stichting
mathematisch
centrum

MC

AFDELING MATHEMATISCHE BESLISKUNDE BW 43/75 APRIL

P. BRUCKER, J.K. LENSTRA, A.H.G. RINNOOY KAN

COMPLEXITY OF MACHINE SCHEDULING PROBLEMS

BA

2e boerhaavestraat 49 amsterdam

Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O), by the Municipality of Amsterdam, by the University of Amsterdam, by the Free University at Amsterdam, and by industries.

AMS (MOS) subject classification scheme (1970): 90B35, 68A20

COMPLEXITY OF MACHINE SCHEDULING PROBLEMS

P. BRUCKER
University of Regensburg, Regensburg

J.K. LENSTRA
Mathematisch Centrum, Amsterdam

A.H.G. RINNOOY KAN
Graduate School of Management, Delft

ABSTRACT

We survey and extend results on the complexity of machine scheduling problems. After a brief review of the central concept of NP-completeness we give a classification of machine scheduling problems and study the influence of various parameters on their complexity. NP-completeness is established for a large number of machine scheduling problems. We finally discuss some questions that remain unanswered.

KEY WORDS AND PHRASES: *complexity, polynomial-bounded algorithms, NP-completeness, reducibility, machine scheduling problems, flow-shop, permutation-shop, job-shop, parallel-shop.*

1. INTRODUCTION

In this paper we study the complexity of machine scheduling problems. Section 2 contains a brief review of recent relevant developments in the theory of computational complexity, centering around the concept of NP-completeness. A classification of machine scheduling problems is given in section 3. In section 4 we present the results on the complexity of these problems: a large number of them turns out to be NP-complete. Quite often a minor change in some parameter transforms an NP-complete problem into one for which a polynomial-bounded algorithm is available. Thus, we have obtained a reasonable insight into the location of the borderline between "easy" and "hard" machine scheduling problems, although some questions remain open. They are briefly discussed in section 5.

2. COMPLEXITY THEORY

Recent developments in the theory of computational complexity as applied to combinatorial problems have aroused the interest of many researchers. The main credit for this must go to S.A. Cook [6] and R.M. Karp [19], who first explored the relation between the classes P and NP of problems solvable by deterministic and non-deterministic Turing machines respectively, within a number of steps bounded by a polynomial in the length of the input. Roughly speaking, P contains all problems for which a *polynomial-bounded*, *good* [7] or *efficient algorithm* exists, whereas all problems in NP can be solved by *polynomial-depth backtrack search*.

In this context, all problems are stated in terms of *recognition* problems which require a yes-no answer. An *optimization* problem of, say, minimizing some criterion, is therefore replaced by the problem of determining the existence of a solution with value $\leq y$, for some y.

For two problems P' and P, we say that P' *is reducible to* P (notation: $P' \propto P$) if for any instance of P' an instance of P can be constructed in polynomial time such that solving the instance of P will solve the instance of P' as well. P' *and* P *are equivalent* if $P' \propto P$ and $P \propto P'$. P is called *NP-complete* [20] if $P \in NP$ and every problem in NP is reducible to P.

It is clear that $P \subset NP$, and the question arises if this inclusion is a proper one or if, on the contrary, $P = NP$. Although this is still an open problem, the equality of P and NP is considered to be very unlikely and most bets [20] have been going in the other direction. This is mainly due to the following remarkable results.

Cook proved that each problem in NP is reducible to the SATISFIABILITY problem, which consists of determining if a given boolean formula in conjunctive normal form is true for some truth assignment to its variables. It was next shown by Karp that SATISFIABILITY is in turn reducible to a large number of notorious combinatorial problems in NP. It follows that all these problems are NP-complete. A polynomial-bounded algorithm for any of them thus would yield good algorithms for all problems in NP. Given the fact that difficult problems such as TRAVELLING SALESMAN, SET PARTITIONING and KNAPSACK are typically NP-complete, the existence of such an algorithm (and thereby an affirmative answer to the $P = NP$ question) seems highly unlikely.

Karp's work has led to a large amount of research on the location of the borderline between problems in P and NP-complete problems. It turns out that a minor change in a problem parameter (notably - for some mystical reason - an increase from two to three) often transforms a problem solvable in polynomial time (e.g., BIPARTITE MATCHING) into an NP-complete one (3-DIMENSIONAL MATCHING). 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 for practitioners facing these problems: establishing NP-completeness can be interpreted as a formal justification to use enumerative solution methods such as branch-and-bound, since no substantially better method is likely to exist. Embarrassing incidents like the presentation in a standard text-book of an implicit enumeration approach to the CHINESE POSTMAN problem, for which a good algorithm had already been developed [8] (see also [9]), will then occur less readily.

The class of machine scheduling problems seems an especially attractive object 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.

3. MACHINE SCHEDULING PROBLEMS

Machine scheduling problems can be verbally formulated as follows [5;31]:

> "A *job* J_j ($j = 1,\ldots,n$) consists of a sequence of *operations*, each of which corresponds to the processing of J_j on some *machine* M_i ($i = 1,\ldots,m$) during a given period of time. Each machine can handle at most one job at a time. What is according to some overall criterion the optimal processing order on each machine?"

The following data can be specified for each job J_j:

- a *number of operations* m_j;
- a *machine order* μ_j, i.e., an ordered m_j-tuple of machines;
- a *processing time* p_{jr} of its r-th operation, $r = 1,\ldots,m_j$;
- a *weight* w_j;
- a *release date* or *ready time* r_j;
- a *due date* d_j.

We assume that all data (except μ_j) are nonnegative integers, and, unless stated otherwise, that all jobs are available at time 0 (i.e., $r_j = 0$ for each J_j). Given a processing order on each machine, we can compute for each job J_j:

- the *starting time* B_j;
- the *completion time* C_j;
- the *lateness* $L_j = C_j - d_j$;
- the *tardiness* $T_j = \max\{0, C_j - d_j\}$;
- $U_j = \underline{\text{if}}\ C_j \leq d_j\ \underline{\text{then}}\ 0\ \underline{\text{else}}\ 1$.

Machine scheduling problems are traditionally classified by means of four parameters n, m, ℓ, k. The first two parameters are integer variables, denoting the numbers of jobs and machines respectively; the cases in which m is constant and equal to 1, 2, or 3 will be studied separately. If $m > 1$, the third parameter takes on one of the following values:

- $\ell = F$ in a *flow-shop* where $m_j = m$ and $\mu_j = (M_1,\ldots,M_m)$ for each job J_j;
- $\ell = P$ in a *permutation-shop*, i.e., a flow-shop where passing is not permitted so that each machine has to process the jobs in the same order;
- $\ell = G$ in a *(general) job-shop* where m_j and μ_j may vary per job J_j;
- $\ell = I$ in a *parallel-shop* where each job has to be processed on just one of m *identical* machines, i.e., $m_j = 1$ and μ_j is not defined for any J_j.

Extensions to more general situations where *several groups of identical machines* are available will not be considered.

The fourth parameter indicates the optimality criterion. The following objective functions have frequently been chosen to be minimized:

- $k = C_{max} = \max_{1\le j\le n} C_j$;
- $k = \sum w_j C_j = \sum_{j=1}^{j=n} w_j C_j$;
- $k = L_{max} = \max_{1\le j\le n} L_j$;
- $k = \sum w_j T_j = \sum_{j=1}^{j=n} w_j T_j$;
- $k = \sum w_j U_j = \sum_{j=1}^{j=n} w_j U_j$.

We refer to [31] for relations between these and other optimality criteria.

Some relevant problem variations are characterized by the presence of one or more elements in a parameter set λ, such as

- $r_j \ge 0$ (possibly non-equal ready times for the jobs);
- $r_n \ge 0$ (a possibly non-zero ready time for one job, say J_n);
- $L_{max} \le 0$ (only schedules whereby all due dates are met are to be considered; in this case we assume that $k \in \{C_{max}, \sum w_j C_j\}$);
- *prec* (precedence constraints between the jobs, where "J_j precedes J_k" (notation: $J_j < J_k$) implies $C_j \le B_k$);
- *tree* (precedence constraints between the jobs such that the associated precedence graph can be given as a *branching*, i.e., a set of directed trees with either indegree or outdegree at most one for all vertices);
- *no wait* (no waiting time for the jobs between their starting and finishing times; hence, $C_j = B_j + \sum_r p_{jr}$ for each job J_j);
- $m_j \le m_*$ (a constant upper bound for the number of operations per job);
- $1 \le p_{jr} \le p_*$ (constant lower and upper bounds for the processing times);
- $w_j = 1$ (equality of the weights).

In view of the above discussion, we can use the notation

$$n|m|\ell,\lambda|k$$

to indicate specific machine scheduling problems in the remaining sections.

4. COMPLEXITY OF MACHINE SCHEDULING PROBLEMS

All machine scheduling problems of the type defined in section 3 can be solved by polynomial-depth backtrack search and thus are members of *NP*.

The results on their complexity are summarized in Table I. The problems which are marked by an asterisk (*) are solvable in polynomial time; in Table II we provide for most of these problems references where the algorithm in question can be found. The problems marked by a note of exclamation (!) are NP-complete. Question-marks (?) indicate open problems; we will return to them in section 5 to motivate our typographical suggestion that these problems are likely to be NP-complete.

Table I contains the "hardest" problems that are known to be in *P* and the "easiest" ones that have been proved to be NP-complete. In this respect, Table I indicates to the best of our knowledge the location of the borderline between easy and hard machine scheduling problems.

We will give a simple example of the interaction between the tables and theorems in this section by examining the status of the general job-shop problem, indicated by $n|m|G|C_{max}$.

In Table I, we see that the $n|2|G,m_j\leq 2|C_{max}$ problem is a member of *P* and that two minor extensions of this problem, $n|2|G,m_j\leq 3|C_{max}$ and $n|3|G,m_j\leq 2|C_{max}$, are NP-complete. By Theorem 1(*c*,*h*), these problems are special cases of the general job-shop problem, which is thus shown to be NP-complete by Theorem 1(*b*). Table II refers to Jackson's polynomial bounded algorithm [17] for the $n|2|G,m_j\leq 2|C_{max}$ problem. Table III tells us that reductions from KNAPSACK to both NP-complete problems are presented in Theorem 4(*i*,*j*); the NP-completeness of KNAPSACK is recalled in Theorem 2(*b*).

Theorem 1 gives some elementary results on reducibility among machine scheduling problems. It can be used to establish either membership of *P* or NP-completeness for problems that are, roughly speaking, either not harder that the polynomially solvable ones or not easier than the NP-complete ones in Table I.

THEOREM 1.

(*a*) *If* $n'|m'|\ell',\lambda'|k' \propto n|m|\ell,\lambda|k$ *and* $n|m|\ell,\lambda|k \in P$, *then* $n'|m'|\ell',\lambda'|k' \in P$.

(*b*) *If* $n'|m'|\ell',\lambda'|k' \propto n|m|\ell,\lambda|k$ *and* $n'|m'|\ell',\lambda'|k'$ *is NP-complete, then* $n|m|\ell,\lambda|k$ *is NP-complete.*

(*c*) $n|m'|\ell,\lambda|k \propto n|m|\ell,\lambda|k$ *if* $m' \leq m$ *or if* m' *is constant and* m *is variable.*

(*d*) $n|2|F|k$ *and* $n|2|P|k$ *are equivalent.*

(*e*) $n|3|F|C_{max}$ *and* $n|3|P|C_{max}$ *are equivalent.*

(*f*) $n|m|F,\lambda|k \propto n|m|G,\lambda|k$.

(*g*) $n|m|\ell,\lambda|k \propto n|m|\ell,\lambda\cup\lambda'|k$ *if* $\lambda' \subset \{r_j \geq 0, r_n \geq 0, L_{max} \leq 0, prec, tree\}$.

(*h*) $n|m|\ell,\lambda\cup\lambda'|k \propto n|m|\ell,\lambda|k$ *if* $\lambda' \subset \{m_j \leq m_*, 1 \leq p_{jr} \leq p_*, w_j = 1\}$.

(*i*) $n|m|\ell,\lambda|C_{max} \propto n|m|\ell,\lambda|L_{max}$.

(*j*) $n|m|\ell,\lambda|\sum w_j C_j \propto n|m|\ell,\lambda|\sum w_j T_j$.

(*k*) $n|m-1|F,r_n \geq 0,\lambda|k \propto n|m|F,\lambda|k$ *if* $\lambda \subset \{w_j = 1\}$.

(*l*) $n'|m|I,prec,1 \leq p_{j1} \leq p_*|C_{max} \propto n|m|I,prec,1 \leq p_{j1} \leq p_*, w_j = 1|\sum w_j C_j$.

Proof. Let P' and P denote the problems on the left-hand side and right-hand side respectively.

(*a*,*b*) Clear from the definition of reducibility.

(*c*) Trivial.

(*d*,*e*) P' has an optimal solution with the same processing order on each machine [5;31].

(*f*,*g*,*h*) In each case P' obviously is a special case of P.

(*i*,*j*) Take $d_j = 0$ $(j = 1,\ldots,n)$ in P.

(*k*) Suppose that in P' the machines and the operations of each job are indexed from 2 to m. We specify P by adding a machine M_1 and defining $p_{j1} = r_j$ $(j = 1,\ldots,n)$ (i.e., $p_{11} = \ldots = p_{n-1\ 1} = 0$, $p_{n1} \geq 0$). Any solution to P' corresponds to a solution to P with the same value for k, and vice versa.

(*l*) Any instance of P' has a solution with value $C_{max} \leq n'p_*$. We construct a corresponding instance of P by choosing y', $0 \leq y' \leq n'p_*$, defining

$$n'' = (n' - 1)y'$$
$$n = n' + n''$$
$$y = ny' + \tfrac{1}{2}n''(n'' + 1)$$

and adding n" jobs $J_{n'+k}$ $(k = 1,\ldots,n'')$ to P' with

$$p_{n'+k\ 1} = 1$$
$$J_j < J_{n'+k} \quad (j = 1,\ldots,n'+k-1)$$

Now P' has a solution with value $\leq y'$ if and only if P has a solution with value $\leq y$:

$$C_{max} \leq y' \Rightarrow \sum_{j=1}^{j=n} C_j \leq n'y' + \sum_{k=n'+1}^{k=n}(y' + k) = y$$
$$C_{max} > y' \Rightarrow \sum_{j=1}^{j=n} C_j > y' + \sum_{k=n'+1}^{k=n}(y' + 1 + k) = y$$

□

TABLE I. COMPLEXITY OF MACHINE SCHEDULING PROBLEMS

n jobs	1 machine	2 machines	m machines
C_{max}	* $r_j \geq 0$, *prec*	* F * F, *no wait* ! F, $r_n \geq 0$! F, *tree*	! $\underline{m=3}$:F ? $\underline{m=3}$:F, *no wait* ! F, *no wait*
		* G, $m_j \leq 2$! G, $m_j \leq 3$	* $\underline{n=2}$:G ! $\underline{m=3}$:G, $m_j \leq 2$
		! I * I, *prec*, $p_{j1}=1$! I, *prec*, $1 \leq p_{j1} \leq 2$	* I, *tree*, $p_{j1}=1$? $\underline{m=3}$:I, *prec*, $p_{j1}=1$! I, *prec*, $p_{j1}=1$
$\sum w_j C_j$	* *tree* ? *prec*, $w_j=1$! $r_n \geq 0$, $w_j=1$ * $L_{max} \leq 0$, $w_j=1$! $L_{max} \leq 0$	! F, $w_j=1$? F, *no wait*, $w_j=1$	! F, *no wait*, $w_j=1$
		! I ! I, *prec*, $1 \leq p_{j1} \leq 2$, $w_j=1$	* I, $r_j \geq 0$, $p_{j1}=1$ * I, $w_j=1$! I, *prec*, $p_{j1}=1$, $w_j=1$
L_{max}	* *prec* * $r_j \geq 0$, $p_{j1}=1$! $r_n \geq 0$	! F	
$\sum w_j T_j$	* $r_j \geq 0$, $p_{j1}=1$? $w_j=1$! ! $r_n \geq 0$, $w_j=1$	! F, $w_j=1$	
$\sum w_j U_j$	* $r_j \geq 0$, $p_{j1}=1$ * $w_j=1$ * $1 \leq p_{j1} \leq p_*$! ! $r_n \geq 0$, $w_j=1$	! F, $w_j=1$	

* problem in $\mathcal{P}$: see Table II

? open problem: see section 5

! NP-complete problem: see Table III

TABLE II. REFERENCES TO POLYNOMIAL-BOUNDED ALGORITHMS

Problem	Reference
$n\|1\|r_j \geq 0, prec\|C_{max}$	-
$n\|1\|tree\|\sum w_j C_j$	Horn [13]; Sidney [33;34]
$n\|1\|L_{max} \leq 0, w_j=1\|\sum w_j C_j$	Smith [35]
$n\|1\|prec\|L_{max}$	Lawler [23]
$n\|1\|r_j \geq 0, p_{j1}=1\|L_{max}$	Horn [15]
$n\|1\|r_j \geq 0, p_{j1}=1\|\sum w_j T_j$	Lawler [22]
$n\|1\|r_j \geq 0, p_{j1}=1\|\sum w_j U_j$	Lawler [24,Ch.7]; Lageweg [21]
$n\|1\|w_j=1\|\sum w_j U_j$	Moore [27]
$n\|1\|1 \leq p_{j1} \leq p_*\|\sum w_j U_j$	Lawler & Moore [25]
$n\|2\|F\|C_{max}$	Johnson [18]
$n\|2\|F, no\ wait\|C_{max}$	Gilmore & Gomory [11]
$n\|2\|G, m_j \leq 2\|C_{max}$	Jackson [17]
$n\|2\|I, prec, p_{j1}=1\|C_{max}$	Coffman & Graham [4]
$2\|m\|G\|C_{max}$	Szwarc [36]; Hardgrave & Nemhauser [12]
$n\|m\|I, tree, p_{j1}=1\|C_{max}$	Hu [16]
$n\|m\|I, r_j \geq 0, p_{j1}=1\|\sum w_j C_j$	-
$n\|m\|I, w_j=1\|\sum w_j C_j$	Conway et al. [5]; Horn [14]; Bruno et al. [3]

Remark. The proofs of Theorem $1(c,k)$ involve processing times equal to 0, implying that the operations in question require an infinitesimally small amount of time. Whenever these reductions are applied, the processing times can be transformed into positive integers by sufficiently (but polynomially) inflating the problem data. Examples of such constructions can be found in the proofs of Theorem $4(g,h)$.

The remaining part of this section will be devoted to the NP-completeness proofs for the problems, marked by a note of exclamation in Table I. The reductions for these problems are listed in Table III. They involve four results from Karp [19] and a minor extension, as collected in Theorem 2.

TABLE III. REDUCTIONS TO NP-COMPLETE MACHINE SCHEDULING PROBLEMS

Reduction	Reference
PARTITION $\propto$ $n\|2\|I\|C_{max}$	Bruno et al. [3]; h.l., Theorem 3(*a*)
PARTITION $\propto$ $n\|2\|I\|\sum w_j C_j$	Bruno et al. [3], h.l., Theorem 3(*b*)
KNAPSACK $\propto$ $n\|1\|r_n \geq 0, w_j=1\|\sum w_j C_j$	h.l., Theorem 4(*a*)
KNAPSACK $\propto$ $n\|1\|L_{max} \leq 0\|\sum w_j C_j$	h.l., Theorem 4(*b*)
KNAPSACK $\propto$ $n\|1\|r_n \geq 0\|L_{max}$	h.l., Theorem 4(*c*)
KNAPSACK $\propto$ $n\|1\|\|\sum w_j T_j$	h.l., Theorem 4(*d*)
KNAPSACK $\propto$ $n\|1\|\|\sum w_j U_j$	Karp [19]; h.l., Theorem 4(*e*)
KNAPSACK $\propto$ $n\|1\|r_n \geq 0, w_j=1\|\sum w_j U_j$	h.l., Theorem 4(*f*)
KNAPSACK $\propto$ $n\|2\|F, r_n \geq 0\|C_{max}$	h.l., Theorem 4(*g*)
KNAPSACK $\propto$ $n\|2\|F, *tree*\|C_{max}$	h.l., Theorem 4(*h*)
KNAPSACK $\propto$ $n\|2\|G, m_j \leq 3\|C_{max}$	h.l., Theorem 4(*i*)
KNAPSACK $\propto$ $n\|3\|G, m_j \leq 2\|C_{max}$	h.l., Theorem 4(*j*)
DIRECTED HAMILTONIAN PATH $\propto$ $n\|m\|F, *no\ wait*\|C_{max}$	h.l., Theorem 5(*a*)
DIRECTED HAMILTONIAN PATH $\propto$ $n\|m\|F, *no\ wait*, w_j=1\|\sum w_j C_j$	h.l., Theorem 5(*b*)
3-SATISFIABILITY $\propto$ $n\|2\|I, *prec*, 1 \leq p_{j1} \leq 2\|C_{max}$	Ullman [37]
3-SATISFIABILITY $\propto$ $n\|m\|I, *prec*, p_{j1}=1\|C_{max}$	Ullman [37]
$n\|1\|r_n \geq 0, w_j=1\|\sum w_j C_j \propto n\|1\|r_n \geq 0, w_j=1\|\sum w_j T_j$	h.l., Theorem 1(*j*)
$n\|1\|r_n \geq 0, w_j=1\|\sum w_j C_j \propto n\|2\|F, w_j=1\|\sum w_j C_j$	h.l., Theorem 1(*k*)
$n\|1\|r_n \geq 0\|L_{max} \propto n\|2\|F\|L_{max}$	h.l., Theorem 1(*k*)
$n\|1\|r_n \geq 0, w_j=1\|\sum w_j T_j \propto n\|2\|F, w_j=1\|\sum w_j T_j$	h.l., Theorem 1(*k*)
$n\|1\|r_n \geq 0, w_j=1\|\sum w_j U_j \propto n\|2\|F, w_j=1\|\sum w_j U_j$	h.l., Theorem 1(*k*)
$n\|2\|F, r_n \geq 0\|C_{max} \propto n\|3\|F\|C_{max}$	h.l., Theorem 1(*k*)
$n'\|2\|I, *prec*, 1 \leq p_{j1} \leq 2\|C_{max} \propto n\|2\|I, *prec*, 1 \leq p_{j1} \leq 2, w_j=1\|\sum w_j C_j$	h.l., Theorem 1(*l*)
$n'\|m\|I, *prec*, p_{j1}=1\|C_{max} \propto n\|m\|I, *prec*, p_{j1}=1, w_j=1\|\sum w_j C_j$	h.l., Theorem 1(*l*)

THEOREM 2. *The following problems are NP-complete:*

(*a*) PARTITION

Given positive integers $a_1,\ldots,a_t$, *does there exist a subset* $S \subset T = \{1,\ldots,t\}$ *such that* $\sum_{j\in S} a_j = \sum_{j\in T-S} a_j$?

(*b*) KNAPSACK

Given positive integers $a_1,\ldots,a_t,b$, *does there exist a subset* $S \subset T = \{1,\ldots,t\}$ *such that* $\sum_{j\in S} a_j = b$?

(*c*) DIRECTED HAMILTON CIRCUIT

Given a directed graph $G' = (V',A')$, *does* G' *have a hamilton circuit (i.e., a directed cycle passing through each vertex exactly once)?*

(*d*) DIRECTED HAMILTON PATH

Given a directed graph $G = (V,A)$, *does* G *have a hamilton path (i.e., a directed path passing through each vertex exactly once)?*

(*e*) 3-SATISFIABILITY

Given clauses $C_1,\ldots,C_m$, *each consisting of at most three* literals *from the set* $X = (x_1,\ldots,x_n,\bar{x}_1,\ldots,\bar{x}_n)$, *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* $(x_j,\bar{x}_j)$;
- $S \cap C_i \neq \emptyset$ *for* $i = 1,\ldots,m$?

Proof.

(*a*,*b*,*c*,*e*) See Karp [19].

(*d*) DIRECTED HAMILTON CIRCUIT $\propto$ DIRECTED HAMILTON PATH

Given $G' = (V',A')$, choose $v' \in V'$ and construct $G = (V,A)$ with

$$V = V' \cup \{v''\}$$

$$A = \{(u,v) \mid (u,v) \in A',\ v \neq v'\} \cup \{(u,v'') \mid (u,v') \in A'\}$$

G' has a hamilton circuit if and only if G has a hamilton path. □

In Theorems 3, 4, and 5 we present a large number of reductions of the form $P \propto n|m|\ell,\lambda|k$ by specifying $n|m|\ell,\lambda|k$ and some y such that P has a solution if and only if $n|m|\ell,\lambda|k$ has a solution with value $k \leq y$. This equivalence is proved for some principal reductions; in other cases, it is trivial or clear from the analogy to a reduction given previously. The NP-completeness of $n|m|\ell,\lambda|k$ then follows from the NP-completeness of P as established in Theorem 2.

First, we briefly deal with the problems on *identical* machines. Theorem 3 presents two reductions which are simplified versions of the reductions given in [3].

THEOREM 3. PARTITION *is reducible to the following problems:*

(*a*) $n|2|I|C_{max}$

(*b*) $n|2|I|\sum w_j C_j$

Proof. Define $A = \sum_{j\in T} a_j$.

(*a*) PARTITION $\propto n|2|I|C_{max}$

$n = t$

$p_{j1} = a_j \quad (j \in T)$

$y = \frac{1}{2}A$

(*b*) PARTITION $\propto n|2|I|\sum w_j C_j$

$n = t$

$p_{j1} = w_j = a_j \quad (j \in T)$

$y = \sum_{j,k\in T, j\le k} a_j a_k - \frac{1}{4}A^2$

Suppose that $\{J_j | j \in S\}$ is assigned to M_1 and $\{J_j | j \in T\text{-}S\}$ to M_2; let $c = \sum_{j\in S} a_j - \frac{1}{2}A$. Since $p_{j1} = w_j$ for all j, the value of $\sum w_j C_j$ is not influenced by the ordering of the jobs on the machines and only depends on the choice of S (cf. [5;31]):

$$\sum w_j C_j = k(S).$$

It is easily seen (cf. Figure 1) that

$$k(S) = k(T) - \left(\sum_{j\in S} a_j\right)\left(\sum_{j\in T-S} a_j\right) =$$

$$= \sum_{j,k\in T, j\le k} a_j a_k - (\tfrac{1}{2}A + c)(\tfrac{1}{2}A - c) = y + c^2,$$

and it follows that PARTITION has a solution if and only if this $n|2|I|\sum w_j C_j$ problem has a solution with value $\le y$. □

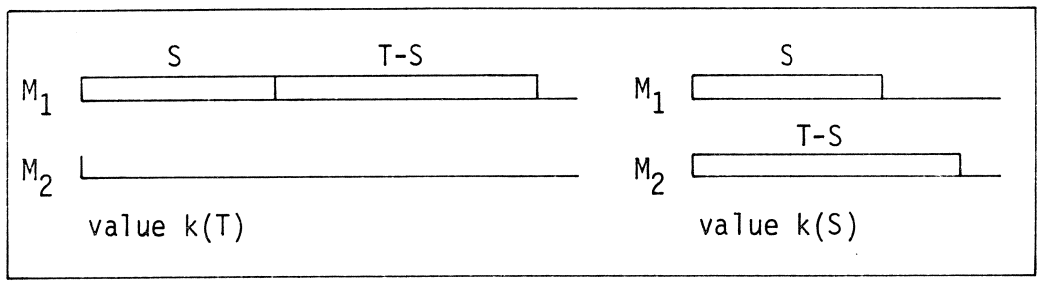

Figure 1

Most of our results on *different* machines involve the KNAPSACK problem, as demonstrated by Theorem 4.

THEOREM 4. KNAPSACK *is reducible to the following problems:*

(*a*) $n|1|r_n \geq 0, w_j = 1|\sum w_j C_j$

(*b*) $n|1|L_{max} \leq 0|\sum w_j C_j$

(*c*) $n|1|r_n \geq 0|L_{max}$

(*d*) $n|1||\sum w_j T_j$

(*e*) $n|1||\sum w_j U_j$

(*f*) $n|1|r_n \geq 0, w_j = 1|\sum w_j U_j$

(*g*) $n|2|F, r_n \geq 0|C_{max}$

(*h*) $n|2|F, tree|C_{max}$

(*i*) $n|2|G, m_j \leq 3|C_{max}$

(*j*) $n|3|G, m_j \leq 2|C_{max}$

Proof. Define $A = \sum_{j \in T} a_j$, $a_* = \max_{j \in T} a_j$. We may assume that $0 < b < A$. The reductions will be presented roughly in order of increasing complexity.

(*i*) KNAPSACK $\propto n|2|G, m_j \leq 3|C_{max}$

$n = t + 1$

$\mu_j = (M_1)$, $p_{j1} = a_j$ $(j \in T)$

$\mu_n = (M_2, M_1, M_2)$, $p_{n1} = b$, $p_{n2} = 1$, $p_{n3} = A - b$

$y = A + 1$

If KNAPSACK has a solution, then there exists a schedule with value $C_{max} = y$, as illustrated in Figure 2. If KNAPSACK has no solution, then $\sum_{j \in S} a_j - b = c \neq 0$ for each $S \subset T$, and we have for a processing order $(\{J_j | j \in S\}, J_n, \{J_j | j \in T\text{-}S\})$ on M_1 that

$$c > 0 \Rightarrow C_{max} \geq \sum_{j \in S} p_{j1} + p_{n2} + p_{n3} = A + c + 1 > y$$

$$c < 0 \Rightarrow C_{max} \geq p_{n1} + p_{n2} + \sum_{j \in T-S} p_{j1} = A - c + 1 > y$$

It follows that KNAPSACK has a solution if and only if this $n|2|G, m_j \leq 3|C_{max}$ problem has a solution with value $\leq y$.

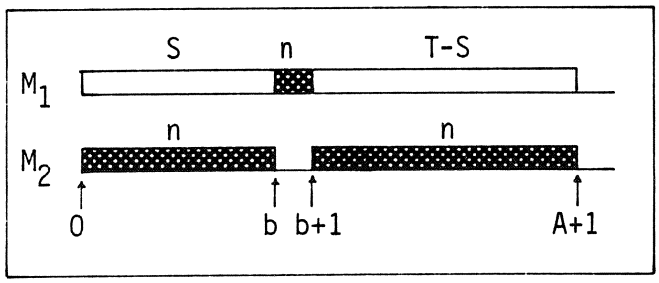

Figure 2

(j) KNAPSACK $\propto$ $n|3|G,m_j\leq 2|C_{max}$

$n = t + 2$

$\mu_j = (M_1,M_3)$, $p_{j1} = p_{j2} = a_j$ $(j \in T)$

$\mu_{n-1} = (M_1,M_2)$, $p_{n-1\,1} = b$, $p_{n-1\,2} = 2(A - b)$

$\mu_n = (M_2,M_3)$, $p_{n1} = 2b$, $p_{n2} = A - b$

$y = 2A$

If KNAPSACK has a solution, then there exists a schedule with value $C_{max} = y$, as illustrated in Figure 3. If KNAPSACK has no solution, then $\sum_{j\in S} a_j - b = c \neq 0$ for each $S \subset T$, and we have for a processing order $(\{J_j | j \in S\}, J_{n-1}, \{J_j | j \in T\text{-}S\})$ on M_1 that

$$c > 0 \Rightarrow C_{max} \geq \sum_{j\in S} p_{j1} + p_{n-1\,1} + p_{n-1\,2} = 2A + c > y$$

$$c < 0 \Rightarrow C_{max} \geq \min\{\sum_{j\in S} p_{j1} + p_{n-1\,1} + 1, p_{n1}\} + p_{n2} + \sum_{j\in T-S} p_{j2} = 2A + 1 > y$$

which completes the equivalence proof.

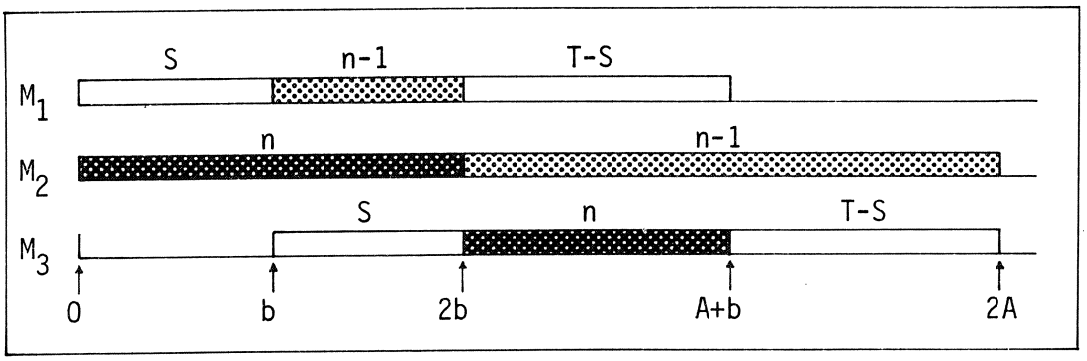

Figure 3

(c) KNAPSACK $\propto$ $n|1|r_n\geq 0|L_{max}$ and

(f) KNAPSACK $\propto$ $n|1|r_n\geq 0, w_j=1|\sum w_j U_j$

$n = t + 1$

$r_j = 0$, $p_{j1} = a_j$, $d_j = A + 1$ $(j \in T)$

$r_n = b$, $p_{n1} = 1$, $d_n = b + 1$

$y = 0$

Cf. reduction (i) and Figure 4.

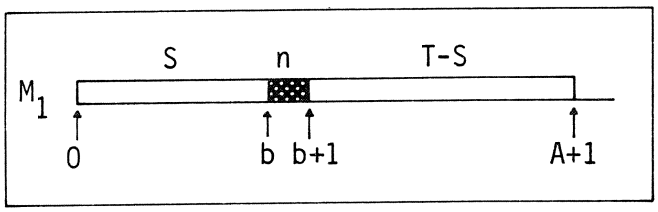

Figure 4

(*g*) KNAPSACK $\propto$ $n|2|F,r_n \geq 0|C_{max}$

$n = t + 1$

$r_j = 0, \quad p_{j1} = ta_j, \quad p_{j2} = 1 \quad (j \in T)$

$r_n = tb, \quad p_{n1} = 1, \quad p_{n2} = t(A - b)$

$y = t(A + 1)$

If KNAPSACK has a solution, then there exists a schedule with value $C_{max} \leq y$, as illustrated in Figure 5. If KNAPSACK has no solution, then $\sum_{j \in S} a_j - b = c \neq 0$ for each $S \subset T$, and we have for a processing order $(\{J_j | j \in S\}, J_n, \{J_j | j \in T\text{-}S\})$ on M_1 that

$$c > 0 \Rightarrow C_{max} \geq \sum_{j \in S} p_{j1} + p_{n1} + p_{n2} = t(A + c) + 1 > y$$

$$c < 0 \Rightarrow C_{max} \geq r_n + p_{n1} + \sum_{j \in T\text{-}S} p_{j1} = t(A - c) + 1 > y$$

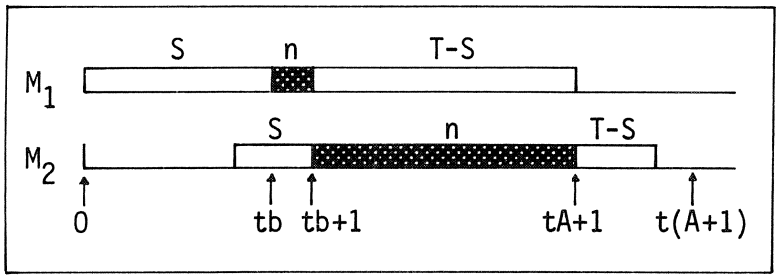

Figure 5

(*h*) KNAPSACK $\propto$ $n|2|F,tree|C_{max}$

$n = t + 2$

$p_{j1} = ta_j, \quad p_{j2} = 1 \quad (j \in T)$

$p_{n-1\,1} = 1, \quad p_{n-1\,2} = tb$

$p_{n1} = 1, \quad p_{n2} = t(A - b)$

$J_{n-1} < J_n$

$y = t(A + 1) + 1$

Cf. Figure 6. We have for a processing order $(\{J_j | j \in R\}, J_{n-1}, \{J_j | j \in S\}, J_n, \{J_j | j \in T\text{-}S\text{-}R\})$ on M_1 that

$$R \neq \emptyset \Rightarrow C_{max} \geq t + p_{n-1\,1} + p_{n-1\,2} + p_{n1} + p_{n2} = t(A + 1) + 2 > y$$

The remainder of the equivalence proof is analogous to that of reduction (*g*).

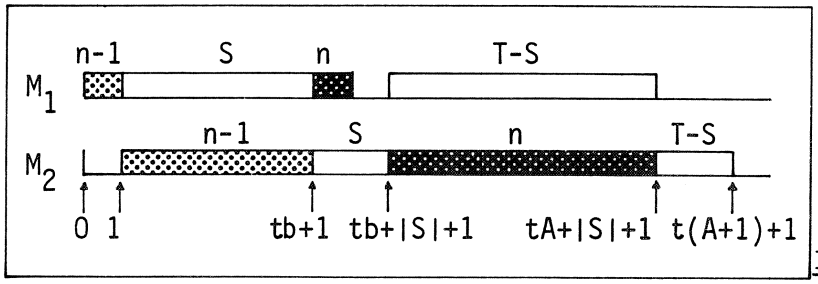

Figure 6

(*e*) KNAPSACK $\propto n|1||\sum w_j U_j$

$n = t$

$p_{j1} = w_j = a_j$, $d_j = b$ $(j \in T)$

$y = A - b$

Cf. Karp [19] and Figure 7.

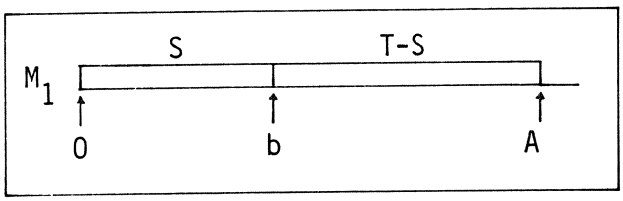

Figure 7

(*b*) KNAPSACK $\propto n|1|L_{max} \leq 0|\sum w_j C_j$

$n = t + 1$

$p_{j1} = w_j = a_j$, $d_j = A + 1$ $(j \in T)$

$p_{n1} = 1$, $w_n = 0$, $d_n = b + 1$

$y = \sum_{j,k \in T, j \leq k} a_j a_k + A - b$

Cf. Figure 4. We have for a feasible processing order $(\{J_j | j \in S\}, J_n, \{J_j | j \in T\text{-}S\})$ on M_1 that $\sum_{j \in S} a_j - b = L_n \leq 0$. Since $p_{j1} = w_j$ for all $j \in T$, the value of $\sum w_j C_j$ is not influenced by the ordering of S and T-S (cf. the proof of Theorem 3(*b*)):

$$\sum w_j C_j = \sum_{j \in T} a_j C_j =$$

$$= \sum_{j,k \in T, j \leq k} a_j a_k + \sum_{j \in T\text{-}S} a_j =$$

$$= y - L_n \geq y$$

The equivalence follows immediately.

(*d*) KNAPSACK $\propto n|1||\sum w_j T_j$

$$n = t + t'$$

$$p_{j1} = \tau + a_j,\ w_j = \tau + a_j + 1,\ d_j = t\tau + b \quad (j \in T)$$

$$p_{j1} = \tau,\ w_j = \tau + 1,\ d_j = t\tau + b \quad (j \notin T)$$

$$y = \tfrac{1}{2}t'(t' + 1)\tau(\tau + 1) + (t' + 1)\tau(A - b) + t'$$

where

$$t' = \tfrac{1}{2}(t + 1)(A - b) + \tfrac{1}{2}t(t + 1)a_*^2$$

$$\tau > 2t' + A$$

Given a processing order $\pi = (\pi(1),\ldots,\pi(n))$, we may assume that the jobs are scheduled without interruption from 0 to $\sum_j p_{j1} = n\tau + A$; any other schedule can be improved by removal of the idle machine time.

It is easily seen that KNAPSACK has a solution if and only if there exists a processing order π such that $C_{\pi(t)} = t\tau + b$. Defining $c_\pi = C_{\pi(t)} - (t\tau + b)$, we have $-b \le c_\pi \le A - b$ for any schedule π. We will show that

(*A*) $c_\pi = 0 \Rightarrow \exists\pi'\colon c_{\pi'} = 0 \wedge \sum w_j T_j \le y$

(*B*) $c_\pi > 0 \Rightarrow \sum w_j T_j > y$

(*C*) $c_\pi < 0 \Rightarrow \sum w_j T_j > y$

which proves the equivalence of KNAPSACK and this $n|1||\sum w_j T_j$ problem.

First, we will derive some important inequalities. Defining $a_j = 0$ for $j \notin T$, we note that $\sum_{j>t} a_{\pi(j)} = A - b - c_\pi$. Since $w_j = p_{j1} + 1$ for all j, we have

$$(1)\quad \sum_{j>t} w_{\pi(j)}\left(C_{\pi(j)} - C_{\pi(t)}\right) = \sum_{j>t} p_{\pi(j)1}\left(C_{\pi(j)} - C_{\pi(t)}\right) + \sum_{j>t}\left(C_{\pi(j)} - C_{\pi(t)}\right)$$

In the first term, the processing times appear as weights (cf. reductions (*e*,*b*)). This implies for any processing order $(\pi(t+1),\ldots,\pi(n))$:

$$(2)\quad \sum_{j>t} p_{\pi(j)1}\left(C_{\pi(j)} - C_{\pi(t)}\right) = \sum_{t<j\le k\le n}\left(\tau + a_{\pi(j)}\right)\left(\tau + a_{\pi(k)}\right) = \tfrac{1}{2}t'(t' + 1)\tau^2 + (t' + 1)\tau(A - b - c_\pi) + \sum_{t<j\le k\le n} a_{\pi(j)}a_{\pi(k)}$$

where the last term is trivially bounded by

$$(3)\quad 0 \le \sum_{t<j\le k\le n} a_{\pi(j)}a_{\pi(k)} \le \tfrac{1}{2}t(t + 1)a_*^2$$

As to the second term in (1), we have for any schedule π that

$$C_{\pi(j)} \geq C_{\pi(t)} + (j - t)\tau \qquad (t < j \leq n)$$

and there exists a schedule π such that

$$C_{\pi(j)} = C_{\pi(t)} + (j - t)\tau \qquad (t < j \leq t')$$

$$C_{\pi(j)} \leq C_{\pi(t)} + (j - t)\tau + (j - t')(A - b - c_{\pi})/t \qquad (t' < j \leq n)$$

Straightforward calculations now show that

(4) $\sum_{j>t}\left(C_{\pi(j)} - C_{\pi(t)}\right) \geq \frac{1}{2}t'(t' + 1)\tau$ for any π;

(5) $\sum_{j>t}\left(C_{\pi(j)} - C_{\pi(t)}\right) \leq \frac{1}{2}t'(t' + 1)\tau + \frac{1}{2}(t + 1)(A - b - c_{\pi})$ for some π.

Combining (1,2,3,4,5) and the definitions of y and t', we obtain

(6) $\sum_{j>t} w_{\pi(j)}\left(C_{\pi(j)} - C_{\pi(t)}\right) \geq y - (t' + 1)\tau c_{\pi} - t'$ for any π;

(7) $\sum_{j>t} w_{\pi(j)}\left(C_{\pi(j)} - C_{\pi(t)}\right) \leq y - (t' + 1)\tau c_{\pi} - \frac{1}{2}(t + 1)c_{\pi}$ for some π.

Given a processing order π, the tardiness of job $J_{\pi(j)}$ is equal to

$$T_{\pi(j)} = \max\{0, C_{\pi(j)} - C_{\pi(t)} + c_{\pi}\}$$

(A) If $c_{\pi} = 0$, then the jobs $J_{\pi(1)},\ldots,J_{\pi(t)}$ are completed before the due date and the remaining ones are completed thereafter. By virtue of (7), these late jobs can be ordered in such a way that

$$\sum w_j T_j = \sum_{j>t} w_{\pi(j)}\left(C_{\pi(j)} - C_{\pi(t)}\right) \leq y$$

(B) If $1 \leq c_{\pi} \leq A - b$, then the jobs $J_{\pi(t)},\ldots,J_{\pi(n)}$ are late. Applying (6) we find

$$\sum w_j T_j \geq \sum_{j>t} w_{\pi(j)}\left(C_{\pi(j)} - C_{\pi(t)}\right) + c_{\pi}\sum_{j\geq t} w_{\pi(j)} \geq$$

$$\geq y - (t' + 1)\tau c_{\pi} - t' + (t' + 1)(\tau + 1)c_{\pi} =$$

$$= y - t' + (t' + 1)c_{\pi} > y$$

(C) If $-b \leq c_{\pi} \leq -1$, then the jobs $J_{\pi(1)},\ldots,J_{\pi(t)}$ finish in time. It follows from (6) and the choice of τ that

$$\sum w_j T_j \geq \sum_{j>t} w_{\pi(j)}\left(C_{\pi(j)} - C_{\pi(t)}\right) + c_{\pi}\sum_{j>t} w_{\pi(j)} \geq$$

$$\geq y - (t' + 1)\tau c_{\pi} - t' + (t'(\tau + 1) + A)c_{\pi} =$$

$$= y - t' - (\tau - t' - A)c_{\pi} > y$$

(*a*) KNAPSACK $\propto n|1|r_n \geq 0, w_j = 1|\sum w_j C_j$

$n = t + t' + u + 1$

$r_j = 0, \quad p_{j1} = \tau + a_j \quad (j \in T = \{1,\ldots,t\})$

$r_j = 0, \quad p_{j1} = \tau \quad (j \in T' = \{t+1,\ldots,t+t'\})$

$r_j = 0, \quad p_{j1} = \upsilon \quad (j \in U = \{t+t'+1,\ldots,t+t'+u\})$

$r_n = t\tau + b, \; p_{n1} = 1$

$y = \upsilon + \frac{1}{2}u(u+1)\upsilon$

where

$t' = t(t+1)a_*$

$\tau = (t'+1)(b+1) + t'$

$u = \frac{1}{2}(t+t')(t+t'+1)\tau + (t+1)\tau$

$\upsilon = u(\sigma + 1)$

$\sigma = \sum_{j \notin U} p_{j1} = (t+t')\tau + A + 1$

If KNAPSACK has a solution, then $\sum_{j \in S} a_j = b$ for some $S \subset T$. Defining $S' = \{t + j \mid j \in T\text{-}S\} \subset T'$, we have for a processing order $(\{J_j \mid j \in S'\}, \{J_j \mid j \in S\}, J_n, \{J_j \mid j \in T'\text{-}S'\}, \{J_j \mid j \in T\text{-}S\}, \{J_j \mid j \in U\})$ that

$$\sum_{j \notin U} C_j = \sum_{j \in S' \cup S} C_j + C_n + \sum_{j \in (T'-S') \cup (T-S)} C_j \leq$$

$$\leq \sum_{j=1}^{j=t} j(\tau + a_*) + t\tau + b + 1 + \sum_{j=t+1}^{j=t+t'} (j\tau + b + 1) + \sum_{j=1}^{j=t} j a_* =$$

$$= \tfrac{1}{2}(t+t')(t+t'+1)\tau + t\tau + (t'+1)(b+1) + t(t+1)a_* = u$$

and

$$\sum_{j \in U} C_j = \sum_{j=1}^{j=u} (\sigma + j\upsilon) = u\sigma + \tfrac{1}{2}u(u+1)\upsilon$$

Hence,

$$\sum_j C_j \leq u + u\sigma + \tfrac{1}{2}u(u+1)\upsilon = y$$

Conversely, if $\sum_j C_j \leq y$ for some schedule, we claim that

(*A*) $\{J_j \mid j \notin U\}$ precedes $\{J_j \mid j \in U\}$;

(*B*) $B_j \leq \sigma$ for some $j \in U$;

(*C*) exactly t jobs precede J_n;

(*D*) $B_n = t\tau + b$.

It follows from (*A*) and (*B*) that $\{J_j \mid j \notin U\}$ is scheduled without interruption from 0 to σ. By (*C*) and (*D*), exactly t jobs from $\{J_j \mid j \in T \cup T'\}$ occupy a period of length $t\tau + b$. This implies that KNAPSACK has a solution, as is easily seen.

We now turn to the proofs of (A), (B), (C), and (D).

(A) If $J_{j'}$, $j' \notin U$, succeeds some J_j, $j \in U$, then we have

$$\sum_j C_j \geq C_{j'} + \sum_{j \in U} C_j > \upsilon + \tfrac{1}{2}u(u+1)\upsilon = y$$

(B) If $B_j > \sigma$ for all $j \in U$, then we have

$$\sum_j C_j > \sum_{j \in U} C_j \geq u(\sigma + 1) + \tfrac{1}{2}u(u+1)\upsilon = y$$

Since $\sum_j C_j \leq y$ and, by (A) and (B), $\sum_{j \in U} C_j \geq u\sigma + \frac{1}{2}u(u+1)\upsilon$, we now know that $\sum_{j \notin U} C_j \leq u$.

(C) Let exactly s jobs from $\{J_j \mid j \in T \cup T'\}$ precede J_n.

If $0 \leq s \leq t - 1$, then we have

$$\sum_{j \notin U} C_j \geq \sum_{j=1}^{j=s} j\tau + t\tau + b + 1 + \sum_{j=s+1}^{j=t+t'} (j\tau + (t-s)\tau + b + 1) =$$

$$= u - t' + (t + t' - s)(t - s)\tau + (t - s)(b + 1) >$$

$$> u - t' + (t' + 1)\tau > u$$

If $t + 1 \leq s \leq t + t'$, then we have

$$\sum_{j \notin U} C_j \geq \sum_{j=1}^{j=s} j\tau + s\tau + 1 + \sum_{j=s+1}^{j=t+t'} (j\tau + 1) =$$

$$= u + (s - t - 1)\tau + t + t' - s + 1 \geq$$

$$\geq u + 1 > u$$

(D) Note that $B_n \geq r_n = t\tau + b$. If $B_n > t\tau + b$, then we have

$$\sum_{j \notin U} C_j \geq \sum_{j=1}^{j=t} j\tau + t\tau + b + 2 + \sum_{j=t+1}^{j=t+t'} (j\tau + b + 2) =$$

$$= u + 1 > u$$

This completes the proof of Theorem 4. □

Remark. In the last two reductions the size of the scheduling problem depends on A and a_*. It can be questioned if these reductions are truly polynomial-bounded: in some encodings the length of the scheduling input string is polynomial in A and a_* and thus exponential in the length of the KNAPSACK input string, the latter one being proportional to $\log_2 a_*$. We may settle this question, however, by characterizing a subset of jobs with identical data (p_{jr}, w_j, r_j, d_j) by its cardinality and a single copy of the data.

The NP-completeness proofs for the problems with a *no wait* assumption are based on the well-known relation between these problems and the TRAVELLING SALESMAN problem of finding a minimum weight hamilton circuit in the complete directed graph on the vertex set V with weights on the arcs.

Given an $n|m|F,no\ wait|k$ problem, we define c_{jk} to be the minimum length of the time interval between B_j and B_k if J_k is scheduled directly after J_j. If we define

$$q_{\ell i} = \sum_{r=1}^{r=i} p_{\ell r} \tag{8}$$

it is easily proved (cf. [29;30;38;26]) that

$$c_{jk} = \max_{1\le i\le m}\{q_{ji} - q_{k\,i-1}\} \tag{9}$$

Finding a schedule that minimizes C_{max} is now equivalent to solving the TRAVELLING SALESMAN problem with $V = \{0,\ldots,n\}$ and weights c_{jk} defined by (9) and by $c_{0\ell} = 0$, $c_{\ell 0} = q_{\ell m}$ for $\ell \neq 0$.

THEOREM 5. DIRECTED HAMILTON PATH *is reducible to the following problems:*

(*a*) $n|m|F,no\ wait|C_{max}$

(*b*) $n|m|F,no\ wait,w_j=1|\sum w_j C_j$

Proof.

(*a*) DIRECTED HAMILTON PATH $\propto n|m|F,no\ wait|C_{max}$

Given $G = (V,A)$, we define

$n = |V|$

$m = n(n-1) + 2$

All jobs have the same machine order $(M_1,M_2,\ldots,M_{m-1},M_m)$. To each pair of jobs (J_j,J_k) $(j,k = 1,\ldots,n,\ j \neq k)$ there corresponds one machine $M_i = M_{\iota(j,k)}$ $(i = 2,\ldots,m-1)$, such that for no J_ℓ some $M_{\iota(j,\ell)}$ directly follows an $M_{\iota(\ell,k)}$. Such an ordering of the pairs (j,k) can easily be constructed. Due to this property of the ordering, partial sums of the processing times can be defined unambiguously by

$$q_{\ell i} = \begin{cases} i\mu + \lambda & \text{if } i = \iota(\ell,k) \text{ and } (\ell,k) \in A \\ i\mu + \lambda + 1 & \text{if } i = \iota(\ell,k) \text{ and } (\ell,k) \notin A \\ i\mu - \lambda & \text{if } i+1 = \iota(j,\ell) \text{ and } (j,\ell) \in A \\ i\mu - \lambda - 1 & \text{if } i+1 = \iota(j,\ell) \text{ and } (j,\ell) \notin A \\ i\mu & \text{otherwise} \end{cases}$$

for $i = 1,\ldots,m$, $\ell = 1,\ldots,n$, where

$$\lambda \geq 1$$
$$\mu \geq 2\lambda + 3$$

The processing times are given by (cf. (8))

$$p_{\ell 1} = q_{\ell 1}$$
$$p_{\ell i} = q_{\ell i} - q_{\ell\ i-1} \quad (i = 2,\ldots,m)$$

Through the choice of μ, these processing times are all strictly positive integers.

We can now compute the c_{jk}, as defined by (9). Through the choice of λ, it is immediate that $q_{ji} - q_{k\ i-1}$ is maximal for $i = \iota(j,k)$. Hence,

$$c_{jk} = \begin{cases} \mu + 2\lambda & \text{if } (j,k) \in A \\ \mu + 2\lambda + 2 & \text{if } (j,k) \notin A \end{cases}$$

Since $q_{\ell m} = m\mu$ for all J_ℓ, it now follows that G has a hamilton path if and only if this $n|m|F,\textit{no wait}|C_{max}$ problem has a solution with value

$$C_{max} \leq (n - 1)(\mu + 2\lambda) + m\mu$$

(*b*) DIRECTED HAMILTON PATH $\propto n|m|F,\textit{no wait},w_j{=}1|\sum w_jC_j$

G has a hamilton path if and only if the $n|m|F,\textit{no wait},w_j{=}1|\sum w_jC_j$ problem, constructed as in (*a*), has a solution with value

$$\sum_j C_j \leq \tfrac{1}{2}n(n - 1)(\mu + 2\lambda) + nm\mu$$

□

5. CONCLUDING REMARKS

The results presented in section 4 offer a reasonable insight into the location of the borderline between "easy" and "hard" machine scheduling problems. Computational experience with many problems proved to be NP-complete confirms the impression that a polynomial-bounded algorithm for one and thus for all of them is highly unlikely to exist. As indicated previously, NP-completeness thus functions as a formal justification to use enumerative solution methods such as branch-and-bound.

Most classical machine scheduling problems have now been shown to be polynomially solvable or NP-complete. Some notable exceptions are indicated by question-marks in Table I. They are briefly discussed below.

As to the $n|1|w_j=1|\sum w_jT_j$ problem, there is impressive computational evidence that this problem is extremely difficult. Although the equality of the weights leads to stronger elimination criteria, problems with thirty jobs may already require large amounts of computer time when attacked by the best available branch-and-bound methods [10;32]. We strongly suspect that this problem is NP-complete. This would indicate a major difference between the $\sum w_jT_j$ and $\sum w_jU_j$ problems, as demonstrated by Table I. Our NP-completeness proof for the $n|1||\sum w_jT_j$ problem depends on the permitted inequality of the weights and furthermore all due dates are equal, in which case we can solve the $n|1|w_j=1|\sum w_jT_j$ problem by ordering the jobs according to nondecreasing processing times (cf. [32]). Thus no straightforward extension of this proof is possible.

With respect to the $n|1|prec,w_j=1|\sum w_jC_j$ problem, it has been indicated that W.A. Horn [13] and J.B. Sidney [33;34] have developed good algorithms to cover precedence constraints that are as a matter of fact slightly more complicated than *tree*. In both cases, no extension to general precedence constraints seems possible and our guess is that these researchers have indeed reached the aforementioned borderline.

The complexity of the $n|3|F,no\ wait|C_{max}$ and $n|2|F,no\ wait,w_j=1|\sum w_jC_j$ problems is not clear. To stimulate further research, we will award a wooden shoe to the first scientist who finds a polynomial-bounded algorithm for any one of these problems.

The question of the complexity of the $n|3|I,prec,p_{j1}=1|C_{max}$ problem has

been raised already in the remarkable paper by J.D. Ullman [37]. Knowledge about the complexity of the $n|2|I,prec,p_{j1}=1,w_j=1|\sum w_jC_j$ problem, not included in Table I, would also more precisely locate the borderline for problems on identical machines.

The results of section 4 could be extended and refined in numerous other ways as well. To mention just one possibility, it might be investigated if problems that are in P if $p_{jr}=1$ remain in P if $1\leq p_{jr}\leq p_*$. However, any such result would probably be more of theoretical than of practical use, whereas good algorithms for problems with equal processing times often find application in lower bound computations for more complicated problems (e.g., see [32]). Similar remarks apply to a number of other refinements.

Finally, the general observation that membership of P versus NP-completeness only yields a very coarse measure of complexity is valid with regards to sequencing problems as well. On one hand, the question has been raised whether polynomial algorithms are really good [1]. On the other hand, there are important differences in complexity within the class of NP-complete problems; we note that a branch-and bound approach to the $n|1|r_j\geq 0|L_{max}$ problem has figured successfully within a lower bound computation for the $n|m|G|C_{max}$ problem [2]. Developing a measure that allows further distinction withing the class of hard problems remains a major research challenge.

ACKNOWLEDGEMENTS

We gratefully acknowledge the valuable cooperation with H.W. Lenstra, Jr. and E.L. Lawler.

REFERENCES

1. J.M. ANTHONISSE, P. VAN EMDE BOAS, Are Polynomial Algorithms Really Good? Report BW 40, Mathematisch Centrum, Amsterdam, 1974.

2. P. BRATLEY, M. FLORIAN, P. ROBILLARD, On Sequencing with Earliest Starts and Due Dates with Application to Computing Bounds for the $(n|m|G|F_{\max})$ Problem, *Naval Res. Logist. Quart.* 20(1973)57-67.

3. J. BRUNO, E.G. COFFMAN Jr., R. SEHTI, Scheduling Independent Tasks to Reduce Mean Finishing Time, *Comm. ACM* 17(1974)382-387.

4. E.G. COFFMAN, R.L. GRAHAM, Optimal Scheduling for Two-Processor Systems, *Acta Informat.* 1(1972)200-213.

5. R.W. CONWAY, W.L. MAXWELL, L.W. MILLER, *Theory of Scheduling*, Addison-Wesley, Reading, Mass., 1967.

6. S.A. COOK, The Complexity of Theorem-Proving Procedures, pp.151-158 in *Proc. 3rd Annual ACM Symposium on Theory of Computing*, May 1971.

7. J. EDMONDS, Paths, Trees, and Flowers, *Canad. J. Math.* 17(1965)449-467.

8. J. EDMONDS, The Chinese Postman's Problem, *Operations Res.* 13 Suppl.1 (1965)B-73.

9. J. EDMONDS, E.L. JOHNSON, Matching, Euler Tours and the Chinese Postman, *Math. Programming* 5(1973)88-124.

10. M.L. FISHER, A Dual Algorithm for the One-Machine Scheduling Problem, Technical Report No. 243, Department of Operations Research, College of Engineering, Cornell University, Ithaca, 1974.

11. P.C. GILMORE, R.E. GOMORY, Sequencing a One State-Variable Machine: A Solvable Case of the Traveling Salesman Problem, *Operations Res.* 12 (1964)655-679.

12. W.W. HARDGRAVE, G.L. NEMHAUSER, A Geometric Method and a Graphical Algorithm for a Sequencing Problem, *Operations Res.* 11(1963)889-900.

13. W.A. HORN, Single-Machine Job Sequencing with Treelike Precedence Ordering and Linear Delay Penalties, *SIAM J. Appl. Math.* 23(1972)189-202.

14. W.A. HORN, Minimizing Average Flow Time with Parallel Machines, *Operations Res.* 21(1973)846-847.

15. W.A. HORN, Some Simple Scheduling Algorithms, *Naval Res. Logist. Quart.* 21(1974)177-185.

16. T.C. HU, Parallel Sequencing and Assembly Line Problems, *Operations Res.* 9(1961)841-848.

17. J.R. JACKSON, An Extension of Johnson's Results on Job Lot Scheduling, *Naval Res. Logist. Quart.* 3(1956)201-203.

18. S.M. JOHNSON, Optimal Two- and Three-Stage Production Schedules with Setup Times Included, *Naval Res. Logist. Quart.* 1(1954)61-68.

19. R.M. KARP, Reducibility among Combinatorial Problems, pp. 85-103 in R.E. MILLER, J.W. THATCHER (eds.), *Complexity of Computer Computations*, Plenum Press, New York, 1972.

20. D.E. KNUTH, A Terminological Proposal, *SIGACT News* 6.1(1974)12-18.

21. B.J. LAGEWEG, Private Communication, January 22, 1975.

22. E.L. LAWLER, On Scheduling Problems with Deferral Costs, *Management Sci.* 11(1964)280-288.

23. E.L. LAWLER, Optimal Sequencing of a Single Machine Subject to Precedence Constraints, *Management Sci.* 19(1973)544-546.

24. E.L. LAWLER, *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart, and Winston, New York, to appear.

25. E.L. LAWLER, J.M. MOORE, A Functional Equation and its Application to Resource Allocation and Sequencing Problems, *Management Sci.* 16(1969) 77-84.

26. J.K. LENSTRA, A.H.G. RINNOOY KAN, Some Simple Applications of the Travelling Salesman Problem, Report BW 38, Mathematisch Centrum, Amsterdam, 1974; Working Paper WP/74/12, Graduate School of Management, Delft, 1974.

27. J.M. MOORE, An *n* Job, One Machine Sequencing Algorithm for Minimizing the Number of Late Jobs, *Management Sci.* 15(1968)102-109.

28. H. MÜLLER-MERBACH, *Optimale Reihenfolgen*, Springer, Berlin etc., 1970.

29. J. PIEHLER, Ein Beitrag zum Reihenfolgeproblem, *Unternehmensforschung* 4(1960)138-142.

30. S.S. REDDI, C.V. RAMAMOORTHY, On the Flow-Shop Sequencing Problem with No Wait in Process, *Operational Res. Quart.* 23(1972)323-331.

31. A.H.G. RINNOOY KAN, The Machine Scheduling Problem, Report BW 27, Mathematisch Centrum, Amsterdam, 1973; Report R/73/4, Graduate School of Management, Delft, 1973.

32. A.H.G. RINNOOY KAN, B.J. LAGEWEG, J.K. LENSTRA, Minimizing Total Costs in One-Machine Scheduling, *Operations Res.*, to appear.

33. J.B. SIDNEY, One-Machine Sequencing with Precedence Relations and Deferral Costs - Part I, Working Paper 124, Faculty of Commerce and Business Administration, University of British Columbia, Vancouver, 1972.

34. J.B. SIDNEY, One-Machine Sequencing with Precedence Relations and Deferral Costs - Part II, Working Paper 125, Faculty of Commerce and Business Administration, University of British Columbia, Vancouver, 1972.

35. W.E. SMITH, Various Optimizers for Single-State Production, *Naval Res. Logist. Quart.* 3(1956)59-66.

36. W. SZWARC, Solution of the Akers-Friedman Scheduling Problem, *Operations Res.* 8(1960)782-788.

37. J.D. ULLMAN, Polynomial Complete Scheduling Problems, pp.96-100 in *Proc. 4th Symposium on Operating System Principles*, October 1973.

38. D.A. WISMER, Solution of the Flowshop-Scheduling Problem with No Intermediate Queues, *Operations Res.* 20(1972)689-697.