Annals of Discrete Mathematics 5 (1979) 287–326
© North-Holland Publishing Company

OPTIMIZATION AND APPROXIMATION IN DETERMINISTIC SEQUENCING AND SCHEDULING: A SURVEY

R.L. GRAHAM

Bell Laboratories, Murray Hill, NJ, U.S.A.

E.L. LAWLER

University of California, Berkeley, CA, U.S.A.

J.K. LENSTRA

Mathematisch Centrum, Amsterdam, The Netherlands

A.H.G. RINNOOY KAN

Erasmus University, Rotterdam, The Netherlands

The theory of deterministic sequencing and scheduling has expanded rapidly during the past years. In this paper 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. We indicate some problems for future research and include a selective bibliography.

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, 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 more recent 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].

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 [Karp 1972, 1975]. An introductory survey of this area appears elsewhere in this journal [Lenstra & Rinnooy Kan 1978B] 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 3-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;* $p_{ij} = p_j$ $(i = 1, \ldots, m)$;

$\alpha_1 = Q$: *uniform parallel machines;* $p_{ij} = q_i p_j$ for a given *speed factor* q_i of M_i $(i = 1, \ldots, m)$;

$\alpha_1 = R$: *unrelated parallel machines.*

If $\alpha_1 = O$, we have an *open shop*, in which each J_j consists of a set of operations $\{O_{1j}, \ldots, O_{mj}\}$. O_{ij} has to be processed on M_j 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 $\alpha_1 = F$, we have a *flow shop*, in which each J_j consists of a chain $(O_{1j}, \ldots, O_{mj})$. O_{ij} has to be processed on M_i during p_{ij} time units.

If $\alpha_1 = J$, we have a *job shop*, in which each J_j consists of a chain $(O_{1j}, \ldots, O_{m_j j})$. 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 $i = 2, \ldots, m_j$.

If α_2 is a positive integer, then m is *constant* and equal to α_2. If $\alpha_2 = \circ$, then m is assumed to be *variable*. Obviously, $\alpha_1 = \circ$ if and only if $\alpha_2 = 1$.

2.4. *Job characteristics*

The second field $\beta \subset \{\beta_1, \ldots, \beta_6\}$ indicates a number of job characteristics, which are defined as follows.

(1) $\beta_1 \in \{pmtn, \circ\}$.
$\beta_1 = pmtn$: *Preemption* (job splitting) is allowed; the processing of any operation may be interrupted and resumed at a later time.
$\beta_1 = \circ$: No preemption is allowed.

(2) $\beta_2 \in \{res, res1, \circ\}$.
$\beta_2 = res$: The presence of s limited *resources* R_h $(h = 1, \ldots, s)$ is assumed, with the property that each J_j requires the use of r_{hj} units of R_h at all times during its execution. Of course, at no time may more than 100% of any resource be in use.
$\beta_2 = res1$: The presence of only a *single resource* is assumed.
$\beta_2 = \circ$: No resource constraints are specified.

(3) $\beta_3 \in \{prec, tree, \circ\}$.
$\beta_3 = prec$: A *precedence relation* $<$ between the jobs is specified. It is derived from a directed acyclic graph G with vertex set $\{1, \ldots, n\}$. If G contains a directed path from j to k, we write $J_j < J_k$ and require that J_j is completed before J_k can start.
$\beta_3 = tree$: G is a *rooted tree* with either outdegree at most one for each vertex or indegree at most one for each vertex.
$\beta_3 = \circ$: No precedence relation is specified.

(4) $\beta_4 \in \{r_j, \circ\}$.
$\beta_4 = r_j$: *Release dates* that may differ per job are specified.
$\beta_4 = \circ$: We assume that $r_j = 0$.

(5) $\beta_5 \in \{m_j \leqslant \bar{m}, \circ\}$.
$\beta_5 = m_j \leqslant \bar{m}$: A *constant upper bound* on m_j is specified (only if $\alpha_1 = J$).
$\beta_5 = \circ$: No such bound is specified.

(6) $\beta_6 \in \{p_{ij} = 1, \underline{p} \leqslant p_{ij} \leqslant \bar{p}, \circ\}$.
$\beta_6 = p_{ij} = 1$: Each operation has *unit processing time*.

$\beta_6 = \underline{p} \leqslant p_{ij} \leqslant \bar{p}$: *Constant lower and upper bounds on* p_{ij} are specified.
$\beta_6 = \circ$: No such bounds are specified.

2.5. *Optimality criteria*

The third field $\gamma \in \{f_{\max}, \sum f_j\}$ 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* $U_j =$ if $C_j \leqslant d_j$, then 0, else 1.

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

$$\sum f_j \in \left\{\sum C_j, \sum T_j, \sum U_j, \sum w_j C_j, \sum w_j T_j, \sum w_j U_j\right\}$$

where $\sum f_j = \sum_{j=1}^{n} f_j(C_j)$ 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 $\sum w_j C_j$ and $\sum w_j L_j$ differ by a constant $\sum w_j d_j$ 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 (†).

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).

$R\,|pmtn|\,\sum C_j$: 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 8-tuple $(v_i)_{i=1}^{8}$, where v_i is a vertex of graph G_i, drawn in Fig. 2.*i* ($i = 1, \ldots, 8$). For two problems $P' = (v_i')_{i=1}^{8}$ and $P = (v_i)_{i=1}^{8}$, we write $P' \to P$ if either $v_i' = v_i$ or G_i contains a directed path from v_i' to v_i, for $i = 1, \ldots, 8$. The reader should verify that $P' \to P$ implies $P' \propto P$. The graphs thus define elementary reductions among

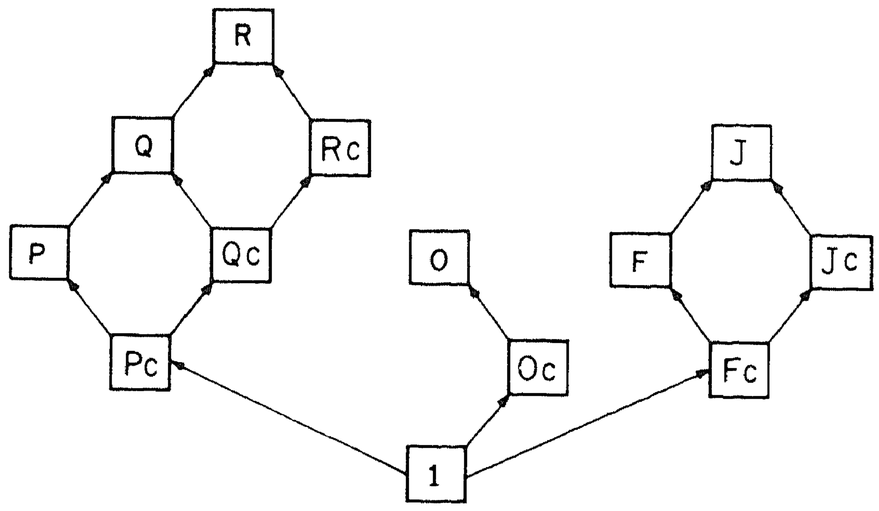

Fig. 2.1. G_1; c denotes an integer constant.

Fig. 2.2. G_2.

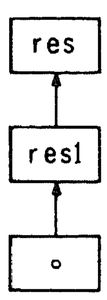

Fig. 2.3. G_3.

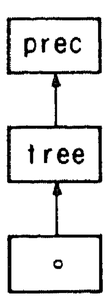

Fig. 2.4. G_4.

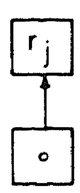

Fig. 2.5. G_5.

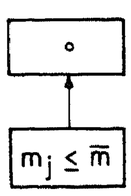

Fig. 2.6. G_6.

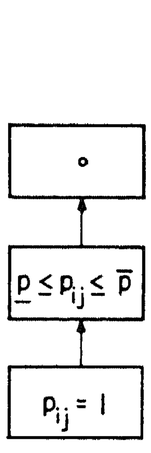

Fig. 2.7. G_7.

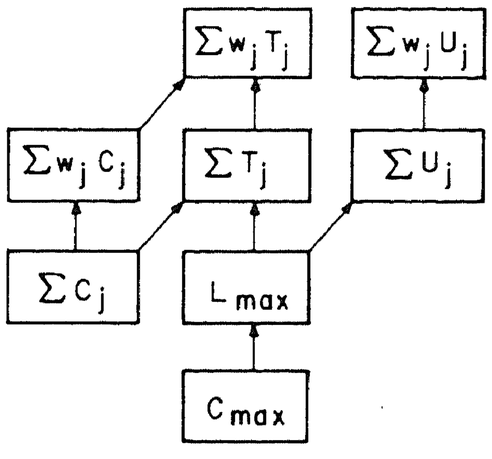

Fig. 2.8. G_8.

scheduling problems. It follows that

if $P' \rightarrow P$ and $P \in \mathcal{P}$, then $P' \in \mathcal{P}$;

if $P' \rightarrow P$ and P' is NP-hard, then P is NP-hard.

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*

The most general 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 $p(S)=\sum_{j\in S} p_j$, and let $S'\subset S$ indicate the jobs all whose successors have been scheduled. One selects J_k for the last position among $\{J_j \mid j\in S\}$ by requiring that $f_k(p(S))\leqslant f_j(p(S))$ for all $j\in S'$.

For $1\,\|\,L_{\max}$, this procedure specializes to *Jackson's rule:* schedule the jobs according to nondecreasing due dates [Jackson 1955]. Introduction of release dates turns this problem into a unary NP-hard one [Lenstra et al. 1977].

$1\,|prec, r_j, p_j=1|\,L_{\max}$ and $1\,|pmtn, prec, r_j|\,L_{\max}$ can still be solved in polynomial time: first update release and due dates so that they suitably reflect the precedence constraints and then apply Jackson's rule continually to the set of available jobs [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 problems with up to 80 jobs [Lageweg et al. 1976].

3.3. *Minimizing total cost*

3.3.1. $1\,|\beta|\,\sum w_jC_j$

The case $1\,\|\,\sum w_jC_j$ 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* [Knuth 1973; 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 $J_j<J_k$ 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 1978A].

If release dates are introduced, $1\,|r_j|\sum C_j$ is already unary NP-hard [Lenstra et al. 1977]. In the preemptive case, $1\,|pmtn, r_j|\sum C_j$ can be solved by an obvious extension of Smith's rule, but, surprisingly, $1\,|pmtn, r_j|\sum w_jC_j$ is unary NP-hard [Labetoulle et al. 1978].

3.3.2. $1\,|\beta|\sum w_jT_j$

$1\,\|\sum w_jT_j$ is a unary NP-hard problem [Lawler 1977; Lenstra et al. 1977], for which various enumerative solution methods have been proposed, some of which can be extended to cover arbitrary nondecreasing cost functions. Lower bounds developed for the problem involve 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 $O(n^4\sum p_j)$ time [Lawler 1977]; the computational complexity of $1\,\|\sum T_j$ with respect to a binary encoding remains an open question.

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

If we introduce release dates, $1\,|r_j, p_j = 1|\sum w_jT_j$ can again be solved as a linear assignment problem, whereas $1\,|r_j|\sum T_j$ is obviously unary NP-hard (cf. Section 2.7).

3.3.3. $1\,|\beta|\sum w_jU_j$

An algorithm due to Moore [Moore 1968] allows solution of $1\,\|\sum U_j$ 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 problem also remains solvable in polynomial time if we add *agreeable weights* (i.e., $p_j < p_k \Rightarrow w_j \geqslant w_k$) [Lawler 1976A] or *agreeable release dates* (i.e., $d_j < d_k \Rightarrow r_j \leqslant r_k$) [Kise et al. 1978]. $1\,\|\sum w_jU_j$ is binary NP-hard [Karp 1972], but can be solved by dynamic programming in $O(n\sum p_j)$ time [Lawler & Moore 1969].

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

Of course, $1\,|r_j|\sum U_j$ is unary NP-hard. The preemptive case $1\,|pmtn, r_j|\sum U_j$ is an intriguing open problem.

Very little work has been done on worst-case analysis of approximation algorithms for single machine problems. For $1\,\|\sum w_jU_j$, Sahni [Sahni 1976] presents algorithms A_k with $O(n^3k)$ running time such that

$$\sum w_j\bar{U}_j(A_k)\Big/\sum w_j\bar{U}_j^* \geqslant 1-\frac{1}{k},$$

where $\bar{U}_j = 1-U_j$. For $1\,|tree|\sum w_jU_j$, Ibarra and Kim [Ibarra & Kim 1975] 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 $P2\,\|\sum w_jC_j$ 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 $\sum C_j$ 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 $\sum C_j$, $C_{\max}$ and $L_{\max}$ criteria have been obtained quite recently. These are summarized in Section 4.4. With respect to other criteria, $P2\,|pmtn|\sum w_jC_j$ turns out to be NP-hard (see Section 4.4.1). Little is known about $P\,|pmtn|\sum T_j$ and $P\,|pmtn|\sum U_j$; these problems remain open. However, we know from Section 3 that $1\,|pmtn|\sum w_jT_j$ and $1\,|pmtn|\sum w_jU_j$ are already NP-hard.

4.2. *Nonpreemptive scheduling: unit processing times*

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

A simple transportation network model provides an efficient solution method for $Q\,|p_j = 1|\sum f_j$ 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 $c_{ijk} = f_j(kq_i)$. The arc flow x_{ijk} is

to have the interpretation:

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

Then the problem is to minimize

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

subject to

$$\sum_{i,k} x_{ijk} = 1 \quad \text{for all } j,$$

$$\sum_{j} x_{ijk} \leqslant 1 \quad \text{for all } i, k,$$

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

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 $m \leqslant n$, the overall running time is $O(n^3)$.

It may be noted that some special cases can be solved more efficiently. For instance, $P\,|p_j = 1|\sum U_j$ can be solved in $O(n \log n)$ time [Lawler 1976A].

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 1978A]. It is an open question whether this remains true for any constant value of $m \geqslant 3$. The problem is in $\mathcal{P}$, 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].

$P2\,|prec, p_j = 1|\,C_{\max}$ can be solved in $O(n^2)$ time [Coffman & Graham 1972]. Previous polynomial-time algorithms for this problem are given in [Fujii et al. 1969, 1971; Muraoka 1971].

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., $J_j \not< J_k$ and $J_k \not< J_j$. 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 a list algorithm. First the jobs are labelled in the following way. Suppose labels $1, \ldots, k$ 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 successors 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 $n+a$, where a is the number of arcs in the *transitive reduction* of the precedence graph (all arcs implied by transitivity removed) [Sethi 1976B]. Note, however, that construction of such a representation requires $O(n^{2.8})$ time [Aho et al. 1972].

Garey and Johnson present polynomial algorithms for $P2\,|prec, p_j = 1|\,C_{\max}$ where, in addition, each job becomes available at its *release date* and has to meet a given *deadline*. In this approach, one obtains an optimal schedule by processing the jobs in order of increasing modified deadlines. This modification requires $O(n^2)$ time if all $r_j = 0$ [Garey & Johnson 1976A] and $O(n^3)$ time in the general case [Garey & Johnson 1977].

We note that $P\,|prec, p_j = 1|\sum C_j$ is NP-hard [Lenstra & Rinnooy Kan 1978A]. Hu's algorithm does not yield an optimal $\sum C_j$ schedule in the case of intrees, but in the case of outtrees critical path scheduling minimizes both $C_{\max}$ and $\sum C_j$ [Rosenfeld –]. The Coffman-Graham algorithm also minimizes $\sum C_j$ [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 1978A] implies that, unless $\mathcal{P} = \mathcal{NP}$, the best possible worst-case bound for a polynomial-time algorithm would be $\frac{4}{3}$. 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

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

In [Kaufman 1972] an example is constructed for which *no* CP schedule is optimal.

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

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

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

$$C_{\max}(\mathrm{SS})/C^*_{\max} \leqslant \tfrac{4}{3} \quad \text{for} \quad m = 2. \tag{†}$$

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

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

$$C_{\max}(\mathrm{GCG})/C^*_{\max} \leqslant \begin{cases} \tfrac{4}{3} & \text{for} \quad k = 2, \\ \tfrac{3}{2} - \dfrac{1}{2k} & \text{for} \quad k \geqslant 3. \end{cases} \tag{†}$$

4.2.3. $P\,|res, \beta, p_j = 1|\ C_{\max}$

We now take up the variation in which *resource* constraints enter the model. $P2\,|res, p_j = 1|\ C_{\max}$ can be formulated and solved as a maximum cardinality matching problem in an obvious way. However, $P2\,|res\,1, tree, p_j = 1|\ C_{\max}$ and $P3\,|res\,1, p_j = 1|\ C_{\max}$ are unary NP-hard [Garey & Johnson 1975].

For the case $P\,|res, prec, p_j = 1, m \geqslant n|\ C_{\max}$, the following results for list scheduling (LS) using an arbitrary priority list are known [Garey et al. 1976A]:

$$C_{\max}(\mathrm{LS})/C^*_{\max} \leqslant \tfrac{1}{2}sC^*_{\max} + \tfrac{1}{2}s + 1$$

and examples exist with

$$C_{\max}(\mathrm{LS})/C^*_{\max} \geqslant \tfrac{1}{2}sC^*_{\max} + \tfrac{1}{2}s + 1 - 2s/C^*_{\max}.$$

For the CP scheduling algorithm, the bound improves considerably:

$$C_{\max}(\mathrm{CP})/C^*_{\max} \leqslant \tfrac{17}{10}s + 1 \quad (s \geqslant 0). \tag{†}$$

Let DMR denote the algorithm which schedules jobs according to decreasing maximum resource requirement. Then

$$C_{\max}(\mathrm{DMR})/C^*_{\max} \leqslant \tfrac{17}{10}s + 1.$$

In the other direction, examples are given in [Garey et al. 1976A] for any $\varepsilon > 0$ with

$$C_{\max}(\mathrm{DMR})/C^*_{max} > \sum_{i=1}^{\infty} \frac{1}{a_i} - \varepsilon = 1.69\cdots - \varepsilon$$

where $a_1 = 1$ and $a_{i+1} = a_i(a_i + 1)$ for $i \geqslant 1$.

An even better bound applies to the case of independent jobs, i.e., $P\,|res, p_j = 1, m \geqslant n|\ C_{\max}$:

$$C_{\max}(\mathrm{LS}) \leqslant (s + \tfrac{7}{10})C^*_{\max} + \tfrac{7}{2} \quad (s \geqslant 1),$$

where the coefficient of $C^*_{\max}$ is best possible.

The case $P|res\ 1, p_j = 1, m \geqslant n| C_{\max}$ has been the subject of intensive study (under the name of *bin packing*) during the past few years. The problem can be viewed as one of placing a number of items with weights r_{1j} into a minimum number of bins of capacity 1. It is also known as the *one-dimensional cutting stock* problem. It is for this scheduling model that some of the deepest results have been obtained. Rather than giving a complete survey of what is known for this model, we shall instead give a sample of typical results and refer the reader to the literature for details [Johnson 1973, 1974; Johnson et al. 1974; Graham 1976; Garey & Johnson 1976B].

Given a list L of items, the *first-fit* (FF) algorithm packs the items successively in the order in which they occur in L, always placing each item into the first bin into it will validly fit (i.e., so that the sum of the weights in the bin does not exceed its capacity 1). The number of bins required by the packing is just the time required to execute the jobs using L as a priority list. If instead of choosing the first bin into which an item will fit, we always choose the bin for which the unused capacity is minimized, then the resulting procedure is called the *best-fit* (BF) algorithm. Finally, when L is first ordered by decreasing weights and then first-fit or best-fit packed, the resulting algorithm is called *first-fit decreasing* (FFD) or *best-fit decreasing* (BFD), respectively.

The basic results which apply to these algorithms are the following [Johnson et al. 1974; Garey et al. 1976A]:

$$C_{\max}(\mathrm{FF}) \leqslant \lceil \tfrac{17}{10} C^*_{\max} \rceil;$$

$$C_{\max}(\mathrm{BF}) \leqslant \lceil \tfrac{17}{10} C^*_{\max} \rceil;$$

$$C_{\max}(\mathrm{FFD}) \leqslant \tfrac{11}{9} C^*_{\max} + 4;$$

$$C_{\max}(\mathrm{BFD}) \leqslant \tfrac{11}{9} C^*_{\max} + 4.$$

The only known proofs of the last two inequalities are extremely lengthy. Examples can be given which show that the coefficients $\frac{17}{10}$ and $\frac{11}{9}$ are best possible.

If constraints are made on the resource requirements, i.e., $\underline{r} \leqslant r_{1j} \leqslant \bar{r}$ for all j, then the following results hold:

$$\text{if } \underline{r} \geqslant \tfrac{1}{6}, \quad \text{then} \quad C_{\max}(\mathrm{BFD}) \leqslant C_{\max}(\mathrm{FFD});$$

$$\text{if } \underline{r} \geqslant \tfrac{1}{5}, \quad \text{then} \quad C_{\max}(\mathrm{BFD}) = C_{\max}(\mathrm{FFD});$$

$$\text{if } \bar{r} \leqslant \tfrac{1}{2}, \quad \text{then} \quad C_{\max}(\mathrm{FF})/C^*_{\max} \leqslant 1 + \lfloor \bar{r}^{-1} \rfloor^{-1};$$

$$\text{if } \bar{r} \in (\tfrac{8}{29}, \tfrac{1}{2}], \quad \text{then} \quad C_{\max}(\mathrm{FFD}) \leqslant \tfrac{71}{60} C^*_{\max} + c \quad \text{for some constant } c.$$

For these and a number of similar results, the reader is referred to [Graham 1976].

Krause [Krause 1973] (see also [Krause et al. 1975, 1977]) considers the case

$P\,|res\ 1, p_j = 1|\ C_{\max}$. He proves that

$$(C_{\max}(\mathrm{LS}) - 2)/C^*_{\max} < \tfrac{27}{10} - \tfrac{24}{20}/m,$$

$$(C_{\max}(\mathrm{DMR}) - 1)/C^*_{\max} \leqslant 2 - \frac{2}{m} \quad (m \geqslant 2),$$

and he gives examples for which

$$C_{\max}(\mathrm{LS})/C^*_{\max} \geqslant \tfrac{27}{10} - \tfrac{37}{10}/m.$$

Krause also proves several bounds for the *preemptive* case $P\,|pmtn, res\ 1|\ C_{\max}$, one of which is

$$C_{\max}(\mathrm{DMR})/C^*_{\max} < 3 - \frac{3}{m} \quad (m \geqslant 2).$$

Goyal [Goyal 1977A] studies the case $P\,|res\ 1, prec, p_j = 1|\ C_{\max}$ with the restriction that each resource requirement is either zero or 100%. Thus, two jobs both requiring the use of the resource can never be executed simultaneously. This problem is already NP-hard for $m = 2$ [Coffman 1976]. Goyal proves that

$$C_{\max}(\mathrm{LS})/C^*_{\max} \leqslant 3 - \frac{2}{m}, \qquad (\dagger)$$

$$C_{\max}(\mathrm{CG})/C^*_{\max} \leqslant \tfrac{3}{2} \quad \text{for} \quad m = 2,$$

where in the latter case a priority list is formed according to the CG labelling algorithm described earlier.

4.3. *Nonpreemptive scheduling: general processing times*

4.3.1. $P \,\|\, \sum w_j C_j$

The following generalization of the SPT rule for $1 \,\|\, \sum C_j$ (see Section 3.3.1) solves $P \,\|\, \sum C_j$ 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 $p_1 \leqslant \cdots \leqslant p_n$. 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.

Bruno, Coffman and Sethi [Bruno et al. 1974] consider the algorithm RPT: first apply list scheduling on the basis of largest processing time first (LPT), then *reverse* the order of jobs on each machine, and finally left justify the schedule. RPT has the same behavior as LPT with respect to the $C_{\max}$ criterion (see Section 4.3.5.1); however, it only yields

$$\sum C_j(\mathrm{RPT}) \,/\, \sum C^*_j \leqslant m. \qquad (\dagger)$$

With respect to $P \,\|\, \sum w_j C_j$, similar heuristics are described and tested empirically by Baker and Merten [Baker & Merten 1973].

Eastman, Even and Isaacs [Eastman et al. 1964] show that after renumbering

the jobs according to nonincreasing ratios w_j/p_j

$$\sum w_j C_j(\mathrm{LS}) - \tfrac{1}{2}\sum_{j=1}^{n} w_j p_j \geqslant \frac{1}{m}\left(\sum_{j=1}^{n}\sum_{k=1}^{j} w_j p_k - \tfrac{1}{2}\sum_{j=1}^{n} w_j p_j\right). \qquad (\dagger)$$

It follows from this inequality that

$$\sum w_j C_j^* \geqslant \frac{m+n}{m(n+1)}\sum_{j=1}^{n}\sum_{k=1}^{j} w_j p_k.$$

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 $1\|\sum w_j U_j$ mentioned in Section 3.3.3) with $O(n(n^2k)^{m-1})$ running time for which

$$\sum w_j C_j(A_k) \Big/ \sum w_j C_j^* \leqslant 1 + \frac{1}{k}.$$

For $m=2$, the running time of A_2 can be improved to $O(n^2k)$.

4.3.2. $Q\|\sum C_j$

The algorithm for solving $P\|\sum C_j$ given in the previous section can be generalized to the case of uniform machines [Conway et al. 1967]. If J_j is the kth last job executed on M_i, a cost contribution $kp_{ij} = kq_i p_j$ is incurred. $\sum C_j$ is a weighted sum of the p_j and is minimized by matching the n smallest weights kq_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. $R\|\sum C_j$

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

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

Then the problem is to minimize

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

subject to

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

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

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

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 $P2 \| \sum w_j C_j$ are NP-hard. Hence it seems fruitless to attempt to find polynomial-time optimization algorithms for criteria other than $\sum C_j$. Moreover, $P2 \,|tree|\, \sum C_j$ 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 $C_{\max}$, $L_{\max}$, $\sum w_j C_j$ and $\sum w_j U_j$ optimality criteria, and even to problems with the $\sum w_j T_j$ 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,

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

and in the case of $\sum f_j$ criteria,

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

In both cases, the initial conditions are

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

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 $Q \| \sum w_j C_j$. $P \| \sum w_j U_j$ can be solved in $O(mn(\max_j \{d_j\})^m)$ time.

Still other dynamic programming approaches can be used to solve $P \| \sum f_j$ and $P \| f_{\max}$ in $O(m \min \{3^n, n2^n C\})$ time, but these are probably of little practical importance.

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]:

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

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

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

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 ith 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

$$C_{\max}(\mathrm{MF}_k)/C^*_{\max} \leqslant 1.22+\frac{1}{2^k}.$$

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

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

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

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

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

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

For $m=2$, algorithm A_2 can be improved to run in time $O(n^2k)$. As in the cases of $1 \| \sum w_jU_j$ (Section 3.3.3) and $P \| \sum w_jC_j$ (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

$$C_{\max}(\mathrm{LS})/C^*_{\max} \leqslant 1+(m-1)\max_j\{p_j\}\Big/\sum_j p_j.$$

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

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

The following *local interchange* (LI) algorithm gives a slight improvement over the original $2-1/m$ bound: assign jobs to machines arbitrarily, then move individual jobs and interchange pairs of jobs as long as $C_{\max}$ can be decreased by any such change. It then follows [Graham –] that

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

In [Bruno et al. 1974] the Conway–Maxwell–Miller (CMM) algorithm for solving $P\|\sum C_j$ (see Section 4.3.1) is considered. Let $C^*_{\max}(\mathrm{CMM})$ be the minimum completion time among all schedules that can be generated by CMM. Then

$$C^*_{\max}(\mathrm{CMM})/C_{\max}(\mathrm{LPT}) \leqslant 2-\frac{1}{m}, \qquad (\dagger)$$

$$C^*_{\max}(\mathrm{CMM})/C^*_{\max} \leqslant 2-\frac{1}{m}. \qquad (\dagger)$$

An interesting variation on the $C_{\max}$ criterion arises in the work of Chandra and Wong [Chandra & Wong 1975]. They consider the case $P\|\sum B_i^2$, where B_i denotes the completion time of the job executed last on M_i, and establish the surprisingly good behavior of LPT:

$$\sum B_i^2(\mathrm{LPT}) \Big/ \sum B_i^{2*} \leqslant \tfrac{25}{24}.$$

They also construct examples for which

$$\sum B_i^2(\mathrm{LPT}) \Big/ \sum B_i^{2*} \geqslant \tfrac{37}{36}-\tfrac{1}{36}/m..$$

Finally, we mention the following result [Garey et al. –]. For any LPT schedule, let $t_{\max}$ denote the latest possible time at which a machine can become idle and let $t_{\min}$ denote the earliest time a machine can be idle. Then

$$t_{\max}/t_{\min} \leqslant \frac{4m-2}{3m-1}$$

and this bound is best possible.

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 4.3.5.2–6. 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 $C^*_{\max}(\mathrm{UI})$.

Liu and Liu [Liu & Liu 1974A, 1974B, 1974C] study numerous questions dealing with uniform machines. We outline some of their results.

For the case that $q_1 = \cdots = q_{m-1} = 1$, $q_m = q \geqslant 1$, they prove

$$C_{\max}(\mathrm{LPT})/C^*_{\max}(\mathrm{UI}) \leqslant \begin{cases} \dfrac{2(m-1+q)}{q+2} & \text{for } q \leqslant 2, \\ \dfrac{m-1+q}{2} & \text{for } q > 2. \end{cases}$$

For the general case, 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

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

where all $q_i \geqslant 1$ and

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

This is best possible when the q_i are integers and $\sum_i q_i$ 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.

For the case that $q_1 = \cdots = q_{m-1} = 1$, $q_m \geqslant 1$, they show that

$$C_{\max}(\mathrm{LPT}')/C^*_{\max} \leqslant \begin{cases} \dfrac{1+\sqrt{17}}{4} & \text{for } m = 2, \\ \frac{3}{2} - \dfrac{1}{2m} & \text{for } m > 2. \end{cases} \qquad (\dagger)$$

For the general case, they show

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

Also, examples are given for which $C_{\max}(\mathrm{LPT}')/C^*_{\max}$ approaches $\frac{3}{2}$ 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 six 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 six algorithms A they prove

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

with equality possible for four of the six. For the other two, they conjecture

$$C_{\max}(A)/C^*_{\max} \overset{?}{\leqslant} 2.$$

For the special case $R2 \| C_{\max}$, they give a complicated algorithm G (however, with $O(n \log n)$ running time) such that

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

In a variation on $R \| C_{\max}$, we assume that each J_j has a processing time p_j and a fixed *memory requirement* $|J_j|$ and that each M_i has a *memory capacity* $|M_i|$. We require that $|M_i| \geqslant |J_j|$ in order for M_i to be able to execute J_j, i.e.,

$$p_{ij} = \begin{cases} p_j & \text{if } |M_i| \geqslant |J_j|, \\ \infty & \text{otherwise.} \end{cases}$$

Kafura and Shen [Kafura & Shen 1977] show

$$C_{\max}(\mathrm{LS})/C^*_{\max} \leqslant 1 + \log m.$$

They also note that when m is a power of 2, the bound can be achieved.

Suppose a list is formed in order of decreasing $|J_j|$; this algorithm is denoted by LMF (largest memory first). It can be shown [Kafura & Shen 1977] that

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

A refinement of LMF is LMTF where ties in $|J_j|$ are broken by decreasing order of p_j. In this case,

$$C_{\max}(\mathrm{LMTF})/C^*_{\max} \leqslant \begin{cases} \frac{5}{4} & \text{for} \quad m = 2, \\ 2 - \dfrac{1}{m-1} & \text{for} \quad m \geqslant 3. \end{cases} \qquad (\dagger)$$

Kafura and Shen also give a complicated (but polynomial-time) algorithm 2D for which

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

Other results for this model may be found in [Kafura & Shen 1978].

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.,

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

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 $p'_j \leqslant p_j$, weakened precedence constraints, m' machines and a (possibly different) priority list. Then [Graham 1966]

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

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

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

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

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

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

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

Liu [Liu 1972] shows that

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

Relatively little is known in the way of approximation algorithms for the more special case $P\,|tree|\,C_{\max}$. It is conjectured in [Denning & Scott Graham 1973] that

$$C_{\max}(\mathrm{CP})/C^*_{\max} \overset{?}{\leqslant} 2-\frac{2}{m+1}.$$

If true this would be best possible as examples show. For the special case that the precedence constraints form an *intree*, Kaufman [Kaufman 1974] shows that

$$C_{\max}(\mathrm{CP}) \leqslant C^*_{\max}(pmtn)+\max_j\{p_j\}-\left\lceil\frac{1}{m}\max_j\{p_j\}\right\rceil.$$

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,

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

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

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

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

We mention here two rather special results of Baer [Baer 1974]. He constructs an algorithm B based on the CG labelling algorithm which has the following behavior. For $Q2\,|tree|\,C_{\max}$ with $q_2/q_1 = 3$,

$$C_{\max}(B) \leqslant C^*_{\max} + 1;$$

for $Q2\,|prec|\,C_{\max}$ with $q_2/q_1 = 2$,

$$C_{\max}(B)/C^*_{\max} \leqslant \tfrac{6}{5}.$$

4.3.5.6. $P\,|res, prec|\,C_{\max}$. The most general bound for $P\,|res, prec|\,C_{\max}$ is given in [Garey & Graham 1975]. It states

$$C_{\max}(\mathrm{LS})/C^*_{\max} \leqslant m \qquad (\dagger)$$

and, in fact, examples with $s = 1$ are given which achieve this bound. Thus, the addition of even a single resource in the presence of precedence constraints can have a drastic effect on the worst-case behavior of an arbitrary priority list.

For $P\,|res|\,C_{\max}$, it is shown in [Garey & Graham 1975] that for $m \geqslant 2$

$$C_{\max}(\mathrm{LS})/C^*_{\max} \leqslant \min\left\{\frac{m+1}{2},\ s+2-\frac{2s+1}{m}\right\}. \qquad (\dagger)$$

With the restriction that $m \geqslant n$, $s \geqslant 1$, this can be improved to

$$C_{\max}(\mathrm{LS})/C^*_{\max} \leqslant s+1. \qquad (\dagger)$$

The techniques used to prove this inequality involve an interesting application of Ramsey theory, a branch of combinatorics.

4.4. *Preemptive scheduling*

4.4.1. $P\,|pmtn|\sum C_j$

A theorem of McNaughton [McNaughton 1959] states that for $P\,|pmtn|\sum w_jC_j$ there is no schedule with a finite number of preemptions which yields a smaller criterion value than an optimal nonpreemptive 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 $P\,|pmtn|\sum C_j$. It also follows that $P2\,|pmtn|\sum w_jC_j$ is NP-hard, since $P2\,\|\sum w_jC_j$ is known to be NP-hard.

4.4.2. $Q\,|pmtn|\sum C_j$

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

One can show that there exists an optimal preemptive schedule in which $C_j \leqslant C_k$ 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 $(m-1)\times(n-m/2)$ preemptions. It has been extended to cover the case in which $\sum C_j$ is minimized subject to a common deadline for all jobs [Gonzalez 1977].

4.4.3. $R\,|pmtn|\sum C_j$

Very little is known about $R\,|pmtn|\sum C_j$. We conjecture that the problem is NP-hard. However, 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

$$\max\left\{\max_j\{p_j\}, \frac{1}{m}\sum_j p_j\right\}.$$

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 $l_j(t)$ 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 $l_j(t)$. 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 $m' \geqslant n'$, 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 1977].

Recently, Gonzalez and Johnson [Gonzalez & Johnson 1977] 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

$$C_{\max}(\mathrm{MC})/C^*_{\max} \leqslant 2 - \frac{2}{m} \quad (m \geqslant 2). \tag{†}$$

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 in an optimal schedule with no more than $(m-1)n^2$ preemptions.

A complicated, but computationally 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 nondecreasing 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

$$\max\left\{\max_{1 \leqslant k \leqslant m-1}\left\{\sum_{j=1}^{k} p_j \Big/ \sum_{i=1}^{k} \frac{1}{q_i}\right\}, \sum_{j=1}^{n} p_j \Big/ \sum_{i=1}^{m} \frac{1}{q_i}\right\},$$

where $p_1 \geqslant \cdots \geqslant p_n$ and $q_1 \leqslant \cdots \leqslant q_m$. 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 $Q2\,|pmtn, tree|\,C_{\max}$.

In [Horvath et al. 1977] it is shown that for $Q\,|pmtn, prec|\,C_{\max}$, critical path

scheduling has the bound

$$C_{\max}(\mathrm{CP})/C^*_{\max} \leqslant \left(\frac{3m}{2}\right)^{\frac{1}{2}}$$

and examples are given for which the bound $(m/8)^{\frac{1}{2}}$ 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

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

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

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

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

In this formulation x_{ij} represents the total time spent by J_j on M_i. Given a solution to the linear program, 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 $\frac{7}{2}m^2$ 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. 1978].

We note that a similar linear programming formulation can be given for the minimization of $L_{\max}$ [Lawler & Labetoulle 1978].

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

$P\,|pmtn|\,L_{\max}$ 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 1977].

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. 1978].

4.4.8: $Q\,|pmtn, r_j|\,L_{\max}$

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 1977A]. More generally, $Q\,|pmtn, r_j|\,C_{\max}$ and, by symmetry, $Q\,|pmtn|\,L_{\max}$ are solvable in $O(n^2)$ time; the number of preemptions generated is $O(n^2)$ [Sahni & Cho 1977B; Labetoulle et al. 1978].

The 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. 1978].

It appears not unlikely that the Gonzalez–Johnson algorithm for $P\,|pmtn, tree|\,C_{\max}$ and the above mentioned algorithm for $Q\,|pmtn, r_j|\,C_{\max}$ allow a common generalization that will make $Q\,|pmtn, tree|C_{\max}$ solvable in 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.

An obvious extension of this type of problem involves machines which can process more than one job at the same time. The resulting *resource constrained project scheduling* problems are extremely hard to solve. We refer to surveys by Davis [Davis 1966, 1973] that contain an extensive bibliography.

We shall be dealing exclusively with the $C_{\max}$ criterion. Other optimality criteria lead usually to NP-hard problems, even for $m = 2$ [Garey et al. 1976B; Lenstra et al. 1977]; a notable exception is $O2\,\|\,\sum C_j$, which is open. Only a few enumerative algorithms for problems involving criteria other than $C_{\max}$ have been developed, e.g., for $F2\,\|\,\sum C_j$ [Ignall & Schrage 1965], $F\,\|\,\sum w_j C_j$ [Townsend 1977A], $F\,\|\,L_{\max}$ [Townsend 1977B], and $J\,\|\,\sum w_j T_j$ [Fisher 1973].

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 $A = \{J_j \mid a_j \geqslant b_j\}$, $B = \{J_j \mid a_j < b_j\}$.

Now choose J_r and J_l to be any two distinct jobs (whether in A or B) such that

$$a_r \geqslant \max_{J_j \in A} \{b_j\},$$

$$b_l \geqslant \max_{J_j \in B} \{a_j\}.$$

Let $A' = A - \{J_r, J_l\}$, $B' = B - \{J_r, J_l\}$. We assert that it is possible to form feasible schedules for $B' \cup \{J_l\}$ and for $A' \cup \{J_r\}$ as indicated in Fig. 5.1, 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 $T_1 = \sum_j a_j$, $T_2 = \sum_j b_j$. Suppose $T_1 - a_l \geqslant T_2 - b_r$ (the case $T_1 - a_l < T_2 - b_r$ being symmetric). We then combine the two schedules as shown in Fig. 5.2, pushing the jobs in $B' \cup \{J_l\}$ 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) $a_r \leqslant T_2 - b_r$. The resulting schedule is as in Fig. 5.3. The length of the schedule is $\max\{T_1, T_2\}$.

(2) $a_r > T_2 - b_r$. The resulting schedule is as in Fig. 5.4. The length of the schedule is $\max\{T_1, a_r + b_r\}$.

For any feasible schedule we obviously have that

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

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

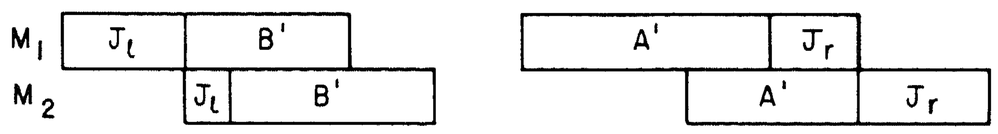

Fig. 5.1.

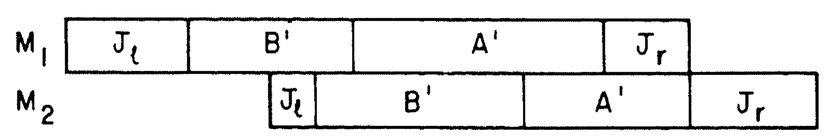

Fig. 5.2.

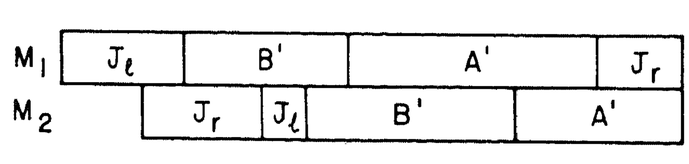

Fig. 5.3.

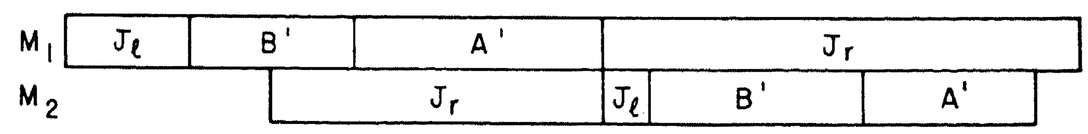

Fig. 5.4.

There is a little hope of finding polynomial-time algorithms for nonpreemptive open shop problems more complicated than $O2 \| C_{\max}$. The case $O3 \| C_{\max}$ 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 [Lenstra –].

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]. We already had occasion to refer to this result in Section 4.4.6. An outline of the algorithm, adapted from [Lawler & Labetoulle 1978], follows below.

Let $P = (p_{ij})$ be the matrix of processing times and

$$C = \max \left\{ \max_j \left\{ \sum_i p_{ij} \right\}, \max_i \left\{ \sum_i p_{ij} \right\} \right\}.$$

Call row i (column j) of P *tight* if $\sum_j p_{ij} = C$ ($\sum_i p_{ij} = C$), *slack* otherwise.

We clearly have $C^*_{\max} \geqslant C$. It is possible to construct a feasible schedule for which $C_{\max} = C$. Hence this schedule will be optimal.

Suppose we can find a subset S of strictly positive elements of P, with exactly one element of S in each tight row and in each tight column, and at most one element of S in each slack row and in each slack column. We shall call such a subset a *decrementing set*, and use it to construct a *partial schedule* of length δ, for some $\delta > 0$. The constraints on the choice of δ are as follows.

(1) If $p_{ij} \in S$ and either row i or column j is tight, then $\delta \leqslant p_{ij}$.

(2) If $p_{ij} \in S$ and row i (column j) is slack, then $\delta \leqslant p_{ij} + C - \sum_k p_{ik}$ ($\delta \leqslant p_{ij} + C - \sum_k p_{kj}$).

(3) If row i (column j) contains no element in S (and is therefore necessarily slack), then $\delta \leqslant C - \sum_k p_{ik}$ ($\delta \leqslant C - \sum_k p_{kj}$).

For a given decrementing set S, let δ be the maximum subject to (1), (2), (3). Then the partial schedule constructed is such that for each $p_{ij} \in S$, M_i processes J_j for $\min \{p_{ij}, \delta\}$ units of time.

We then obtain the matrix P' from P by replacing each $p_{ij} \in S$ by $\max \{0, p_{ij} - \delta\}$, and repeat the procedure until after a finite number of times $P' = (0)$. Joining together the partial schedules obtained for successive decrementing sets then yields an optimal preemptive schedule for P.

By suitably embedding P in a doubly stochastic matrix and appealing to the Birkhoff-Von Neumann theorem, it can be shown that a decrementing set can be found by solving a linear assignment problem; see [Lawler & Labetoulle 1978] for details.

Other network formulations of the problem are possible. An analysis of various possible computations reveals that $O | pmtn | C_{\max}$ can be solved in $O(r + \min \{m^4, n^4, r^2\})$ time, where r is the number of nonzero elements in P [Gonzalez 1976].

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 $\min\{p_{1j}, p_{2k}\} \leqslant \min\{p_{2j}, p_{1k}\}$. It follows that the problem can be solved in $O(n \log n)$ time: arrange first the jobs with $p_{1j} \leqslant p_{2j}$ 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

$$l_j = \min\{l_{1j} - p_{1j}, l_{2j} - p_{2j}\}$$

and applying Johnson's algorithm to processing times $(p_{1j} + l_j, p_{2j} + l_j)$ 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 −].

Similarly, some $F3\,\|\,C_{\max}$ problems can be solved by applying Johnson's algorithm to processing times $(p_{1j} + p_{2j}, p_{2j} + p_{3j})$, e.g., if there exists a $\theta \in [0, 1]$ such that

$$\theta p_{1j} + (1-\theta) p_{3j} \geqslant p_{2k} \quad \text{for all} \quad (j, k)$$

[Johnson 1954; Burns & Rooker 1976] or if M_2 can process any number of jobs at the same time [Conway et al. 1967]. We refer to [Monma 1977] for further generalizations.

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. 1976B; Lenstra et al. 1977].

It should be noted that an interpretation of precedence constraints which differs from our definition is possible. If $J_j <' J_k$ 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 1977]. 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 1977]. It is an open question whether $F2\,|prec'|\,C_{\max}$ is NP-hard.

Gonzalez and Sahni [Gonzalez & Sahni 1978A] consider the case of preemptive flow shop scheduling. They show that preemptions on M_1 and M_m can be removed without increasing $C_{\max}$. Hence, Johnson's algorithm solves $F2\,|pmtn|\,C_{\max}$ as well. $F3\,|pmtn|\,C_{\max}$ turns 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.

Except for some rather simple worst-case results for heuristics, obtained by Gonzalez and Sahni [Gonzalez & Sahni 1978A], that are to be mentioned in Section 5.4.2, all research in this area has focused on enumerative methods.

The usual enumeration scheme is to assign jobs to the lth position in the schedule at the lth level of the search tree. Thus, at a node at that level a partial schedule $(J_{\sigma(1)}, \ldots, J_{\sigma(l)})$ has been formed and the jobs with index set $S = \{1, \ldots, n\} - \{\sigma(1), \ldots, \sigma(l)\}$ are candidates for the $(l+1)$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. 1978B].

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 $(1 \leqslant u \leqslant v \leqslant m)$. We then obtain a problem of scheduling $\{J_j \mid j \in S\}$ on five machines $N_{\cdot u}$, 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 $J_{\sigma(l)}$ on M_i. $N_{\cdot u}$, N_{uv} and N_v. have infinite capacity; the processing times on these machines are defined by

$$q_{\cdot uj} = \max_{1 \leqslant i \leqslant u} \left\{ C(\sigma, i) + \sum_{h=i}^{u-1} p_{hj} \right\},$$

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

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

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

Any of the machines $N_{\cdot u}$, 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 $N_{\cdot u}$ and N_u. from the problem produces the *machine-based bound* used in [Ignall & Schrage 1965; McMahon 1971]:

$$\max_{1 \leqslant u \leqslant m} \left\{ \min_{j \in S} \{q_{\cdot uj}\} + \sum_{j \in S} p_{uj} + \min_{j \in S} \{q_{u \cdot j}\} \right\}.$$

Removing only $N_{\cdot u}$ results in a $1 \parallel L_{\max}$ 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 $N_{\cdot u}$, N_{uv} and $N_{v\cdot}$ 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.

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 $(J_{\sigma(1)}, \ldots, J_{\sigma(l)}, J_j)$ can be eliminated because a schedule at least as good exists among the completions of $(J_{\sigma(1)}, \ldots, J_{\sigma(l)}, J_k, J_j)$. 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 lth 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. 1978B]. It may be worthwile to consider further extensions that, for instance, involve the processing times of the unscheduled jobs [Gupta & Reddi 1978; Szwarc 1978].

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

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

where $p_{i0} = 0$ $(i = 1, \ldots, m)$ [Piehler 1960; Reddi & Ramamoorthy 1972; Wismer 1972]. We refer to [Lenstra & Rinnooy Kan 1975] for an extension of this formulation to certain job shop systems and to [Van Deman & Baker 1974] for a branch-and-bound approach to $F |no\ wait| \sum C_j$.

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]. Both $F |no\ wait| C_{\max}$

and $F|no\ wait|\sum C_j$ are unary NP-hard [Lenstra et al. 1977], and the same is true for $O2|no\ wait|C_{max}$ and $J2|no\ wait|C_{max}$ [Sahni & Cho 1977C]. In spite of challenging prizes awarded for their solution [Lenstra et al. 1977], $F3|no\ wait|C_{max}$ and $F2|no\ wait|\sum C_j$ are still open.

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

$$C^*_{max}(no\ wait)/C^*_{max} < m \quad \text{for} \quad m \geqslant 2. \tag{†}$$

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 $J2|m_j \leqslant 2|C_{max}$ in $O(n \log n)$ time [Jackson 1956]. Let $\mathcal{J}_i$ be the set of jobs with operations on M_i only $(i = 1, 2)$ and $\mathcal{J}_{hi}$ 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 $(\mathcal{J}_{12}, \mathcal{J}_1, \mathcal{J}_{21})$ and on M_2 in the order $(\mathcal{J}_{21}, \mathcal{J}_2, \mathcal{J}_{12})$.

This, however, is probably as far as we can get. Unary NP-hardness of $J2\|C_{max}$ results as soon as we allow one job to have more than two operations [Garey et al. 1976B; Lenstra et al. 1977]. In fact, $J2|1 \leqslant p_{ij} \leqslant 2|C_{max}$ and $J3|p_{ij} = 1|C_{max}$ are already NP-hard [Lenstra & Rinnooy Kan 1978B].

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 $u = \sum_{k=1}^{j-1} m_k + i$ 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 Fig. 5.5 and 5.6 for examples.

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

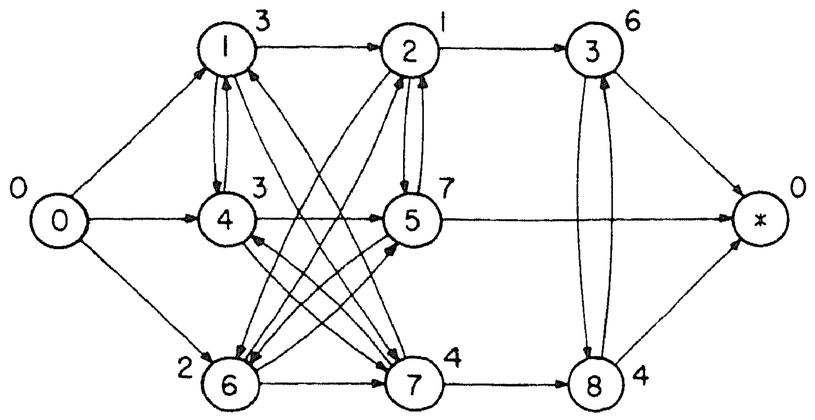

Fig. 5.5. Job shop problem, represented as a disjunctive graph.

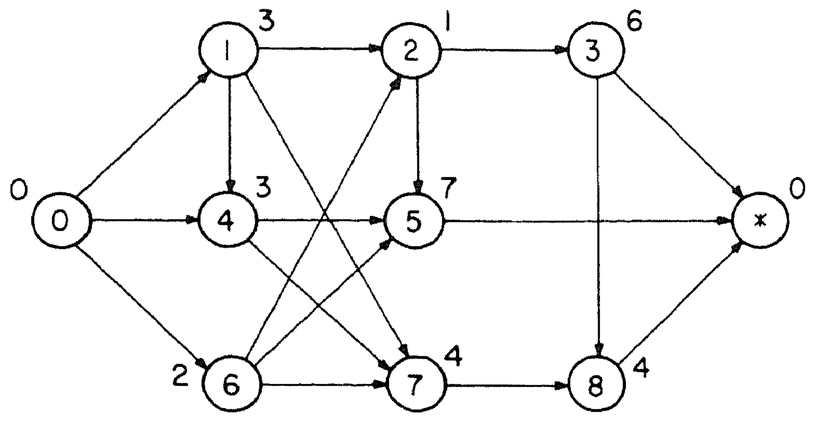

Fig. 5.6. Job shop schedule, represented as an acyclic directed graph.

to u and from u to $*$, excluding p_u, be denoted by r_u ang 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, 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].

The applicability of Lagrangean techniques to obtain stronger lower bounds is the subject of ongoing research. Either the precedence constraints fixing the machine orders for the jobs or the capacity constraints of the machines can be multiplied by a Lagrangean variable and added to the objective function. For fixed values of the multipliers, the resulting problems can be solved in (pseudo)polynomial time. Computational experiments will have to reveal if this approach, combined with subgradient optimization or another suitable technique, will lead to any substantially better job shop algorithm.

As far as approximation algorithms are concerned, 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.

Not much has been done in the way of worst-case analysis of approximation algorithms for flow shop and job shop problems. Gonzalez and Sahni [Gonzalez & Sahni 1978A] show that for any active flow shop or job shop schedule (AS)

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

This bound is tight even for LPT schedules, in which the jobs are ordered according to nonincreasing sums of processing times. They give an $O(mn \log n)$ algorithm H for $F \| C_{\max}$ based on Johnson's Algorithm for $F2 \| C_{\max}$ with

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

With SPT defined similarly as LPT, it is also shown that for $F \| \sum C_j$ and $J \| \sum C_j$

$$\sum C_j(\mathrm{AS}) \Big/ \sum C_j^* \leqslant n, \qquad (\dagger)$$

$$\sum C_j(\mathrm{SPT}) \Big/ \sum C_j^* \leqslant m. \qquad (\dagger)$$

It thus appears that, in general, the obvious algorithms can deviate quite substantially from the optimum for this class of problems.

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. 1978A]. 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, although for any practical problem it probably will remain true that a successful heuristic will have to exploit whatever special structure and properties the problem at hand may have.

Thus, the area of deterministic scheduling theory appears as one of the more fruitful interfaces between computer science and operations research. Much progress has been made and more can be expected in the near future.

Note. The last three authors are currently engaged in writing a book on scheduling problems and would very much appreciate being informed about new algorithmic and complexity results in this area.

Acknowledgments

The authors gratefully acknowledge the useful comments from M.L. Fisher. The research by the last three authors was supported by NSF Grant MCS76–17605 and by NATO Special Research Grant 9.2.02 (SRG.7).

References

D. Adolphson (1977), Single machine job sequencing with precedence constraints, SIAM J. Comput. 6, 40–54.

D. Adolphson and T.C. Hu (1973), Optimal linear ordering, SIAM J. Appl. Math. 25, 403–423.

A.V. Aho, M.R. Garey and J.D. Ullman (1972), The transitive reduction of a directed graph, SIAM J. Comput. 1, 131–137.

J.L. Baer (1974), Optimal scheduling on two processors with different speeds, in: E. Gelenbe and R. Mahl, eds., Computer Architectures and Networks (North-Holland, Amsterdam, 1974) 27–45.

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

K.R. Baker and A.G. Merten (1973), Scheduling with parallel processors and linear delay costs, Naval Res. Logist. Quart. 20, 793–804.

K.R. Baker and L.E. Schrage (1978), Finding an optimal sequence by dynamic programming: an extension to precedence-related tasks, Operations Res. 26, 111–120.

K.R. Baker and 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 and S.R. Arora (1969), The sequencing problem, Management Sci. 16, B247–263.

J.W. Barnes and J.J. Brennan (1977), An improved algorithm for scheduling jobs on identical machines, AIIE Trans. 9, 25–31.

P. Bratley, M. Florian and 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 and D.S. Johnson (1977), Scheduling equal-length tasks under treelike precedence constraints to minimize maximum lateness, Math. Operations Res. 2, 275–284.

J. Bruno, E.G. Coffman, Jr. and R. Sethi (1974), Scheduling independent tasks to reduce mean finishing time, Comm. ACM 17, 382–387.

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

F. Burns and J. Rooker (1976), $3 \times n$ Flow-shops with convex external stage time dominance (unpublished manuscript).

A.K. Chandra and C.K. Wong (1975), Worst-case analysis of a placement algorithm related to storage allocation, SIAM J. Comput. 4, 249–263.

J.M. Charlton and C.C. Death (1970), A generalized machine scheduling algorithm, Operational 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 and C.L. Liu (1975), On a class of scheduling algorithms for multiprocessors computing systems, in: T.–Y. Feng, ed., Parallel Processing, Lecture Notes in Computer Science 24 (Springer, Berlin, 1975) 1–16.

E.G. Coffman, Jr. (ed.) (1976), Computer and Job-Shop Scheduling Theory (Wiley, New York).

E.G. Coffman, Jr., M.F. Garey and D.S. Johnson (1978), An application of bin-packing to multiprocessor scheduling, SIAM J. Comput., to appear.

E.G. Coffman, Jr. and R.L. Graham (1972), Optimal scheduling for two-processor systems, Acta Informat. 1, 200–213.

R.W. Conway, W.L. Maxwell and L.W. Miller (1967), Theory of Scheduling (Addison–Wesley, Reading, MA).

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 and M.P. Hottenstein (1970), Review of scheduling research, Naval Res. Logist. Quart. 17, 11–39.

P.J. Denning and G. Scott Graham (1973), A note on subexpression ordering in the execution of arithmetic expressions, Comm. ACM 16, 700–702.

W.L. Eastman, S. Even and I.M. Isaacs (1964), Bounds for the optimal scheduling of n jobs on m processors, Management Sci. 11, 268–279.

S.E. Elmaghraby and S.H. Park (1974), Scheduling jobs on a number of identical machines, AIIE Trans. 6, 1–12.

M.L. Fisher (1973), Optimal solution of scheduling problems using Lagrange multipliers, part I, Operations Res. 21, 1114–1127.

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

M. Fujii, T. Kasami and K. Ninomiya (1969, 1971), Optimal sequencing of two equivalent processors, SIAM J. Appl. Math. 17, 784–789; Erratum. 20, 141.

M.R. Garey (–), Unpublished.

M.R. Garey and R.L. Graham (1975), Bounds for multiprocessor scheduling with resource constraints, SIAM J. Comput. 4, 187–200.

M.R. Garey, R.L. Graham and D.S. Johnson (1978), Performance guarantees for scheduling algorithms, Operations Res. 26, 3–21.

M.R. Garey, R.L. Graham and D.S. Johnson (–), Unpublished.

M.R. Garey, R.L. Graham, D.S. Johnson and A.C.–C. Yao (1976A), Resource constrained scheduling as generalized bin packing, J. Combinatorial Theory Ser. A 21, 257–298.

M.R. Garey and D.S. Johnson (1975), Complexity results for multiprocessor scheduling under resource constraints, SIAM J. Comput. 4, 397–411.

M.R. Garey and D.S. Johnson (1976A), Scheduling tasks with nonuniform deadlines on two processors, J. Assoc. Comput. Mach. 23, 461–467.

M.R. Garey and D.S. Johnson (1976B), Approximation algorithms for combinatorial problems: an annotated bibliography, in: J.F. Traub, ed., Algorithms and Complexity: New Directions and Recent Results (Academic Press, New York 1976) 41–52.

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

M.R. Garey, D.S. Johnson and R. Sethi (1976B), The complexity of flowshop and jobshop scheduling, Math. Operations Res. 1, 117–129.

L. Gelders and P. R. Kleindorfer (1974), Coordinating aggregate and detailed scheduling decisions in the one-machine job shop: part I, Theory, Operations Res. 22, 46–60.

L. Gelders and P.R. Kleindorfer (1975), Coordinating aggregate and detailed scheduling in the one-machine job shop: II — computation and structure, Operations Res. 23, 312–324.

W.S. Gere (1966), Heuristics in job shop scheduling, Management Sci. 13, 167–190.

B. Giffler and G.L. Thompson (1960), Algorithms for solving production-scheduling problems, Operations Res. 8, 487–503.

P.C. Gilmore and R.E. Gomory (1964), Sequencing a one-state variable machine: a solvable case of the traveling salesman problem, Operations Res. 12, 655–679.

T. Gonzalez (1976), A note on open shop preemptive schedules, Technical Report 214, Computer Science Department, Pennsylvania State University.

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

T. Gonzalez, O.H. Ibarra and S. Sahni (1977), Bounds for LPT schedules on uniform processors, SIAM J. Comput. 6, 155–166.

T. Gonzalez and D.B. Johnson (1977), A new algorithm for preemptive scheduling of trees, Technical Report 222, Computer Science Department, Pennsylvania State University.

T. Gonzalez, E.L. Lawler and S. Sahni (1978), Optimal preemptive scheduling of a fixed number of unrelated processors in linear time (to appear).

T. Gonzalez and S. Sahni (1976), Open shop scheduling to minimize finish time, J. Assoc. Comput. Mach. 23, 665–679.

T. Gonzalez and S. Sahni (1978A), Flowshop and jobshop schedules: complexity and approximation, Operations Res. 26, 36–52.

T. Gonzalez and S. Sahni (1978B), Preemptive scheduling of uniform processor systems, J. Assoc. Comput. Mach. 25, 92–101.

D.K. Goyal (1977A), Scheduling equal execution time tasks under unit resource restriction (to appear).

D.K. Goyal (1977B), 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 (1976), Bounds on the performance of scheduling algorithms, in: [Coffman 1976], 165–227.

R.L. Graham (–), Unpublished.

J.N.D. Gupta and S.S. Reddi (1978), Improved dominance conditions for the three-machine flowshop scheduling problem, Operations Res. 26, 200–203.

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, Operations Res. 21, 846–847.

W.A. Horn (1974), Some simple scheduling algorithms, Naval Res. Logist. Quart. 21, 177–185.

E. Horowitz and S. Sahni (1976), Exact and approximate algorithms for scheduling nonidentical processors, J. Assoc. Comput. Mach. 23, 317–327.

E.C. Horvath, S. Lam and 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, Operations Res. 9, 841–848.

O.H. Ibarra and C.E. Kim (1975), Scheduling for maximum profit, Technical Report, Computer Science Department, University of Minnesota, Minneapolis.

O.H. Ibarra and 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 and C.E. Kim (1977), Heuristic algorithms for scheduling independent tasks on nonidentical processors, J. Assoc. Comput. Mach. 24, 280–289.

E. Ignall and L. Schrage (1965), Application of the branch-and-bound technique to some flow-shop scheduling problems, Operations 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.

D.S. Johnson (1973), Near-optimal bin packing algorithms, Report MAC TR–109, Massachusetts Institute of Technology, Cambridge, MA.

D.S. Johnson (1974), Fast algorithms for bin packing, J. Comput. System Sci. 8, 272–314.

D.S. Johnson, A. Demers, J.D. Ullman, M.R. Garey and R.L. Graham (1974), Worst-case performance bounds for simple one-dimensional packing algorithms, SIAM J. Comput. 3, 299–325.

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.

D.G. Kafura and V.Y. Shen (1977), Task scheduling on a multiprocessor system with independent memories, SIAM J. Comput. 6, 167–187.

D.G. Kafura and V. Y. Shen (1978), An algorithm to design the memory configuration of a computer network, J. Assoc. Comput. Mach. 25, 365–377.

R.M. Karp (1972), Reducibility among combinatorial problems, in: R.E. Miller and J.W. Thatcher, eds., Complexity of Computer Computations (Plenum Press, New York, 1972) 85–103.

R.M. Karp (1975), On the computational complexity of combinatorial problems, Networks 5, 45–68.

M.T. Kaufman (1972), Anomalies in scheduling unit-time tasks, Technical Report 34, Stanford Electronic Laboratory.

M.T. Kaufman (1974), An almost-optimal algorithm for the assembly line scheduling problem, IEEE Trans. Computers C–23, 1169–1174.

H. Kise, T. Ibaraki and H. Mine (1978), A solvable case of the one-machine scheduling problem with ready and due times, Operations Res. 26, 121-126.

D. Knuth (1973), Private communication to T.C. Hu, July 23 (1973).

K.L. Krause (1973), Analysis of computer scheduling with memory constraints, Doctoral Dissertation, Computer Science Department, Purdue University, West Lafayette.

K.L. Krause, V.Y. Shen and H.D. Schwetman (1975, 1977), Analysis of several task-scheduling algorithms for a model of multiprogramming computer systems, J. Assoc. Comput. Mach. 22, 522–550; 24, 527.

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

J. Labetoulle, E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan (1978), Preemptive scheduling of uniform machines subject to release dates (to appear).

B.J. Lageweg, E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan (1978A), Computer aided complexity classification of deterministic scheduling problems (to appear).

B.J. Lageweg, J.K. Lenstra and A.H.G. Rinnooy Kan (1976), Minimizing maximum lateness on one machine: computational experience and some applications, Statistica Neerlandica 30, 25–41.

B.J. Lageweg, J.K. Lenstra and A.H.G. Rinnooy Kan (1977), Job-shop scheduling by implicit enumeration, anagement Sci. 24, 441–450.

B.J. Lageweg, J.K. Lenstra and A.H.G. Rinnooy Kan (1978B), A general bounding scheme for the permutation flow-shop problem, Operations 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, Rev. Française Automat. Informat. Recherche Opérationnelle 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 and J. Labetoulle (1978). On preemptive scheduling of unrelated parallel processors, J. Assoc. Comput. Mach., to appear.

E.L. Lawler and 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 Tract 69, Mathematisch Centrum, Amsterdam.

J.K. Lenstra (–), Unpublished.

J.K. Lenstra and A.H.G. Rinnooy Kan (1975), Some simple applications of the travelling salesman problem, Operational Res. Quart. 26, 717–733.

J.K. Lenstra and A.H.G. Rinnooy Kan (1978A), Complexity of scheduling under precedence constraints, Operations Res. 26, 22–35.

J.K. Lenstra and A.H.G. Rinnooy Kan (1978B), Computational complexity of discrete optimization problems, Ann. Discrete Math., 4 (1979), Discrete Optimization I.

J.K. Lenstra, A.H.G. Rinnooy Kan and P. Brucker (1977), Complexity of machine scheduling problems, Ann. Discrete Math, 4, 281–300.

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 and C.L. Liu (1974A), Bounds on scheduling algorithms for heterogeneous computing systems, in: J.L. Rosenfeld, ed., Information Processing 74 (North-Holland, Amsterdam, 1974) 349–353.

J.W.S. Liu and C.L. Liu (1974B), Bounds on schedulling 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 and C.L. Liu (1974C), Performance analysis of heterogeneous multiprocessor computing systems, in: E. Gelenbe and R. Mahl, eds., Computer Architectures and Networks (North-Holland, Amsterdam, 1974) 331–343.

G.B. McMahon (1969), Optimal production schedules for flow shops, Canad. Operational Res. Soc. J. 7, 141–151.

G.B. McMahon (1971), A study of algorithms for industrial scheduling problems, Thesis, University of New South Wales, Kensington.

G.B. McMahon and M. Florian (1975), On scheduling with ready times and due dates to minimize maximum lateness, Operations 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 (1977), Optimal $m \times n$ flow shop sequencing with precedence constraints and lag times, School of Operations Research, Cornell University, Ithaca, NY.

C.L. Monma and J.B. Sidney (1977), A general algorithm for optimal job sequencing with series-parallel precedence constraints, Technical Report 347, School of Operations Research, Cornell University, Ithaca, NY.

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 and E.G. Coffman, Jr. (1969), Optimal preemptive scheduling on two-processor systems, IEEE Trans. Computers C–18, 1014–1020.
R.R. Muntz and E.G. Coffman, Jr. (1970), Preemptive scheduling of real time tasks on multiprocessor systems, J. Assoc. Comput. Mach. 17, 324–338.
Y. Muraoka (1971), Parallelism, exposure and exploitation in programs, Ph.D. Thesis, Department of Computer Science, University of Illinois at Urbana–Champaign.
J.F. Muth and G.L. Thompson (eds.) (1963), Industrial Scheduling (Prentice–Hall, Englewood Cliffs, NJ) 236.
I. Nabeshima (1963), Sequencing on two machines with start lag and stop lag, J. Operations Res. Soc. Japan 5, 97–101.
S.S. Panwalkar and W. Iskander (1977), A survey of scheduling rules, Operations Res. 25, 45–61.
J. Piehler (1960), Ein Beitrag zum Reihenfolgeproblem, Unternehmensforschung 4, 138–142.
S.S. Reddi and C.V. Ramamoorthy (1972), On the flow-shop sequencing problem with no wait in process, Operational Res. Quart. 23, 323–331.
A.H.G. Rinnooy Kan (1976), Machine Scheduling Problems: Classification, Complexity and Computations (Nijhoff, The Hague).
A.H.G. Rinnooy Kan, B.J. Lageweg and J.K. Lenstra (1975), Minimizing total costs in one-machine scheduling, Operations Res. 23, 908–927.
P. Rosenfeld (–), Unpublished.
M.H. Rothkopf (1966), Scheduling independent tasks on parallel processors, Management Sci. 12, 437–447.
B. Roy and 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 and Y. Cho (1977A), Scheduling independent tasks with due times on a uniform processor system, Computer Science Department, University of Minneapolis.
S. Sahni and Y. Cho (1977B), Nearly on line scheduling of a uniform processor system with release times, Computer Science Department, University of Minnesota, Minneapolis.
S. Sahni and Y. Cho (1977C), Complexity of scheduling jobs with no wait in process, Technical Report 77–20, Computer Science Department, University of Minnesota, Minneapolis.
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. Operations Res., 2, 320–330.
J.B. Sidney (1973), An extension of Moore's due date algorithm, in: S.E. Elmaghraby, ed., Symposium on the Theory of Scheduling and its Applications, Lecture Notes in Economics and Mathematical Systems 86 (Springer, Berlin, 1973) 393–398.
J.B. Sidney (1975), Decomposition algorithms for single-machine sequencing with precedence relations and deferral costs, Operations Res. 23, 283–298.
J.B. Sidney (1977), The two-machine maximum flow time problem with series-parallel precedence constraints, Faculty of Management Sciences, University of Ottawa.
W.E. Smith (1956), Various optimizers for single-stage production, Naval Res. Logist. Quart. 3, 59–66.
H.E. Stern (1976), Minimizing makespan for independent jobs on nonidentical 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, Operations Res. 21, 1250–1259.
W. Szwarc (1978), Dominance conditions for the three machine flow-shop problem, Operations Res. 26, 203–206.
W. Townsend (1977A), A branch-and-bound method for sequencing problems with linear and exponential penalty functions, Operational Res. Quart. 28, 191–200.
W. Townsend (1977B), Sequencing n jobs on m machines to minimise maximum tardiness: a branch-and-bound solution, Management Sci. 23, 1016–1019.

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.
J.M. Van Deman and K.R. Baker (1974), Minimizing mean flowtime in the flow shop with no intermediate queues, AIIE Trans. 6, 28–34.
D.A. Wismer (1972), Solution of the flowshop-scheduling problem with no intermediate queues, Operations Res. 20, 689–697.