, we define vertex jobs, edge jobs and dummy jobs. The dummy jobs create an appropriate pattern of idle machines during three unit periods, available for vertex and edge jobs. CLIQUE has a solution if and only if we are able to schedule the clique vertex and edge jobs in the first and second period, respectively, and the remaining vertex and edge jobs in the second and third period, respectively. Thus, we define the following $m|prec|1|C_{\max}$ problem: $n=3m$; $m=\max\{k, l+k', l'\}+1$; vertex jobs $J_i$, $i\in V$; edge jobs $J_{\{i,j\}}$, $\{i,j\}\in E$; dummy jobs $J_{ht}$, $h\in D_t$, $t=1, 2, 3$: $D_1=\{1, \cdots, m-k\}$, $D_2=\{1, \cdots, m-(l+k')\}$, $D_3=\{1, \cdots, m-l'\}$; $J_i<J_{\{i,j\}}$, $i\in V$, $\{i,j\}\in E$; $J_{gt}<J_{h,t+1}$, $g\in D_t$, $h\in D_{t+1}$, $t=1, 2$; $y=3$. We may again assume that the dummy jobs $J_{ht}, h\in D_t$, are scheduled in period $t$.

If CLIQUE has a solution, we schedule the $k$ clique vertex jobs in period 1, the $l$ clique edge jobs and the $k'$ remaining vertex jobs in period 2 and the $l'$ remaining edge jobs in period 3 (cf. Figure 2e). If CLIQUE has no solution, then at most $l-1$ edge jobs can be scheduled in period 2; and since there are not enough vertex jobs to fill the idle machines in periods 1 and 2, we have for any schedule that $C_{\max}>3$.

Note that the problem of determining the existence of an $m|prec|1|C_{\max}$ schedule with $C_{\max}\leqq y$ is *NP*-complete for the case $y=3$, whereas there is an obvious $O(n^2)$ algorithm for the case $y=2$.

THEOREM 7. CLIQUE $\propto 2|prec|1, 2|\sum C_i$.

*Proof.* We start from the $2|prec|1, 2|C_{\max}$ problem constructed in the proof of Theorem 5. CLIQUE has a solution if and only if there exists a schedule with value $C_{\max}\leqq y=2v+2e$. Let $z$ denote the (fixed) value of the $\sum C_i$ criterion with respect to such a schedule. We now define a corresponding $2|prec|1, 2|\sum C_i$ problem by adding a chain of dummy jobs that has to follow all original jobs and defining a new threshold value $y'$: dummy jobs $J_{1t}$, $t=y+1, \cdots, z$: $p_{1t}=1$; $J_{\{i,j\}}<J_{1,y+1}$, $\{i,j\}\in E$; $J_{1,t-1}<J_{1t}$, $t=y+1, \cdots, z$; $y'=z+(z-y)y+\frac{1}{2}(z-y)(z-y+1)$.

The original problem has a schedule with value $C_{\max}\leqq y$ if and only if the new problem has a schedule with value $\sum C_i\leqq y'$:

$$C_{\max}\leqq y \Rightarrow \sum C_i\leqq z+\sum_{t=1}^{z-y}(y+t)=y';$$
$$C_{\max}>y \Rightarrow \sum C_i>y+\sum_{t=1}^{z-y}(y+1+t)=y'.$$

THEOREM 8. CLIQUE $\propto m|prec|1|\sum C_i$.

*Proof.* Taking the $m|prec|1|C_{\max}$ problem constructed in the proof of Theorem 6, we define a threshold value on $\sum C_i$ by $y'=6m$. CLIQUE has a solution if and only if $3m$ unit jobs can be scheduled within three unit periods on $m$ machines. This is possible if and only if there exists a schedule with $C_{\max}\leqq 3=y$ or, equivalently, with $\sum C_i\leqq 6m=y'$.

## 6. CONCLUDING REMARKS

Table I summarizes the complexity results discussed in Sections 3, 4, and 5. Columns and rows in the table indicate the various values of $\alpha$ and $\delta$, respectively; each entry specifies the values of $\beta$ and $\gamma$. An asterisk denotes that the corresponding $\alpha|\beta|\gamma|\delta$ problem is polynomially solvable; an exclamation mark indicates that it has been proved $NP$-complete.

The borderline between polynomially solvable and $NP$-complete problems is clearly visible in the table. As usual, the detailed problem classification employed in this paper also serves to reveal a large number of

TABLE I

SURVEY OF COMPLEXITY RESULTS

| | 1 machine | 2 machines | m machines |
|---|---|---|---|
| $C_{\max}$ | * *prec*\| | ! \|<br>* *prec*\|1<br>! *prec*\|1, 2 | * *tree*\|1<br>! *prec*\|1 |
| $\sum C_i$ | ! *prec*\|0, 1 | * *prec*\|1<br>! *prec*\|1, 2 | * \|<br>! *prec*\|1 |
| $\sum w_i C_i$ | * *tree*\|<br>! *prec*\|1 | | |
| $\sum T_i$ | * \|1, $\cdots$ , $P(n)$<br>! *prec*\|1 | | |
| $\sum U_i$ | * \|<br>! *prec*\|1 | | |

* Polynomially solvable problem.
! $NP$-complete problem.

challenging open problems. Among these we find prominent examples such as $1|||\sum T_i$ and $c|prec|1|C_{\max}$ for any constant $c \geqq 3$, and a large number of problems with precedence constraints of type *tree*, such as $m|tree|1|\sum C_i$, $1|tree|1|\sum T_i$ and $1|tree|1|\sum U_i$. No doubt the reader can easily add many other problems to this intriguing list.

We hope we have conveyed the flavor of some proof techniques that are currently used to settle such problems. The applicability of complexity theory in the area of combinatorial optimization should stimulate further research in this interface between computer science and operations research.

*Note added in proof.* The authors have recently established $NP$-completeness for the $1|$ *tree* $|\ 1\ |\ U_i$ problem, even for the case of *chain*-like precedence constraints.

## ACKNOWLEDGMENTS

We gratefully acknowledge many fruitful discussions with Eugene L. Lawler. This research was partially supported by grants from the Netherlands Organization for the Advancement of Pure Research (Z.W.O.) and the National Research Council of Canada.

## REFERENCES

1. D. Adolphson, "Single Machine Job Sequencing with Precedence Constraints," *SIAM J. Comput.* **6,** 40–54 (1977).
2. D. Adolphson and T. C. Hu, "Optimal Linear Ordering," *SIAM J. Appl. Math.* **25,** 403–423 (1973).
3. J. Bruno, E. G. Coffman, Jr., and R. Sethi, "Scheduling Independent Tasks to Reduce Mean Finishing Time," *Comm. ACM* **17,** 382–387 (1974).
4. J. Bruno and R. Sethi, "On the Complexity of Mean Flow-Time Scheduling," Computer Science Department, Pennsylvania State University, 1975.
5. E. G. Coffman, Jr. and R. L. Graham, "Optimal Scheduling for Two-Processor Systems," *Acta Informat.* **1,** 200–213 (1972).
6. R. W. Conway, W. L. Maxwell, and L. W. Miller, *Theory of Scheduling*, Addison-Wesley, Reading, Mass., 1967.
7. S. A. Cook, "The Complexity of Theorem-Proving Procedures," 151–158 in *Proc. 3rd Annual ACM Symp. Theory Comput.*, 1971.
8. M. R. Garey, "Optimal Task Sequencing with Precedence Constraints," *Discrete Math.* **4,** 37–56 (1973).
9. M. R. Garey and R. L. Graham, Private communications.
10. M. R. Garey and D. S. Johnson, "Complexity Results for Multiprocessor Scheduling under Resource Constraints," *SIAM J. Comput.* **4,** 397–411 (1975).
11. M. R. Garey and D. S. Johnson, "Two-Processor Scheduling with Start-Times and Deadlines," Bell Laboratories, Murray Hill, N.J., 1975.
12. M. R. Garey and D. S. Johnson, "Scheduling Tasks with Nonuniform Deadlines on Two Processors," *J. Assoc. Comput. Mach.* **23,** 461–467 (1976).
13. M. R. Garey and D. S. Johnson, "Approximation Algorithms for Combinatorial Problems: An Annotated Bibliography," in *Algorithms and Complexity: New Directions and Recent Results*, pp. 41–52, J. F. Traub (Ed.). Academic Press, New York, 1976.
14. M. R. Garey, D. S. Johnson, and R. Sethi, "The Complexity of Flowshop and Jobshop Scheduling," *Math. Opns. Res.* **1,** 117–129 (1976).
15. M. R. Garey, D. S. Johnson, and L. Stockmeyer, "Some Simplified *NP*-Complete Graph Problems," *Theoret. Comput. Sci.* **1,** 237–267 (1976).
16. W. A. Horn, "Single-Machine Job Sequencing with Treelike Precedence Ordering and Linear Delay Penalties," *SIAM J. Appl. Math.* **23,** 189–206 (1972).
17. T. C. Hu, "Parallel Sequencing and Assembly Line Problems," *Opns. Res.* **9,** 841–848 (1961).
18. H. Kise, T. Ibaraki, and H. Mine, "A Solvable Case of the One Machine

Scheduling Problem with Ready and Due Times," *Opns. Res.* **26,** 121–126 (1978).

19. R. M. KARP, "Reducibility Among Combinatorial Problems," in *Complexity of Computer Computations*, pp. 85–103, R. E. Miller and J. W. Thatcher (Eds.). Plenum Press, New York, 1972.
20. R. M. KARP, "On the Computational Complexity of Combinatorial Problems," *Networks* **5,** 45–68 (1975).
21. R. M. KARP, "The Probabilistic Analysis of Some Combinatorial Search Algorithms," in *Algorithms and Complexity: New Directions and Recent Results*, pp. 1–19, J. F. Traub (Ed.). Academic Press, New York, 1976.
22. D. KNUTH, Private communication to T. C. Hu, July 23, 1973.
23. B. J. LAGEWEG, E. L. LAWLER, AND J. K. LENSTRA, "Machine Scheduling Problems: Computations, Complexity and Classification," Report BN 30, Mathematisch Centrum, Amsterdam, 1976.
24. E. L. LAWLER, "Optimal Sequencing of a Single Machine Subject to Precedence Constraints," *Management Sci.* **19,** 544–546 (1973).
25. E. L. LAWLER, "Sequencing to Minimize the Weighted Number of Tardy Jobs," *Rev. Franç. Automat. Informat. Rech. Opnnelle.* **10.5** Suppl. 27–33 (1976).
26. E. L. LAWLER, "A 'Pseudopolynomial' Algorithm for Sequencing Jobs to Minimize Total Tardiness," *Ann. Discrete Math.* **1,** 331–342 (1977).
27. E. L. LAWLER, "Sequencing Jobs to Minimize Total Weighted Completion Time Subject to Precedence Constraints," Symposium on Algorithmic Aspects of Combinatorics, Qualicum Beach, May 1977; to appear in *Ann. Discrete Math.*
28. E. L. LAWLER, H. W. LENSTRA, A. H. G. RINNOOY KAN, AND T. J. WANSBEEK (Eds.), *Een Tuyltje Boskruyd*, Amsterdam, 1976.
29. J. K. LENSTRA, *Sequencing by Enumerative Methods*, Mathematical Centre Tract 69, Mathematisch Centrum, Amsterdam, 1977.
30. J. K. LENSTRA, A. H. G. RINNOOY KAN, AND P. BRUCKER, "Complexity of Machine Scheduling Problems," *Ann. Discrete Math.* **1,** 343–362 (1977).
31. J. M. MOORE, "An $n$ Job, One Machine Sequencing Algorithm for Minimizing the Number of Late Jobs," *Management Sci.* **15,** 102–109 (1968).
32. A. H. G. RINNOOY KAN, *Machine Scheduling Problems: Classification, Complexity and Computations*, Nijhoff, The Hague, 1976.
33. J. B. SIDNEY, "Decomposition Algorithms for Single-Machine Sequencing with Precedence Relations and Deferral Costs," *Opns. Res.* **23,** 283–298 (1975).
34. W. E. SMITH, "Various Optimizers for Single-Stage Production," *Naval Res. Logist. Quart.* **3,** 59–66 (1956).
35. J. D. ULLMAN, "*NP*-Complete Scheduling Problems," *J. Comput. System. Sci.* **10,** 384–393 (1975).