5tat.

Netherlands school of economics.

ECONOMETRIC INSTITUTE

COMPLEXITY RESULTS FOR SCHEDULING CHAINS ON A SINGLE MACHINE

J.K. LENSTRA and A.H.G. RINNOOY KAN

REPORT 7907/O

ERASMUS UNIVERSITY ROTTERDAM, P.O. BOX 1738, ROTTERDAM, THE NETHERLANDS

COMPLEXITY RESULTS FOR SCHEDULING CHAINS ON A SINGLE MACHINE

J.K. LENSTRA

Mathematisch Centrum, Amsterdam

A.H.G. RINNOOY KAN

Erasmus University, Rotterdam

ABSTRACT

We investigate the computational complexity of deterministic sequencing problems in which unit-time jobs have to be scheduled on a single machine subject to chain-like precedence constraints. NP-hardness is established for the cases in which the number of late jobs or the total weighted tardiness is to be minimized, and for several related problems involving the total weighted completion time criterion.

KEY WORDS & PHRASES: *NP-hardness, single machine, unit-time jobs, chain-like precedence constraints, number of late jobs, total weighted tardiness, total weighted completion time.*

NOTE: This report is not for review; it has been submitted for publication in a journal.

1. INTRODUCTION

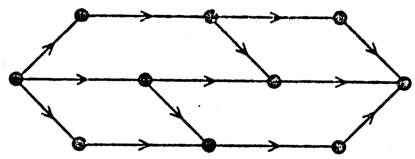
The theory of the computational complexity of combinatorial problems has been applied on various occasions to provide fundamental insights into their inherent difficulty, notably in the area of sequencing and scheduling [7]. Rather than reviewing this theory in detail, we refer to [9;15] for informal introductions and to [6] for a thorough exposition. Suffice it to say that the theory has allowed the identification of a large class of *NP-complete* problems, with the following two important properties:

(i) no NP-complete problem is known to be *easy*, i.e., solvable by an algorithm whose running time is bounded by a polynomial function of problem size;

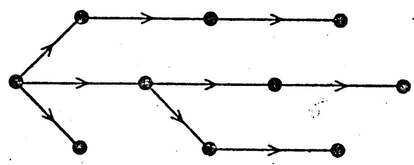
(ii) if any NP-complete problem would turn out to be easy, then they would all be easy.

All these problems are *recognition* problems, which require a yes/no answer. The *optimization* problems that correspond to many of them are at least as difficult and will be called *NP-hard*. Many notorious problems such as 0-1 programming, traveling salesman, plant location and job shop scheduling problems are NP-hard. Hence, establishing NP-hardness of a problem yields strong circumstantial evidence against the existence of a polynomial-time algorithm for its solution. This makes it easier to accept the inevitability of tedious enumerative optimization methods or of fast approximation algorithms.

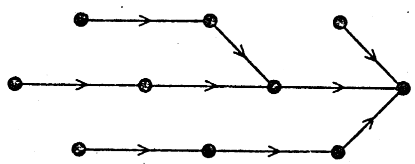
In this paper we shall be mainly concerned with the complexity of scheduling unit-time jobs on a single machine subject to chain-like precedence constraints. The scheduling model is defined as follows. There are n *jobs* $J_1,\ldots,J_n$ that have to be processed on a *single machine*. The machine can execute at most one job at a time; each job is available at time zero and requires *one unit of uninterrupted processing time*. The ordering of the jobs has to respect a given *precedence relation* $\rightarrow$. This relation is derived from an acyclic directed graph with vertices corresponding to jobs; if there is a directed path from J_j to J_k, we write $J_j \rightarrow J_k$ and require that J_j is completed before J_k can start. Some important types of precedence relations are defined and illustrated in Figure 1. We shall assume that the constraints are *chain*-like, i.e., each job has at most one immediate predecessor and at



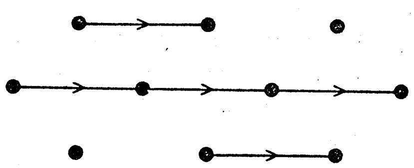
(a) *arbitrary.*



(b) *outtree*: each vertex has indegree at most one.



(c) *intree*: each vertex has outdegree at most one.



(d) *chain*: each vertex has indegree at most one and outdegree at most one.

Figure 1 Types of precedence relations.

most one immediate successor.

Each feasible schedule defines a *completion time* C_j for J_j $(j = 1,\ldots,n)$. The *optimality criteria* that will be considered are all nondecreasing functions of $C_1,\ldots,C_n$. Given a *due date* d_j and a *weight* w_j for each J_j, we define its *tardiness* $T_j = \max\{0, C_j - d_j\}$ and its *unit penalty* $U_j = 1$ if $C_j > d_j$, $U_j = 0$ otherwise, and we may require the minimization of the *total weighted completion time* $\sum w_jC_j$, the *total weighted tardiness* $\sum w_jT_j$, the *total tardiness* $\sum T_j$, or the *number of late jobs* $\sum U_j$.

In Sections 2 and 3 we establish NP-hardness for the minimization of $\sum U_j$ or $\sum w_jT_j$ in the described model. Weaker results for the $\sum U_j$ criterion have been reported in [4;8]; the case of the $\sum T_j$ criterion remains open. In

Section 4 we prove NP-hardness for the minimization of $\sum w_j C_j$ in various related scheduling environments. In Section 5 we summarize our results in the compact notation of [7] and offer some concluding remarks.

2. THE NUMBER OF LATE JOBS

The main result of this paper concerns the minimization of $\sum U_j$ on a single machine.

THEOREM 1. *The problem of scheduling unit-time jobs on a single machine subject to chain-like precedence constraints so as to minimize $\sum U_j$ is NP-hard.*

The case in which there are no precedence constraints but arbitrary processing times is solvable in $O(n \log n)$ time [19]. Thus, imposition of a very simple type of precedence relation on the jobs has a dramatic effect on the computational complexity of the problem. Theorem 1 dominates previous NP-hardness results for the case of arbitrary precedence constraints [4] and for the case of chain-like constraints and arbitrary release dates (i.e., lower bounds on the starting times of the jobs) [8].

Proof of Theorem 1. We have to show that some known NP-complete problem is reducible to the $\sum U_j$ problem. Our starting point will be the following NP-complete problem [9;6;16]:

> SET 3-PARTITION: Given a set $S = \{1,\ldots,3t\}$ and a family $\underline{\underline{S}} = \{S_1,\ldots, S_s\}$ of 3-element subsets of S, does $\underline{\underline{S}}$ include a partition if S, i.e., a subfamily of t subsets such that each element in S is contained in exactly one of them?

Given any instance of SET 3-PARTITION, we construct an instance of the $\sum U_j$ problem, but with nonequal processing times, as follows:

- there are $4s$ jobs;
- for each occurrence of an element $j \in S$ in a subset $S_i \in \underline{\underline{S}}$, there is a job J_{ij} with processing time $p_{ij} = sj$ and due date $d_{ij} = t+\frac{1}{2}sj(j+1)$ $(j \in S_i,\ i = 1,\ldots,s)$;
- for each subset $S_i \in \underline{\underline{S}}$, there is a job J_i with processing time $p_i = 1$ and due date $d_i = d$, where $d = s + s\sum_{i=1}^{s}\sum_{j\in S_i} j$ $(i = 1,\ldots,s)$;
- for each subset $S_i = \{j,j',j''\} \in \underline{\underline{S}}$, where $j < j' < j''$, there are chain-like precedence constraints $J_i \to J_{ij} \to J_{ij'} \to J_{ij''}$ $(i = 1,\ldots,s)$.

We shall prove the following propositions.

1(a) This problem can be polynomially transformed into an equivalent problem with unit processing times.

1(b) SET 3-PARTITION has a solution if and only if there exists a feasible schedule with value $\sum U_j \le 3(s-t)$.

Propositions 1(a) and 1(b) together imply Theorem 1.

Proof of Proposition 1(a). We replace each job J_{ij} by a chain $J_{ij}^{(1)} \to \ldots \to J_{ij}^{(p_{ij})}$ of unit-time jobs with due dates $d_{ij}^{(1)} = \ldots = d_{ij}^{(p_{ij}-1)} = d$, $d_{ij}^{(p_{ij})} = d_{ij}$ $(j \in S_i,\ i = 1,\ldots,s)$. The resulting problem has d unit-time jobs. Given any feasible schedule in which $J_{ij}^{(1)},\ldots,J_{ij}^{(p_{ij})}$ are not scheduled consecutively, we can obtain another schedule by moving $J_{ij}^{(1)},\ldots,J_{ij}^{(p_{ij}-1)}$ to the right, up to $J_{ij}^{(p_{ij})}$, thereby moving some other jobs to the left. This schedule is still feasible, since no precedence constraints are violated, and it has no more late jobs, due to our choice of due dates. Hence, each chain $J_{ij}^{(1)} \to \ldots \to J_{ij}^{(p_{ij})}$ can be considered as a single job J_{ij} with processing time p_{ij} and due date d_{ij}.

Proof of Proposition 1(b). Suppose that SET 3-PARTITION has a solution, i.e., $\underline{\underline{S}}$ includes a partition $\underline{\underline{S}}'$ of S. A feasible schedule in which no more than 3(s-t) jobs are late is then obtained as follows (cf. Figure 2). First, the t "subset jobs" J_i with $S_i \in \underline{\underline{S}}'$ are scheduled in the interval [0,t]. For each element $j \in S$, it is now possible to select exactly one "occurrence job" from $\underline{\underline{J}}_j = \{J_{ij'} | j' = j,\ i = 1,\ldots,s\}$ that is preceded by one of these subset

SET 3-PARTITION instance with t = 2, s = 4:

S	1	2	3	4	5	6
S_1	1	2		4		
S_2		2	3		5	
S_3		2		4	5	
S_4			3		5	6

partition of S: $\{S_1, S_4\}$

feasible schedule with 3(s-t) late jobs:

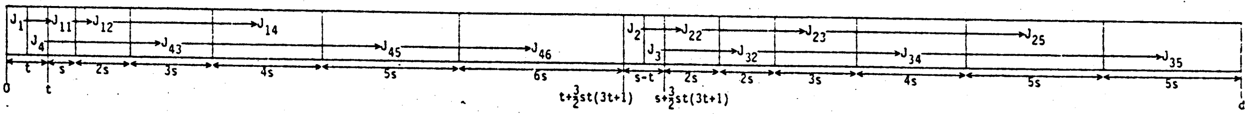


Figure 2 Illustration of the reduction of SET 3-PARTITION to the $\sum U_j$ problem.

jobs, and to schedule it in the interval $[t+\frac{1}{2}s(j-1)j, t+\frac{1}{2}sj(j+1)]$; in this way, 3t occurrence jobs are completed at their due dates. Finally, the remaining s-t subset jobs are scheduled in $[t+\frac{3}{2}st(3t+1), s+\frac{3}{2}st(3t+1)]$ and the remaining 3(s-t) occurrence jobs in $[s+\frac{3}{2}st(3t+1), d]$; the latter occurrence jobs are late.

Conversely, suppose that there exists a feasible schedule in which at most 3(s-t) jobs are late, or, equivalently, in which at least 3t occurrence jobs are on time. It will be shown below that this implies that exactly one job from each set $\underline{\underline{J}}_j$ ($j \in S$) is on time. This, in turn, implies that the amount of time available for processing subset jobs that precede at least one of these occurrence jobs is bounded from above by

$$\max_{j\in S}\{d_{ij}\} - \sum_{j\in S} p_{ij} = t + \frac{3}{2}st(3t+1) - \frac{3}{2}st(3t+1) = t.$$

The subsets corresponding to these jobs constitute a subfamily $\underline{S}' \subset \underline{S}$ of size at most t such that each element in S is contained in at least one of them. Hence, $\underline{S}'$ defines a partition of S.

It remains to be shown that if 3t occurrence jobs are on time, then exactly one job from each set $\underline{\underline{J}}_j$ ($j \in S$) is on time. It is clearly sufficient to prove that the following assertion A(j) holds for $j = 1,\ldots,3t$.

A(j): If j occurrence jobs are on time and completed not later than $t+\frac{1}{2}sj(j+1)$, then exactly one job from the set $\underline{\underline{J}}_k$ is on time, for $k = 1,\ldots,j$.

Note that A(j) implies that no set of j on-time jobs can be completed before $\sum_{k=1}^{j} sk = \frac{1}{2}sj(j+1)$.

Obviously, A(1) and A(2) are true. We will show that A(1),...,A(j-1) together imply A(j). Suppose that j jobs are on time and completed not later than $t+\frac{1}{2}sj(j+1)$. Let x ($0 \le x \le j$) of these jobs belong to $\underline{\underline{J}}_j$. If $x = 0$, then at least one job from a set $\underline{\underline{J}}_k$ with $k \ge j+1$ has to be completed not later than $t+\frac{1}{2}sj(j+1)$, and j-1 other jobs have to be on time and completed not later than

$$t + \tfrac{1}{2}sj(j+1) - s(j+1) = t + \tfrac{1}{2}s(j-1)j - s < \tfrac{1}{2}s(j-1)j.$$

A(j-1) implies that this is impossible. It follows that $x \ge 1$, and j-x other jobs have to be on time and completed not later than

$$t + \tfrac{1}{2}sj(j+1) - xsj = t + \tfrac{1}{2}s(j-x)(j-x+1) - \tfrac{1}{2}s(x-1)x$$

$$\begin{cases} = t + \tfrac{1}{2}s(j-1)j & \text{for } x = 1, \\ < \tfrac{1}{2}s(j-x)(j-x+1) & \text{for } x \geq 2. \end{cases}$$

If $x \geq 2$, $A(j-x)$ implies that this is impossible. It follows that $x = 1$, and $A(j-1)$ asserts that exactly one job from the set $\underline{\underline{J}}_k$ is on time, for $k = 1,\ldots,j-1$. This is equivalent to $A(j)$. □

3. TOTAL WEIGHTED TARDINESS

We next consider the minimization of $\sum w_jT_j$ on a single machine.

THEOREM 2. *The problem of scheduling unit-time jobs on a single machine subject to chain-like precedence constraints so as to minimize* $\sum w_jT_j$ *is NP-hard.*

The case in which there are no precedence constraints is simply solvable as a linear assignment problem in $O(n^3)$ time [7]; for arbitrary processing times it is NP-hard [18;17;12]. When all weights are equal, the problem of Theorem 2 is NP-hard for arbitrary precedence constraints [14], but the case of chain-like constraints remains open. We strongly suspect that even this problem is NP-hard: minimizing $\sum T_j$ seems much harder than minimizing $\sum U_j$, and so far all complexity results have confirmed this intuition.

Proof of Theorem 2. Our proof is of the same form as the proof of Theorem 1. We will start from the following NP-complete problem [6]:

3-PARTITION: Given a set $S = \{1,\ldots,3t\}$ and positive integers $a_1,\ldots,a_{3t}$, b with $\frac{b}{4} < a_j < \frac{b}{2}$ 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 $\sum_{j\in S_i} a_j = b$ $(i = 1,\ldots,t)$?

Given any instance of 3-PARTITION, we construct an instance of the $\sum w_jT_j$ problem, again with nonequal processing times, 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 $i \in \{1,\ldots,t-1\}$, there is a job J_i' with processing time $p_i' = 1$, due date $d_i' = i(b+1)$ and weight $w_i' = 2$;
- there are no precedence constraints.

It clearly suffices to prove the following propositions.

2(a) This problem can be polynomially transformed into an equivalent problem with unit processing times and chain-like precedence constraints.

2(b) 3-PARTITION has a solution if and only if there exists a schedule with value $\sum w_jT_j \le y$, where $y = \sum_{1\le j\le k\le 3t} a_ja_k + \frac{1}{2}(t-1)tb$.

It is easily verified that the entire transformation is polynomial-bounded. This crucially depends on the fact that 3-PARTITION is NP-complete even when the numerical problem data are encoded in unary rather than binary, i.e., when the problem size is $O(tb)$ instead of $O(t \log b)$ [5].

Proof of Proposition 2(a). We replace each job J_j by a chain $J_j^{(1)} \rightarrow \ldots \rightarrow J_j^{(p_j)}$ of unit-time jobs with due dates $d_j^{(1)} = \ldots = d_j^{(p_j)} = d_j$ and weights $w_j^{(1)} = \ldots = w_j^{(p_j-1)} = 0$, $w_j^{(p_j)} = w_j$ $(j \in S)$. As in the proof of Proposition 1(a), we can apply a simple interchange argument to show that, due to our choice of weights, each chain $J_j^{(1)} \rightarrow \ldots \rightarrow J_j^{(p_j)}$ can be considered as a single job J_j with processing time p_j, due date d_j and weight w_j.

Proof of Proposition 2(b). Let us first ignore the jobs J_i' $(i = 1,\ldots, t-1)$. Since $d_j = 0$ for all $j \in S$, we have $\sum_{j\in S} w_jT_j = \sum_{j\in S} w_jC_j$; moreover, since $p_j = w_j$ for all $j \in S$, the value of $\sum_{j\in S} w_jC_j$ is not influenced by the ordering of S [2]. It follows that for any schedule of the jobs J_j $(j \in S)$ without machine idle time we have

$$\sum_{j\in S} w_jT_j = \sum_{1\le j\le k\le 3t} a_ja_k.$$

Let us now calculate the effect of inserting job J_1' in such a schedule. Suppose that J_1' is completed at time C_1' and define $L_1' = C_1'-d_1'$. Since all jobs J_j $(j \in S)$ that are processed after J_1' are completed one time unit later, the value of $\sum_{j\in S} w_jT_j$ is increased by the total weight of these jobs. It follows that

$$\begin{aligned}\sum_{j\in S} w_jT_j + w_1'T_1' &= \left(\sum_{1\le j\le k\le 3t} a_ja_k + (t-1)b - L_1'\right) + 2\max\{0,L_1'\}\\ &= \sum_{1\le j\le k\le 3t} a_ja_k + \left((t-1)b + |L_1'|\right).\end{aligned}$$

It is easily seen that insertion of all jobs J_i' resulting in completion times $C_i' = d_i'+L_i'$ $(i = 1,\ldots,t-1)$ yields a schedule with value

$$\begin{aligned}\sum w_jT_j &= \sum_{1\le j\le k\le 3t} a_ja_k + \sum_{i=1}^{t-1}\left((t-i)b + |L_i'|\right)\\ &= y + \sum_{i=1}^{t-1}|L_i'|.\end{aligned}$$

It follows that a schedule has value $\sum w_jT_j \le y$ if and only if there is no idle time and moreover the jobs J_i' are completed at times $C_i' = d_i' = i(b+1)$ $(i = 1,\ldots,t-1)$. 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. □

4. TOTAL WEIGHTED COMPLETION TIME

We finally extend the result of the previous section to the minimization of $\sum w_j C_j$ in various scheduling environments. Our results are stated without proof; they can easily be derived by a straightforward application of the techniques employed to prove Theorem 2.

Theorem 3 deals with the single machine model where, in addition, the jobs have either release dates (i.e., lower bounds on their starting times) or deadlines (i.e., upper bounds on their completion times).

THEOREM 3. *The problems of scheduling unit-time jobs on a single machine subject to chain-like precedence constraints and either arbitrary release dates or arbitrary deadlines so as to minimize* $\sum w_j C_j$ *are both NP-hard.*

The case in which there are no precedence constraints but both release dates and deadlines is solvable as a linear assignment problem in $O(n^3)$ time; the reverse case in which there are arbitrary precedence constraints but neither release dates nor deadlines is NP-hard [13;14]. When all weights are equal, the case of arbitrary precedence constraints, release dates and deadlines can be solved in $O(n^2)$ time through the Coffman-Graham algorithm [1] [11].

Theorem 4 extends these results to the situation of two parallel identical machines, where each job can be processed on either machine. Chain-like precedence constraints and release dates (deadlines) on one of the machines can be simulated by outtree(intree)-like constraints, including a single chain on the other machine, in an obvious way.

THEOREM 4. *The problems of scheduling unit-time jobs on two parallel identical machines subject to either outtree- or intree-like precedence constraints so as to minimize* $\sum w_j C_j$ *are both NP-hard.*

When all weights are equal, the case of arbitrary precedence constraints can be solved in $O(n^2)$ time by the Coffman-Graham algorithm [1] [3].

Theorem 5 states analogous results for a two-machine flow shop, where each job J_j consists of a chain of two operations $O_{1j} \rightarrow O_{2j}$ which have to be processed on the first and the second machine respectively. Note that a precedence constraint $J_j \rightarrow J_k$ implies that O_{2j} has to be completed before O_{1k} can start.

THEOREM 5. *The problems of scheduling jobs with unit-time operations in a two-machine flow shop subject to either outtree- or intree-like precedence constraints so as to minimize* $\sum w_j C_j$ *are both NP-hard.*

When all weights are equal, these problems can be solved in polynomial time [10], but the case of arbitrary precedence constraints then remains open.

5. CONCLUDING REMARKS

For those who are familiar with the classification of deterministic sequencing problems introduced by Graham, Lawler, et al. [7], we list the problems which have been shown to be NP-hard in this paper using their notation.

Theorem 1: $1|\mathit{chain},p_j=1|\sum U_j$;
Theorem 2: $1|\mathit{chain},p_j=1|\sum w_jT_j$;
Theorem 3: $1|\mathit{chain},r_j,p_j=1|\sum w_jC_j$; $1|\mathit{chain},d_j,p_j=1|\sum w_jC_j$;
Theorem 4: $P2|\mathit{outtree},p_j=1|\sum w_jC_j$; $P2|\mathit{intree},p_j=1|\sum w_jC_j$;
Theorem 5: $F2|\mathit{outtree},p_j=1|\sum w_jC_j$; $F2|\mathit{intree},p_j=1|\sum w_jC_j$.

The remaining major open problem in the area of scheduling chains of unit-time jobs on a single machine is $1|\mathit{chain},p_j=1|\sum T_j$.

Proposition 2(b) in Section 3 basically establishes NP-hardness for $1||\sum w_jT_j$. The same reduction was already given, without proof, in [17,p.359]; a more complicated transformation can be found in [12]. NP-hardness proofs for this problem that are weaker in the sense that they are valid only with respect to the standard *binary* encoding of the numerical problem data appeared in [17,p.357] and, surprisingly, [18]. All the above NP-hardness results are "strong" in the sense that they hold even with respect to a *unary* encoding [5;6;15].

ACKNOWLEDGMENTS

This research was partially supported by NSF Grant MCS76-17605 and by NATO Special Research Grant 9.2.02 (SRG.7).

REFERENCES

1. E.G. COFFMAN, JR., R.L. GRAHAM (1972) Optimal scheduling for two-processor systems. *Acta Informat.* 1,200-213.
2. R.W. CONWAY, W.L. MAXWELL, L.W. MILLER (1967) *Theory of Scheduling*, Addison-Wesley, Reading, Mass.
3. M.R. GAREY (1975) Personal communication.
4. M.R. GAREY, D.S. JOHNSON (1976) Scheduling tasks with nonuniform deadlines on two processors. *J. Assoc. Comput. Mach.* 23,461-467.
5. M.R. GAREY, D.S. JOHNSON (1978) "Strong" NP-completeness results: motivation, examples and implications. *J. Assoc. Comput. Mach.* 25,499-508.
6. M.R. GAREY, D.S. JOHNSON (1979) *Computers and Intractability: a Guide to the Theory of NP-Completeness*, Freeman, San Francisco.
7. 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, to appear.
8. T. IBARAKI, H. KISE, H. MINE (1976) Parallel-machine scheduling problem with unit processing time when jobs have ready and due times. *Trans. IECE Japan* E59.7,1-6.
9. R.M. KARP (1975) On the computational complexity of combinatorial problems. *Networks* 5,45-68.
10. B.J. LAGEWEG (1976) Personal communication.
11. E.L. LAWLER (1976) Personal communication.
12. E.L. LAWLER (1977) A "pseudopolynomial" algorithm for sequencing jobs to minimize total tardiness. *Ann. Discrete Math.* 1,331-342.
13. E.L. LAWLER (1978) Sequencing jobs to minimize total weighted completion time subject to precedence constraints. *Ann. Discrete Math.* 2,75-90.
14. J.K. LENSTRA, A.H.G. RINNOOY KAN (1978) Complexity of scheduling under precedence constraints. *Operations Res.* 26,22-35.
15. J.K. LENSTRA, A.H.G. RINNOOY KAN (1979) Computational complexity of discrete optimization problems. *Ann. Discrete Math.* 4,281-300.
16. J.K. LENSTRA, A.H.G. RINNOOY KAN (1979) Complexity of packing, covering and partitioning problems. In: A. SCHRIJVER (ed.) (1979) *Packing and Covering in Combinatorics*, Mathematical Centre Tracts 106, Mathematisch Centrum, Amsterdam, 275-291.