stichting
mathematisch
centrum

MC

AFDELING MATHEMATISCHE BESLISKUNDE
(DEPARTMENT OF OPERATIONS RESEARCH)

BW 146/81

AUGUSTUS

E.L. LAWLER, J.K. LENSTRA, A.H.G. RINNOOY KAN

RECENT DEVELOPMENTS IN DETERMINISTIC SEQUENCING AND SCHEDULING: A SURVEY

Preprint

kruislaan 413 1098 SJ amsterdam

Printed at the Mathematical Centre, 413 Kruislaan, 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.).

1980 Mathematics Subject Classification: 90B35, 68C15, 68C25

RECENT DEVELOPMENTS IN DETERMINISTIC SEQUENCING AND SCHEDULING: A SURVEY

E.L. LAWLER
University of California, Berkeley

J.K. LENSTRA
Mathematisch Centrum, Amsterdam

A.H.G. RINNOOY KAN
Erasmus University, Rotterdam

ABSTRACT

The theory of deterministic sequencing and scheduling has expanded rapidly during the past years. We survey the state of the art with respect to optimization and approximation algorithms and interpret these in terms of computational complexity theory. Special cases considered are single machine scheduling, identical, uniform and unrelated parallel machine scheduling, and open shop, flow shop and job shop scheduling. This paper is a revised version of the survey by Graham *et al.* (*Ann. Discrete Math.* 5(1979)287-326), with emphasis on recent developments.

KEY WORDS & PHRASES: *deterministic scheduling, single machine, parallel machines, open shop, flow shop, job shop, polynomial-time algorithm, NP-hardness, worst-case analysis.*
NOTE: This report will appear in *Deterministic and Stochastic Scheduling*, edited by M.A.H. Dempster, J.K. Lenstra and A.H.G. Rinnooy Kan, to be published by Reidel, Dordrecht, in 1982.

RECENT DEVELOPMENTS IN DETERMINISTIC SEQUENCING AND SCHEDULING: A SURVEY

E.L. Lawler[1], J.K. Lenstra[2], A.H.G. Rinnooy Kan[3]

[1]University of California, Berkeley
[2]Mathematisch Centrum, Amsterdam
[3]Erasmus University, Rotterdam

ABSTRACT

The theory of deterministic sequencing and scheduling has expanded rapidly during the past years. We survey the state of the art with respect to optimization and approximation algorithms and interpret these in terms of computational complexity theory. Special cases considered are single machine scheduling, identical, uniform and unrelated parallel machine scheduling, and open shop, flow shop and job shop scheduling. This paper is a revised version of the survey by Graham *et al.* (*Ann. Discrete Math.* 5(1979)287-326), with emphasis on recent developments.

1. INTRODUCTION

In this paper we attempt to survey the rapidly expanding area of deterministic scheduling theory. Although the field only dates back to the early fifties, an impressive amount of literature has been created and the remaining open problems are currently under heavy attack. An exhaustive discussion of all available material would be impossible - we will have to restrict ourselves to the most significant results, paying special attention to recent developments and omitting detailed theorems and proofs. For further information the reader is referred to the classic book by Conway, Maxwell and Miller [Conway *et al.* 1967], the introductory textbook by Baker [Baker 1974], the advanced expository articles collected by Coffman [Coffman 1976] and a few survey papers and theses [Bakshi & Arora 1969; Lenstra 1977; Liu 1976; Rinnooy Kan 1976]. This paper itself is a revised and updated version of a recent survey [Graham *et al.* 1979].

The outline of the paper is as follows. Section 2 introduces the essential notation and presents a detailed problem classification. Sections 3, 4 and 5 deal with single machine, parallel machine, and open shop, flow shop and job shop problems, respectively. In each section we briefly outline the relevant complexity results and optimization and approximation algorithms. Section 6 contains some concluding remarks.

We shall be making extensive use of concepts from the *theory of computational complexity* [Cook 1971; Karp 1972]. Several introductory surveys of this area are currently available [Karp 1975; Garey & Johnson 1979; Lenstra & Rinnooy Kan 1979] and hence terms like *(pseudo)polynomial-time algorithm* and *(binary* and *unary) NP-hardness* will be used without further explanation.

2. PROBLEM CLASSIFICATION

2.1. Introduction

Suppose that n *jobs* J_j $(j = 1,\ldots,n)$ have to be processed on m *machines* M_i $(i = 1,\ldots,m)$. Throughout, we assume that each machine can process at most one job at a time and that each job can be processed on at most one machine at a time. Various job, machine and scheduling characteristics are reflected by a three-field problem classification $\alpha|\beta|\gamma$, to be introduced in this section.

2.2. Job data

In the first place, the following data can be specified for each J_j:

- a *number of operations* m_j;
- one or more *processing times* p_j or p_{ij}, that J_j has to spend on the various machines on which it requires processing;
- a *release date* r_j, on which J_j becomes available for processing;
- a *due date* d_j, by which J_j should ideally be completed;
- a *weight* w_j, indicating the relative importance of J_j;
- a nondecreasing real *cost function* f_j, measuring the cost $f_j(t)$ incurred if J_j is completed at time t.

In general, m_j, p_j, p_{ij}, r_j, d_j and w_j are integer variables.

2.3. Machine environment

We shall now describe the first field $\alpha = \alpha_1\alpha_2$ specifying the machine environment. Let $\circ$ denote the empty symbol.

If $\alpha_1 \in \{\circ,P,Q,R\}$, each J_j consists of a single operation that can be processed on any M_i; the processing time of J_j on M_i is p_{ij}. The four values are characterized as follows:

- $\alpha_1 = \circ$: *single machine*; $p_{1j} = p_j$;
- $\alpha_1 = P$: *identical parallel machines*; $p_{ij} = p_j$ $(i = 1,\ldots,m)$;

- α_1 = Q: *uniform parallel machines*; $p_{ij} = p_j/q_i$ for a given *speed* q_i of M_i (i = 1,...,m);
- α_1 = R: *unrelated parallel machines*.

If α_1 = O, we have an *open shop*, in which each J_j consists of a set of operations $\{O_{1j},...,O_{mj}\}$. O_{ij} has to be processed on M_i during p_{ij} time units, but the order in which the operations are executed is immaterial. If $\alpha_1 \in \{F,J\}$, an ordering is imposed on the set of operations corresponding to each job. If α_1 = F, we have a *flow shop*, in which each J_j consists of a chain $(O_{1j},...,O_{mj})$. O_{ij} has to be processed on M_i during p_{ij} time units. If α_1 = J, we have a *job shop*, in which each J_j consists of a chain $(O_{1j},...,O_{m_jj})$. O_{ij} has to be processed on a given machine μ_{ij} during p_{ij} time units, with $\mu_{i-1,j} \neq \mu_{ij}$ for i = 2,...,m_j.

If α_2 is a positive integer, then m is *constant* and equal to α_2. If α_2 = °, then m is assumed to be *variable*. Obviously, α_1 = ° if and only if α_2 = 1.

2.4. Job characteristics

The second field $\beta \subset \{\beta_1,...,\beta_5\}$ indicates a number of job characteristics, which are defined as follows.

1. $\beta_1 \in \{pmtn,°\}$
   - β_1 = *pmtn* : *Preemption* (job splitting) is allowed: the processing of any operation may be interrupted and resumed at a later time.
   - β_1 = ° : No preemption is allowed.
2. $\beta_2 \in \{prec,tree,°\}$
   - β_2 = *prec* : A *precedence relation* $\rightarrow$ between the jobs is specified. It is derived from a directed acyclic graph G with vertex set {1,...,n}. If G contains a directed path from j to k, we write $J_j \rightarrow J_k$ and require that J_j is completed before J_k can start.
   - β_2 = *tree* : G is a *rooted tree* with either outdegree at most one for each vertex or indegree at most one for each vertex.
   - β_2 = ° : No precedence relation is specified.
3. $\beta_3 \in \{r_j,°\}$
   - $\beta_3 = r_j$: *Release dates* that may differ per job are specified.
   - β_3 = ° : All r_j = 0.
4. $\beta_4 \in \{m_j \leq \bar{m},°\}$
   - $\beta_4 = m_j \leq \bar{m}$: A *constant upper bound* on m_j is specified (only if α_1 = J).
   - β_4 = ° : All m_j are arbitrary integers.
5. $\beta_5 \in \{p_{ij}=1,°\}$
   - $\beta_5 = p_{ij}=1$: Each operation has *unit processing time* (if $\alpha_1 \in \{°,P,Q\}$, we write p_j=1; if α_1 = R, p_{ij}=1 will not occur).
   - β_5 = ° : All p_{ij} (p_j) are arbitrary integers.

2.5. Optimality criteria

The third field $\gamma \in \{f_{max}, \Sigma f_j\}$ refers to the optimality criterion chosen. Given a schedule, we can compute for each J_j:
- the *completion time* C_j;
- the *lateness* $L_j = C_j - d_j$;
- the *tardiness* $T_j = \max\{0, C_j - d_j\}$;
- the *unit penalty* $U_j = 0$ if $C_j \leq d_j$, $U_j = 1$ otherwise.

The optimality criteria most commonly chosen involve the minimization of

$$f_{max} \in \{C_{max}, L_{max}\}$$

where $f_{max} = \max_j\{f_j(C_j)\}$ with $f_j(C_j) = C_j, L_j$, respectively, or

$$\Sigma f_j \in \{\Sigma C_j, \Sigma T_j, \Sigma U_j, \Sigma w_j C_j, \Sigma w_j T_j, \Sigma w_j U_j\}$$

where $\Sigma f_j = \Sigma_{j=1}^{n} f_j(C_j)$ with $f_j(C_j) = C_j, T_j, U_j, w_j C_j, w_j T_j, w_j U_j$, respectively.

It should be noted that $\Sigma w_j C_j$ and $\Sigma w_j L_j$ differ by a constant $\Sigma w_j d_j$ and hence are *equivalent*. Furthermore, any schedule minimizing L_{max} also minimizes T_{max} and U_{max}, but not *vice versa*.

The optimal value of γ will be denoted by γ^*, the value produced by an (approximation) algorithm A by $\gamma(A)$. If a known upper bound ρ on $\gamma(A)/\gamma^*$ is best possible in the sense that examples exist for which $\gamma(A)/\gamma^*$ equals or asymptotically approaches ρ, this will be denoted by a dagger ($\dagger$).

2.6. Examples

$1|prec|L_{max}$: minimize maximum lateness on a single machine subject to general precedence constraints. This problem can be solved in polynomial time (Section 3.2).

$R|pmtn|\Sigma C_j$: minimize total completion time on a variable number of unrelated parallel machines, allowing preemption. The complexity of this problem is unknown (Section 4.4.3).

$J3|p_{ij}=1|C_{max}$: minimize maximum completion time in a 3-machine job shop with unit processing times. This problem is NP-hard (Section 5.4.1).

2.7. Reducibility among scheduling problems

Each scheduling problem in the class outlined above corresponds to an 7-tuple $(v_0,\ldots,v_6)$, where v_i is a vertex of graph G_i drawn in Figure 1 ($i = 0,\ldots,6$). For two problems $P' = (v_0',\ldots,v_6')$ and $P = (v_0,\ldots,v_6)$, we write $P' \rightarrow P$ if either $v_i' = v_i$ or G_i contains a directed path from v_i' to v_i, for $i = 0,\ldots,6$. The reader should verify that $P' \rightarrow P$ implies $P' \propto P$. The graphs thus define elementary reductions among scheduling problems. It follows that
- if $P' \rightarrow P$ and P is well solved, then P' is well solved;
- if $P' \rightarrow P$ and P' is NP-hard, then P is NP-hard.

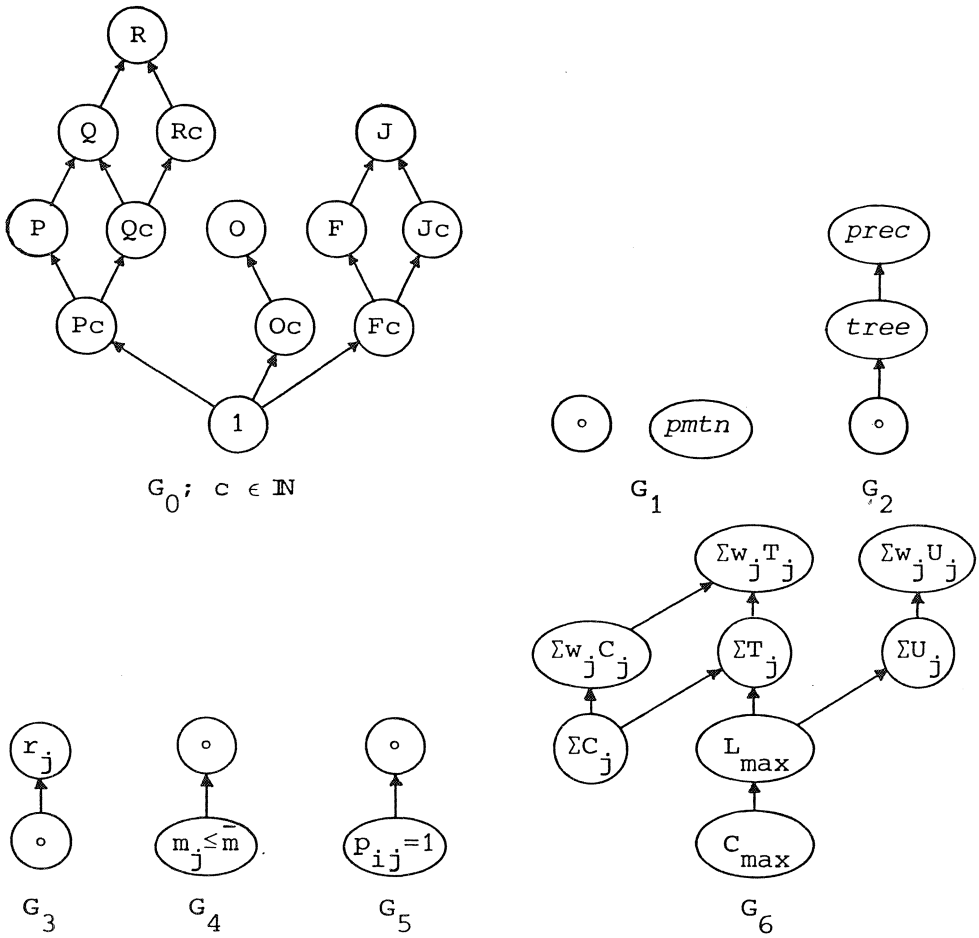


Figure 1

3. SINGLE MACHINE PROBLEMS

3.1. Introduction

The single machine case has been the object of extensive research ever since the seminal work by Jackson [Jackson 1955] and Smith [Smith 1956]. We will give a brief survey of the principal results, classifying them according to the optimality criterion chosen. As a general result, we note that if all $r_j = 0$ we need only consider schedules without preemption and without machine idle time [Conway *et al.* 1967].

3.2. Minimizing maximum cost

A crucial result in this section is an $O(n^2)$ algorithm to solve $1|prec|f_{max}$ for arbitrary nondecreasing cost functions [Lawler 1973]. At each step of the algorithm, let S denote the index set of unscheduled jobs, let $p(S) = \Sigma_{j \in S} p_j$, and let $S' \subset S$ indicate

the jobs all whose successors have been scheduled. One selects J_k for the last position among $\{J_j | j \in S\}$ by requiring that $f_k(p(S)) \leq f_j(p(S))$ for all $j \in S'$.

This method has been generalized to an $O(n^2)$ algorithm for $1|pmtn,prec,r_j|f_{max}$ [Baker *et al.* 1982]. First, the release dates are modified such that $r_j+p_j \leq r_k$ whenever $J_j \rightarrow J_k$. Next, the jobs are scheduled in order of nondecreasing release dates; this creates a number of *blocks* that can be considered separately. From among the jobs without successors in a certain block, a job J_k that yields minimum cost when put in the last position is selected, the other jobs in the block are rescheduled in order of nondecreasing release dates, and J_k is assigned to the remaining time intervals. By repeated application of this procedure to each of the resulting subblocks, one obtains an optimal schedule with at most n-1 preemptions in $O(n^2)$ time.

The remainder of this section deals with nonpreemptive L_{max} problems. The general $1|r_j|L_{max}$ problem is unary NP-hard [Lenstra *et al.* 1977]. However, polynomial algorithms exist if all r_j are equal, all d_j are equal, or all p_j are equal. The first case is solved by a specialization of Lawler's method, known as *Jackson's rule* [Jackson 1955]: schedule the jobs in order of nondecreasing due dates. The second case is solved similarly by scheduling the jobs in order of nondecreasing release dates.

As to the third case, $1|r_j,p_j=1|L_{max}$ is solved by the *extended Jackson's rule*: at any time, schedule an available job with smallest due date. The problem $1|r_j,p_j=p|L_{max}$, where p is an arbitrary integer, requires a more sophisticated approach [Simons 1978]. Let us first consider the simpler problem of finding a feasible schedule with respect to given release dates r_j and deadlines d_j. If application of the extended Jackson's rule yields such a schedule, we are finished; otherwise, let J_ℓ be the first late job and let J_k be the last job preceding J_ℓ such that $d_k > d_\ell$. If J_k does not exist, there is no feasible schedule; otherwise, the only hope of obtaining such a schedule is to postpone J_k by forcing it to yield precedence to the set of jobs currently between J_k and J_ℓ. This is achieved by declaring the interval between the starting time of J_k and the smallest release date of this set to be a *forbidden region* in which no job is allowed to start and applying the extended Jackson's rule again subject to this constraint. Since at each iteration at least one starting time of the form r_j+hp $(1 \leq j,h \leq n)$ is excluded, at most n^2 iterations will occur and the feasibility question is answered in $O(n^3 \log n)$ time. An improved implementation requires only $O(n \log n)$ time [Garey *et al.* 1981A]. Bisection search over the possible L_{max} values leads to a polynomial algorithm for $1|r_j,p_j=p|L_{max}$.

These three special cases remain well solved in the presence of precedence constraints. It suffices to update release and due dates such that $r_j < r_k$ and $d_j < d_k$ whenever $J_j \rightarrow J_k$ [Lageweg *et al.* 1976].

Various elegant enumerative methods exist for solving

$1|prec,r_j|L_{max}$. Baker and Su [Baker & Su 1974] obtain a lower bound by allowing preemption; their enumeration scheme simply generates all *active schedules*, *i.e.* schedules in which one cannot decrease the starting time of an operation without increasing the starting time of another one. McMahon and Florian [McMahon & Florian 1975] propose a more ingenious approach; a slight modification of their algorithm allows very fast solution of quite large problems [Lageweg *et al.* 1976]. Carlier [Carlier 1980] describes a related method of comparable efficiency.

Very little work has been done on worst-case analysis of approximation algorithms for single machine problems. For $1|r_j|L_{max}$, Potts [Potts 1980B] presents an iterative version of the extended Jackson's rule (IJ) and shows that, if $r_j \geq 0$ and $d_j \leq 0$ $(j = 1,\ldots,n)$,

$$L_{max}(IJ)/L^*_{max} \leq \frac{3}{2}. \qquad (\dagger)$$

3.3. Minimizing total cost

3.3.1. $1|\beta|\Sigma w_jC_j$

The case $1||\Sigma w_jC_j$ can be solved in O(n log n) time by *Smith's rule*: schedule the jobs according to nonincreasing ratios w_j/p_j [Smith 1956]. If all weights are equal, this amounts to the SPT rule of executing the jobs on the basis of shortest processing time first, a rule that is often used in more complicated situations without much empirical, let alone theoretical, support for its superior quality (*cf.* Section 5.4.2).

This result has been extended to O(n log n) algorithms that deal with *tree*-like [Horn 1972; Adolphson & Hu 1973; Sidney 1975] and even *series-parallel* [Lawler 1978] precedence constraints; see [Adolphson 1977] for an $O(n^3)$ algorithm covering a slightly more general case. The crucial observation to make here is that, if $J_j \rightarrow J_k$ with $w_j/p_j < w_k/p_k$ and if all other jobs either have to precede J_j, succeed J_k, or are incomparable with both, then J_j and J_k are adjacent in at least one optimal schedule and can effectively be treated as one job with processing time p_j+p_k and weight w_j+w_k. By successive application of this device, starting at the bottom of the precedence tree, one will eventually obtain an optimal schedule. Addition of general precedence constraints results in NP-hardness, even if all $p_j = 1$ or all $w_j = 1$ [Lawler 1978; Lenstra & Rinnooy Kan 1978].

If release dates are introduced, $1|r_j|\Sigma C_j$ is already unary NP-hard [Lenstra *et al.* 1977]. In the preemptive case, $1|pmtn,r_j|\Sigma C_j$ can be solved by an obvious extension of Smith's rule, but, surprisingly, $1|pmtn,r_j|\Sigma w_jC_j$ is unary NP-hard [Labetoulle *et al.* 1979].

For $1|r_j|\Sigma w_jC_j$, several elimination criteria and branch-and-bound algorithms have been proposed [Rinaldi & Sassano 1977; Bianco & Ricciardelli 1981; Hariri & Potts 1981].

3.3.2. $1|\beta|\Sigma w_j T_j$

$1||\Sigma w_j T_j$ is a unary NP-hard problem [Lawler 1977; Lenstra *et al.* 1977], for which various enumerative solution methods have been proposed. Elimination criteria developed for the problem [Emmons 1969; Shwimer 1972] can be extended to the case of arbitrary nondecreasing cost functions [Rinnooy Kan *et al.* 1975]. Lower bounds can be based on a linear assignment relaxation using an underestimate of the cost of assigning J_j to position k [Rinnooy Kan *et al.* 1975], a fairly similar relaxation to a transportation problem [Gelders & Kleindorfer 1974, 1975], and relaxation of the requirement that the machine can process at most one job at a time [Fisher 1976]. In the latter approach, one attaches "prices" (*i.e.*, Lagrangean multipliers) to each unit-time interval. Multiplier values are sought for which a cheapest schedule does not violate the capacity constraint. The resulting algorithm is quite successful on problems with up to 50 jobs, although a straightforward but cleverly implemented dynamic programming approach [Baker & Schrage 1978] offers a surprisingly good alternative.

If all $p_j = 1$, we have a simple linear assignment problem, the cost of assigning J_j to position k being given by $f_j(k)$. If all $w_j = 1$, the problem can be solved by a pseudopolynomial algorithm in $O(n^4 \Sigma p_j)$ time [Lawler 1977]; the computational complexity of $1||\Sigma T_j$ with respect to a binary encoding remains an open question.

Addition of precedence constraints yields NP-hardness, even for $1|prec,p_j=1|\Sigma T_j$ [Lenstra & Rinnooy Kan 1978].

If we introduce release dates, $1|r_j,p_j=1|\Sigma w_j T_j$ can again be solved as a linear assignment problem, whereas $1|r_j|\Sigma T_j$ is obviously unary NP-hard.

3.3.3. $1|\beta|\Sigma w_j U_j$

An algorithm due to Moore [Moore 1968] allows solution of $1||\Sigma U_j$ in O(n log n) time: jobs are added to the schedule in order of nondecreasing due dates, and if addition of J_j results in this job being completed after d_j, the scheduled job with the largest processing time is marked to be late and removed. This procedure can be extended to cover the case in which certain specified jobs have to be on time [Sidney 1973]; the further generalization in which jobs have to meet given *deadlines* occurring at or after their due dates is binary NP-hard [Lawler -]. The problem also remains solvable in O(n log n) time if we add *agreeable weights* (*i.e.*, $p_j < p_k \Rightarrow w_j \geq w_k$) [Lawler 1976A] or *agreeable release dates* (*i.e.*, $d_j < d_k \Rightarrow r_j \leq r_k$) [Kise *et al.* 1978]. $1||\Sigma w_j U_j$ is binary NP-hard [Karp 1972], but can be solved by dynamic programming in $O(n\Sigma p_j)$ time [Lawler & Moore 1969].

Again, $1|prec,p_j=1|\Sigma U_j$ is NP-hard [Garey & Johnson 1976], even for *chain*-like precedence constraints [Lenstra & Rinnooy Kan 1980].

Of course, $1|r_j|\Sigma U_j$ is unary NP-hard, but dynamic programming

techniques can be applied to solve $1|pmtn,r_j|\Sigma U_j$ in $O(n^6)$ time and $1|pmtn,r_j|\Sigma w_j U_j$ in $O(n^3(\Sigma w_j)^3)$ time [Lawler -].

For $1||\Sigma w_j U_j$, Sahni [Sahni 1976] presents algorithms A_k with $O(n^3 k)$ running time such that

$$\Sigma w_j \bar{U}_j(A_k)/\Sigma w_j \bar{U}_j^* \geq 1 - \frac{1}{k},$$

where $\bar{U}_j = 1-U_j$. For $1|tree|\Sigma w_j U_j$, Ibarra and Kim [Ibarra & Kim 1978] give algorithms B_k of order $O(kn^{k+2})$ with the same worst-case error bound.

4. PARALLEL MACHINE PROBLEMS

4.1. Introduction

Recall from Section 2.3 the definitions of *identical*, *uniform* and *unrelated* machines, denoted by P, Q and R, respectively.

Nonpreemptive parallel scheduling problems tend to be difficult. This can be inferred immediately from the fact that $P2||C_{max}$ and $P2||\Sigma w_j C_j$ are binary NP-hard [Bruno *et al.* 1974; Lenstra *et al.* 1977]. If we are to look for polynomial algorithms, it follows that we should either restrict attention to the special case $p_j = 1$, as we do in Section 4.2, or concern ourselves with the ΣC_j criterion, as we do in the first three subsections of Section 4.3. The remaining part of Section 4.3 is entirely devoted to enumerative optimization methods and approximation algorithms for various NP-hard problems.

The situation is much brighter with respect to *preemptive* parallel scheduling. For example, $P|pmtn|C_{max}$ has long been known to admit a simple $O(n)$ algorithm [McNaughton 1959]. Many new results for the ΣC_j, C_{max}, L_{max}, ΣU_j and $\Sigma w_j U_j$ criteria have been obtained quite recently. These are summarized in Section 4.4. With respect to other criteria, $P2|pmtn|\Sigma w_j C_j$ turns out to be NP-hard (see Section 4.4.1). Little is known about $P|pmtn|\Sigma T_j$, but we know from Section 3.3.2 that $1|pmtn|\Sigma w_j T_j$ is already NP-hard.

4.2. Nonpreemptive scheduling: unit processing times

4.2.1. $Q|p_j=1|\Sigma f_j$, $Q|p_j=1|f_{max}$

A simple transportation network model provides an efficient solution method for $Q|p_j=1|\Sigma f_j$ and $Q|p_j=1|f_{max}$.

Let there be n sources j ($j = 1,\ldots,n$) and mn sinks (i,k) ($i = 1,\ldots,m$, $k = 1,\ldots,n$). Set the cost of arc (j,(i,k)) equal to $c_{ijk} = f_j(k/q_i)$. The arc flow x_{ijk} is to have the interpretation:

$$x_{ijk} = \begin{cases} 1 & \text{if } J_j \text{ is executed on } M_i \text{ in the k-th position,} \\ 0 & \text{otherwise.} \end{cases}$$

Then the problem is to minimize

$$\Sigma_{i,j,k}\, c_{ijk}x_{ijk} \quad \text{or} \quad \max_{i,j,k}\{c_{ijk}x_{ijk}\}$$

subject to

$$\Sigma_{i,k}\, x_{ijk} = 1 \qquad \text{for all } j,$$

$$\Sigma_{j}\, x_{ijk} \leq 1 \qquad \text{for all } i,k,$$

$$x_{ijk} \geq 0 \qquad \text{for all } i,j,k.$$

The time required to prepare the data for this transportation problem is $O(mn^2)$. A careful analysis reveals that the problem can be solved (in integers) in $O(n^3)$ time. Since we may assume that $m \leq n$, the overall running time is $O(n^3)$.

We note that the special case $P|p_j=1|\Sigma U_j$ can be solved in $O(n \log n)$ time [Lawler 1976A]. The problem $P|r_j,p_j=p|L_{max}$ is solvable in polynomial time by an extension of the corresponding single machine algorithm (see Section 3.2) [Simons 1980].

4.2.2. $P|prec,p_j=1|C_{max}$

$P|prec,p_j=1|C_{max}$ is known to be NP-hard [Ullman 1975; Lenstra & Rinnooy Kan 1978]. It is an open question whether this remains true for any constant value of $m \geq 3$. The problem is well solved, however, if the precedence relation is of the *tree*-type or if $m = 2$.

$P|tree,p_j=1|C_{max}$ can be solved in $O(n)$ time by Hu's algorithm [Hu 1961; Hsu 1966; Sethi 1976A]. The *level* of a job is defined as the number of jobs in the unique path to the root of the precedence tree. At the beginning of each time unit, as many available jobs as possible are scheduled on the m machines, where highest priority is granted to the jobs with the largest levels. Thus, Hu's algorithm is a nonpreemptive *list scheduling* algorithm, whereby at each step the available job with the highest ranking on a priority list is assigned to the first machine that becomes available. It can also be viewed as a *critical path* scheduling algorithm: the next job chosen is the one which heads the longest current chain of unexecuted jobs.

If the precedence constraints are in the form of an *intree* (each job has at most one successor), then Hu's algorithm can be adapted to minimize L_{max}; in the case of an *outtree* (each job has at most one predecessor), the L_{max} problem turns out to be NP-hard [Brucker *et al.* 1977]. There are some recent algorithmic and NP-hardness results concerning $P|prec,p_j=1|C_{max}$ for precedence constraints other than intrees or outtrees, such as *opposing forests* (combinations of intrees and outtrees), *level graphs*, and so forth [Dolev 1981; Garey *et al.* 1981B; Warmuth 1980].

$P2|prec,p_j=1|C_{max}$ can be solved by various polynomial algo-

rithms [Fujii *et al.* 1969, 1971; Coffman & Graham 1972; Gabow 1980].

In the approach due to Fujii *et al.*, an undirected graph is constructed with vertices corresponding to jobs and edges {j,k} whenever J_j and J_k can be executed simultaneously, *i.e.*, $J_j \neq J_k$ and $J_k \neq J_j$. An optimal schedule is then derived from a maximum cardinality matching in the graph. Such a matching can be found in $O(n^3)$ time [Lawler 1976B].

The Coffman-Graham approach leads to an $O(n^2)$ list algorithm. First the jobs are labelled in the following way. Suppose labels 1,...,k have been applied and S is the subset of unlabelled jobs all of whose successors have been labelled. Then a job in S is given the label k+1 if the labels of its immediate succesors are *lexicographically minimal* with respect to all jobs in S. The priority list is given by ordering the jobs according to decreasing labels. It is possible to execute this algorithm in time almost linear in n plus the number of arcs in the precedence graph, if the graph is given in the form of a *transitive reduction* [Sethi 1976B].

Recently, Gabow developed an algorithm which has the same running time, but which does not require such a representation of the precedence graph.

Garey and Johnson present polynomial algorithm for this problem where, in addition, each job becomes available at its *release date* and has to meet a given *deadline*. In this approach, one processes the jobs in order of increasing modified deadlines. This modification requires $O(n^2)$ time if all $r_j = 0$ [Garey & Johnson 1976] and $O(n^3)$ time in the general case [Garey & Johnson 1977].

We note that $P|prec,p_j=1|\Sigma C_j$ is NP-hard [Lenstra & Rinnooy Kan 1978]. Hu's algorithm does not yield an optimal ΣC_j schedule in the case of intrees, but in the case of outtrees critical path scheduling minimizes both C_{max} and ΣC_j [Rosenfeld -]. The Coffman-Graham algorithm also minimizes ΣC_j [Garey -].

As far as approximation algorithms for $P|prec,p_j=1|C_{max}$ are concerned, the NP-hardness proof given in [Lenstra & Rinnooy Kan 1978] implies that, unless $P = NP$, the best possible worst-case bound for a polynomial-time algorithm would be 4/3. The performance of both Hu's algorithm and the Coffman-Graham algorithm has been analyzed.

When critical path (CP) scheduling is used, Chen and Liu [Chen 1975; Chen & Liu 1975] and Kunde [Kunde 1976] show that

$$C_{max}(CP)/C^*_{max} \leq \begin{cases} \frac{4}{3} & \text{for } m = 2, \\ 2 - \frac{1}{m-1} & \text{for } m \geq 3. \end{cases} \qquad (\dagger)$$

Lam and Sethi [Lam & Sethi 1977] use the Coffman-Graham (CG) algorithm to generate lists and show that

$$C_{max}(CG)/C^*_{max} \leq 2 - \frac{2}{m} \quad (m \geq 2). \qquad (\dagger)$$

If SS denotes the algorithm which schedules as the next job the

one having the greatest number of successors then it can be shown [Ibarra & Kim 1976] that

$$C_{max}(SS)/C^*_{max} \leq \frac{4}{3} \quad \text{for } m = 2. \qquad (\dagger)$$

Examples show that this bound does not hold for $m \geq 3$.

Finally, we mention some results for the more general case in which all $p_j \in \{1,k\}$. Both $P2|prec,p_j\in\{1,2\}|C_{max}$ and $P2|prec,p_j\in\{1,2\}|\Sigma C_j$ are NP-hard [Ullman 1975; Lenstra & Rinnooy Kan 1978]. For $P2|prec,p_j\in\{1,k\}|C_{max}$, Goyal [Goyal 1977] proposes a generalized version of the Coffman-Graham algorithm (GCG) and shows that

$$C_{max}(GCG)/C^*_{max} \leq \begin{cases} \frac{4}{3} & \text{for } k = 2, \\ \frac{3}{2} - \frac{1}{2k} & \text{for } k \geq 3. \end{cases} \qquad (\dagger)$$

4.3. Nonpreemptive scheduling: general processing times

4.3.1. $P||\Sigma w_jC_j$

The following generalization of the SPT rule for $1||\Sigma C_j$ (see Section 3.3.1) solves $P||\Sigma C_j$ in $O(n \log n)$ time [Conway *et al.* 1967]. Assume $n = km$ (dummy jobs with zero processing times can be added if not) and suppose $p_1 \leq \ldots \leq p_n$. Assign the m jobs $J_{(j-1)m+1}$, $J_{(j-1)m+2},\ldots,J_{jm}$ to m different machines ($j = 1,\ldots,k$) and execute the k jobs assigned to each machine in SPT order.

With respect to $P||\Sigma w_jC_j$, Eastman, Even and Isaacs [Eastman *et al.* 1964] show that after renumbering the jobs according to nonincreasing ratios w_j/p_j

$$\Sigma w_jC_j(LS) - \tfrac{1}{2}\Sigma_{j=1}^{n} w_jp_j \geq \tfrac{1}{m}(\Sigma_{j=1}^{n}\Sigma_{k=1}^{j} w_jp_k - \tfrac{1}{2}\Sigma_{j=1}^{n} w_jp_j). \qquad (\dagger)$$

It follows from this inequality that

$$\Sigma w_jC^*_j \geq \frac{m+n}{m(n+1)} \Sigma_{j=1}^{n} \Sigma_{k=1}^{j} w_jp_k.$$

In [Elmaghraby & Park 1974; Barnes & Brennan 1977] branch-and-bound algorithms based on this lower bound are developed.

Sahni [Sahni 1976] constructs algorithms A_k (in the same spirit as his approach for $1||\Sigma w_jU_j$ mentioned in Section 3.3.3) with $O(n(n^2k)^{m-1})$ running time for which

$$\Sigma w_jC_j(A_k)/\Sigma w_jC^*_j \leq 1 + \frac{1}{k}.$$

For $m = 2$, the running time of A_k can be improved to $O(n^2k)$.

4.3.2. $Q||\Sigma C_j$

The algorithm for solving $P||\Sigma C_j$ given in the previous section can be generalized to the case of uniform machines [Conway *et al.* 1967]. If J_j is the k-th last job executed on M_i, a cost contribution $kp_{ij} = kp_j/q_i$ is incurred. ΣC_j is a weighted sum of the p_j and is minimized by matching the n smallest weights k/q_i in nondecreasing order with the p_j in nonincreasing order. The procedure can be implemented to run in O(n log n) time [Horowitz & Sahni 1976].

4.3.3. $R||\Sigma C_j$

$R||\Sigma C_j$ can be formulated and solved as an m×n transportation problem [Horn 1973; Bruno *et al.* 1974]. Let

$$x_{ijk} = \begin{cases} 1 & \text{if } J_j \text{ is the k-th last job executed on } M_i, \\ 0 & \text{otherwise.} \end{cases}$$

Then the problem is to minimize

$$\Sigma_{i=1}^{m} \Sigma_{j=1}^{n} \Sigma_{k=1}^{n} kp_{ij}x_{ijk}$$

subject to

$$\Sigma_{i=1}^{m} \Sigma_{k=1}^{n} x_{ijk} = 1 \quad \text{for all } j,$$

$$\Sigma_{j=1}^{n} x_{ijk} \leq 1 \quad \text{for all } i,k,$$

$$x_{ijk} \geq 0 \quad \text{for all } i,j,k.$$

This problem, like the similar one in Section 4.2.1, can be solved in $O(n^3)$ time.

4.3.4. *Other cases: enumerative optimization methods*

As we noted in Section 4.1, $P2||C_{max}$ and $P2||\Sigma w_jC_j$ are NP-hard. Hence it seems fruitless to attempt to find polynomial-time optimization algorithms for criteria other than ΣC_j. Moreover, $P2|tree|\Sigma C_j$ is known to be NP-hard, both for intrees and outtrees [Sethi 1977]. It follows that it is also not possible to extend the above algorithms to problems with precedence constraints. The only remaining possibility for optimization methods seems to be implicit enumeration.

$R||C_{max}$ can be solved by a branch-and-bound procedure described in [Stern 1976]. The enumerative approach for identical machines in [Bratley *et al.* 1975] allows inclusion of release dates and deadlines as well.

A general dynamic programming technique [Rothkopf 1966; Lawler & Moore 1969] is applicable to parallel machine problems with the C_{max}, L_{max}, Σw_jC_j and Σw_jU_j optimality criteria, and even to

problems with the $\Sigma w_j T_j$ criterion in the special case of a common due date.

Let us define $F_j(t_1,\ldots,t_m)$ as the minimum cost of a schedule without idle time for $J_1,\ldots,J_j$ subject to the constraint that the last job on M_i is completed at time t_i, for $i = 1,\ldots,m$. Then, in the case of f_{max} criteria,

$$F_j(t_1,..,t_m) = \min_{1\le i\le m}\{\max\{f_j(t_i),F_{j-1}(t_1,..,t_i-p_{ij},..,t_m)\}\},$$

and in the case of Σf_j criteria,

$$F_j(t_1,..,t_m) = \min_{1\le i\le m}\{f_j(t_i)+F_{j-1}(t_1,..,t_i-p_{ij},..,t_m)\}.$$

In both cases, the initial conditions are

$$F_0(t_1,..,t_m) = \begin{cases} 0 & \text{if } t_i = 0 \text{ for } i = 1,..,m, \\ \infty & \text{otherwise.} \end{cases}$$

Appropriate implementation of these equations yields $O(mnC^{m-1})$ computations for a variety of problems, where C is an upper bound on the completion time of any job in an optimal schedule. Among these problems are $P|r_j|C_{max}$, $Q||L_{max}$ and $Q||\Sigma w_j C_j$. $P||\Sigma w_j U_j$ can be solved in $O(mn(\max_j\{d_j\})^m)$ time.

Still other dynamic programming approaches can be used to solve $P||\Sigma f_j$ and $P||f_{max}$ in $O(m\cdot\min\{3^n,n2^nC\})$ time.

4.3.5. *Other cases: approximation algorithms*

4.3.5.1. $P||C_{max}$

By far the most studied scheduling model from the viewpoint of approximation algorithms is $P||C_{max}$. We refer to [Garey *et al.* 1978] for an easily readable introduction into the techniques involved in many of the "performance guarantees" mentioned below.

Perhaps the earliest and simplest result on the worst-case performance of list scheduling is given in [Graham 1966]:

$$C_{max}(LS)/C^*_{max} \le 2 - \frac{1}{m}. \qquad (\dagger)$$

If the jobs are selected in LPT order, then the bound can be considerably improved, as is shown in [Graham 1969]:

$$C_{max}(LPT)/C^*_{max} \le \frac{4}{3} - \frac{1}{3m}. \qquad (\dagger)$$

A somewhat better algorithm, called *multifit* (MF) and based on a completely different principle, is given in [Coffman *et al.* 1978]. The idea behind MF is to find (by binary search) the smallest "capacity" a set of m "bins" can have and still accommodate all jobs when the jobs are taken in order of nonincreasing p_j and each job is placed into the first bin into which it will fit. The set

of jobs in the i-th bin will be processed by M_i. If k packing attempts are made, the algorithm (denoted by MF_k) runs in time $O(n \log n + knm)$ and satisfies

$$C_{max}(MF_k)/C^*_{max} \leq 1.22 + 2^{-k}.$$

We note that if the jobs are not ordered by decreasing p_j then all that can be guaranteed by this method is

$$C_{max}(MF)/C^*_{max} \leq 2 - \frac{2}{m+1}. \qquad (\dagger)$$

The following algorithm Z_k was introduced in [Graham 1969]: schedule the k largest jobs optimally, then list schedule the remaining jobs arbitrarily. It is shown in [Graham 1969] that

$$C_{max}(Z_k)/C^*_{max} \leq 1 + (1 - \frac{1}{m})/(1 + \left[\frac{k}{m}\right])$$

and that when m divides k, this is best possible. Thus, we can make the bound as close to 1 as desired by taking k sufficiently large. Unfortunately, the best bound on the running time is $O(n^{km})$.

A very interesting algorithm for $P||C_{max}$ is given by Sahni [Sahni 1976]. He presents algorithms A_k with $O(n(n^2k)^{m-1})$ running time which satisfy

$$C_{max}(A_k)/C^*_{max} \leq 1 + \frac{1}{k}.$$

For $m = 2$, algorithm A_2 can be improved to run in time $O(n^2k)$. As in the cases of $1||\Sigma w_jU_j$ (Section 3.3.3) and $P||\Sigma w_jC_j$ (Section 4.3.1), the algorithms A_k are based on a clever combination of dynamic programming and rounding and are beyond the scope of the present discussion.

Several bounds are available which take into account the processing times of the jobs. In [Graham 1969] it is shown that

$$C_{max}(LS)/C^*_{max} \leq 1 + (m-1)\max_j\{p_j\}/\Sigma_j p_j.$$

For the case of LPT, Ibarra and Kim [Ibarra & Kim 1977] prove that

$$C_{max}(LPT)/C^*_{max} \leq 1 + \frac{2(m-1)}{n} \quad \text{for } n \geq 2(m-1)\max_j\{p_j\}/\min_j\{p_j\}.$$

4.3.5.2. $Q||C_{max}$

In the literature on approximation algorithms for scheduling problems, it is usually assumed that *unforced idleness* (UI) of machines is *not* allowed, *i.e.*, a machine cannot be idle when jobs are available. In the case of identical machines, UI need not occur in an optimal schedule if there are no precedence constraints or if all $p_j = 1$. Allowing UI may yield better solutions, however, in the cases which are to be discussed in Sections 4.3.5.2-5. The optimal value of C_{max} under the restriction of *no* UI will be denoted by

C^*_{max}, the optimum if UI *is* allowed by $C^*_{max}(UI)$.

Liu and Liu [Liu & Liu 1974A, 1974B, 1974C] study numerous questions dealing with uniform machines. They define the algorithm A_k as follows: schedule the k longest jobs first, resulting in a completion time of $C_k(A_k)$, and schedule the remaining tasks for a total completion time of $C_{max}(A_k)$. If $C_{max}(A_k) > C_k(A_k)$, then

$$C_{max}(A_k)/C^*_{max}(UI) \leq 1 + \frac{1}{Q} - \frac{1}{Q\Sigma_i q_i}$$

where all $q_i \geq 1$ and

$$Q = \max\{\min_j\{\left\lceil \frac{k+1}{\Sigma_i \lceil q_i \rceil} \right\rceil \cdot \frac{\lceil q_j \rceil}{q_j} - \frac{1}{\lceil q_j \rceil q_j}\}, \frac{k+1}{\Sigma_i q_i}\}.$$

This is best possible when the q_i are integers and $\Sigma_i q_i$ divides k.

Gonzalez, Ibarra and Sahni [Gonzalez *et al.* 1977] consider the following generalization LPT' of LPT: assign each job, in order of nonincreasing processing time, to the machine on which it will be completed soonest. Thus, unforced idleness may occur in the schedule. They show

$$C_{max}(LPT')/C^*_{max} \leq 2 - \frac{2}{m+1}.$$

Also, examples are given for which $C_{max}(LPT')/C^*_{max}$ approaches 3/2 as m tends to infinity.

4.3.5.3. $R||C_{max}$

Very little is known about approximation algorithms for this model. Ibarra and Kim [Ibarra & Kim 1977] consider five algorithms, typical of which is to schedule J_j on the machine that executes it fastest, *i.e.*, on an M_i with minimum p_{ij}. For all five algorithms A they prove

$$C_{max}(A)/C^*_{max} \leq m$$

with equality possible for four of the five. For the special case $R2||C_{max}$, they give an O(n log n) algorithm G such that

$$C_{max}(G)/C^*_{max} \leq \frac{1+\sqrt{5}}{2}. \qquad (\dagger)$$

Potts [Potts -] proposes an $R||C_{max}$ algorithm based on linear programming (LP), the running time of which is polynomial only for fixed m. He proves

$$C_{max}(LP)/C^*_{max} \leq 2. \qquad (\dagger)$$

4.3.5.4. $P|prec|C_{max}$

In the presence of precedence constraints it is somewhat unexpected [Graham 1966] that the 2-(1/m) bound still holds, *i.e.*,

$$C_{max}(LS)/C^*_{max} \leq 2 - \frac{1}{m}.$$

Now, consider executing the set of jobs *twice*: the first time using processing times p_j, precedence constraints, m machines and an arbitrary priority list, the second time using processing times $p'_j \leq p_j$, weakened precedence constraints, m' machines and a (possibly different) priority list. Then [Graham 1966]

$$C'_{max}(LS)/C_{max}(LS) \leq 1 + \frac{m-1}{m'}. \qquad (\dagger)$$

Even when critical path (CP) scheduling is used, examples exist [Graham -] for which

$$C_{max}(CP)/C^*_{max} = 2 - \frac{1}{m}.$$

It is known [Graham -] that unforced idleness (UI) has the following behavior:

$$C_{max}(LS)/C^*_{max}(UI) \leq 2 - \frac{1}{m}. \qquad (\dagger)$$

Let $C^*_{max}(pmtn)$ denote the optimal value of C_{max} if preemption is allowed. As in the case of UI, it is known [Graham -] that

$$C_{max}(LS)/C^*_{max}(pmtn) \leq 2 - \frac{1}{m}. \qquad (\dagger)$$

Liu [Liu 1972] shows that

$$C^*_{max}(UI)/C^*_{max}(pmtn) \leq 2 - \frac{2}{m+1}. \qquad (\dagger)$$

4.3.5.5. $Q|prec|C_{max}$

Liu and Liu [Liu & Liu 1974B] also consider the presence of precedence constraints in the case of uniform machines. They show that, when unforced idleness or preemption is allowed,

$$C_{max}(LS)/C^*_{max}(UI) \leq 1+\max_i\{q_i\}/\min_i\{q_i\}-\max_i\{q_i\}/\Sigma_i q_i, \qquad (\dagger)$$

$$C_{max}(LS)/C^*_{max}(pmtn) \leq 1+\max_i\{q_i\}/\min_i\{q_i\}-\max_i\{q_i\}/\Sigma_i q_i. \qquad (\dagger)$$

When all $q_i = 1$ this reduces to the earlier 2-(1/m) bounds for these questions on identical machines.

Suppose that the jobs are executed twice: the first time using m machines of speeds $q_1,\ldots,q_m$, the second time using m' machines of speeds $q'_1,\ldots,q'_{m'}$. Then

$$C'_{max}(LS)/C^*_{max}(UI) \leq \max_i\{q_i\}/\min_i\{q'_i\}+(\Sigma_i q_i-\max_i\{q_i\})/\Sigma_i q'_i. \qquad (\dagger)$$

Jaffe [Jaffe 1979] develops an algorithm LSi, that uses list scheduling on the fastest i machines for an appropriately chosen value of i. It is shown that

$$C_{max}(LSi)/C^*_{max}(UI) \leq \sqrt{m} + O(m^{1/4})$$

and examples are given for which the bound $\sqrt{m-1}$ is approached arbitrarily closely.

4.4. Preemptive scheduling

4.4.1. $P|pmtn|\Sigma C_j$

A theorem of McNaughton [McNaughton 1959] states that for $P|pmtn|\Sigma w_jC_j$ there is no schedule with a finite number of preemptions which yields a smaller criterion value than an optimal non-preemptive schedule. The finiteness restriction can be removed by appropriate application of results from open shop theory. It therefore follows that the procedure of Section 4.3.1 can be applied to solve $P|pmtn|\Sigma C_j$. It also follows that $P2|pmtn|\Sigma w_jC_j$ is NP-hard, since $P2||\Sigma w_jC_j$ is known to be NP-hard.

4.4.2. $Q|pmtn|\Sigma C_j$

McNaughton's theorem does not apply to uniform machines, as can be demonstrated by a simple counterexample. There is, however, a polynomial algorithm for $Q|pmtn|\Sigma C_j$.

One can show that there exists an optimal preemptive schedule in which $C_j \leq C_k$ if $p_j < p_k$ [Lawler & Labetoulle 1978]. Accordingly, first place the jobs in SPT order. Then obtain an optimal schedule by preemptively scheduling each successive job in the available time on the m machines so as to minimize its completion time [Gonzalez 1977]. This procedure can be implemented in $O(n \log n + mn)$ time and yields an optimal schedule with no more than $(m-1)(n-\frac{1}{2}m)$ preemptions. It has been extended to cover the case in which ΣC_j is minimized subject to a common deadline for all jobs [Gonzalez 1977].

4.4.3. $R|pmtn|\Sigma C_j$

Very little is known about $R|pmtn|\Sigma C_j$. This remains one of the more vexing questions in the area of preemptive scheduling.

4.4.4. $P|pmtn,prec|C_{max}$

An obvious lower bound on the value of an optimal $P|pmtn|C_{max}$ schedule is given by

$$\max\{\max_j\{p_j\}, \frac{1}{m}\Sigma_j p_j\}.$$

A schedule meeting this bound can be constructed in $O(n)$ time [McNaughton 1959]: just fill the machines successively, scheduling the jobs in any order and splitting a job whenever the above time bound is met. The number of preemptions occurring in this schedule

is at most m-1. It is possible to design a class of problems for which this number is minimal, but the general problem of minimizing the number of preemptions is easily seen to be NP-hard.

In the case of precedence constraints, $P|pmtn,prec,p_j=1|C_{max}$ turns out to be NP-hard [Ullman 1976], but $P|pmtn,tree|C_{max}$ and $P2|pmtn,prec|C_{max}$ can be solved by a polynomial-time algorithm due to Muntz and Coffman [Muntz & Coffman 1969, 1970]. This is as follows.

Define $\ell_j(t)$ to be the level of a J_j wholly or partly unexecuted at time t. Suppose that at time t m' machines are available and that n' jobs are currently maximizing $\ell_j(t)$. If $m' < n'$, we assign m'/n' machines to each of the n' jobs, which implies that each of these jobs will be executed at speed m'/n'. If $m' \geq n'$, we assign one machine to each job, consider the jobs at the next highest level, and repeat. The machines are reassigned whenever a job is completed or threatens to be processed at a higher speed than another one at a currently higher level. Between each pair of successive reassignment points, jobs are finally rescheduled by means of McNaughton's algorithm for $P|pmtn|C_{max}$. The algorithm requires $O(n^2)$ time [Gonzalez & Johnson 1980].

Gonzalez and Johnson [Gonzalez & Johnson 1980] have developed a totally different algorithm that solves $P|pmtn,tree|C_{max}$ by starting at the roots rather than the leaves of the tree and determines priority by considering the total remaining processing time in subtrees rather than by looking at critical paths. The algorithm runs in O(n log m) time and introduces at most n-2 preemptions into the resulting optimal schedule.

Lam and Sethi [Lam & Sethi 1977], much in the same spirit as their work mentioned in Section 4.2.2, analyze the performance of the Muntz-Coffman (MC) algorithm for $P|pmtn,prec|C_{max}$. They show

$$C_{max}(MC)/C^*_{max} \leq 2 - \frac{2}{m} \quad (m \geq 2). \qquad (\dagger)$$

4.4.5. $Q|pmtn,prec|C_{max}$

Horvath, Lam and Sethi [Horvath *et al.* 1977] adapt the Muntz-Coffman algorithm to solve $Q|pmtn|C_{max}$ and $Q2|pmtn,prec|C_{max}$ in $O(mn^2)$ time. This results is an optimal schedule with no more than $(m-1)n^2$ preemptions.

A computationally more efficient algorithm due to Gonzalez and Sahni [Gonzalez & Sahni 1978B] solves $Q|pmtn|C_{max}$ in O(n) time, if the jobs are given in order of nonincreasing p_j and the machines in order of nonincreasing q_i. This procedure yields an optimal schedule with no more than 2(m-1) preemptions, which can be shown to be a tight bound.

The optimal value of C_{max} is given by

$$\max\{\max_{1\leq k\leq m-1}\{\Sigma_{j=1}^{k} p_j/\Sigma_{i=1}^{k} q_i\}, \Sigma_{j=1}^{n} p_j/\Sigma_{i=1}^{m} q_i\},$$

where $p_1 \geq \ldots \geq p_n$ and $q_1 \geq \ldots \geq q_m$. This result generalizes the one given in Section 4.4.4.

The Gonzalez-Johnson algorithm for $P|pmtn,tree|C_{max}$ mentioned in the previous section can be adapted to the case $Q2|pmtn,tree|C_{max}$.

Jaffe [Jaffe 1980] studies the performance of *maximal usage schedules* (MUS) for $Q|pmtn,prec|C_{max}$, *i.e.*, schedules without unforced idleness in which at any time the jobs being processed are assigned to the fastest machines. It is shown that

$$C_{max}(MUS)/C^*_{max} \leq \sqrt{m} + \frac{1}{2}$$

and examples are given for which the bound $\sqrt{m-1}$ is approached arbitrarily closely.

4.4.6. $R|pmtn|C_{max}$

Many preemptive scheduling problems involving independent jobs on unrelated machines can be formulated as linear programming problems [Lawler & Labetoulle 1978]. For instance, solving $R|pmtn|C_{max}$ is equivalent to minimizing

$$C_{max}$$

subject to

$$\Sigma_{i=1}^{m} x_{ij}/p_{ij} = 1 \quad (j = 1,\ldots,n),$$

$$\Sigma_{i=1}^{m} x_{ij} \leq C_{max} \quad (j = 1,\ldots,n),$$

$$\Sigma_{j=1}^{n} x_{ij} \leq C_{max} \quad (i = 1,\ldots,m),$$

$$x_{ij} \geq 0 \quad (i = 1,\ldots,m,\ j = 1,\ldots,n).$$

In this formulation x_{ij} represents the total time spent by J_j on M_i. The linear program can be solved in polynomial time [Khachiyan 1979], and a feasible schedule can be constructed in polynomial time by applying the algorithm for $O|pmtn|C_{max}$, discussed in Section 5.2.2.

This procedure can be modified to yield an optimal schedule with no more than about $7m^2/2$ preemptions. It remains an open question as to whether $O(m^2)$ preemptions are necessary for an optimal preemptive schedule.

For fixed m, it seems to be possible to solve the linear program in linear time. Certainly, the special case $R2|pmtn|C_{max}$ can be solved in O(n) time [Gonzalez *et al.* 1981].

We note that a similar linear programming formulation can be given for $R|pmtn,r_j|L_{max}$ [Lawler & Labetoulle 1978].

4.4.7. $P|pmtn,prec,r_j|L_{max}$

$P|pmtn|L_{max}$ and $P|pmtn,r_j|C_{max}$ can be solved by a procedure due to Horn [Horn 1974]. The $O(n^2)$ running time has been reduced to $O(mn)$ [Gonzalez & Johnson 1980].

More generally, the existence of a feasible preemptive schedule with given release dates and deadlines can be tested by means of a network flow model in $O(n^3)$ time [Horn 1974]. A binary search can then be conducted on the optimal value of L_{max}, with each trial value of L_{max} inducing deadlines which are checked for feasibility by means of the network computation. It can be shown that this yields an $O(n^3 \min\{n^2, \log n + \log \max_j\{p_j\}\})$ algorithm [Labetoulle *et al.* 1979].

In the case of precedence constraints, the algorithms of Brucker, Garey and Johnson for $P|intree,p_j=1|L_{max}$, $P2|prec,p_j=1|L_{max}$ and $P2|prec,r_j,p_j=1|L_{max}$ (see Section 4.2.2) have preemptive counterparts. *E.g.*, $P|pmtn,intree|L_{max}$ can be solved in $O(n^2)$ time [Lawler 1980]; see also the next section.

4.4.8. $Q|pmtn,prec,r_j|L_{max}$

In the case of uniform machines, the existence of a feasible preemptive schedule with given release dates and a common deadline can be tested in $O(n \log n + mn)$ time; the algorithm generates $O(mn)$ preemptions in the worst case [Sahni & Cho 1980]. More generally, $Q|pmtn,r_j|C_{max}$ and, by symmetry, $Q|pmtn|L_{max}$ are solvable in $O(n \log n + mn)$ time; the number of preemptions generated is $O(mn)$ [Sahni & Cho 1979B; Labetoulle *et al.* 1979].

The first feasibility test mentioned in the previous section has been adapted to the case of two uniform machines [Bruno & Gonzalez 1976] and extended to a polynomial-time algorithm for $Q2|pmtn,r_j|L_{max}$ [Labetoulle *et al.* 1979].

Most recently, Martel has found a polynomial-time algorithm for $Q|pmtn,r_j|L_{max}$ [Martel 1981]. This method is in fact a special case of a more general algorithm for computing maximal *polymatroidal* network flows [Lawler & Martel 1980].

In the case of precedence constraints, $Q2|pmtn,prec|L_{max}$ and $Q2|pmtn,prec,r_j|L_{max}$ can be solved in $O(n^2)$ and $O(n^6)$ time, respectively [Lawler 1980].

4.4.9. $Q|pmtn|\Sigma w_jU_j$

Binary NP-hardness has been established for $1|pmtn|\Sigma w_jU_j$ (see Section 3.3.3) and $P|pmtn|\Sigma U_j$ [Lawler 1981]. For any *fixed* number of uniform machines, $Qm|pmtn|\Sigma w_jU_j$ can be solved in pseudopolynomial time: $O(n^2(\Sigma w_j)^2)$ if $m = 2$ and $O(n^{3m-5}(\Sigma w_j)^2)$ if $m \geq 3$ [Lawler 1981]. Hence, $Qm|pmtn|\Sigma U_j$ is solvable in strictly polynomial time.

5. OPEN SHOP, FLOW SHOP AND JOB SHOP PROBLEMS

5.1. Introduction

We now pass on to problems in which each job requires execution on more than one machine. Recall from Section 2.3 that in an *open shop* (denoted by O) the order in which a job passes through the machines is immaterial, whereas in a *flow shop* (F) each job has the same machine ordering $(M_1,\ldots,M_m)$ and in a *job shop* (J) possibly different machine orderings are specified for the jobs. We survey these problem classes in Sections 5.2, 5.3 and 5.4, respectively.

We shall be dealing exclusively with the C_{max} criterion. Other optimality criteria lead usually to NP-hard problems, such as:

- $O2||L_{max}$ [Lawler *et al.* 1981],
- $O||\Sigma C_j$, $O|pmtn|\Sigma C_j$ [Gonzalez 1979B],
- $F2||L_{max}$ [Lenstra *et al.* 1977], $F2|pmtn|L_{max}$ [Cho & Sahni 1978],
- $F2||\Sigma C_j$ [Garey *et al.* 1976], $F3|pmtn|\Sigma C_j$, $J2|pmtn|\Sigma C_j$ [Lenstra -].

Notable exceptions are $O|pmtn,r_j|L_{max}$, which is solvable in polynomial time by linear programming [Cho & Sahni 1978], and $O2||\Sigma C_j$ and $F2|pmtn|\Sigma C_j$, which are open.

5.2. Open shop scheduling

5.2.1. *Nonpreemptive case*

The case $O2||C_{max}$ admits of an O(n) algorithm [Gonzalez & Sahni 1976]. A simplified exposition is given below.

For convenience, let $a_j = p_{1j}$, $b_j = p_{2j}$. Let $A = \{J_j|a_j \geq b_j\}$, $B = \{J_j|a_j < b_j\}$. Now choose J_r and J_ℓ to be any two distinct jobs (whether in A or B) such that

$$a_r \geq \max_{J_j\in A}\{b_j\}, \quad b_\ell \geq \max_{J_j\in B}\{a_j\}.$$

Let $A' = A-\{J_r,J_\ell\}$, $B' = B-\{J_r,J_\ell\}$. We assert that it is possible to form feasible schedules for $B'\cup\{J_\ell\}$ and for $A'\cup\{J_r\}$ as indicated in Figure 2(a), the jobs in A' and B' being ordered arbitrarily. In each of these separate schedules, there is no idle time on either machine, from the start of the first job on that machine to the completion of the last job on that machine.

Let $T_1 = \Sigma_j a_j$, $T_2 = \Sigma_j b_j$. Suppose $T_1-a_\ell \geq T_2-b_r$ (the case $T_1-a_\ell < T_2-b_r$ being symmetric). We then combine the two schedules as shown in Figure 2(b), pushing the jobs in $B'\cup\{J_\ell\}$ on M_2 to the right. Again, there is no idle time on either machine, from the start of the first job to the completion of the last job.

We finally propose to move the processing of J_r on M_2 to the first position on that machine. There are two cases to consider.

(1) $a_r \leq T_2 - b_r$. The resulting schedule is as in Figure 2(c). The length of the schedule is $\max\{T_1, T_2\}$.
(2) $a_r > T_2 - b_r$. The resulting schedule is as in Figure 2(d). The length of the schedule is $\max\{T_1, a_r + b_r\}$.

For any feasible schedule we obviously have that

$$C_{max} \geq \max\{T_1, T_2, \max_j\{a_j + b_j\}\}.$$

Since, in all cases, we have met this lower bound, it follows that the schedules constructed are optimal.

There is little hope of finding polynomial-time algorithms for nonpreemptive open shop problems more complicated than $O2||C_{max}$. $O3||C_{max}$ is binary NP-hard [Gonzalez & Sahni 1976] and $O2|r_j|C_{max}$, $O2|tree|C_{max}$ and $O||C_{max}$ are unary NP-hard [Lawler *et al.* 1981; Lenstra -].

The special case of $O3||C_{max}$ is which $\max_j\{p_{hj}\} \leq \min_j\{p_{ij}\}$ for some pair (M_h, M_i) $(h \neq i)$ is likely to be solvable in polynomial time [Adiri & Hefetz 1980].

5.2.2. *Preemptive case*

The result on $O2||C_{max}$ presented in the previous section shows that there is no advantage to preemption for $m = 2$, and hence $O2|pmtn|C_{max}$ can be solved in $O(n)$ time. More generally, $O|pmtn|C_{max}$ is solvable in polynomial time as well [Gonzalez & Sahni 1976; Lawler & Labetoulle 1978; Gonzalez 1979A]. We had already occasion to refer to this result in Section 4.4.6.

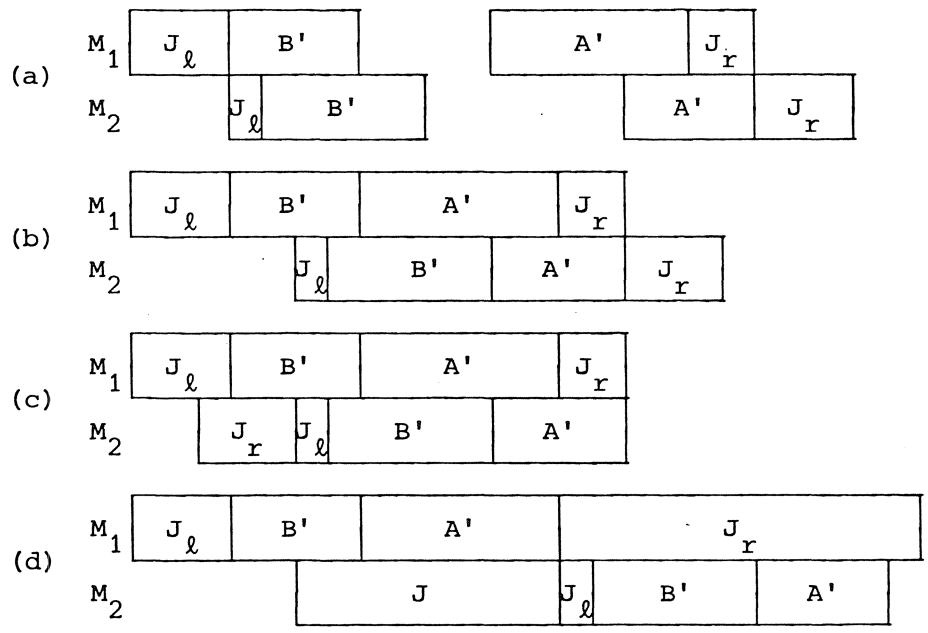


Figure 2

If release dates are introduced, $O2|pmtn,r_j|C_{max}$ is still solvable in O(n) time [Lawler *et al.* 1981]. As mentioned in Section 5.1, even $O|pmtn,r_j|L_{max}$ is well solved [Cho & Sahni 1978].

5.3. Flow shop scheduling

5.3.1. $F2|\beta|C_{max}$, $F3|\beta|C_{max}$

A fundamental algorithm for solving $F2||C_{max}$ is due to Johnson [Johnson 1954]. He shows that there exists an optimal schedule in which J_j precedes J_k if $\min\{p_{1j},p_{2k}\} \leq \min\{p_{2j},p_{1k}\}$. It follows that the problem can be solved in O(n log n) time: arrange first the jobs with $p_{1j} \leq p_{2j}$ in order of nondecreasing p_{1j} and subsequently the remaining jobs in order of nonincreasing p_{2j}.

Some special cases involve *start lags* ℓ_{1j} and *stop lags* ℓ_{2j} for J_j, that represent minimum time intervals between starting times on M_1 and M_2 and between completion times on M_1 and M_2, respectively [Mitten 1958; Johnson 1958; Nabeshima 1963; Szwarc 1968]. Defining $\ell_j = \min\{\ell_{1j}-p_{1j},\ell_{2j}-p_{2j}\}$ and applying Johnson's algorithm to processing times $(p_{1j}+\ell_j,p_{2j}+\ell_j)$ will produce an optimal *permutation schedule*, *i.e.*, one with identical processing orders on all machines [Rinnooy Kan 1976]. If we drop the latter restriction, the problem is unary NP-hard [Lenstra -].

A fair amount of effort has been devoted to identifying special flow shop problems that can still be solved in polynomial time. The crucial notion here is that of a *nonbottleneck machine*, that can effectively be treated as though it can process any number of jobs at the same time. For example, $F3||C_{max}$ can be solved by applying Johnson's algorithm to processing times $(p_{1j}+p_{2j},p_{2j}+p_{3j})$ if $\max_j\{p_{2j}\} \leq \max\{\min_j\{p_{1j}\},\min_j\{p_{3j}\}\}$ [Johnson 1954]. Many other special cases appearing in the literature, including some of the work on *ordered flow shops* [Smith *et al.* 1975, 1976], can be discussed in this framework [Monma & Rinnooy Kan 1981; Achuthan 1980].

The general $F3||C_{max}$ problem, however, is unary NP-hard, and the same applies to $F2|r_j|C_{max}$ and $F2|tree|C_{max}$ [Garey *et al.* 1976; Lenstra *et al.* 1977].

It should be noted that an interpretation of precedence constraints which differs from our definition is possible. If $J_j \rightarrow' J_k$ only means that O_{ij} should precede O_{ik} for i = 1,2, then $F2|tree'|C_{max}$ can be solved in O(n log n) time [Sidney 1979]. In fact, Sidney's algorithm applies even to series-parallel precedence constraints. The arguments used to establish this result are very similar to those referred to in Section 3.3.1 and apply to a larger class of scheduling problems [Monma & Sidney 1979]. The general case $F2|prec'|C_{max}$ is unary NP-hard [Monma 1980].

Gonzalez, Cho and Sahni [Gonzalez & Sahni 1978A; Cho & Sahni 1978] consider the case of preemptive flow shop scheduling. Since preemptions on M_1 and M_m can be removed without increasing C_{max}, Johnson's algorithm solves $F2|pmtn|C_{max}$ as well. $F3|pmtn|C_{max}$ and $F2|pmtn,r_j|C_{max}$ turn out to be unary NP-hard.

5.3.2. $F||C_{max}$

As a general result, we note that there exists an optimal flow shop schedule with the same processing order on M_1 and M_2 and the same processing order on M_{m-1} and M_m [Conway *et al.* 1967]. It is, however, not difficult to construct a 4-machine example in which a job "passes" another one between M_2 and M_3 in the optimal schedule. Nevertheless, it has become tradition in the literature to assume identical processing orders on all machines, so that in effect only the best permutation schedule has to be determined.

Most research in this area has focused on enumerative methods. The usual enumeration scheme is to assign jobs to the ℓ-th position in the schedule at the ℓ-th level of the search tree. Thus, at a node at that level a partial schedule $(J_{\sigma(1)},\ldots,J_{\sigma(\ell)})$ has been formed and the jobs with index set $S = \{1,\ldots,n\} - \{\sigma(1),\ldots,\sigma(\ell)\}$ are candidates for the $(\ell+1)$-st position. One then needs to find a lower bound on the value of all possible completions of the partial schedule. It turns out that almost all lower bounds developed so far are generated by the following bounding scheme [Lageweg *et al.* 1978].

Let us relax the capacity constraint that each machine can process at most one job at a time, for all machines but at most two, say, M_u and M_v $(1 \leq u \leq v \leq m)$. We then obtain a problem of scheduling $\{J_j | j \in S\}$ on five machines $N_{*u},M_u,N_{uv},M_v,N_{v*}$ in that order, which is specified as follows. Let $C(\sigma,i)$ denote the completion time of $J_{\sigma(\ell)}$ on M_i. N_{*u}, N_{uv} and N_{v*} have infinite capacity; the processing times on these machines are defined by

$$q_{*uj} = \max_{1\leq i\leq u}\{C(\sigma,i) + \sum_{h=i}^{u-1} p_{hj}\},$$

$$q_{uvj} = \sum_{h=u+1}^{v-1} p_{hj},$$

$$q_{v*j} = \sum_{h=v+1}^{m} p_{hj}.$$

M_u and M_v have capacity 1 and processing times p_{uj} and p_{vj}, respectively. Note that we can interpret N_{*u} as yielding release dates q_{*uj} on M_u and N_{v*} as setting due dates $-q_{v*j}$ on M_v, with respect to which L_{max} is to be minimized.

Any of the machines N_{*u},N_{uv},N_{v*} can be removed from this problem by underestimating its contribution to the lower bound to be the minimum processing time on that machine. Valid lower bounds are obtained by adding these contributions to the optimal solution value of the remaining problem.

For the case that $u = v$, removing N_{*u} and N_{u*} from the problem produces the *machine-based bound* used in [Ignall & Schrage 1965; McMahon 1971]:

$$\max_{1\leq u\leq m}\{\min_{j\in S}\{q_{*uj}\} + \sum_{j\in S} p_{uj} + \min_{j\in S}\{q_{u*j}\}\}.$$

Removing only N_{*u} results in a $1||L_{max}$ problem on M_u, which can be

solved by Jackson's rule (Section 3.2) and provides a slightly stronger bound.

If $u \neq v$, removal of N_{*u}, N_{uv} and N_{v*} yields an $F2||C_{max}$ problem, to be solved by Johnson's algorithm (Section 5.3.1). As pointed out in that section, solution in polynomial time remains possible if N_{uv} is taken fully into account; the resulting bound dominates the *job-based bound* proposed in [McMahon 1971] and is the best one currently available.

All other variations on this theme (*e.g.*, taking $u = v$ and considering the resulting $1|r_j|L_{max}$ problem) would involve the solution of NP-hard problems. The development of fast algorithms or strong lower bounds for these problems thus emerges as a possibly fruitful research area.

An alternative and somewhat more efficient enumeration scheme [Potts 1980A] builds up a schedule from the front and from the back at the same time. The adaptation of the above bounding scheme to this approach is straightforward.

The computational performance of branch-and-bound algorithms for $F||C_{max}$ might be improved by the use of *elimination criteria*. Particular attention has been paid to conditions under which all completions of $(J_{\sigma(1)},\ldots,J_{\sigma(\ell)},J_j)$ can be eliminated because a schedule at least as good exists among the completions of $(J_{\sigma(1)},\ldots,J_{\sigma(\ell)},J_k,J_j)$. If all information obtainable from the processing times of the other jobs is disregarded, the strongest condition under which this is allowed is as follows. Defining $\Delta_i = C(\sigma kj,i)-C(\sigma j,i)$, we can exclude J_j for the $(\ell+1)$-st position if

$$\max\{\Delta_{i-1},\Delta_i\} \leq p_{ij} \quad (i = 2,\ldots,m)$$

[McMahon 1969; Szwarc 1971, 1973]. Inclusion of these and similar dominance rules can be very helpful from a computational point of view, depending on the lower bound used [Lageweg *et al.* 1978]. It may be worthwhile to consider further extensions that, for instance, involve the processing times of the unscheduled jobs [Gupta & Reddi 1978; Szwarc 1978].

Not much has been done in the way of worst-case analysis of approximation algorithms for $F||C_{max}$. It is not hard to see that for any active schedule (AS)

$$C_{max}(AS)/C^*_{max} \leq \max_{i,j}\{p_{ij}\}/\min_{i,j}\{p_{ij}\}. \qquad (\dagger)$$

Gonzalez and Sahni [Gonzalez & Sahni 1978A] show that

$$C_{max}(AS)/C^*_{max} \leq m. \qquad (\dagger)$$

This bound is tight even for LPT schedules, in which the jobs are ordered according to nonincreasing sums of processing times. They also give an $O(mn \log n)$ algorithm H based on Johnson's algorithm with

$$C_{max}(H)/C^*_{max} \leq \lceil \frac{m}{2} \rceil .$$

It thus appears that, in general, the obvious algorithms can deviate quite substantially from the optimum.

5.3.3. *No wait in process*

In a variation on the flow shop problem, each job, once started, has to be processed without interruption until it is completed. This *no wait* constraint may arise out of certain job characteristics (*e.g.*, the "hot ingot" problem in which metal has to be processed at continuously high temperature) or out of the unavailability of intermediate storage in between machines.

The resulting $F|no\ wait|C_{max}$ problem can be formulated as a *traveling salesman* problem with cities $0,1,\ldots,n$ and intercity distances

$$c_{jk} = \max_{1\leq i\leq m}\{\Sigma_{h=1}^{i} p_{hj} - \Sigma_{h=1}^{i-1} p_{hk}\} \quad (j,k = 0,1,\ldots,n),$$

where $p_{i0} = 0$ $(i = 1,\ldots,m)$ [Piehler 1960; Reddi & Ramamoorthy 1972; Wismer 1972].

For the case $F2|no\ wait|C_{max}$, the traveling salesman problem assumes a special structure and the results from [Gilmore & Gomory 1964] can be applied to yield an $O(n^2)$ algorithm [Reddi & Ramamoorthy 1972]. $F4|no\ wait|C_{max}$ is unary NP-hard [Papadimitriou & Kanellanis 1980], and the same is true for $O2|no\ wait|C_{max}$ and $J2|no\ wait|C_{max}$ [Sahni & Cho 1979A]. In spite of a challenging prize awarded for its solution [Lenstra *et al.* 1977], $F3|no\ wait|C_{max}$ is still open.

The *no wait* constraint may lengthen the optimal flow shop schedule considerably. It can be shown [Lenstra -] that

$$C^*_{max}(no\ wait)/C^*_{max} < m \quad \text{for } m \geq 2. \qquad (\dagger)$$

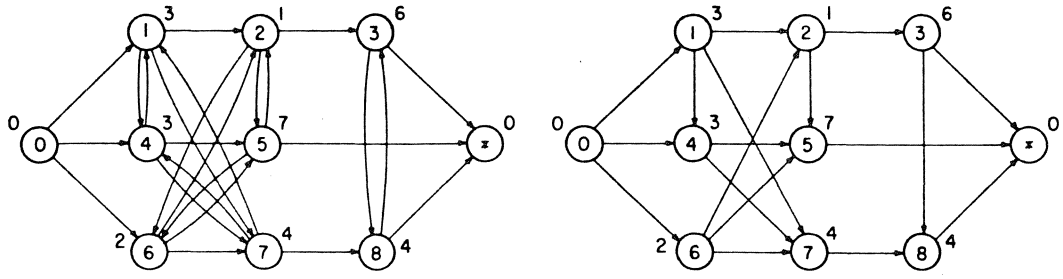
5.4. Job shop scheduling

5.4.1. $J2|\beta|C_{max}$, $J3|\beta|C_{max}$

A simple extension of Johnson's algorithm for $F2||C_{max}$ allows solution of $J2|m_j\leq 2|C_{max}$ in $O(n \log n)$ time [Jackson 1956]. Let J_i be the set of jobs with operations on M_i only $(i = 1,2)$ and J_{hi} the set of jobs that go from M_h to M_i $(hi = 12,21)$. Order the latter two sets by means of Johnson's algorithm and the former two sets arbitrarily. One then obtains an optimal schedule by executing the jobs on M_1 in the order (J_{12},J_1,J_{21}) and on M_2 in the order (J_{21},J_2,J_{12}).

Another special case, $J2|p_{ij}=1|C_{max}$, is solvable in $O(n \log n)$ time as well [Hefetz & Adiri 1979].

This, however, is probably as far as we can get. $J2|m_j\leq 3|C_{max}$ and $J3|m_j\leq 2|C_{max}$ are binary NP-hard [Lenstra *et al.* 1977; Gonzalez



Job shop problem, represented as a disjunctive graph

Job shop schedule, represented as an acyclic directed graph

Figure 3

& Sahni 1978A], J2|$p_{ij}\in\{1,2\}$|C_{max} and J3|$p_{ij}=1$|C_{max} are unary NP-hard [Lenstra & Rinnooy Kan 1979], and these results are still true if preemption is allowed.

5.4.2. J||C_{max}

The general job shop problem is extremely hard to solve optimally. An indication of this is given by the fact that a 10-job 10-machine problem, formulated in 1963 [Muth & Thompson 1963], still has not been solved.

A convenient problem representation is provided by the *disjunctive graph* model, introduced by Roy and Sussmann [Roy & Sussmann 1964]. Assume each operation O_{ij} being renumbered as O_u with $u = \Sigma_{k=1}^{j-1} m_k + i$ and add two fictitious initial and final operations O_0 and O_* with $p_0 = p_* = 0$. The disjunctive graph is then defined as follows. There is a vertex u with weight p_u corresponding to each operation O_u. The directed *conjunctive arcs* link the consecutive operations of each job, and link O_0 to all first operations and all last operations to O_*. A pair of directed *disjunctive arcs* connects every two operations that have to be executed on the same machine. A feasible schedule corresponds to the selection of one disjunctive arc of every such pair, granting precedence of one operation over the other on their common machine, in such a way that the resulting directed graph is acyclic. The value of the schedule is given by the weight of the maximum weight path from 0 to *. We refer to Figure 3 for an example.

At a typical stage of any enumerative algorithm, a certain subset D of disjunctive arcs will have been selected. We consider the directed graph obtained by removing all other disjunctive arcs. Let the maximum weights of paths from 0 to u and from u to *, excluding p_u, be denoted by r_u and q_u, respectively. In particular, r_* is an obvious lower bound on the value of any feasible schedule obtainable from the current graph [Charlton & Death 1970]. We can get a far better bound in a manner very similar to the development of flow shop bounds in Section 5.3.2 [Lageweg *et al.* 1977].

Let us relax the capacity constraints for all machines except M_i. We then obtain a problem of scheduling the operations O_u on M_i with release dates r_u, processing times p_u, due dates $-q_u$ and precedence constraints defined by the directed graph, so as to minimize maximum lateness. As pointed out in Section 3.2, this $1|prec,r_j|L_{max}$ problem is NP-hard, but there exist fast enumerative methods for its solution on each M_i. Again, almost all lower bounds proposed in the literature appear as special cases of the above one by underestimating the contribution of r_u, q_u or both, by ignoring the precedence constraints, or by restricting the set of machines over which maximization is to take place.

The currently best job shop algorithm [McMahon & Florian 1975] involves the $1|r_j|L_{max}$ bound combined with the enumeration of *active schedules*. Starting from O_0, we consider at each stage the subset S of operations all of whose predecessors have been scheduled and calculate their earliest possible completion times r_u+p_u. It can be shown [Giffler & Thompson 1960] that it is sufficient to consider only a machine on which the minimum value of r_u+p_u is achieved and to branch by successively scheduling next on that machine all O_v for which $r_v < \min_{O_u \in S}\{r_u+p_u\}$. In this scheme, several disjunctive arcs are added to D at each stage. An alternative approach whereby at each stage one disjunctive arc of some *crucial* pair is selected leads to a computationally inferior approach [Lageweg *et al.* 1977].

Surrogate duality relaxations of the job shop problem are investigated in [Fisher *et al.* 1981]. Either the precedence constraints fixing the machine orders for the jobs or the capacity constraints of the machines can be weighted and aggregated to a single constraint. For fixed values of the multipliers, the resulting problems can be solved in (pseudo)polynomial time. Although this approach leads to stronger lower bounds, it appears to be too time consuming to have much computational value.

As far as approximation algorithms are concerned, the performance guarantees due to [Gonzalez & Sahni 1978A] for flow shop algorithms AS and LPT (see Section 5.3.2) also apply to the case of a job shop.

A considerable effort has been invested in the empirical testing of various priority rules [Gere 1966; Conway *et al.* 1967; Day & Hottenstein 1970; Panwalkar & Iskander 1977]. No rule appears to be consistently better than any other and in practical situations one would be well advised to exploit any special structure that the problem at hand has to offer.

6. CONCLUDING REMARKS

If one thing emerges from the preceding survey, it is the amazing success of complexity theory as a means of differentiating between easy and hard problems. Within the very detailed problem

classification developed especially for this purpose, surprisingly few open problems remain. For an extensive class of scheduling problems, a computer program has been developed that classifies these problems according to their computational complexity [Lageweg *et al.* 1981A, 1981B]. It employs elementary reductions such as those defined in Section 2.7 in order to deduce the consequences of the development of a new polynomial-time algorithm or a new NP-hardness proof.

As far as polynomial-time algorithms are concerned, the most impressive recent advances have occurred in the area of parallel machine scheduling and are due to researchers with a computer science background, recognizable as such by their use of terms like *tasks* and *processors* rather than *jobs* and *machines*. Single machine, flow shop and job shop scheduling has been traditionally the domain of operations researchers; here, an analytical approach to the performance of approximation algorithms is badly needed.

Several extensions of the problem class considered in this paper appear to be worthy of further study. A quite natural one involves the presence of additional limited *resources*, with the property that each job requires the use of a part of each resource during its execution. These problems turn out to be fairly complicated. We refer to [Davis 1966, 1973] for surveys and extensive bibliographies on resource constrained project scheduling, to [Blazewicz *et al.* 1980] for a partial complexity classification of this problem class, and to [Garey & Johnson 1981] for the famous special case of *bin packing* models.

More fundamentally, the strictly deterministic character of our models represents one of their major shortcomings. The investigation of their *stochastic* counterparts is of obvious interest and forms the subject of several contributions to this volume.

The area of deterministic sequencing and scheduling has emerged as one of the more fruitful interfaces between computer science and operations research. Proper consideration for the practical relevance of further theoretical work should continue to make it a challenging research area for many years to come.

ACKNOWLEDGMENT

The research by the first author was supported by NSF grant MCS78-20054.

REFERENCES

N.R. ACHUTHAN (1980) *Flow-Shop Scheduling Problems*, Ph.D. Thesis, Indian Statistical Institute, Calcutta.

I. ADIRI, N. HEFETZ (1980) Subproblems of openshop - more than two machines - schedule length problem. Operations Research, Statistics and Economics Mimeograph Series 260, Technion, Haifa.

D. ADOLPHSON (1977) Single machine job sequencing with precedence constraints. *SIAM J. Comput.* 6,40-54.

D. ADOLPHSON, T.C. HU (1973) Optimal linear ordering. *SIAM J. Appl. Math.* 25,403-423.

K.R. BAKER (1974) *Introduction to Sequencing and Scheduling*, Wiley, New York.

K.R. BAKER, E.L. LAWLER, J.K. LENSTRA, A.H.G. RINNOOY KAN (1982) Preemptive scheduling of a single machine to minimize maximum cost subject to release dates and precedence constraints. *Oper. Res.*, to appear.

K.R. BAKER, L.E. SCHRAGE (1978) Finding an optimal sequence by dynamic programming: an extension to precedence-related tasks. *Oper. Res.* 26,111-120.

K.R. BAKER, Z.-S. SU (1974) Sequencing with due-dates and early start times to minimize maximum tardiness. *Naval Res. Logist. Quart.* 21,171-176.

M.S. BAKSHI, S.R. ARORA (1969) The sequencing problem. *Management Sci.* 16,B247-263.

J.W. BARNES, J.J. BRENNAN (1977) An improved algorithm for scheduling jobs on identical machines. *AIIE Trans.* 9,25-31.

L. BIANCO, S. RICCIARDELLI (1981) Scheduling of a single machine to minimize total weighted completion time subject to release dates. Istituto di Analisi dei Sistemi ed Informatica, CNR, Rome.

J. BLAZEWICZ, J.K. LENSTRA, A.H.G. RINNOOY KAN (1980) Scheduling subject to resource constraints: classification and complexity. Report BW 127, Mathematisch Centrum, Amsterdam.

P. BRATLEY, M. FLORIAN, P. ROBILLARD (1975) Scheduling with earliest start and due date constraints on multiple machines. *Naval Res. Logist. Quart.* 22,165-173.

P. BRUCKER, M.R. GAREY, D.S. JOHNSON (1977) Scheduling equal-length tasks under tree-like precedence constraints to minimize maximum lateness. *Math. Oper. Res.* 2,275-284.

J. BRUNO, E.G. COFFMAN, JR., R. SETHI (1974) Scheduling independent tasks to reduce mean finishing time. *Comm. ACM* 17,382-387.

J. BRUNO, T. GONZALEZ (1976) Scheduling independent tasks with release dates and due dates on parallel machines. Technical Report 213, Computer Science Department, Pennsylvania State University.

J. CARLIER (1980) Problème à une machine. Manuscript, Institut de programmation, Université Paris VI.

J.M. CHARLTON, C.C. DEATH (1970) A generalized machine scheduling algorithm. *Oper. Res. Quart.* 21,127-134.

N.-F. CHEN (1975) An analysis of scheduling algorithms in multiprocessing computing systems. Technical Report UIUCDCS-R-75-724, Department of Computer Science, University of Illinois at Urbana-Champaign.

N.-F. CHEN, C.L. LIU (1975) On a class of scheduling algorithms for multiprocessors computing systems. In: T.-Y. FENG (ed.) (1975) *Parallel Processing*, Lecture Notes in Computer Science 24,

Springer, Berlin, 1-16.

Y. CHO, S. SAHNI (1978) Preemptive scheduling of independent jobs with release and due times on open, flow and job shops. Technical Report 78-5, Computer Science Department, University of Minnesota, Minneapolis.

E.G. COFFMAN, JR. (ed.) (1976) *Computer and Job-Shop Scheduling Theory*, Wiley, New York.

E.G. COFFMAN, JR., M.R. GAREY, D.S. JOHNSON (1978) An application of bin-packing to multiprocessor scheduling. *SIAM J. Comput.* 7,1-17.

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

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

S.A. COOK (1971) The complexity of theorem-proving procedures. *Proc. 3rd Annual ACM Symp. Theory of Computing*, 151-158.

E.W. DAVIS (1966) Resource allocation in project network models - a survey. *J. Indust. Engrg.* 17,177-188.

E.W. DAVIS (1973) Project scheduling under resource constraints - historical review and categorization of procedures. *AIIE Trans.* 5,297-313.

J. DAY, M.P. HOTTENSTEIN (1970) Review of scheduling research. *Naval Res. Logist. Quart.* 17,11-39.

D. DOLEV (1981) Scheduling wide graphs. Unpublished manuscript.

W.L. EASTMAN, S. EVEN, I.M. ISAACS (1964) Bounds for the optimal scheduling of *n* jobs on *m* processors. *Management Sci.* 11, 268-279.

S.E. ELMAGHRABY, S.H. PARK (1974) Scheduling jobs on a number of identical machines. *AIIE Trans.* 6,1-12.

H. EMMONS (1969) One-machine sequencing to minimize certain functions of job tardiness. *Oper. Res.* 17,701-715.

M.L. FISHER (1976) A dual algorithm for the one-machine scheduling problem. *Math. Programming* 11,229-251.

M.L. FISHER, B.J. LAGEWEG, J.K. LENSTRA, A.H.G. RINNOOY KAN (1981) Surrogate duality relaxation for job shop scheduling. Report, Mathematisch Centrum, Amsterdam.

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 (1980) An almost-linear algorithm for two-processor scheduling. Technical Report CU-CS-169-80, Department of Computer Science, University of Colorado, Boulder.

M.R. GAREY (-) Unpublished.

M.R. GAREY, R.L. GRAHAM, D.S. JOHNSON (1978) Performance guarantees for scheduling algorithms. *Oper. Res.* 26,3-21.

M.R. GAREY, D.S. JOHNSON (1976) Scheduling tasks with nonuniform deadlines on two processors. *J. Assoc. Comput. Mach.* 23, 461-467.

M.R. GAREY, D.S. JOHNSON (1977) Two-processor scheduling with start-times and deadlines. *SIAM J. Comput.* 6,416-426.

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 (1981) Approximation algorithms for bin packing problems: a survey. In: G. AUSIELLO, M. LUCERTINI (eds.) (1981) *Analysis and Design of Algorithms in Combinatorial Optimization*, CISM Courses and Lectures 266, Springer, Vienna, 147-172.

M.R. GAREY, D.S. JOHNSON, R. SETHI (1976) The complexity of flowshop and jobshop scheduling. *Math. Oper. Res.* 1,117-129.

M.R. GAREY, D.S. JOHNSON, B.B. SIMONS, R.E. TARJAN (1981A) Scheduling unit-time tasks with arbitrary release times and deadlines. *SIAM J. Comput.* 10,256-269.

M.R. GAREY, D.S. JOHNSON, R.E. TARJAN, M. YANNAKAKIS (1981B) Scheduling opposing forests. Unpublished manuscript.

L. GELDERS, P.R. KLEINDORFER (1974) Coordinating aggregate and detailed scheduling decisions in the one-machine job shop: part I. Theory. *Oper. Res.* 22,46-60.

L. GELDERS, P.R. KLEINDORFER (1975) Coordinating aggregate and detailed scheduling in the one-machine job shop: II - computation and structure. *Oper. Res.* 23,312-324.

W.S. GERE (1966) Heuristics in job shop scheduling. *Management Sci.* 13,167-190.

B. GIFFLER, G.L. THOMPSON (1960) Algorithms for solving production-scheduling problems. *Oper. Res.* 8,487-503.

P.C. GILMORE, R.E. GOMORY (1964) Sequencing a one-state variable machine: a solvable case of the traveling salesman problem. *Oper. Res.* 12,655-679.

T. GONZALEZ (1977) Optimal mean finish time preemptive schedules. Technical Report 220, Computer Science Department, Pennsylvania State University.

T. GONZALEZ (1979A) A note on open shop preemptive schedules. *IEEE Trans. Computers* C-28,782-786.

T. GONZALEZ (1979B) NP-Hard shop problems. Report CS-79-35, Department of Computer Science, Pennsylvania State University, University Park.

T. GONZALEZ, O.H. IBARRA, S. SAHNI (1977) Bounds for LPT schedules on uniform processors. *SIAM J. Comput.* 6,155-166.

T. GONZALEZ, D.B. JOHNSON (1980) A new algorithm for preemptive scheduling of trees. *J. Assoc. Comput. Mach.* 27,287-312.

T. GONZALEZ, E.L. LAWLER, S. SAHNI (1981) Optimal preemptive scheduling of a fixed number of unrelated processors in linear time. To appear.

T. GONZALEZ, S. SAHNI (1976) Open shop scheduling to minimize finish time. *J. Assoc. Comput. Mach.* 23,665-679.

T. GONZALEZ, S. SAHNI (1978A) Flowshop and jobshop schedules: complexity and approximation. *Oper. Res.* 26,36-52.

T. GONZALEZ, S. SAHNI (1978B) Preemptive scheduling of uniform processor systems. *J. Assoc. Comput. Mach.* 25,92-101.

D.K. GOYAL (1977) Non-preemptive scheduling of unequal execution time tasks on two identical processors. Technical Report

CS-77-039, Computer Science Department, Washington State University, Pullman.

R.L. GRAHAM (1966) Bounds for certain multiprocessing anomalies. *Bell System Tech. J.* 45,1563-1581.

R.L. GRAHAM (1969) Bounds on multiprocessing timing anomalies. *SIAM J. Appl. Math.* 17,263-269.

R.L. GRAHAM (-) Unpublished.

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.

J.N.D. GUPTA, S.S. REDDI (1978) Improved dominance conditions for the three-machine flowshop scheduling problem. *Oper. Res.* 26, 200-203.

A.M.A. HARIRI, C.N. POTTS (1981) An algorithm for single machine sequencing with release dates to minimise total weighted completion time. Report BW 143, Mathematisch Centrum, Amsterdam.

N. HEFETZ, I. ADIRI (1979) An efficient optimal algorithm for the two-machines, unit-time, job-shop, schedule-length, problem. Operations Research, Statistics and Economics Mimeograph Series 237, Technion, Haifa.

W.A. HORN (1972) Single-machine job sequencing with treelike precedence ordering and linear delay penalties. *SIAM J. Appl. Math.* 23,189-202.

W.A. HORN (1973) Minimizing average flow time with parallel machines. *Oper. Res.* 21,846-847.

W.A. HORN (1974) Some simple scheduling algorithms. *Naval Res. Logist. Quart.* 21,177-185.

E. HOROWITZ, S. SAHNI (1976) Exact and approximate algorithms for scheduling nonidentical processors. *J. Assoc. Comput. Mach.* 23,317-327.

E.C. HORVATH, S. LAM, R. SETHI (1977) A level algorithm for preemptive scheduling. *J. Assoc. Comput. Mach.* 24,32-43.

N.C. HSU (1966) Elementary proof of Hu's theorem on isotone mappings. *Proc. Amer. Math. Soc.* 17,111-114.

T.C. HU (1961) Parallel sequencing and assembly line problems. *Oper. Res.* 9,841-848.

O.H. IBARRA, C.E. KIM (1976) On two-processor scheduling of one- or two-unit time tasks with precedence constraints. *J. Cybernet.* 5,87-109.

O.H. IBARRA, C.E. KIM (1977) Heuristic algorithms for scheduling independent tasks on nonidentical processors. *J. Assoc. Comput. Mach.* 24,280-289.

O.H. IBARRA, C.E. KIM (1978) Approximation algorithms for certain scheduling problems. *Math. Oper. Res.* 3,197-204.

E. IGNALL, L. SCHRAGE (1965) Application of the branch-and-bound technique to some flow-shop scheduling problem. *Oper. Res.* 13,400-412.

J.R. JACKSON (1955) Scheduling a production line to minimize maximum tardiness. Research Report 43, Management Science Research Project, University of California, Los Angeles.

J.R. JACKSON (1956) An extension of Johnson's results on job lot scheduling. *Naval Res. Logist. Quart.* 3,201-203.

J.M. JAFFE (1979) Efficient scheduling of tasks without full use of processor resources. Report MIT/LCS/TM-122, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge.

J.M. JAFFE (1980) An analysis of preemptive multiprocessor job scheduling. *Math. Oper. Res.* 5,415-421.

S.M. JOHNSON (1954) Optimal two- and three-stage production schedules with setup times included. *Naval Res. Logist. Quart.* 1, 61-68.

S.M. JOHNSON (1958) Discussion: sequencing *n* jobs on two machines with arbitrary time lags. *Management Sci.* 5,299-303.

R.M. KARP (1972) Reducibility among combinatorial problems. In: R. E. MILLER, J.W. THATCHER (eds.) (1972) *Complexity of Computer Computations*, Plenum Press, New York, 85-103.

R.M. KARP (1975) On the computational complexity of combinatorial problems. *Networks* 5,45-68.

L.G. KHACHIYAN (1979) A polynomial algorithm in linear programming. *Soviet Math. Dokl.* 20,191-194.

H. KISE, T. IBARAKI, H. MINE (1978) A solvable case of the one-machine scheduling problem with ready and due times. *Oper. Res.* 26,121-126.

M. KUNDE (1976) Beste Schranken beim LP-Scheduling. Bericht 7603, Institut für Informatik und Praktische Mathematik, Universität Kiel.

J. LABETOULLE, E.L. LAWLER, J.K. LENSTRA, A.H.G. RINNOOY KAN (1979) Preemptive scheduling of uniform machines subject to release dates. Report BW 99, Mathematisch Centrum, Amsterdam.

B.J. LAGEWEG, E.L. LAWLER, J.K. LENSTRA, A.H.G. RINNOOY KAN (1981A) Computer aided complexity classification of combinatorial problems. Report BW 137, Mathematisch Centrum, Amsterdam.

B.J. LAGEWEG, E.L. LAWLER, J.K. LENSTRA, A.H.G. RINNOOY KAN (1981B) Computer aided complexity classification of deterministic scheduling problems. Report BW 138, Mathematisch Centrum, Amsterdam.

B.J. LAGEWEG, J.K. LENSTRA, A.H.G. RINNOOY KAN (1976) Minimizing maximum lateness on one machine: computational experience and some applications. *Statist. Neerlandica* 30,25-41.

B.J. LAGEWEG, J.K. LENSTRA, A.H.G. RINNOOY KAN (1977) Job-shop scheduling by implicit enumeration. *Management Sci.* 24,441-450.

B.J. LAGEWEG, J.K. LENSTRA, A.H.G. RINNOOY KAN (1978) A general bounding scheme for the permutation flow-shop problem. *Oper. Res.* 26,53-67.

S. LAM, R. SETHI (1977) Worst case analysis of two scheduling algorithms. *SIAM J. Comput.* 6,518-536.

E.L. LAWLER (1973) Optimal sequencing of a single machine subject to precedence constraints. *Management Sci.* 19,544-546.

E.L. LAWLER (1976A) Sequencing to minimize the weighted number of tardy jobs. *RAIRO Rech. Opér.* 10.5 Suppl.27-33.

E.L. LAWLER (1976B) *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and 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 (1978) Sequencing jobs to minimize total weighted completion time subject to precedence constraints. *Ann. Discrete Math.* 2,75-90.

E.L. LAWLER (1980) Preemptive scheduling of precedence-constrained jobs on parallel machines. Report BW 132, Mathematisch Centrum, Amsterdam.

E.L. LAWLER (1981) Preemptive scheduling of uniform parallel machines to minimize the number of late jobs. Report, Mathematisch Centrum, Amsterdam, to appear.

E.L. LAWLER (-) Unpublished.

E.L. LAWLER, J. LABETOULLE (1978) On preemptive scheduling of unrelated parallel processors by linear programming. *J. Assoc. Comput. Mach.* 25,612-619.

E.L. LAWLER, J.K. LENSTRA, A.H.G. RINNOOY KAN (1981) Minimizing maximum lateness in a two-machine open shop. *Math. Oper. Res.* 6,153-158.

E.L. LAWLER, C.U. MARTEL (1980) Computing "polymatroidal" network flows. Research Memorandum ERL M80/52, Electronics Research Laboratory, University of California, Berkeley.

E.L. LAWLER, J.M. MOORE (1969) A functional equation and its application to resource allocation and sequencing problems. *Management Sci.* 16,77-84.

J.K. LENSTRA (1977) *Sequencing by Enumerative Methods*, Mathematical Centre Tracts 69, Mathematisch Centrum, Amsterdam.

J.K. LENSTRA (-) Unpublished.

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 (1979) Computational complexity of discrete optimization problems. *Ann. Discrete Math.* 4, 121-140.

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.

C.L. LIU (1972) Optimal scheduling on multi-processor computing systems. *Proc. 13th Annual IEEE Symp. Switching and Automata Theory*, 155-160.

C.L. LIU (1976) Deterministic job scheduling in computing systems. Department of Computer Science, University of Illinois at Urbana-Champaign.

J.W.S. LIU, C.L. LIU (1974A) Bounds on schedulling algorithms for heterogeneous computing systems. In: J.L. ROSENFELD (ed.) (1974) *Information Processing 74*, North-Holland, Amsterdam, 349-353.

J.W.S. LIU, C.L. LIU (1974B) Bounds on scheduling algorithms for heterogeneous computing systems. Technical Report UIUCDCS-R-74-632, Department of Computer Science, University of Illinois at Urbana-Champaign, 68 pp.

J.W.S. LIU, C.L. LIU (1974C) Performance analysis of heterogeneous multi-processor computing systems. In: E. GELENBE, R. MAHL (eds.) (1974) *Computer Architectures and Networks*, North-Holland, Amsterdam, 331-343.

C. MARTEL (1981) Scheduling uniform machines with release times, deadlines and due times. *J. Assoc. Comput. Mach.*, to appear.

G.B. McMAHON (1969) Optimal production schedules for flow shops. *Canad. Oper. Res. Soc. J.* 7,141-151.

G.B. McMAHON (1971) *A Study of Algorithms for Industrial Scheduling Problems*, Ph.D. Thesis, University of New South Wales, Kensington.

G.B. McMAHON, M. FLORIAN (1975) On scheduling with ready times and due dates to minimize maximum lateness. *Oper. Res.* 23,475-482.

R. McNAUGHTON (1959) Scheduling with deadlines and loss functions. *Management Sci.* 6,1-12.

L.G. MITTEN (1958) Sequencing *n* jobs on two machines with arbitrary time lags. *Management Sci.* 5,293-298.

C.L. MONMA (1980) Sequencing to minimize the maximum job cost. *Oper. Res.* 28,942-951.

C.L. MONMA, A.H.G. RINNOOY KAN (1981) Efficiently solvable special cases of the permutation flow-shop problem. Report 8105, Erasmus University, Rotterdam.

C.L. MONMA, J.B. SIDNEY (1979) Sequencing with series-parallel precedence constraints. *Math. Oper. Res.* 4,215-224.

J.M. MOORE (1968) An *n* job, one machine sequencing algorithm for minimizing the number of late jobs. *Management Sci.* 15,102-109.

R.R. MUNTZ, E.G. COFFMAN, JR. (1969) Optimal preemptive scheduling on two-processor systems. *IEEE Trans. Computers* C-18,1014-1020.

R.R. MUNTZ, E.G. COFFMAN, JR. (1970) Preemptive scheduling of real time tasks on multiprocessor systems. *J. Assoc. Comput. Mach.* 17,324-338.

J.F. MUTH, G.L. THOMPSON (eds.) (1963) *Industrial Scheduling*, Prentice-Hall, Englewood Cliffs, N.J., 236.

I. NABESHIMA (1963) Sequencing on two machines with start lag and stop lag. *J. Oper. Res. Soc. Japan* 5,97-101.

S.S. PANWALKAR, W. ISKANDER (1977) A survey of scheduling rules. *Oper. Res.* 25,45-61.

C.H. PAPADIMITRIOU, P.C. KANELLAKIS (1980) Flowshop scheduling with limited temporary storage. *J. Assoc. Comput. Mach.* 27,533-549.

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

C.N. POTTS (1980A) An adaptive branching rule for the permutation flow-shop problem. *European J. Oper. Res.* 5,19-25.

C.N. POTTS (1980B) Analysis of a heuristic for one machine sequencing with release dates and delivery times. *Oper. Res.* 28,1436-1441.

C.N. POTTS (-) Unpublished.

S.S. REDDI, C.V. RAMAMOORTHY (1972) On the flow-shop sequencing problem with no wait in process. *Oper. Res. Quart.* 23,323-331.

G. RINALDI, A. SASSANO (1977) On a job scheduling problem with different ready times: some properties and a new algorithm to determine the optimal solution. Report R.77-24, Istituto di Automatica, Università di Roma.

A.H.G. RINNOOY KAN (1976) *Machine Scheduling Problems: Classification, Complexity and Computations*, Nijhoff, The Hague.

A.H.G. RINNOOY KAN, B.J. LAGEWEG, J.K. LENSTRA (1975) Minimizing total costs in one-machine scheduling. *Oper. Res.* 23,908-927.

P. ROSENFELD (-) Unpublished.

M.H. ROTHKOPF (1966) Scheduling independent tasks on parallel processors. *Management Sci.* 12,437-447.

B. ROY, B. SUSSMANN (1964) Les problèmes d'ordonnancement avec contraintes disjonctives. Note DS no.9 bis, SEMA, Montrouge.

S. SAHNI (1976) Algorithms for scheduling independent tasks. *J. Assoc. Comput. Mach.* 23,116-127.

S. SAHNI, Y. CHO (1979A) Complexity of scheduling jobs with no wait in process. *Math. Oper. Res.* 4,448-457.

S. SAHNI, Y. CHO (1979B) Nearly on line scheduling of a uniform processor system with release times. *SIAM J. Comput.* 8,275-285.

S. SAHNI, Y. CHO (1980) Scheduling independent tasks with due times on a uniform processor system. *J. Assoc. Comput. Mach.* 27,550-563.

R. SETHI (1976A) Algorithms for minimal-length schedules. In: [Coffman 1976], 51-99.

R. SETHI (1976B) Scheduling graphs on two processors. *SIAM J. Comput.* 5,73-82.

R. SETHI (1977) On the complexity of mean flow time scheduling. *Math. Oper. Res.* 2,320-330.

J. SHWIMER (1972) On the *N*-job, one-machine, sequence-independent scheduling problem with tardiness penalties: a branch-and-bound solution. *Management Sci.* 18,B301-313.

J.B. SIDNEY (1973) An extension of Moore's due date algorithm. In: S.E. ELMAGHRABY (ed.) (1973) *Symposium on the Theory of Scheduling and its Applications*, Lecture Notes in Economics and Mathematical Systems 86, Springer, Berlin, 393-398.

J.B. SIDNEY (1975) Decomposition algorithms for single-machine sequencing with precedence relations and deferral costs. *Oper. Res.* 23,283-298.

J.B. SIDNEY (1979) The two-machine maximum flow time problem with series parallel precedence relations. *Oper. Res.* 27,782-791.

B. SIMONS (1978) A fast algorithm for single processor scheduling. *Proc. 19th Annual IEEE Symp. Foundations of Computer Science*, 246-252.

B. SIMONS (1980) A fast algorithm for multiprocessor scheduling. *Proc. 21st Annual IEEE Symp. Foundations of Computer Science*, 50-53.

M.L. SMITH, S.S. PANWALKAR, R.A. DUDEK (1975) Flow shop sequencing

with ordered processing time matrices. *Management Sci.* 21, 544-549.

M.L. SMITH, S.S. PANWALKAR, R.A. DUDEK (1976) Flow shop sequencing problem with ordered processing time matrices: a general case. *Naval Res. Logist. Quart.* 23,481-486.

W.E. SMITH (1956) Various optimizers for single-stage production. *Naval Res. Logist. Quart.* 3,59-66.

H.I. STERN (1976) Minimizing makespan for independent jobs on non-identical parallel machines - an optimal procedure. Working Paper 2/75, Department of Industrial Engineering and Management, Ben-Gurion University of the Negev, Beer-Sheva.

W. SZWARC (1968) On some sequencing problems. *Naval Res. Logist. Quart.* 15,127-155.

W. SZWARC (1971) Elimination methods in the $m \times n$ sequencing problem. *Naval Res. Logist. Quart.* 18,295-305.

W. SZWARC (1973) Optimal elimination methods in the $m \times n$ sequencing problem. *Oper. Res.* 21,1250-1259.

W. SZWARC (1978) Dominance conditions for the three-machine flow-shop problem. *Oper. Res.* 26,203-206.

J.D. ULLMAN (1975) *NP*-Complete scheduling problems. *J. Comput. System Sci.* 10,384-393.

J.D. ULLMAN (1976) Complexity of sequencing problems. In: [Coffman 1976], 139-164.

M.K. WARMUTH (1980) M Processor unit-execution-time scheduling reduces to M-1 weakly connected components. M.S. Thesis, Department of Computer Science, University of Colorado, Boulder.

D.A. WISMER (1972) Solution of the flowshop-scheduling problem with no intermediate queues. *Oper. Res.* 20,689-697.