

## TWO OPEN PROBLEMS IN PRECEDENCE CONSTRAINED SCHEDULING

**J.K. Lenstra**

*Centre for Mathematics and Computer Science  
P.O. Box 4079, 1009 AB Amsterdam*

**A.H.G. Rinnooy Kan**

*Erasmus University  
P.O. Box 1738, 3000 DR Rotterdam*

The computational complexity of a machine scheduling problem can be affected in various ways if a partial order is imposed on the set of jobs that has to be executed. Some typical complexity results for such problems are discussed in the light of two prominent open problems in this area: the minimization of total tardiness for unit-time jobs on a single machine subject to chain-like precedence constraints, and the minimization of maximum completion time for unit-time jobs on three identical parallel machines subject to arbitrary precedence constraints.

### RESUME

La théorie de l'*ordonnement* traite de l'affectation temporelle de *resources* limitées à des *activités*. Dans ce contexte, des *ordres partiels* apparaissent de façon naturelle: un ordre partiel sur l'ensemble des activités impose des contraintes sur l'ordre dans lequel les activités doivent être exécutées et il réduit ainsi l'ensemble des affectations admissibles. L'intérêt du problème consiste à incorporer ces *contraintes de précédence* de manière aussi efficace que possible dans des algorithmes destinés à déterminer une affectation admissible qui soit optimale par rapport à un certain critère.

L'effet des contraintes de précédence peut être double. Si le problème sans contraintes peut être résolu par une méthode efficace, alors leur adjonction conduira à modifier l'algorithme. Dans certains cas, cette adaptation n'affecte pas l'efficacité de l'algorithme. Dans d'autres, l'efficacité est diminuée au point

que parfois la nouvelle méthode de résolution revient à énumérer entièrement les affectations admissibles. Si le problème sans contraintes de précédence est déjà si compliqué qu'une approche par énumération semble inévitable, on peut tabler sur le fait que l'addition de contraintes de précédence peut réduire le nombre d'affectations admissibles. Dans le premier cas, les contraintes rendent le problème plus difficile; dans le second, le problème devient un peu plus simple.

En termes de *complexité de calcul*, l'addition de contraintes de précédence peut transformer un problème *bien soluble* en un problème *NP-dur*, ou encore rendre un problème d'ordonnement *NP-dur* plus facile à résoudre en pratique.

Nous nous concentrerons sur le premier phénomène et nous l'illustrerons dans le cas où les ressources limitées correspondent à des *machines*  $M_1, \dots, M_m$  dont chacune peut traiter au plus l'une des activités ou *tâches*  $J_1, \dots, J_n$  à la fois. De nombreux problèmes spécifiques ont pu être formulés et étudiés dans ce cadre général.

Le *critère d'optimalité* qui doit être minimisé joue un rôle proéminent dans ce contexte. Pour chaque ordonnancement admissible conduisant à un temps de fin d'exécution  $C_j$  pour  $J_j$  ( $j = 1, \dots, n$ ), on fait l'hypothèse fondamentale que le critère est une fonction non décroissante de chacune des variables  $C_1, \dots, C_n$ . Nous en donnerons des exemples plus loin.

Parmi les caractéristiques relatives aux tâches qui définissent des types particuliers de problèmes, on peut avoir des *contraintes de précédence* de la forme  $J_j \rightarrow J_k$  c'est-à-dire que  $J_j$  doit être terminée avant que  $J_k$  puisse commencer. De telles contraintes ont été abondamment étudiées et certains types de contraintes de précédence ont été mis en évidence. En termes du graphe de précédence  $G$  dont les sommets sont  $1, \dots, n$  et dont les arcs sont les paires  $(j, k)$  telles que  $J_j \rightarrow J_k$ , on s'est surtout intéressé au cas où  $G$  est une collection de *chaînes*, une *forêt* ou un *graphe série-parallèle*. Beaucoup d'autres cas particuliers entre un graphe *sans arcs* et un *graphe arbitraire* ont aussi été examinés.

En général, l'effort principal a été mis sur la détermination d'une frontière aussi bien délimitée que possible entre les problèmes bien solubles et les problèmes *NP-durs*. Ceci a été fait en déterminant le cas le plus général de contraintes de précédence qui peuvent être traitées en temps polynomial et le cas le plus simple qui conduit à un problème *NP-dur*. Dans cette note, nous nous concentrerons sur deux problèmes ouverts connus: d'une part la minimisation du retard total pour des tâches de durée unité à exécuter sur une machine avec des contraintes de précédence définies par des chaînes; d'autre part, la minimisation du maximum des temps d'exécution sur trois machines parallèles identiques avec des contraintes de précédence quelconques. Les résultats connus pour des problèmes voisins seront passés en revue et généralisés à l'aide de deux nouvelles preuves de *NP-difficulté*.

## 1. INTRODUCTION

The theory of *scheduling* is concerned with the allocation over time of scarce *resources* to *activities*. In this context, *partial orders* arise in a natural fashion: a partial order on the activity set imposes constraints on the order in which the activities can be executed and as such delimits the set of feasible allocations. The challenge is to incorporate these *precedence constraints* as efficiently as possible in algorithms designed to determine a feasible allocation that is optimal with respect to some criterion.

The effect of precedence constraints can be twofold. If the problem without precedence constraints can be solved efficiently, their addition will generally require the algorithm to be adapted. In some cases, this adaptation does not affect the efficiency of the algorithm; in other cases, it does, possibly to the point that the new solution method amounts to complete enumeration of all feasible allocations. If the unconstrained problem is already so difficult in itself that an enumerative approach seems unavoidable, one may capitalize on the addition of precedence constraints by exploiting the fact that they reduce the number of feasible allocations. In the former case, precedence constraints make the problem harder to solve; in the latter case, it becomes a little easier.

The theory of *computational complexity* of combinatorial problems has served to formalize the preceding informal discussion. We will settle here for a very brief review of the main concepts of this theory and refer the reader for more details to [Cook 1971; Karp 1972] (the first two papers on the subject), [Garey & Johnson 1979] (a comprehensive textbook) and [Lawler & Lenstra 1982] (a survey likely to be readily available to the current readership).

The *size* of a combinatorial problem is defined as the number of bits needed to encode its data, and the *running time* of an algorithm as the number of elementary operations (such as additions and comparisons) required for its solution.

If a problem of size  $s$  can be solved by an algorithm with running time  $O(p(s))$  where  $p$  is a *polynomial* function, then the problem is said to be *well solvable*; there are good theoretical and practical justifications for this notion. Many problems have been shown to be well solvable, simply by the construction of a polynomial-time algorithm.

Only few problems have been proved to be *not* well solvable, but there is a large class of problems for which it is *strongly suspected* that this is indeed the case. These are the *NP-hard* problems, which share a notorious reputation for computational intractability as well as the property that a polynomial-time algorithm for any one of them would yield polynomial-time algorithms for all problems in an important subclass, the *NP-complete* problems — a very unlikely event.

One establishes *NP*-hardness of a problem  $P$  by taking another problem  $Q$  and showing that  $Q$  is *reducible* to  $P$  ( $Q \propto P$ ), i.e., that instance of  $Q$  a corresponding instance of  $P$  can be constructed in polynomial time such that solving the latter will solve the former as well. This implies that  $Q$  is a special case of  $P$ , and since  $Q$  is *NP*-hard,  $P$  is *NP*-hard to solve. This recipe obviously does not apply to the *first NP*-hardness proof — for [Cook 1971].)

Rephrased more formally, then, the addition of precedence constraints can turn a well-solvable problem into an *NP*-hard one, or may make an scheduling problem easier to solve in practice.

We shall focus on the former phenomenon, and illustrate it for a problem where the scarce resources correspond to *machines*  $M_1, \dots, M_m$ , each of which can handle at most one of the activities or *jobs*  $J_1, \dots, J_n$  at a time. In this general setting, many specific problem types have been formulated and studied. For a detailed problem classification and a survey of the known results in this area, we refer to [Graham *et al.* 1979; Lawler *et al.* 1982].

A prominent role in this classification is played by the *optimality criterion* to be minimized. With every feasible schedule leading to a *completion time* for  $J_j$  ( $j=1, \dots, n$ ), the basic assumption is that the criterion is a function  $C_1, \dots, C_n$ , nondecreasing in every variable. We shall encounter several examples below.

Among the various job characteristics that further specify a problem, there may be *precedence constraints* of the form  $J_j \rightarrow J_k$ , signifying that  $J_j$  has to be completed before  $J_k$  can start. Such constraints have long been a major research subject in the area, whereby several types of precedence constraints have been distinguished. In terms of the precedence graph  $G$  with vertices  $v \in \{1, \dots, n\}$  and arc set  $\{(j, k): J_j \rightarrow J_k\}$ , separate attention has been paid to the case that  $G$  is a collection of *chains*, a *forest*, or *series-parallel*. Many special cases in between an *empty* and an *arbitrary* arc set have been investigated as well.

In general, the effort has been to draw as sharp a borderline as possible between well-solvable and *NP*-hard problems, by identification of a general type of precedence constraints that can be coped with in polynomial time versus the simplest type that leads to *NP*-hardness. For a review of the results obtained so far, we refer to [Lawler & Lenstra 1982]. In this paper we concentrate on two prominent open problems in this area, while several known related results.

## 2. A SINGLE MACHINE PROBLEM

Let us assume that there is a *single* machine ( $m = 1$ ) and that each

jobs  $J_j$  ( $j = 1, \dots, n$ ) has to spend an uninterrupted *processing time* of  $p_j$  time units on the machine. Each  $J_j$  becomes available for processing at time 0 and incurs, upon its completion at time  $C_j$ , a *tardiness cost*  $T_j = \max\{0, C_j - d_j\}$ , where  $d_j$  is a given *due date*. The criterion to be minimized is the *total tardiness*  $\sum_{j=1}^n T_j$ .

This is perhaps the most notorious open problem in single machine scheduling theory. It can be solved by *dynamic programming* techniques in  $O(n^4 \sum p_j)$  time [Lawler 1977]; although the running time is obviously *exponential* in the problem size (which is  $O(\sum(\log p_j + \log d_j))$ ), the algorithm in question is called *pseudopolynomial* since the running time is *polynomial* in the problem data themselves.

We will concentrate on the special case of *unit-time* jobs, i.e.,  $p_j = 1$  ( $j = 1, \dots, n$ ). The cost  $c_{ij}$  of scheduling  $J_j$  in the  $i$ -th position is now given by  $c_{ij} = \max\{0, i - d_j\}$ , and the problem is to find a permutation  $\sigma$  of  $\{1, \dots, n\}$  minimizing  $\sum_{j=1}^n c_{\sigma(j)j}$ . If there are *no* precedence constraints, this is an ordinary *linear assignment* problem, which can be solved in  $O(n^3)$  time (see, e.g., [Lawler 1976]). If *arbitrary* precedence constraints between the jobs are allowed, the problem becomes *NP-hard* [Lenstra & Rinnooy Kan 1978]. It is not known, however, what the effect of *chain-like* precedence constraints is, and this is our first open problem:

*Given a directed graph  $G$  with vertex set  $\{1, \dots, n\}$  in which each vertex  $j$  has an associated integer  $d_j$ , indegree at most one and outdegree at most one, find a permutation  $\sigma$  of  $\{1, \dots, n\}$  satisfying  $\sigma(j) < \sigma(k)$  whenever  $(j, k)$  is an arc of  $G$ , such that  $\sum_{j=1}^n \max\{0, \sigma(j) - d_j\}$  is minimized.*

An optimality criterion related to the total tardiness  $\sum T_j$  is the *number of late jobs*  $\sum U_j$ , where  $U_j = 0$  if  $C_j \leq d_j$ ,  $U_j = 1$  if  $C_j > d_j$ . Since we know of no problem type for which minimizing  $\sum U_j$  is harder than minimizing  $\sum T_j$  and since the problem of minimizing  $\sum U_j$  for unit-time jobs on a single machine subject to chain-like precedence constraints is *NP-hard* [Lenstra & Rinnooy Kan 1980], the most plausible conjecture is that the above problem will eventually turn out to be *NP-hard*.

Three immediate generalizations of our open problem are worth considering:

- (1) The processing times  $p_j$  ( $j = 1, \dots, n$ ) are arbitrary nonnegative integers. The resulting problem is *NP-hard* (Theorem 1).
- (2) Each  $J_j$  ( $j = 1, \dots, n$ ) has to be completed no later than a given *deadline*  $e_j$  (not to be confused with the due date  $d_j$ ). This problem is *NP-hard* as well (Theorem 2).
- (3) Each  $J_j$  ( $j = 1, \dots, n$ ) becomes available for processing at a given *release*

date  $r_j$ . This problem is still open and, of course, also suspected to be *NP*-hard.

As a preparation for the proofs of Theorems 1 and 2, we recall an *NP*-hardness result for the *total weighted tardiness* criterion  $\sum_{j=1}^n w_j T_j$ , where  $w_j$  is a given *weight* of  $J_j$  ( $j = 1, \dots, n$ ).

**Lemma 1** [Lawler 1977; Lenstra *et al.* 1977]. *The problem of scheduling jobs with arbitrary processing times on a single machine in the absence of precedence constraints so as to minimize total weighted tardiness  $\sum w_j T_j$  is *NP*-hard.*

**Proof** [Lenstra & Rinnooy Kan 1980]. We have to show that a known *NP*-hard problem is reducible to the  $\sum w_j T_j$  problem. Our starting point will be the following *NP*-hard problem [Garey & Johnson 1979]:

**3-PARTITION:** Given a set  $S = \{1, \dots, 3t\}$  and positive integers  $a_1, \dots, a_{3t}$ ,  $b$  with  $\frac{1}{4}b < a_j < \frac{1}{2}b$  for all  $j \in S$  and  $\sum_{j \in S} a_j = tb$ , does  $S$  have a partition into  $t$  3-element subsets  $S_i$  such that  $\sum_{j \in S_i} a_j = b$  ( $i = 0, \dots, t-1$ )?

Given any instance of 3-PARTITION, we construct an instance of the  $\sum w_j T_j$  problems as follows:

- there are  $4t - 1$  jobs;
- for each  $j \in S$ , there is a job  $J_j$  with processing time  $p_j = a_j$ , due date  $d_j = 0$  and weight  $w_j = a_j$ ;
- for each  $i \in \{1, \dots, t-1\}$ , there is a job  $J'_i$  with processing time  $p'_i = 1$ , due date  $d'_i = i(b+1)$  and weight  $w'_i = 2$ .

We claim that 3-PARTITION has a solution if and only if there exists a schedule with value  $\sum w_j T_j \leq y$ , where  $y = \sum_{1 \leq j \leq k \leq 3t} a_j a_k + \frac{1}{2}(t-1)tb$ . This would imply that a polynomial-time algorithm for the  $\sum w_j T_j$  problem could be used to solve 3-PARTITION in polynomial time as well and therefore prove the theorem.

Let us first ignore the jobs  $J'_i$  ( $i = 1, \dots, t-1$ ). Since  $d_j = 0$  for all  $j \in S$ , we have  $\sum_{j \in S} w_j T_j = \sum_{j \in S} w_j C_j$ ; moreover, since  $p_j = w_j$  for all  $j \in S$ , the value of  $\sum_{j \in S} w_j C_j$  is not influenced by the ordering of  $S$ . That is, for any schedule of the jobs  $J_j$  ( $j \in S$ ) without machine idle time we have

$$\sum_{j \in S} w_j T_j = \sum_{1 \leq j \leq k \leq 3t} a_j a_k.$$

Let us now calculate the effect of inserting a job  $J'_i$  in such a schedule. Suppose that  $J'_i$  is completed at time  $C'_i$  and define  $L'_i = C'_i - d'_i$ . Since all jobs  $J_j$  ( $j \in S$ ) that are processed after  $J'_i$  are completed one time unit later, the value of  $\sum_{j \in S} w_j T_j$  is increased by the total weight of these jobs, and we have

$$\begin{aligned} \sum_{j \in S} w_j T_j + w'_i T'_i &= \sum_{1 \leq j \leq k \leq 3t} a_j a_k + tb + 1 - d'_i - L'_i + 2\max\{0, L'_i\} \\ &= \sum_{1 \leq j \leq k \leq 3t} a_j a_k + tb + 1 - d'_i + |L'_i|. \end{aligned}$$

More generally, insertion of all jobs  $J'_i$ , resulting in completion times  $C'_i = d'_i + L'_i$  ( $i = 1, \dots, t-1$ ) such that  $C'_{\tau(1)} < \dots < C'_{\tau(t-1)}$  for some permutation  $\tau$  of  $\{1, \dots, t-1\}$ , yields a schedule with value

$$\begin{aligned} \sum w_j T_j &= \sum_{1 \leq j \leq k \leq 3t} a_j a_k + \sum_{i=1}^{t-1} (tb + i - d'_{\tau(i)} + |L'_{\tau(i)}|) \\ &= y + \sum_{i=1}^{t-1} |L'_i|, \end{aligned}$$

irrespective of the permutation  $\tau$ .

It follows that a schedule has value  $\sum w_j T_j \leq y$  if and only if there is no idle time and moreover the jobs  $J'_i$  are completed at times  $C'_i = d'_i = i(b+1)$  ( $i = 1, \dots, t-1$ ). Such a schedule exists if and only if the jobs  $J_j$  ( $j \in S$ ) can be divided into  $t$  groups, each containing 3 jobs and requiring  $b$  units of processing time, i.e., if and only if 3-PARTITION has a solution.  $\square$

The proof of Lemma 1 provides the basis for our proofs of Theorems 1 and 2. We will specify reductions from 3-PARTITION to both  $\Sigma T_j$  problems in which the number of jobs created is  $O(tb)$  and  $O(tb^2)$  respectively. This may raise some eyebrows, as the size of an instance of 3-PARTITION is only  $O(t \log b)$ . However, 3-PARTITION has been shown to be *NP*-hard even when problem size is measured in a pseudopolynomial fashion as  $O(tb)$  [Garey & Johnson 1979], and hence the reductions below suffice to establish *NP*-hardness.

**Theorem 1.** *The problem of scheduling jobs with arbitrary processing times on a single machine subject to chain-like precedence constraints so as to minimize total tardiness  $\Sigma T_j$  is NP-hard.*

**Proof.** Given any instance of 3-PARTITION, we first construct an instance of the  $\Sigma w_j T_j$  problem as in the proof of Lemma 1 and then transform it into an instance of the  $\Sigma T_j$  problem with chain-like precedence constraints as follows. Each job  $J_j$  with processing time  $p_j$ , due date  $d_j$  and weight  $w_j$  (whether it is a "partition" job  $J_j$  ( $j \in S$ ) or a "splitting" job  $J'_i$  ( $i = 1, \dots, t-1$ )) is replaced by a chain of  $w_j$  unit-weight jobs. The first job in the chain has processing time  $p_j$  and due date  $d_j$ , the next  $w_j - 1$  ones have processing times 0 and due dates  $d_j$ .

The resulting problem instance has  $tb + 2(t - 1)$  jobs. Given any feasible schedule in which the jobs of some chain are not scheduled consecutively, one can obtain another schedule by processing all the zero-time jobs of that chain directly after its first job. This schedule is still feasible, and its  $\sum T_j$  value has not increased. Hence, each chain of length  $w_j$  can be considered as a single job with weight  $w_j$ , and we are back at our original construction.

The reader who dislikes zero-time jobs could quite easily replace them by unit-time jobs and multiply the lengths of the other jobs by a factor polynomial in  $t$  and  $b$  such that the equivalence argument still carries through.  $\square$

**Theorem 2.** *The problem of scheduling unit-time jobs on a single machine subject to arbitrary deadlines  $e_j$  and chain-like precedence constraints so as to minimize total tardiness  $\sum T_j$  is NP-hard.*

**Proof.** Our proof is again related to the proof of Lemma 1, although it is not such a straightforward extension as the proof of Theorem 1. Given any instance of 3-PARTITION, we construct an instance of the  $\sum T_j$  problem with unit-time jobs, deadlines and chain-like precedence constraints as follows:

- there are  $n = tb^2 + t - 1$  jobs;
- for each  $j \in S$ , there is a chain  $\bar{J}_j$  of  $ba_j$  unit-time jobs:

$$\bar{J}_j = J_j^{(1)} \rightarrow \cdots \rightarrow J_j^{(ba_j)},$$

with due dates and deadlines defined by

$$d_j^{(k)} = n \quad (k = 1, \dots, (b-1)a_j), \quad d_j^{(ba_j-l)} = -l \quad (l = a_j - 1, \dots, 0),$$

$$e_j^{(k)} = n \quad (k = 1, \dots, ba_j);$$

- for each  $i \in \{1, \dots, t-1\}$ , there is a unit-time job  $J'_i$  with due date and deadline defined by  $d'_i = e'_i = i(b^2 + 1)$ .

We claim that 3-PARTITION has a solution if and only if there exists a feasible schedule with value  $\sum T_j \leq z$ , where  $z = b \sum_{1 \leq j \leq k \leq 3t} a_j a_k + \frac{1}{2}(t-1)tb$ . Before we prove this claim, we make some introductory remarks about the way in which the job weights occurring in the proof of Lemma 1 have been simulated in the present construction. For each chain  $\bar{J}_j$  ( $j \in S$ ), the due dates have been specified such that in any schedule without machine idle time only the last  $a_j$  jobs in the chain contribute to the criterion; if all these jobs are completed one time unit later, this adds  $a_j$  units to  $\sum T_j$ , which corresponds to the original weight  $w_j = a_j$ . For each job  $J'_i$  ( $i = 1, \dots, t-1$ ), we previously used a weight  $w'_i = 2$  in combination with an upper bound  $y$  on  $\sum w_j T_j$  to enforce an implicit deadline  $d'_i$ ; we now simply have an explicit deadline  $e'_i = d'_i$ .

Consider any feasible schedule with value  $\sum T_j \leq z$ . Without loss of gen-



erality, we assume that the schedule contains no machine idle time, that each job  $J'_i$  ( $i = 1, \dots, t-1$ ) is completed at time  $d'_i$ , and that the chains  $\bar{J}_j$  ( $j \in S$ ) do not preempt each other; the latter two statements can be proved by means of simple interchange arguments. The jobs  $J'_i$  ( $i = 1, \dots, t-1$ ) do not contribute to the  $\sum T_j$  value of the schedule. The contribution of the chains  $\bar{J}_j$  ( $j \in S$ ) consists of two terms.

First, there is the total tardiness of all jobs in the chains when the chains are processed from time 0 onwards without interruption. It is not hard to see that this term is given by  $b \sum_{1 \leq j \leq k \leq t} a_j a_k$ , irrespective of the ordering of  $S$ .

Secondly, there is the increase in total tardiness due to the insertion of the jobs  $J'_i$  in the intervals  $[d'_i - 1, d'_i] = [d'_{i-1} + b^2, d'_i]$  ( $i = 1, \dots, t-1$ ), where  $d'_0 = 0$ . Let  $S_i \subset S$  denote the index subset of chains that are completed in the interval  $[d'_i, d'_i + b^2]$ , and let  $A_i = \sum_{j \in S_i} a_j$  ( $i = 0, \dots, t-1$ ). Note that  $bA_{t-1}$  is equal to the total length of all chains completed in the final interval  $[d'_{t-1}, d'_{t-1} + b^2]$ , so that  $A_{t-1} \geq b$ . More generally, we have that  $\sum_{h=i}^{t-1} A_h \geq ib$  ( $i = 1, \dots, t-1$ ). Since all chain lengths as well as the interval lengths are integer multiples of  $b$ , we know that, if  $j \in S_i$ , the last  $b$  jobs of  $\bar{J}_j$  and in particular the last  $a_j$  ones (the only ones that contribute to  $\sum T_j$ ) must be processed in  $[d'_i, d'_i + b^2]$ , so that  $\bar{J}_j$  contributes  $ia_j$  additional units to  $\sum T_j$ . Thus, the second term is given by

$$\sum_{i=0}^{t-1} iA_i = \sum_{i=1}^{t-1} \sum_{h=t-i}^{t-1} A_h \geq \sum_{i=1}^{t-1} ib = \frac{1}{2}(t-1)tb.$$

It follows that  $\sum T_j \leq z$  if and only if  $A_i = b$  ( $i = 0, \dots, t-1$ ), i.e., if and only if 3-PARTITION has a solution.  $\square$

### 3. A PARALLEL MACHINE PROBLEM

We now assume that there are  $m$  machines and  $n$  jobs  $J_j$  ( $j = 1, \dots, n$ ). The machines are *parallel* in the sense that each job can be assigned to any one of them, and they are *identical* in the sense that, when  $J_j$  is assigned to some machine, it requires an uninterrupted processing time  $p_j$ , irrespective of the machine. The criterion to be minimized is the *maximum completion time*  $C_{\max} = \max_{1 \leq j \leq n} \{C_j\}$ .

If *arbitrary* processing times are allowed, the problem is already *NP-hard* if  $m = 2$  and no precedence constraints are specified. This generalizes the PARTITION problem of splitting a set of numbers into two subsets with equal sums, which is known to be *NP-hard* [Karp 1972].

We will, once again, concentrate on the case of *unit-time* jobs. We first state three classical results on minimizing  $C_{\max}$  for unit-time jobs on  $m$  identi-

cal parallel machines subject to precedence constraints, specified in the form of a directed graph  $G$ :

- (1) If  $m$  is arbitrary (i.e., specified as part of the problem instance) and  $G$  is an *inforest* (each vertex has outdegree at most one) or an *outforest* (each vertex has indegree at most one), the problem is solvable in  $O(n)$  time [Hu 1961].
- (2) If  $m = 2$  and  $G$  is arbitrary, the problem is also well solvable; algorithms that have subsequently been developed require  $O(n^3)$  time [Fujii *et al.* 1969, 1971],  $O(n^2)$  time [Coffman & Grahman 1972], "almost linear" time [Gabow 1982], and  $O(n)$  time [Gabow & Tarjan 1983].
- (3) If  $m$  and  $G$  are arbitrary, the problem is *NP*-hard [Ullman 1975].

These results do not resolve the complexity status of the problem if  $G$  is arbitrary and  $m$  is fixed but greater than 2. In particular, the case that  $m = 3$  has withstood all attacks, and this is our second open problem:

*Given a directed graph  $G$  with vertex set  $\{1, \dots, n\}$ , find the minimum value of  $C$  for which there exists a function  $\sigma: \{1, \dots, n\} \rightarrow \{1, \dots, C\}$  satisfying  $\sigma(j) < \sigma(k)$  whenever  $(j, k)$  is an arc of  $G$  and  $|\{j \in \{1, \dots, n\} : \sigma(j) = t\}| \leq 3$  for all  $t \in \{1, \dots, C\}$ .*

In the course of research on this problem, progress has been made for several special types of precedence constraints other than forests. We mention the following results.

- (4) Let the *height*  $h$  of  $G$  be defined as the number of arcs in a longest path in  $G$ . If  $m$  is arbitrary and  $h = 2$ , the problem is still *NP*-hard, and there exists no polynomial-time (approximation) algorithm that guarantees a relative error less than one third of the optimal  $C_{\max}$  value unless all *NP*-complete problems are well solvable [Lenstra & Rinnooy Kan 1978]. If both  $m$  and  $h$  are fixed, the problem is well solvable in  $O(n^{h(m-1)+1})$  time [Dolev & Warmuth 1982B].
- (5) Suppose  $G$  is an *interval order*: each vertex  $j$  corresponds to an interval  $[a_j, b_j]$  on the real line and  $(j, k)$  is an arc of  $G$  whenever  $b_j < a_k$ . In this case, the problem is solvable in  $O(n^2)$  time [Papadimitriou & Yannakakis 1979].
- (6) Suppose  $G$  is a *level order*: any two incomparable vertices with a common predecessor or successor have identical sets of predecessors and successors. If  $m$  is fixed, this problem is well solvable in  $O(n^{m-1})$  time [Dolev & Warmuth 1982C].
- (7) Suppose  $G$  is an *opposing forest*, consisting of the disjoint union of an inforest and an outforest. If  $m$  is arbitrary, this problem is *NP*-hard [Garey *et al.* 1983, Mayr 1981]. If  $m$  is fixed, it is well solvable in  $O(n^{2m-2} \log n)$  time [Dolev & Warmuth 1982C]. If  $m = 3$ , there is an

$O(n)$  algorithm [Garey *et al.* 1983; Dolev & Warmuth 1982A].

These results give little additional insight into the complexity status of the three-machine problem with arbitrary precedence constraints. Recent rumors on a proof of its well-solvability have not been substantiated so far, but such a proof should be extendable to the case that  $m$  is any fixed constant. We believe that the problem stands a good chance to be the seventh one to be removed from the list of twelve open problems in [Garey & Johnson 1979].

#### 4. CONCLUDING REMARKS

The above discussion has illustrated that very detailed insights exist on the way in which partial orders on the job set affect the computational complexity of machine scheduling problems. The two problems considered in the preceding two sections figure prominently on the list of open problems that is produced by the computer program MSPCLASS [Lageweg *et al.* 1981,1982]. This program keeps track of the complexity status of 4,536 machine scheduling problems, 390 of which are currently still open. Resolution of many of these problems, in particular of the two above ones, would seem to require the development of new algorithmic approaches or transformation techniques.

#### ACKNOWLEDGEMENTS

We are grateful to D. de Werra who translated our extended abstract into a *résumé détaillé*. Our research was supported by NSF grant MCS78-20054.

#### REFERENCES

- E.G. COFFMAN, JR. & R.L. GRAHAM (1972) Optimal scheduling for two-processor systems. *Acta Informat.* 1, 200-213.
- S.A. COOK (1971) The complexity of theorem-proving procedures. *Proc. 3rd Annual ACM Symp. Theory of Computing*, 151-158.
- D. DOLEV & M.K. WARMUTH (1982A) Scheduling flat graphs. Research Report RJ3398, IBM, San Jose, CA.
- D. DOLEV & M.K. WARMUTH (1982B) Scheduling precedence graphs of bounded height. Research Report RJ3399, IBM, San Jose, CA; *J. Algorithms*, to appear.
- D. DOLEV & M.K. WARMUTH (1982C) Profile scheduling of opposing forests and level orders. Research Report RJ 3553, IBM, San Jose, CA.

- M. FUJII, T. KASAMI & K. NINOMIYA (1969,1971) Optimal sequencing of two equivalent processors. *SIAM J. Appl. Math.* 17, 784-789. Erratum. 20, 141.
- H.N. GABOW (1982) An almost-linear algorithm for two-processor scheduling. *J. Assoc. Comput. Mach.* 29, 766-780.
- H.N. GABOW & R.E. TARJAN (1983) A linear-time algorithm for a special case of disjoint set union. *Proc. 15th Annual ACM Symp. Theory of Computing*, 246-251.
- M.R. GAREY & D.S. JOHNSON (1979) *Computers and Intractability: a Guide to the Theory of NP-Completeness*, Freeman, San Francisco.
- M.R. GAREY, D.S. JOHNSON, R.E. TARJAN & M. YANNAKAKIS (1983) Scheduling opposing forests. *SIAM J. Algebraic Discrete Methods* 4, 72-93.
- R.L. GRAHAM, E.L. LAWLER, J.K. LENSTRA & A.H.G. RINNOOY KAN (1979) Optimization and approximation in deterministic sequencing and scheduling: a survey. *Ann. Discrete Math.* 5, 287-326.
- T.C. HU (1961) Parallel sequencing and assembly line problems. *Oper. Res.* 9, 841-848.
- R.M. KARP (1972) Reducibility among combinatorial problems. In: R.E. MILLER, J.W. THATCHER (eds.) (1972) *Complexity of Computer Computations*, Plenum, New York, 85-103.
- B.J. LAGEWEG, E.L. LAWLER, J.K. LENSTRA & A.H.G. RINNOOY KAN (1981) Computer aided complexity classification of deterministic scheduling problems. Report BW 138, Mathematisch Centrum, Amsterdam.
- B.J. LAGEWEG, J.K. LENSTRA, E.L. LAWLER & A.H.G. RINNOOY KAN (1982) Computer aided complexity classification of combinatorial problems. *Comm. ACM* 25, 817-822.
- E.L. LAWLER (1976) *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart & Winston, New York.
- E.L. LAWLER (1977) A "pseudopolynomial" algorithm for sequencing jobs to minimize total tardiness. *Ann. Discrete Math.* 1, 331-342.
- E.L. LAWLER & J.K. LENSTRA (1982) Machine scheduling with precedence constraints. In: I. RIVAL (ed.) (1982) *Ordered Sets*, Reidel, Dordrecht, 655-675.
- E.L. LAWLER, J.K. LENSTRA & A.H.G. RINNOOY KAN (1982) Recent developments in deterministic sequencing and scheduling: a survey. In: M.A.H. DEMPSTER, J.K. LENSTRA & A.H.G. RINNOOY KAN

- (eds.) (1982) *Deterministic and Stochastic Scheduling*, Reidel, Dordrecht, 35-73.
- J.K. LENSTRA & A.H.G. RINNOOY KAN (1978) Complexity of scheduling under precedence constraints. *Oper. Res.* 26, 22-35.
- J.K. LENSTRA & A.H.G. RINNOOY KAN (1980) Complexity results for scheduling chains on a single machine. *European J. Oper. Res.* 4, 270-275.
- J.K. LENSTRA, A.H.G. RINNOOY KAN & P. BRUCKER (1977) Complexity of machine scheduling problems. *Ann. Discrete Math.* 1, 343-362.
- E. MAYR (1981) Well structured parallel programs are not easier to schedule. Report STAN-CS-81-880, Department of Computer Science, Stanford University.
- C.H. PAPADIMITRIOU & M. YANNAKAKIS (1979) Scheduling interval-ordered tasks. *SIAM J. Comput.* 8, 405-409.
- J.D. ULLMAN (1975) NP-Complete scheduling problems. *J. Comput. System Sci.* 10, 384-393.