S.C. Graves et al., Eds., *Handbooks in OR & MS, Vol. 4*


Chapter 9

# Sequencing and Scheduling: Algorithms and Complexity

*Eugene L. Lawler*
*University of California, Berkeley, CA, U.S.A.*

*Jan Karel Lenstra*
*Eindhoven University of Technology, Eindhoven, The Netherlands and CWI, Amsterdam, The Netherlands*

*Alexander H.G. Rinnooy Kan*
*Erasmus University, Rotterdam, The Netherlands*

*David B. Shmoys*
*Cornell University, Ithaca, NY, U.S.A.*

Sequencing and scheduling as a research area is motivated by questions that arise in production planning, in computer control, and generally in all situations in which scarce resources have to be allocated to activities over time. In this survey, we concentrate on the area of deterministic machine scheduling. We review complexity results and optimization and approximation algorithms for problems involving a single machine, parallel machines, open shops, flow shops and job shops. We also pay attention to two extensions of this area: resource-constrained project scheduling and stochastic machine scheduling.

## PART I. PRELIMINARIES

Sequencing and scheduling is concerned with the *optimal allocation of scarce resources to activities over time*. Of obvious practical importance, it has been the subject of extensive research since the early 1950's, and an impressive amount of literature has been created. Any discussion of the available material has to be selective. We will concentrate on the area of deterministic machine scheduling. We will also pay attention to two extensions of this area that are of particular interest in the context of production planning, namely resource-constrained project scheduling and stochastic machine scheduling.

The chapter is organized as follows. Part I gives a brief overview of the many

We also have to exclude a number of other areas, each of which would be worth a survey of its own: periodic scheduling, cyclic scheduling, scheduling with fixed starting times, and scheduling with sequence-dependent processing times. The latter area is closely related to the traveling salesman problem and its extensions.

General references on sequencing and scheduling are the classic book by Conway, Maxwell & Miller [1967], the introductory textbooks by Baker [1974] and French [1982], the expository articles collected by Coffman [1976], and the proceedings volume edited by Dempster, Lenstra & Rinnooy Kan [1982]. There are several survey papers that complement the present chapter. We mention the review of the broad area of production planning by Graves [1981], the introductory survey of precedence-constrained scheduling by Lawler & Lenstra [1982], the tutorial on machine scheduling by Lawler [1983], the NP-completeness column on multiprocessor scheduling by Johnson [1983], the annotated bibliography covering the period 1981–1984 by Lenstra & Rinnooy Kan [1985], the discussions of new directions in scheduling by Lenstra & Rinnooy Kan [1984], Blazewicz [1987] and Blazewicz, Finke, Haupt & Schmidt [1988], and the recent overviews of single-machine scheduling by Gupta & Kyparisis [1987] and of multiprocessor and flow shop scheduling by Kawaguchi & Kyan [1988].

References on resource-constrained project scheduling and stochastic scheduling will be given in Sections 15 and 16. For the scheduling areas that are not covered in this chapter, we refer to the bibliography by Lenstra & Rinnooy Kan [1985]. In addition, we mention the survey of due date determination rules by Cheng & Gupta [1989], the reviews on scheduling with nonregular criteria by Raghavachari [1988] and Baker & Scudder [1990], the results in that area by Garey, Tarjan & Wilfong [1988], the survey on bicriterion single-machine scheduling by Dileepan & Sen [1988], and the book on the traveling salesman problem edited by the present authors [Lawler, Lenstra, Rinnooy Kan & Shmoys, 1985].

## 2. Algorithms and complexity

Practical experience makes it clear that some computational problems are easier to solve than others. For some scheduling problems, algorithms have been known for decades that are capable of solving instances with thousands of jobs, whereas for other problems, the best algorithms strain to cope with only a handful of jobs. Complexity theory provides a mathematical framework in which computational problems can be studied so that they can be classified as 'easy' or 'hard'. In this section, we will review the main points of this theory. The reader is referred to the survey articles by Karp [1975], Lenstra & Rinnooy Kan [1979], Shmoys & Tardos [1993], and Stockmeyer [1992], and to the textbook by Garey & Johnson [1979] for a more extensive treatment of this subject.

An NP-*complete* problem is, roughly speaking, a hardest problem in NP, in that if it would be solvable in polynomial time, then each problem in NP would be solvable in polynomial time, so that P would be equal to NP. Thus, the NP-completeness of a particular problem is strong evidence that a polynomial-time algorithm for its solution is unlikely to exist. The principal notion in defining NP-completeness is that of a *reduction*. For two decision problems $P$ and $Q$, we say that $P$ reduces to $Q$ (denoted $P \propto Q$) if there exists a polynomial-time computable function $\tau$ that transforms inputs for $P$ into inputs for $Q$ such that $x$ is a 'yes' input for $P$ if and only if $\tau(x)$ is a 'yes' input for $Q$. A problem is NP-complete if it is in NP and every problem in NP reduces to it. An optimization problem will be called NP-*hard* if the associated decision problem is NP-complete.

Cook showed that a natural problem from logic is NP-complete by exhibiting a 'master reduction' from each problem in NP to it. Given one NP-complete problem $P$, it is a much easier task to prove the NP-completeness of the next one, say $Q$: one need only prove that $Q \in \text{NP}$ and that $P \propto Q$. The *clique* problem is the following problem from graph theory: given a graph $G = (V, E)$ and an integer $k$, does there exist a set of vertices $C \subset V$ such that $|C| = k$ and for each distinct pair $u, v \in C$, $\{u, v\} \in E$? Cook showed that the clique problem is NP-complete. The wide applicability of the notion of NP-completeness was observed by Karp, who proved that 21 basic problems are NP-complete.

Although we have thus far ignored all questions of encoding the inputs, there is one distinction that will play an important role in our discussion. The natural way to encode integers is to use a binary notation; e.g., $5 = \langle 101 \rangle$. However, one may also consider a unary notation; e.g., $5 = \langle 11111 \rangle$. There is an exponential gap between the lengths of both encodings. In the clique problem, there are no large integers to be encoded, and so this distinction is unimportant, but this is not always the case. In the *partition* problem, the input consists of $n$ numbers $a_1, \ldots, a_n$, and the question is if there exists a subset $S \subset \{1, \ldots, n\}$ such that $\Sigma_{j \in S} a_j = \Sigma_j a_j / 2$. This problem is NP-complete under a binary encoding. On the other hand, it can be solved by dynamic programming in $O(n \Sigma_j a_j)$ time, which is polynomial under a unary encoding; the method is therefore called a *pseudopolynomial-time* algorithm. There are also 'number problems' that are NP-complete, even when the numbers are encoded in unary. In the *3-partition* problem, the input consists of $3n$ integers $a_1, \ldots, a_{3n}$, and the question is if there exists a partition of $\{1, \ldots, 3n\}$ into $n$ 3-element sets $S_1, \ldots, S_n$ such that $\Sigma_{j \in S_i} a_j = \Sigma_j a_j / n$ for $i = 1, \ldots, n$. This problem remains NP-complete under a unary encoding and is therefore called *strongly* NP-*complete*.

The NP-hardness of an optimization problem suggests that it is impossible to always find an optimal solution quickly. However, it may still be possible to use an *approximation algorithm* to find solutions that are provably close to the optimum. For a minimization problem $f$, a *$\rho$-approximation algorithm* ($\rho > 1$) delivers a solution with value at most $\rho f(x)$ for each input $x$. Some NP-hard

which is a family of
olynomial-time $(1+\epsilon)$-
depend not only on the
a polynomial in $|x|$ and
*ximation scheme.*
on a *worst-case* analysis
delivered. It would be
s. To do this it appears
e inputs. We shall also
of *probabilistic* analysis.

is

: to process $n$ jobs $J_j$
re time intervals on one
no two time intervals on
located to the same job
f specific requirements
icteristics. A schedule is
e machine environment,
that together define a
sification $\alpha \mid \beta \mid \gamma$, which

ed for each job $J_j$:

operation models, or a
multi-operation models;
r processing;
e cost $f_j(t)$ incurred if $J_j$

n defining $f_j$.
ies.

e machine environment.

operation that can be
will be denoted by $p_{ij}$.

- $\alpha_1 = \circ$: *single machine*; $p_{1j} = p_j$;
- $\alpha_1 = P$: *identical parallel machines*; $p_{ij} = p_j$ for all $M_i$;
- $\alpha_1 = Q$: *uniform parallel machines*; $p_{ij} = p_j/s_i$ for a given *speed* $s_i$ of $M_i$;
- $\alpha_1 = R$: *unrelated parallel machines*; $p_{ij} = p_j/s_{ij}$ for given job-dependent speeds $s_{ij}$ of $M_i$.

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 $M_i$ during $p_{ij}$ time units, but the order in which the operations are executed is immaterial. If $\alpha_1 \in \{F, J\}$, an ordering is imposed on the set of operations corresponding to each job. If $\alpha_1 = F$, 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 $\mu_{ij}$ during $p_{ij}$ time units, with $\mu_{i,j} \neq \mu_{i+1,j}$ for $i = 1, \ldots, m_j - 1$.

If $\alpha_2$ is a positive integer, then $m$ is a constant, equal to $\alpha_2$; it is specified as part of the problem *type*. If $\alpha_2 = \circ$, then $m$ is a variable, the value of which is specified as part of the problem *instance*. Obviously, $\alpha_1 = \circ$ if and only if $\alpha_2 = 1$.

### 3.3. Job characteristics

The second field $\beta \subset \{\beta_1, \ldots, \beta_4\}$ 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) $\beta_2 = \{prec, tree, \circ\}$.

$\beta_2 = prec$. A *precedence relation* $\rightarrow$ between the jobs is specified. It is derived from an acyclic directed graph $G$ with vertex set $\{1, \ldots, n\}$. If $G$ contains a directed path from $j$ to $k$, we write $J_j \rightarrow J_k$ and require that $J_j$ is completed before $J_k$ can start.

$\beta_2 = tree$: $G$ is a *rooted tree* with either outdegree at most one for each vertex or indegree at most one for each vertex.

$\beta_2 = \circ$: No precedence relation is specified.

(3) $\beta_3 \in \{r_j, \circ\}$.

$\beta_3 = r_j$: *Release dates* that may differ per job are specified.

$\beta_3 = \circ$: All $r_j = 0$.

(4) $\beta_4 \in \{p_j = 1, p_{ij} = 1, \circ\}$.

$\beta_4 = p_j = 1$: Each job has a *unit processing requirement*. This will occur only if $\alpha_1 \in \{\circ, P, Q\}$.

$\beta_4 = p_{ij} = 1$: Each operation has *unit processing requirement*. This will occur only if $\alpha_1 \in \{O, F, J\}$.

$\beta_4 = \circ$: All $p_j$ or $p_{ij}$ are arbitrary nonnegative integers.

cteristics such as $m_j \leq 2$
obvious.
s than suggested above.
such as series–parallel
tree, such as intrees,
orders and level orders.

ality criterion. Given a

the minimization of

ctively, or of

$w_jT_j$, $w_jU_j$ respectively.
a constant $\Sigma w_j d_j$ and
nizing $L_{\max}$ also minim-

ıe value produced by an
er bound $\rho$ on $\gamma(A)/\gamma^*$
ists for which $\gamma(A)/\gamma^*$
noted by a dagger (†).

m lateness on a single
can be solved in poly-

completion time on an
owing preemption. Its

ıum completion time in
It is NP-hard (Section

### 3.6. Reducibility among scheduling problems

Each scheduling problem in the class outlined above corresponds to a six-tuple $(u_0, \ldots, u_5)$, where $u_i$ is a vertex of the graph $G_i$ shown in Figure 1 $(i = 0, \ldots, 5)$. For two problems $P = (u_0, \ldots, u_5)$ and $Q = (v_0, \ldots, v_5)$, we write $P \rightarrow Q$ if either $u_i = v_i$ or $G_i$ contains a directed path from $u_i$ to $v_i$, for $i = 0, \ldots, 5$. The reader should verify that $P \rightarrow Q$ implies that the decision version of $P$ reduces to the decision version of $Q$. For example, deciding if $L^*_{\max} \leq k$ can be reduced to the special case where $k = 0$, and this is equivalent to deciding if $\Sigma T^*_j = 0$. The graphs thus define elementary reductions between scheduling problems. It follows that if $P \rightarrow Q$ and $Q$ is solvable in polynomial time, then $P$ is solvable in polynomial time, and if $P \rightarrow Q$ and $P$ is NP-hard, then $Q$ is NP-hard.

These types of reductions play an instrumental role in the computer program MSPCLASS [Lageweg, Lawler, Lenstra & Rinnooy Kan, 1981, 1982]. The program records the complexity status of scheduling problems on the basis of known results and employing simple inference rules as given above. The main application of MSPCLASS concerns a collection of 4536 problems, which only differs from the class described in this section in that $\alpha_2$ is restricted to values from $\{1, 2, 3, \circ\}$, $\beta_1 = pmtn$ excludes $\beta_4 = p_{(i)j} = 1$, and $\beta$ also allows the specification of deadlines, i.e., strict upper bounds on job completion times. At present, 417 of these problems are known to be solvable in polynomial time, 3821 have been proved NP-hard, and 298 are still open. With respect to a

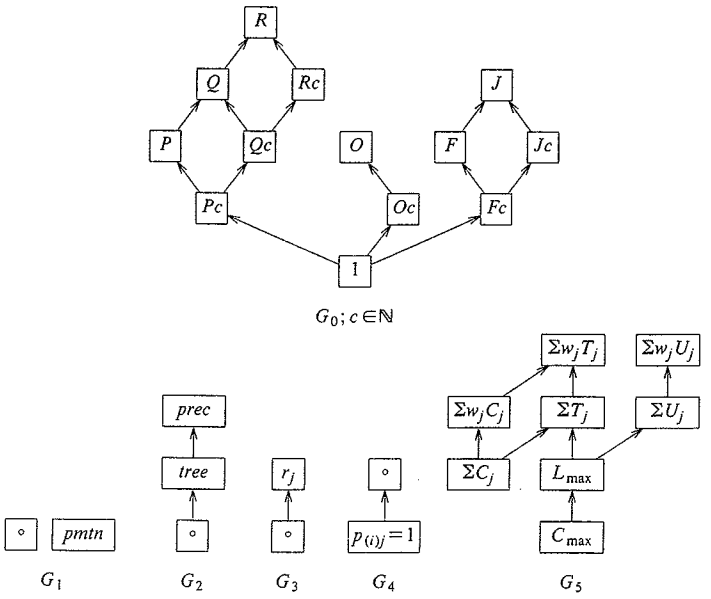



Fig. 1. Problem classification: the graphs $G_i$ $(i = 0, \ldots, 5)$.

l time, 3588 are strongly

nsive research ever since
56]. We will survey the
optimality criterion in
= 0, then only schedules
ed be considered [Con-

e and elegant solution.
arbitrary and different
creasing.
d let $L \subseteq N$ be the index
et $p(S) = \Sigma_{j \in S} p_j$ and let
the jobs indexed by $S$.
s:

the cost of an optimal
ist. It follows that there
position. By repeated
ule in $O(n^2)$ time. This

### 4.1. Maximum cost

Lawler's algorithm has been generalized by Baker, Lawler, Lenstra & Rinnooy Kan [1983] to an $O(n^2)$ algorithm for $1 \mid pmtn, prec, r_j \mid f_{\max}$. First, the release dates are modified such that $r_j + p_j \leq r_k$ whenever $J_j \rightarrow J_k$. Next, the jobs are scheduled in order of nondecreasing release dates; this creates a number of *blocks* that can be considered separately. From among the jobs without successors in a certain block, a job $J_k$ that yields minimum cost when finishing last is selected, the other jobs in the block are rescheduled in order of nondecreasing release dates, and $J_k$ is assigned to the remaining time intervals. By repeated application of this procedure to each of the resulting subblocks, one obtains an optimal schedule with at most $n - 1$ preemptions in $O(n^2)$ time.

Monma [1980] considers a generalization of $1 \mid \mid f_{\max}$. Let $c_j$ indicate the amount of a resource consumed (or, if $c_j < 0$, contributed) by $J_j$. The problem is to find a job permutation minimizing the maximum cumulative cost, $\max_j f_{\pi(j)} (\Sigma_{i=1}^{j-1} c_{\pi(i)})$. An NP-hardness proof and polynomial-time algorithms for special cases are presented.

### 4.2. Maximum lateness

Although Lenstra, Rinnooy Kan & Brucker [1977] show that the general $1 \mid r_j \mid L_{\max}$ problem is strongly NP-hard, polynomial algorithms exist for the cases that all $r_j$ are equal, all $d_j$ are equal or all $p_j$ are equal, and for the preemptive problem. The first case is solved by a specialization of Lawler's method, known as Jackson's rule [Jackson, 1955]: schedule the jobs in order of nondecreasing due dates. This rule, which minimizes the maximum tardiness as well, is also referred to as the *earliest due date* (*EDD*) rule. Note that, if any sequence completes all jobs by their due dates, an *EDD* sequence does. The second case is solved similarly by scheduling the jobs in order of nondecreasing release dates.

Horn [1974] observes that $1 \mid r_j, p_j = 1 \mid L_{\max}$ and $1 \mid pmtn, r_j \mid L_{\max}$ are solved by the extended Jackson's rule: at any time, schedule an available job with smallest due date. Frederickson [1983] gives an $O(n)$ algorithm for the case of unit-time jobs. Simons [1978] presents a more sophisticated approach to solve the problem $1 \mid r_j, p_j = p \mid L_{\max}$, where $p$ is an arbitrary integer. Let us first consider the simpler problem of finding a feasible schedule with respect to given release dates $r_j$ and deadlines $\bar{d}_j$. If application of the extended Jackson's rule yields such a schedule, we are finished; otherwise, let $J_l$ be the first late job and let $J_k$ be the last job preceding $J_l$ such that $\bar{d}_k > \bar{d}_l$. If $J_k$ does not exist, there is no feasible schedule; otherwise, the only hope of obtaining such a schedule is to postpone $J_k$ by forcing it to yield precedence to the set of jobs currently between $J_k$ and $J_l$. This is achieved by declaring the interval between the starting time of $J_k$ and the smallest release date of this set to be a forbidden region in which no job is allowed to start and applying the extended Jackson's rule again subject to this constraint. Since at each iteration at least one starting

Jackson's rule (EJ) is shown to guarantee

$$L_{\max}(\mathrm{EJ})/L^*_{\max} \leqslant 2\,. \qquad (\dagger)$$

Potts [1980b] presents an iterative version of the extended Jackson's rule (IJ), and proves that

$$L_{\max}(\mathrm{IJ})/L^*_{\max} \leqslant \tfrac{3}{2}\,. \qquad (\dagger)$$

Although interchanging the roles of the release times and delivery times does not improve the performance guarantee of algorithms EJ and IJ, Hall & Shmoys [1992] use it as the essential element of a modification of the latter algorithm (MIJ) that guarantees

$$L_{\max}(\mathrm{MIJ})/L^*_{\max} \leqslant \tfrac{4}{3}\,. \qquad (\dagger)$$

The technique of Lageweg, Lenstra & Rinnooy Kan [1976] implies that the results above extend to the case of precedence constraints. Hall & Shmoys [1992] also present two algorithms $A_{1k}$ and $A_{2k}$ that guarantee

$$L_{\max}(A_{lk})/L^*_{\max} \leqslant 1 + \frac{1}{k} \quad \text{for } l = 1, 2\,; \qquad (\dagger)$$

$A_{1k}$ runs in $O(n \log n + nk^{16k^2+8k})$ time, whereas $A_{2k}$ runs in $O(2^{4k}(nk)^{4k+3})$ time.

## 5. Total weighted completion time

### 5.0. Smith's ratio rule for $1\,|\,|\,\Sigma w_j C_j$

For the problem $1\,|\,|\,\Sigma w_j C_j$, any sequence is optimal that puts the jobs in order of nondecreasing ratios $p_j/w_j$ [Smith, 1956]. This rule is established by a simple interchange argument. Consider a sequence in which the jobs are not in order of nondecreasing $p_j/w_j$. Then there is a job $J_k$ that is immediately preceded by a job $J_j$, with $p_j/w_j > p_k/w_k$. If $J_j$ completes at time $C_k$, then $J_j$ completes at time $C_k - p_k$. The effect of interchanging these two jobs in the sequence is to decrease its cost by a positive amount:

$$\begin{aligned}&[w_j(C_k - p_k) + w_k C_k] - [w_k(C_k - p_j) + w_j C_k]\\ &\quad = w_k p_j - w_j p_k\\ &\quad = w_j w_k(p_j/w_j - p_k/w_k) > 0\,.\end{aligned}$$

Hence the sequence cannot be optimal. This confirms Smith's rule.

each node in the tree, and the ordering at the root yields the optimal solution. In fact, Buer & Möhring [1983] give an $O(n^3)$ algorithm that computes the decomposition, and Muller & Spinrad [1989] improve the running time to $O(n^2)$. For series-parallel graphs, each leaf of the decomposition tree corresponds to a single job, and each internal node corresponds to either a series operation, where all jobs in the first module must precede all jobs in the second, or a parallel operation, where no precedence constraints are added between the two modules.

The algorithm works from the bottom of the tree upward, merging sets of strings in the appropriate way. The one remaining observation needed is that for a series operation, if the largest string $\sigma_1$ in the first set (with respect to $\leqslant$) is bigger than the smallest string $\sigma_2$ in the second, then there exists an optimal ordering which contains $\sigma_1\sigma_2$, and so the two strings can be concatenated. By iterating this argument, the two sets of strings can be merged correctly.

Lawler [1978a,b], Monma & Sidney [1979], Monma [1981], Sidney [1981], Lawler & Lenstra [1982] and Monma & Sidney [1987] describe several axiomatic settings for characterizing results of this sort.

Series-parallel graphs can also be viewed as graphs that are iteratively built up by substitution from the two-element chain and from two incomparable elements. Möhring & Radermacher [1985a] generalize this by considering graphs whose prime (undecomposable) modules are of size $k$, giving an $O(n^{k^2})$ algorithm to minimize, for example, $\Sigma w_j C_j$ subject to such precedence constraints. Sidney & Steiner [1986] improve the running time to $O(n^{w+1})$, where $w$ denotes the maximum width of a prime module, by applying a more sophisticated dynamic programming procedure within the decomposition framework. Monma & Sidney [1987] give a partial characterization of objectives for which this combination of decomposition and dynamic programming can be applied.

### 5.2. Arbitrary precedence constraints, release dates and deadlines

Lawler [1978a] and Lenstra & Rinnooy Kan [1978] show that adding arbitrary precedence constraints results in NP-hardness, even if all $p_j = 1$ or all $w_j = 1$. Potts [1980c, 1985c] considers branch and bound methods for $1 \mid prec \mid \Sigma w_j C_j$ and provides empirical evidence that a simple lower bound heuristic based on Smith's rule pales in comparison to Lagrangean techniques.

Lenstra, Rinnooy Kan & Brucker [1977] show that if release dates are specified, $1 \mid r_j \mid \Sigma C_j$ is already strongly NP-hard. Gazmuri [1985] gives a probabilistic analysis of this problem under the assumption that the processing times and release times are independently and identically distributed. For each of two cases characterized by the relation between expected processing time and expected interarrival time, a heuristic is developed whose relative error tends to 0 in probability.

In the preemptive case, $1 \mid pmtn, r_j \mid \Sigma C_j$ can be solved by a simple extension

We may assume that any schedule is of the following form: first, the on-time jobs are processed in order of nondecreasing due dates; next, the late jobs are processed in an arbitrary order. Now suppose that $d_1 \leqslant \cdots \leqslant d_n$, and let $F_j(t)$ denote the minimum criterion value for the first $j$ jobs, subject to the constraint that the total processing time of the on-time jobs is at most $t$. Initializing the recursion by

$$F_j(t) = \infty \quad \text{for } t < 0\,, \quad j = 0, \ldots, n\,,$$
$$F_0(t) = 0 \quad \text{for } t \geqslant 0\,,$$

we have that

$$F_j(t) = \begin{cases} \min\{F_{j-1}(t-p_j),\, F_{j-1}(t) + w_j\} & \text{for } 0 \leqslant t \leqslant d_j\,, \\ F_j(d_j) & \text{for } t > d_j\,, \end{cases} \qquad j = 1, \ldots, n\,.$$

The problem is solved by computing $F_n(\Sigma_j p_j)$, which requires $O(n\Sigma_j p_j)$ time.

### 6.1. Further results

An algorithm due to Moore & Hodgson [Moore, 1968] allows the solution of $1 \mid\mid \Sigma U_j$ in $O(n \log n)$ time: jobs are added to the set of on-time jobs in order of nondecreasing due dates, and if the 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. Maxwell [1970] gives an alternative derivation of this algorithm based on ideas from linear and integer programming. Sidney [1973] extends the procedure to cover the case in which certain specified jobs have to be on time. The further generalization in which jobs have to meet given deadlines occurring at or after their due dates is shown to be NP-hard by Lawler [1982b]. Lawler [1976a] shows that the Moore-Hodgson algorithm is easily adapted to solve $1 \mid\mid \Sigma w_j U_j$ in $O(n \log n)$ time if processing times and weights are oppositely ordered (i.e., $p_j < p_k \Rightarrow w_j \geqslant w_k$).

Not surprisingly, $1 \mid r_j \mid \Sigma U_j$ is strongly NP-hard, but Lawler [1982b, -] shows how to apply dynamic programming techniques to solve $1 \mid pmtn, r_j \mid \Sigma U_j$ in $O(n^5)$ time and $1 \mid pmtn, r_j \mid \Sigma w_j U_j$ in $O(n^3(\Sigma w_j)^2)$ time. Kise, Ibaraki & Mine [1978] provide an $O(n^2)$ algorithm for $1 \mid r_j \mid \Sigma U_j$ in the case that release dates and due dates are similarly ordered (i.e., $r_j < r_k \Rightarrow d_j \leqslant d_k$); Lawler [1982b] shows that a variation of the Moore-Hodgson algorithm solves this problem in $O(n \log n)$ time. Lawler [-] also obtains $O(n \log n)$ solutions for $1 \mid pmtn, r_j \mid \Sigma w_j U_j$ in the case that the $(r_j, d_j)$ intervals are nested and in the case that release dates and processing times are similarly ordered and in opposite order of job weights.

Monma [1982] gives an $O(n)$ algorithm for $1 \mid p_j = 1 \mid \Sigma U_j$. However, Garey & Johnson [1976] prove that $1 \mid prec, p_j = 1 \mid \Sigma U_j$ is NP-hard, and Lenstra & Rinnooy Kan [1980] show that this is true even for *chain*-like precedence constraints.

They also derived a number of *elimination criteria*. These are statements of the following form: if the cost functions and processing times of $J_j$ and $J_k$ satisfy a certain relationship, then there is an optimal schedule in which $J_j$ precedes $J_k$.

Lower bounds and elimination criteria are used to discard partial schedules that are generated by an *enumeration scheme*. For $1\,|\,|\,\Sigma f_j$, it is customary to generate schedules by building them from back to front. That is, at the $l$th level of the search tree, jobs are scheduled in the $(n-l+1)$th position. The justification for this is that, since the cost functions are nondecreasing, the larger terms of the optimality criterion are fixed at an early stage while the smaller terms are estimated by the lower bound.

### 7.1. Further results

Lawler [1977] gives a pseudopolynomial algorithm for the problem $1\,|\,|\,\Sigma\, T_j$ that runs in $O(n^4\Sigma p_j)$ time. Recently, Du & Leung [1990] have shown that the problem is NP-hard in the ordinary sense.

Lenstra & Rinnooy Kan [1978] prove that $1\,|\,prec, p_j = 1\,|\,\Sigma T_j$ is NP-hard, and Leung & Young [1989] show that this is true even for *chain*-like precedence constraints. If we introduce release dates, $1\,|\,r_j, p_j = 1\,|\,\Sigma w_j T_j$ can be solved as a weighted bipartite matching problem, whereas $1\,|\,r_j\,|\,\Sigma T_j$ is obviously strongly NP-hard.

Lawler [1977] and Lenstra, Rinnooy Kan & Brucker [1977] show that $1\,|\,|\,\Sigma w_j T_j$ is strongly NP-hard. Various enumerative solution methods have been proposed for this problem. Elmaghraby [1968] presents the first elimination criteria for the problem, including the observation that any job with due date exceeding the total processing time can be scheduled last in an optimal schedule. Emmons [1969] and Shwimer [1972] develop other elimination criteria, and Rinnooy Kan, Lageweg & Lenstra [1975] extend these to the case of arbitrary nondecreasing cost functions. Rachamadugu [1987] gives an elimination criterion that generates an optimal schedule if there is one in which all jobs are late.

A variety of lower bounds have been studied. As already discussed in Section 7.0, Rinnooy Kan, Lageweg & Lenstra [1975] use a linear assignment relaxation based on an underestimate of the cost of assigning $J_j$ to position $k$, and Gelders & Kleindorfer [1974, 1975] use a fairly similar relaxation to a transportation problem. Fisher [1976] proposes a method in which the requirement that the machine can process at most one job at a time is relaxed. In this approach, one attaches 'prices' (i.e., Lagrangean multipliers) to each unit-time interval, and looks for multiplier values for which a cheapest schedule does not violate the capacity constraint. The resulting algorithm is quite successful on problems with up to 50 jobs. Potts & Van Wassenhove [1985] observe that a more efficiently computable but weaker bound may be preferable. They apply a multiplier adjustment method similar to the one mentioned in Section 5.2; the constraints $T_j \geq C_j - d_j$ are relaxed while associated prices for violating these constraints are introduced.

brighter here, and we will mention a number of polynomial-time algorithms for the minimization of $C_{\max}$ and $L_{\max}$, even subject to release dates. Finally, Section 11 deals with the presence of precedence constraints, with an emphasis on unit-time or preemptable jobs. The more general problems in this section are NP-hard and will lead us again to investigate the performance of approximation algorithms. However, several special cases turn out to be solvable in polynomial time.

## 8. Minsum criteria

### 8.0. A bipartite matching formulation for $R|\ |\Sigma C_j$

Horn [1973] and Bruno, Coffman & Sethi [1974] formulated $R|\ |\Sigma C_j$ as an integer programming problem. The structure of this program is such that it can be solved in polynomial time.

Consider the jobs that are to be performed by a single machine $M_i$, and for simplicity suppose that these are $J_1, J_2, \ldots, J_l$ in that order. For these jobs we have $\Sigma C_j = lp_{i1} + (l-1)p_{i2} + \cdots + p_{il}$. In general, $\Sigma C_j$ is a weighted sum of $p_{ij}$ values, where the weight of $p_{ij}$ is equal to the number of jobs to whose completion time it contributes. We now describe schedules in terms of 0–1 variables $x_{(ik),j}$, where $x_{(ik),j} = 1$ if $J_j$ is the $k$th last job processed on $M_i$, and $x_{(ik),j} = 0$ otherwise. The problem is then to minimize

$$\sum_{i,k} \sum_{j} kp_{ij} x_{(ik),j}$$

subject to

$$\sum_{i,k} x_{(ik),j} = 1 \qquad \text{for } j = 1, \ldots, n,$$

$$\sum_{j} x_{(ik),j} \leq 1 \qquad \text{for } i = 1, \ldots, m,\ k = 1, \ldots, n,$$

$$x_{(ik),j} \in \{0, 1\} \qquad \text{for } i = 1, \ldots, m,\ j, k = 1, \ldots, n.$$

The constraints ensure that each job is scheduled exactly once and that each position on each machine is occupied by at most one job. This is a weighted bipartite matching problem, so that the integrality constraints can be replaced by nonnegativity constraints without altering the feasible set. This matching problem can be solved in $O(n^3)$ time.

A similar approach yields $O(n \log n)$ algorithms for $P|\ |\Sigma C_j$ and $Q|\ |\Sigma C_j$. In the case of identical machines, $\Sigma C_j$ is a weighted sum of $p_j$ values, where each weight is an integer between 1 and $n$, and no weight may be used more than $m$ times. It is obviously optimal to match the smallest weights with the largest processing requirements. This is precisely what the generalized SPT rule

### 8.2. Minsum criteria without preemption

We have seen that $R|\ |\Sigma C_j$ is solvable in polynomial time. Meilijson & Tamir [1984] show that the SPT rule remains optimal for identical machines that increase in speed over time. On the other hand, if the speed decreases, then the problem is NP-hard.

In the case of arbitrary processing requirements, it seems fruitless to attempt to find polynomial algorithms for more general criteria or for $\Sigma C_j$ problems with additional constraints, even when there are only two identical machines. $P2|\ |\Sigma w_j C_j$ is already NP-hard [Bruno, Coffman & Sethi, 1974; Lenstra, Rinnooy Kan & Brucker, 1977], and so is $P2|\,tree\,|\Sigma C_j$, for intrees as well as outtrees [Sethi, 1977] and even for chains [Du, Leung & Young, 1991]. The specification of due dates or release dates does not leave much hope either, as both $P2|\ |C_{\max}$ and $1|r_j|\Sigma C_j$ are NP-hard. In this section, we will therefore be concerned with approximation in polynomial time and with optimization by implicit enumeration.

With respect to $P|\ |\Sigma w_j C_j$, an obvious idea is to list the jobs according to nondecreasing ratios $p_j/w_j$, as specified by Smith's rule for the single-machine case (see Section 5.0), and to schedule the next job whenever a machine becomes available. Eastman, Even & Isaacs [1964] show that this largest ratio (LR) rule gives

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

It follows from this inequality that

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

This lower bound has been the basis for the branch and bound algorithms of Elmaghraby & Park [1974], Barnes & Brennan [1977], and Sarin, Ahn & Bishop [1988]. Kawaguchi & Kyan [1986] have refined the analysis of these bounds to prove that

$$\sum w_j C_j(\mathrm{LR}) \Big/ \sum w_j C_j^* \leqslant \frac{\sqrt{2}+1}{2} . \qquad (\dagger)$$

Sahni [1976] constructs algorithms $A_k$ (in the same spirit as his approach for $1|\ |\Sigma w_j U_j$ mentioned in Section 6.1) 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 $A_k$ can be improved to $O(n^2k)$.

A general dynamic programming technique of Rothkopf [1966] and Lawler

essence of the correctness proof of the following algorithm of Gonzalez [1977]. 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. This procedure can be implemented to run in $O(n \log n + mn)$ time and yields an optimal schedule with no more than $(m-1)(n-\frac{1}{2}m)$ preemptions. Gonzalez also extends it to cover the case in which $\Sigma C_j$ is to be minimized subject to a common deadline for all jobs. McCormick & Pinedo [1989] extend this to handle the problem of minimizing $wC_{\max} + \Sigma C_j$ for an arbitrary weight $w \geq 0$.

Very little is known about $R \mid pmtn \mid \Sigma C_j$. This remains one of the more vexing questions in the area of preemptive scheduling. One approach has been to apply the techniques of Lawler & Labetoulle [1978] to show that if the optimal *order* of completion times is known, then an optimal solution can be constructed in polynomial time.

The problems $1 \mid pmtn \mid \Sigma w_j U_j$ (see Section 6.0) and $P \mid pmtn \mid \Sigma U_j$ are both NP-hard in the ordinary sense; the latter result is due to Lawler [1983]. Lawler [1979a] also shows that, for any fixed number of uniform machines, $Qm \mid pmtn \mid \Sigma w_j U_j$ can be solved in pseudopolynomial time: $O(n^2(\Sigma w_j)^2)$ if $m = 2$ and $O(n^{3m-5}(\Sigma w_j)^2)$ if $m \geq 3$. Hence, $Qm \mid pmtn \mid \Sigma U_j$ is solvable in strictly polynomial time. Lawler & Martel [1989] give an improved algorithm for $m = 2$ that runs in $O(n^2 \Sigma w_j)$ time, and also use this algorithm to derive a fully polynomial approximation scheme for $Q2 \mid pmtn \mid \Sigma w_j U_j$. The remaining minimal open problems are $R2 \mid pmtn \mid \Sigma U_j$ and, only with respect to a unary encoding, $P \mid pmtn \mid \Sigma U_j$.

We know from Section 7.1 that $1 \mid pmtn \mid \Sigma T_j$ and $1 \mid pmtn \mid \Sigma w_j T_j$ are NP-hard in the ordinary sense and in the strong sense, respectively. With respect to a unary encoding, $P2 \mid pmtn \mid \Sigma T_j$ is open.

In the presence of release dates, NP-hardness has been established for $P2 \mid pmtn, r_j \mid \Sigma C_j$ [Du, Leung & Young, 1988], $P2 \mid pmtn, r_j \mid \Sigma U_j$ [Du, Leung & Wong, 1989].

## 9. Minmax criteria without preemption

### 9.0. The performance of list scheduling for $P \mid \mid C_{\max}$

Although $P \mid \mid C_{\max}$ is strongly NP-hard [Garey & Johnson, 1978], there are simple procedures to construct schedules that are provably close to optimal. Consider the *list scheduling* (LS) rule, which schedules the next available job in some prespecified list whenever a machine becomes idle.

In the earliest paper on the worst-case analysis of approximation algorithms, Graham [1966] proves that, for any instance,

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

Since there always is a list ordering for which this simple heuristic produces an optimal schedule, it is natural to consider refinements of the approach. Graham [1969] shows that, if the jobs are selected in *longest processing time* (LPT) order, then the bound can be considerably improved:

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

A somewhat better algorithm, called *multifit* (MF) and based on a completely different principle, is due to Coffman, Garey & Johnson [1978]. The idea behind MF is to find (by binary search) the smallest 'capacity' that a set of $m$ 'bins' can have and still accommodate all jobs when the jobs are taken in order of nonincreasing $p_j$ and each job is placed into the first bin into which it will fit. The set of jobs in the $i$th bin will be processed by $M_i$. Coffman, Garey & Johnsons show that, if $k$ packing attempts are made, the algorithm (denoted by $\mathrm{MF}_k$) runs in time $O(n \log n + kn \log m)$ and satisfies

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

Friesen [1984] subsequently improves this bound from 1.22 to 1.2. Yue [1990] improves it to $\frac{13}{11}$, which is tight. The procedure executed within the binary search 'loop' can be viewed as an approximation algorithm for packing a set of jobs in the fewest number bins of a given capacity. If a more primitive algorithm is used for this, where the jobs are not ordered by decreasing $p_j$, then all that can be guaranteed is

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

Friesen & Langston [1986] refine the iterated approximation algorithm to provide algorithms $\mathrm{MF}'_k$ with running time $O(n \log n + kn \log m)$ (where the constant embedded within the 'big Oh' notation is big indeed) that guarantee

$$C_{\max}(\mathrm{MF}'_k)/C^*_{\max} \leqslant \tfrac{72}{61} + 2^{-k}\,. \qquad (\dagger)$$

The following algorithm $Z_k$ is due to Graham [1969]: schedule the $k$ largest jobs optimally, then list schedule the remaining jobs arbitrarily. Graham shows that

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

and that when $m$ divides $k$, this is best possible. By selecting $k = m/\varepsilon$, we obtain an algorithm with worst-case performance ratio less than $1 + \varepsilon$. Unfortunately, the best bound on the running time is $O(n^{km})$. Thus, for any fixed number of machines, this family of algorithms is a polynomial approximation

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

Significantly less is known about the worst-case performance of approximation algorithms for other minmax criteria. Gusfield [1984] considers the problem $P \mid r_j \mid L_{\max}$, and proves that for the EDD rule (see Section 4.1),

$$L_{\max}(\mathrm{EDD}) - L^*_{\max} \leqslant \frac{2m-1}{m} \max_j p_j . \qquad (\dagger)$$

As in the single machine case, it is natural to consider the relative error in the delivery time model. The translation of the previous bound into this setting provides an unnecessarily weak guarantee. By using a simple extension of the argument of Graham [1966], Hall & Shmoys [1989] observe that

$$L_{\max}(\mathrm{LS})/L^*_{\max} < 2 . \qquad (\dagger)$$

They also develop a polynomial approximation scheme for this problem. Carlier [1987] gives an enumerative method for $P \mid r_j \mid L_{\max}$. Simons [1983] shows that an interesting special case, $P \mid r_j, p_j = p \mid L_{\max}$, can be solved in polynomial time. Simons & Warmuth [1989] give an improved $O(mn^2)$ algorithm based on a generalization of the approach of Garey, Johnson, Simons & Tarjan [1981]. No approximation results are known for minimizing $C_{\max}$ with both release times and deadlines; Bratley, Florian & Robillard [1975] give an enumerative method for this problem.

The simple probabilistic analysis of list scheduling that was discussed in Section 8.0 is also just a first step in a series of results in this area. For example, the bounds of Bruno & Downey [1986] were refined and extended to other distributions by Coffman & Gilbert [1985].

Probabilistic analysis also supports the claim that the LPT heuristic performs better than arbitrary list scheduling. Unlike the relative error of list scheduling, the absolute error $C_{\max}(\mathrm{LS}) - C^*_{\max}$ does not tend to 0 as $n \to \infty$ (with $m$ fixed). Coffman, Flatto & Lueker [1984] observe that, if $I(\mathrm{LPT})$ denotes the total idle time in an LPT schedule, then the absolute error is at most $I(\mathrm{LPT})/m$. For processing times selected independently and uniformly from $[0, 1]$, they prove that $\mathrm{E}[I(\mathrm{LPT})] \leqslant c_m m^2/(n+1)$, where $c_m$ is bounded and $\lim_{m\to\infty} c_m = 1$.

Loulou [1984] and Frenk & Rinnooy Kan [1987] both base their analyses of LPT on the difference $C_{\max}(\mathrm{LPT}) - \Sigma_j p_j/m$, which is an upper bound on $C_{\max}(\mathrm{LPT}) - C^*_{\max}$. Loulou shows that, if the processing times are independent and identically distributed with finite mean, then, for any fixed $m \geqslant 2$, the absolute error of LPT is stochastically smaller than a fixed random variable that does not depend on $n$. Frenk & Rinnooy Kan consider the general situation where the processing times are independently drawn from a distribution that has finite second moment and positive density at zero. They prove that the absolute error converges to 0 not only in expectation but even almost surely; that is, $\Pr[\lim_{n\to\infty} C_{\max}(\mathrm{LPT}) - C^*_{\max} = 0] = 1$.

Friesen & Langston [1983] extend the multifit approach to uniform processors. They prove that, if the bins are ordered in increasing size for each iteration of the binary search, then

$$C_{\max}(\mathrm{MF}_k)/C^*_{\max} \leq 1.4 + 2^{-k},$$

and that there exists an example that has performance ratio 1.341. They also show that the decision to order the bins by increasing size is the correct one, since for decreasing bin sizes there exist examples with performance ratio $\frac{3}{2}$.

Horowitz & Sahni [1976] give a family of algorithms $A_k$ with running time $O(n^{2m}k^{m-1})$ such that

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

so that for any fixed value of $m$, this is a fully polynomial approximation scheme. Extending their dual approximation approach for identical machines, Hochbaum & Shmoys [1988] give a polynomial approximation scheme, where algorithm $D_k$ has running time $O(mn^{10k^2+3})$ and

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

For small values of $k$, the efficiency of this scheme can be improved; Hochbaum & Shmoys provide algorithms with performance guarantee arbitrarily close to $\frac{3}{2}$ that run in $O(n \log n + m)$ time.

The probabilistic results of Frenk & Rinnooy Kan [1986, 1987] also extend to the case of uniform machines. In fact, the naive implementation of the LPT rule (as opposed to the algorithm LPT′ that was discussed above) produces schedules in which the absolute error converges in expectation and almost surely to 0.

### 9.3. Unrelated machines

Unrelated parallel machine problems are perceived to be significantly harder than uniform machine problems, and results concerning the worst-case analysis of approximation algorithms substantiate this distinction. Lenstra, Shmoys & Tardos [1990] show that it is NP-complete to decide if there is a feasible schedule of length 2 for instances of $R|\ |C_{\max}$. This implies that there does not exist a polynomial-time $\rho$-approximation algorithm with $\rho < \frac{3}{2}$ unless P = NP. Although this excludes the possibility of a polynomial approximation scheme, Horowitz & Sahni [1976] show that for any fixed number of machines, there is a *fully* polynomial approximation scheme.

Ibarra & Kim [1977] show that a variety of simple algorithms perform discouragingly poorly; in fact, they were only able to prove that these methods were $m$-approximation algorithms. The first substantial improvement of this

for which any optimal schedule has at least this many preemptions. It is not hard to see that the problem of minimizing the number of preemptions is NP-hard.

### 10.1. Maximum completion time on uniform and unrelated machines

For $Q \mid pmtn \mid C_{\max}$, the length of any schedule is at least

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

where $p_1 \geqslant \cdots \geqslant p_n$ and $s_1 \geqslant \cdots \geqslant s_m$. This generalizes the lower bound given in the previous section.

Horvath, Lam & Sethi [1977] prove that this bound is met by a preemptive variant of the LPT rule, which, at each point in time, assigns the jobs with the largest remaining processing requirement to the fastest available processors. The algorithm runs in $O(mn^2)$ time and generates an optimal schedule with no more than $(m-1)n^2$ preemptions.

Gonzalez & Sahni [1978b] give a more efficient algorithm. It requires $O(n)$ time, if the jobs are given in order of nonincreasing $p_j$ and the machines in order of nonincreasing $s_i$; without this assumption, the running time increases only to $O(n + m \log m)$. The procedure yields an optimal schedule with no more than $2(m-1)$ preemptions, which is a tight bound.

Lawler & Labetoulle [1978] show that many preemptive scheduling problems involving independent jobs on unrelated machines can be formulated as linear programming problems. For $R \mid pmtn \mid C_{\max}$, the length of any schedule is at least equal to the minimum value of $C$ subject to

$$\sum_i x_{ij}/p_{ij} = 1 \quad \text{for } j = 1, \ldots, n,$$

$$\sum_i x_{ij} \leqslant C \quad \text{for } j = 1, \ldots, n,$$

$$\sum_j x_{ij} \leqslant C \quad \text{for } i = 1, \ldots, m,$$

$$x_{ij} \geqslant 0 \quad \text{for } i = 1, \ldots, m,\ j = 1, \ldots, n.$$

In this formulation, $x_{ij}$ represents the total time spent by $J_j$ on $M_i$. The linear program can be solved in polynomial time [Khachiyan, 1979]. A feasible schedule for which $C_{\max}$ equals the optimal value of $C$ can be constructed in polynomial time by applying the algorithm for $O \mid pmtn \mid C_{\max}$, discussed in Section 12.2. This procedure can be modified to yield an optimal schedule with no more than about $7m^2/2$ preemptions. It remains an open question as to whether there is some constant $c > 0$ such that $cm^2$ preemptions are necessary for an optimal preemptive schedule.

For fixed $m$, it seems to be possible to solve the linear program in linear

)] show how to solve the

*ions*

$L_{max}$ and $P \mid pmtn, r_j \mid C_{max}$
ore efficient algorithm that

ence of a feasible preemp-
can be tested by means of
rch can then be conducted
of $L_{max}$ inducing deadlines
ıe network computation.
·] show that this yields an

les have been investigated.
*ines* are only available in
: existence of a feasible
.e. Rayward-Smith [1987b]
is incurred when a job is
:erves that imposing such
ost $k-1$. Thus, for $k=1$,
ıghton's rule. Surprisingly,

980] show how to test the
given release dates and a
gorithm generates $O(mn)$
ahni & Cho [1979b] and
.an [1984] show that
$\mid L_{max}$ are solvable in
tions generated is $O(mn)$.
been adapted by Bruno &
hines and extended to a
Labetoulle, Lawler, Len-

ım for $Q \mid pmtn, r_j \mid L_{max}$.
·al algorithm of Lawler &
ıetwork flows. Federgruen
he problem by reducing it
.re machines of $t$ distinct
) time.
yields a polynomial-time
$, r_j \mid L_{max}$.

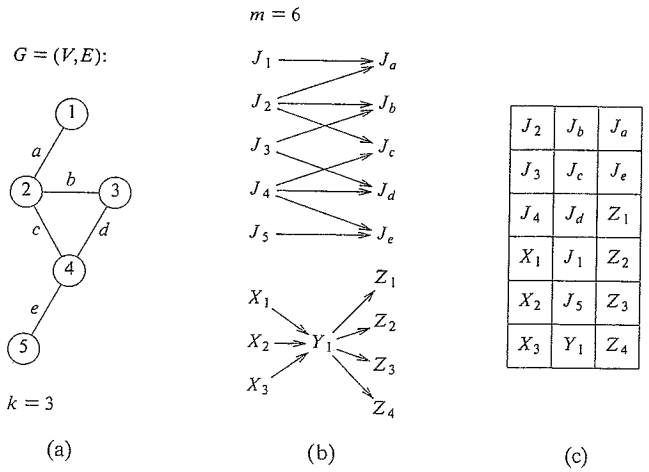
## 11. Precedence constraints

### *11.0. An NP-hardness proof for* $P \mid prec, p_j = 1 \mid C_{max}$

The first NP-hardness proof for $P \mid prec, p_j = 1 \mid C_{max}$ is due to Ullman [1975]. Lenstra & Rinnooy Kan [1978] show that even the problem of deciding if there exists a feasible schedule of length at most 3 is NP-complete; the proof is given below. This result implies that, for $P \mid prec, p_j = 1 \mid C_{max}$, there is no polynomial $\rho$-approximation algorithm for any $\rho < \frac{4}{3}$, unless P = NP. Note that it is trivial to decide if a feasible schedule of length 2 exists.

Recall the NP-complete *clique* problem from Section 2: given a graph $G = (V, E)$ and an integer $k$, does $G$ have a clique (i.e., a complete subgraph) on $k$ vertices? We denote the number of edges in a clique of size $k$ by $l = k(k-1)/2$, and we define $k' = |V| - k$, $l' = |E| - l$. For any instance of the clique problem, we construct a corresponding instance of $P \mid prec, p_j = 1 \mid C_{max}$. The number of machines is given by $m = \max\{k, l + k', l'\} + 1$. We introduce a job $J_v$ for every vertex $v \in V$ and a job $J_e$ for every edge $e \in E$, with $J_v \rightarrow J_e$ whenever $v$ is an endpoint of $e$. We also need dummy jobs $X_x$ $(x = 1, \ldots, m-k)$, $Y_y$ $(y = 1, \ldots, m-l-k')$ and $Z_z$ $(z = 1, \ldots, m-l')$, with $X_x \rightarrow Y_y \rightarrow Z_z$ for all $x, y, z$. Note that the total number of jobs is $3m$.

The reduction is illustrated in Figure 2. The basic idea is the following. In any schedule of length 3 for the dummy jobs, there is a certain pattern of idle machines that are available for the vertex and edge jobs. This pattern is chosen such that a complete feasible schedule of length 3 exists if and only if there is a clique of size $k$.




| | | |
|---|---|---|
| $J_2$ | $J_b$ | $J_a$ |
| $J_3$ | $J_c$ | $J_e$ |
| $J_4$ | $J_d$ | $Z_1$ |
| $X_1$ | $J_1$ | $Z_2$ |
| $X_2$ | $J_5$ | $Z_3$ |
| $X_3$ | $Y_1$ | $Z_4$ |

Fig. 2. The clique problem reduces to $P \mid prec, p_j = 1 \mid C_{max}$. (a) Instance of the clique problem. (b) Corresponding instance of $P \mid prec, p_j = 1 \mid C_{max}$. (c) Feasible schedule for $P \mid prec, p_j = 1 \mid C_{max}$.

graph is an *interval order* and give an $O(n+m)$ list scheduling rule that delivers optimal schedules. Bartusch, Möhring & Radermacher [1988a] give an algorithm that unifies many of the special cases previously known to be polynomially solvable.

In addition to proving interesting structural theorems about optimal schedules, Dolev & Warmuth [1984, 1985a,b] give polynomial-time algorithms for a number of special cases of $Pm \mid prec, p_j = 1 \mid C_{\max}$. Dolev & Warmuth [1985b] give an algorithm for opposing forests with substantially improved running time, that also uses substantially more space. In an arbitrary precedence graph, the *level* of a job is the length of the longest path that starts at that job. A *level order* is a precedence graph in which each pair of incomparable jobs with a common predecessor or successor have identical sets of predecessors and successors. Dolev & Warmuth [1985b] also show that level orders can be solved in $O(n^{m-1})$ time. For precedence graphs in which the longest path has at most $h$ arcs, Dolev & Warmuth [1984] give an $O(n^{h(m-1)+1})$ algorithm. Note that the proof given above shows that the problem is already NP-hard for $h = 2$. Dynamic programming can be used to obtain a polynomial-time algorithm for the case where the width of the precedence graph is bounded; this is one of the many polynomially solvable special cases surveyed by Möhring [1989].

Fujii, Kasami & Ninomiya [1969] present the first polynomial-time algorithm for $P2 \mid prec, p_j = 1 \mid C_{\max}$. An undirected graph is constructed with vertices corresponding to jobs and edges $\{j, k\}$ whenever $J_j$ and $J_k$ can be executed simultaneously. An optimal schedule is then derived from a maximum cardinality matching in the graph, and so the algorithm runs in $O(n^3)$ time [Lawler, 1976b].

Coffman & Graham [1972] give an alternative approach that leads to an $O(n^2)$ list scheduling algorithm. First the jobs are labeled in the following way. Suppose labels $1, \ldots, k$ have been applied and $S$ is the subset of unlabeled jobs all of whose successors have been labeled. 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. Sethi [1976b] shows that it is possible to execute this algorithm in time almost linear in $n$ plus the numbers of arcs in the precedence graph, if the graph is given in the form of a *transitive reduction*.

Gabow [1982] presents an algorithm which has the same running time, but which does not require such a representation of the precedence graph. The running time of the algorithm is dominated by the time to maintain a data structure that represents sets of elements throughout a sequence of so-called union-find operations, and Gabow & Tarjan [1985] improve the running time to linear by exploiting the special structure of the particular union-find problems generated in this way. Consider the following procedure to compute a lower bound on the length of an optimal schedule. Delete jobs and precedence constraints to obtain a precedence graph that can be decomposed into $t$ sets of jobs, $S_1, \ldots, S_t$, such that for each pair of jobs $J_k \in S_i$, $J_l \in S_{i+1}$,

[1988a] show that $P \mid tree, p_j \in \{1, k\} \mid C_{\max}$ (where $k$ is input) is strongly NP-hard, and that $P2 \mid tree, p_j \in \{k^l : l \geq 0\} \mid C_{\max}$ is NP-hard in the ordinary sense for any integer $k > 1$. For $P2 \mid prec, p_j \in \{1, k\} \mid C_{\max}$, Goyal [1977] proposes a generalized version of the Coffman–Graham algorithm (GCG) and shows that

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

Rayward-Smith [1987a] considers a model similar to one discussed in Section 10.2, where there is a unit-time communication delay between any pair of distinct processors. For unit-time jobs, the problem is shown to be NP-complete. The performance of a *greedy* ($G$) algorithm is analyzed, where first a list schedule is generated, and then a local interchange strategy tries to improve the schedule. The algorithm produces schedules such that

$$C_{\max}(G)/C^*_{\max} \leq 3 - \frac{2}{m}. \quad (\dagger)$$

Approximation algorithms in a similar model are also considered by Papadimitriou & Yannakakis [1990].

### 11.2. Precedence constraints and no preemption

The list scheduling rule performs surprisingly well on identical machines, even in the presence of precedence constraints. Graham [1966] shows that precedence constraints do not affect its worst-case performance at all; that is,

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

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

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

Note that this result implies the previous one. Even when critical path (CP) scheduling is used, Graham [-] provides examples for which

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

Gabow [1988] considers $Q2 \mid prec, p_j = 1 \mid C_{\max}$ and analyzes two approximation algorithms. The algorithm $P2$, which ignores the machine speeds and finds an optimal solution to the resulting problem on two identical machines, guarantees

$$C_{\max}(P2)/C^*_{\max} \leq 2 - \min\{s_1, s_2\}/\max\{s_1, s_2\} . \qquad (\dagger)$$

The *highest level first* (HLF) algorithm is shown to be slightly better in special cases:

$$C_{\max}(\mathrm{HLF})/C^*_{\max} \leq \begin{cases} \frac{5}{4} & \text{if } \min\{s_1, s_2\}/\max\{s_1, s_2\} = \frac{1}{2} , \\ \frac{6}{5} & \text{if } \min\{s_1, s_2\}/\max\{s_1, s_2\} = \frac{2}{3} . \end{cases} \qquad (\dagger)$$

Gabow also gives an $O((n+a)2^l)$ algorithm to find an optimal solution if $|1/s_1 - 1/s_2| = 1$, where $1/s_1$ and $1/s_2$ are relatively prime integers, $a$ is the number of arcs and $l$ is the number of levels in the precedence graph.

Nothing is known about approximation algorithms for unrelated machines with precedence constraints.

### 11.3. Precedence constraints and preemption

Ullmann [1976] shows that $P \mid pmtn, prec, p_j = 1 \mid C_{\max}$ is NP-hard, but $P \mid pmtn, tree \mid C_{\max}$ and $P2 \mid pmtn, prec \mid C_{\max}$ can be solved by a polynomial-time algorithm due to Muntz & Coffman [1969, 1970].

The Muntz–Coffman algorithm can be described as follows. Define $l_j(t)$ to be the level of a $J_j$ wholly or partly unexecuted at time $t$, where the level now refers to the length of the path in the precedence graph with maximum total processing requirement. 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 $m' \geq n'$, we assign one machine to each job, consider the jobs at the next highest level, and repeat. The machines are reassigned whenever a job is completed or threatens to be processed at a higher speed than another one at a currently higher level. Between each pair of successive reassignment points, jobs are finally rescheduled by means of McNaughton's algorithm for $P \mid pmtn \mid C_{\max}$. Gonzalez & Johnson [1980] give an implementation of the algorithm that runs in $O(n^2)$ time.

Gonzalez & Johnson [1980] have developed a totally different algorithm that solves $P \mid pmtn, tree \mid C_{\max}$ 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.

This approach can be adapted to the case $Q2 \mid pmtn, tree \mid C_{\max}$. Horvath, Lam & Sethi [1977] give an algorithm to solve $Q2 \mid pmtn, prec \mid C_{\max}$ in $O(mn^2)$ time, similar to the result mentioned in Section 10.1.

focuses on the $C_{\max}$ criterion. A few results for other optimality criteria will be briefly mentioned.

Very few multi-operation scheduling problems can be solved in polynomial time; the main well-solvable cases are $F2 \mid \mid C_{\max}$ [Johnson, 1954], $O2 \mid \mid C_{\max}$ [Gonzalez & Sahni, 1976], and $O \mid pmtn \mid C_{\max}$ [Gonzalez & Sahni, 1976; Lawler & Labetoulle, 1978]. General flow shop and job shop scheduling problems have earned a reputation for intractability. We will be mostly concerned with enumerative optimization methods for their solution and, to a lesser extent, with approximation algorithms. An analytical approach to the performance of methods of the latter type is badly needed.

## 12. Open shops

### 12.0. Gonzalez & Sahni's algorithm for $O2 \mid \mid C_{\max}$

The problem $O2 \mid \mid C_{\max}$ admits of an elegant linear-time algorithm due to Gonzalez & Sahni [1976].

For convenience, let $a_j = p_{1j}$, $b_j = p_{2j}$, $\bar{a} = \Sigma_j a_j$, $\bar{b} = \Sigma_j b_j$. An obvious lower bound on the length of any feasible schedule is given by

$$\max\{\bar{a}, \bar{b}, \max_j a_j + b_j\} .$$

We will show how a schedule matching this bound can be constructed in $O(n)$ time.

Let $A = \{J_j \mid a_j \geqslant b_j\}$ and $B = \{J_j \mid a_j < b_j\}$. Choose $J_r$ and $J_l$ to be any two distinct jobs, whether in $A$ or $B$, such that

$$a_r \geqslant \max_{J_j \in A} b_j , \qquad b_l \geqslant \max_{J_j \in B} a_j .$$

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 Figure 3(a), where the jobs in $A'$ and $B'$ are ordered arbitrarily. In each of these separate schedules, there is no idle time on either machine, from the start of the first operation on that machine to the completion of its last operation.

Suppose $\bar{a} - a_l \geqslant \bar{b} - b_r$ (the case $\bar{a} - a_l < \bar{b} - b_r$ being symmetric). We then combine the two schedules as shown in Figure 3(b), 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 operation to the completion of the last operation.

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. First, if $a_r \leqslant \bar{b} - b_r$, then the resulting schedule is as in Figure 3(c); the length of the schedule is $\max\{\bar{a}, \bar{b}\}$. Secondly, if $a_r > \bar{b} - b_r$, then the schedule in Figure 3(d) results; its length is $\max\{\bar{a}, a_r + b_r\}$. Since, in both cases, we have met our lower bound, the schedules constructed are optimal.

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

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

Call row $i$ (column $j$) *tight* if $\Sigma_j p_{ij} = C$ ($\Sigma_i p_{ij} = C$), and *slack* otherwise. Clearly, we have $C^*_{\max} \geq C$. 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 call such a subset a *decrementing set*, and use it to construct a *partial schedule* of length $\delta$, for some $\delta > 0$. The constraints on the choice of $\delta$ are as follows:

- If $p_{ij} \in S$ and row $i$ or column $j$ is tight, then $\delta \leq p_{ij}$.
- If $p_{ij} \in S$ and row $i$ (column $j$) is slack, then $\delta \leq p_{ij} + C - \Sigma_k p_{ik}$ ($\delta \leq p_{ij} + C - \Sigma_h p_{hi}$).
- If row $i$ (column $j$) contains no element in $S$ (and is therefore necessarily slack), then $\delta \leq C - \Sigma_k p_{ik}$ ($\delta \leq C - \Sigma_h p_{hj}$).

For a given decrementing set $S$, let $\delta$ be the maximum value subject to these constraints. 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 a matrix $P'$ from $P$ by replacing each $p_{ij} \in S$ by $\max\{0, p_{ij} - \delta\}$, with a lower bound $C - \delta$ on the schedule length for the remaining problem. We 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 schedule for $P$.

By suitably embedding $P$ in a doubly stochastic matrix and appealing to the Birkhoff–Von Neumann theorem, one can show that a decrementing set can be found by solving a linear assignment problem; see Lawler & Labetoulle [1978] for details. Other networks formulations of the problem are possible. An analysis of various possible computations reveals that $O \mid pmtn \mid C_{\max}$ is solvable in $O(r + \min\{m^4, n^4, r^2\})$ time, where $r$ is the number of nonzero elements in $P$ [Gonzalez, 1979].

Similar results can be obtained for the minimization of maximum lateness. Lawler, Lenstra & Rinnooy Kan [1981] give an $O(n)$ time algorithm for $O2 \mid pmtn \mid L_{\max}$ and, by symmetry, for $O2 \mid pmtn, r_j \mid C_{\max}$. For $O \mid pmtn, r_j \mid L_{\max}$, Cho & Sahni [1981] show that a trial value of $L_{\max}$ can be tested for feasibility by linear programming; bisection search is then applied to minimize $L_{\max}$ in polynomial time.

The minimization of total completion time appears to be much harder. Liu & Bulfin [1985] provide NP-hardness proofs for $O3 \mid pmtn \mid \Sigma C_j$ and $O2 \mid pmtn, \bar{d}_j \mid \Sigma C_j$, where $\bar{d}_j$ is a deadline for the completion of $J_j$. $O2 \mid pmtn \mid \Sigma C_j$ remains an open problem.

processing times $(p_{1j} + l_j, l_j + p_{2j})$ [Rinnooy Kan, 1976]. Monma & Rinnooy Kan [1983] put many results of this kind in a common framework. Their discussion includes results for problems with an arbitrary number of machines, such as some of the work by Smith, Panwalkar & Dudek [1975, 1976] on ordered flow shops and by Chin & Tsai [1981] on $J$-maximal and $J$-minimal flow shops. In the latter case, there is an $M_i$ for which $p_{ij} = \max_h p_{hj}$ for all $j$ or $p_{ij} = \min_h p_{hj}$ for all $j$. Achugbue & Chin [1982b] analyze $F3 \mid \mid C_{\max}$ in which each machine may be maximal or minimal in this sense and derive an exhaustive complexity classification. It should be noted that, in all this work, there is an implicit restriction to permutation schedules. This is justified for special cases of $F3 \mid \mid C_{\max}$, but not necessarily for its variants. Indeed, the unrestricted $F3 \mid \mid C_{\max}$ problem with a nonbottleneck $M_2$ is strongly NP-hard [Lenstra, -].

NP-hardness in the strong sense has also been established for $F2 \mid r_j \mid C_{\max}$, $F2 \mid \mid L_{\max}$ [Lenstra, Rinnooy Kan & Brucker, 1977] and $F2 \mid \mid \Sigma C_j$ [Garey, Johnson & Sethi, 1976]. Potts [1985b] investigates the performance of five approximation algorithms for $F2 \mid r_j \mid C_{\max}$. The best one of these, called RJ', involves the repeated application of a dynamic variant of Johnson's algorithm to modified versions of the problem, and satisfies

$$C_{\max}(\mathrm{RJ}')/C^*_{\max} \leqslant \tfrac{5}{3} . \qquad (\dagger)$$

Grabowski [1980] presents a branch and bound algorithm for $F2 \mid r_j \mid L_{\max}$. Ignall & Schrage [1965], in one of the earliest papers on branch and bound methods for scheduling problems, propose two lower bounds for $F2 \mid \mid \Sigma C_j$, Kohler & Steiglitz [1975] report on the implementation of these bounds, and Van de Velde [1990] shows that both bounds can be viewed as special cases of a lower bound based on Lagrangean relaxation.

Gonzalez & Sahni [1978a] and Cho & Sahni [1981] consider the case of preemptive flow shop scheduling. Since preemptions on $M_1$ and $M_m$ can be removed without increasing $C_{\max}$, Johnson's algorithm solves $F2 \mid pmtn \mid C_{\max}$ as well. $F3 \mid pmtn \mid C_{\max}$, $F2 \mid pmtn, r_j \mid C_{\max}$ and $F2 \mid pmtn \mid L_{\max}$ are strongly NP-hard. So is $F3 \mid pmtn \mid \Sigma C_j$ [Lenstra, -]; $F2 \mid pmtn \mid \Sigma C_j$ remains open.

As to precedence constraints, $F2 \mid tree \mid C_{\max}$ is strongly NP-hard [Lenstra, Rinnooy Kan & Brucker, 1977], but $F2 \mid tree, p_{ij} = 1 \mid C_{\max}$ and $F2 \mid tree, p_{ij} = 1 \mid \Sigma C_j$ are solvable in polynomial time [Lageweg, -]. We note that an interpretation of precedence constraints that differs from our definition is possible. If $J_j \rightarrow 'J_k$ only means that $O_{ij}$ has to precede $O_{ik}$, for $i = 1, 2$, then $F2 \mid tree' \mid C_{\max}$ and even the problem with series-parallel precedence constraints can be solved in $O(n \log n)$ time [Sidney, 1979; Monma, 1979]. The arguments used to establish this result are very similar to those referred to in Section 5.1 and apply to a larger class of scheduling problems. The general case $F2 \mid prec' \mid C_{\max}$ is strongly NP-hard [Monma, 1980]. Hariri & Potts [1984] develop a branch and bound algorithm for this problem, using a lower bound based on Lagrangean relaxation.

Removal of either $N_{*u}$ or $N_{u*}$ results in a $1|\ |L_{\max}$ or $1|r_j|C_{\max}$ problem on $M_u$. Both problems are solvable in $O(n \log n)$ time (see Section 4.2) and provide slightly stronger bounds.

If $u \neq v$, removal of $N_{*u}$, $N_{uv}$ and $N_{v*}$ yields an $F2|\ |C_{\max}$ problem, which can be solved by Johnson's algorithm. As pointed out in Section 13.1, we can take $N_{uv}$ fully into account and still solve the problem in $O(n \log n)$ time. The resulting bound dominates the *job-based bound* proposed by McMahon [1971] and is currently the most successful bound that can be computed in polynomial time.

All other variations on this theme lead to NP-hard problems. However, this does not necessarily preclude their effectivity for lower bound computations, as will become clear in Section 14.2.

In addition to lower bounds, one may use elimination criteria in order to prune the search tree. In this respect, 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 the following: $J_j$ can be excluded for the $(l+1)$th position if

$$\max\{C(\sigma kj, i-1) - C(\sigma j, i-1),\ C(\sigma kj, i) - C(\sigma j, i)\} \leq p_{ij}$$
$$\text{for } i = 2, \ldots, m$$

[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, Lenstra & Rinnooy Kan, 1978]. It may be worthwhile to consider extensions that, for instance, take the processing times of the unscheduled jobs into account [Gupta & Reddi, 1978; Szwarc, 1978].

A number of alternative and more efficient enumeration schemes has been developed. Potts [1980a] proposes to construct a schedule from the front and from the back at the same time. Grabowski's [1982] *block approach* obtains a complete feasible schedule at each node and bases the branching decision on an analysis of the transformations required to shorten the critical path that determines the schedule length. Grabowski, Skubalska & Smutnicki [1983] extend these ideas to the $F|r_j|L_{\max}$ problem.

Not much has been done in the way of worst-case analysis of *approximation algorithms* for the flow shop scheduling problem. It is not hard to see that for any active schedule (AS)

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

Gonzalez & Sahni [1978a] show that

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

For the case $F2 \mid no\ wait \mid C_{\max}$, the traveling salesman problem assumes a special structure, and results due to Gilmore & Gomory [1964] can be applied to yield an $O(n^2)$ algorithm; see Reddi & Ramamoorthy [1972] and also Gilmore, Lawler & Shmoys [1985]. In contrast, $F4 \mid no\ wait \mid C_{\max}$ is strongly NP-hard [Papadimitriou & Kanellakis, 1980], and so is $F3 \mid no\ wait \mid C_{\max}$ [Röck, 1984a]. The same is true for $F2 \mid no\ wait \mid L_{\max}$ and $F2 \mid no\ wait \mid \Sigma C_j$ [Röck, 1984b], and for $O2 \mid no\ wait \mid C_{\max}$ and $J2 \mid no\ wait \mid C_{\max}$ [Sahni & Cho, 1979a]. Goyal & Sriskandarajah [1988] review complexity results and approximation algorithms for this problem class.

The *no wait* constraint may lengthen the optimal flow shops schedule considerably. Lenstra [-] shows that

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

## 14. Job shops

### 14.0. The disjunctive graph model for $J \mid\ \mid C_{\max}$

The description of $J \mid\ \mid C_{\max}$ in Section 3 does not reveal much of the structure of this problem type. An illuminating problem representation is provided by the disjunctive graph model due to Roy & Sussmann [1964].

Given an instance of $J \mid\ \mid C_{\max}$, the corresponding disjunctive graph is defined as follows. For every operation $O_{ij}$, there is a vertex, with a weight $p_{ij}$. For every two consecutive operations of the same job, there is a (directed) arc. For every two operations that require the same machine, there is an (undirected) edge. Thus, the arcs represent the job precedence constraints, and the edges represent the machine capacity constraints.

The basic scheduling decision is to impose an ordering on a pair of operations on the same machine. In the disjunctive graph, this corresponds to orienting the edge in question, in one way or the other. A schedule is obtained by orienting all of the edges. The schedule is feasible if the resulting directed graph is acyclic, and its length is obviously equal to the weight of maximum weight path in this graph.

The job shop scheduling problem has now been formulated as the problem of finding an orientation of the edges of a disjunctive graph that minimizes the maximum path weight. We refer to Figure 4 for an example.

### 14.1. Two or three machines

A simple extension of Johnson's algorithm for $F2 \mid\ \mid C_{\max}$ allows solution of $J2 \mid m_j \leq 2 \mid C_{\max}$ 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 let $\mathcal{J}_{hi}$ be the set of jobs that go from $M_h$ to $M_i$ $(\{h, i\} = \{1, 2\})$. Order the latter two sets by means of Johnson's algorithm and the former two sets arbitrarily. One then obtains an optimal

edge in a certain subset $E' \subset E$. The question then is how to compute a lower bound on the value of all completions of this partial solution. Németi [1964], Charlton & Death [1970] and Schrage [1970] are among the researchers who obtain a lower bound by simply disregarding $E - E'$ and computing the maximum path weight in the directed graph $(\mathcal{O}, A \cup E')$. A more sophisticated bound, due to Bratley, Florian & Robillard [1973], is based on the relaxation of the capacity constraints of all machines except one. They propose to select a machine $M'$ and to solve the job shop scheduling problem on the disjunctive graph $(\mathcal{O}, A \cup E', \{\{O_{ij}, O_{i'j'}\} \mid \mu_{ij} = \mu_{i'j'} = M'\})$. This is a single-machine problem, where the arcs in $A \cup E'$ define release times and delivery times for the operations that are to be scheduled on machine $M'$. This observation has spawned the subsequent work on the $1 \mid r_j \mid L_{\max}$ problem which was reviewed in Section 4.2 and which has led to fast methods for its solution. As pointed out by Lageweg, Lenstra & Rinnooy Kan [1977], the lower bound problem is, in fact, $1 \mid prec, r_j \mid L_{\max}$, since $A \cup E'$ may define precedence constraints among the operations on $M'$. Again, most other lower bounds appear as special cases of this one, by relaxing the capacity constraint of $M'$ (which gives Németi's longest path bound), by underestimating the contribution of the release and delivery times, by allowing preemption, or by ignoring the precedence constraints. These relaxations, with the exception of the last one, turn an NP-hard single-machine problem into a problem that is solvable in polynomial time.

Fisher, Lageweg, Lenstra & Rinnooy Kan [1983] investigate surrogate duality relaxations, in which either the capacity constraints of the machines or the precedence constraints among the operations of each job are weighted and aggregated into a single constraint. In theory, the resulting bounds dominate the above single-machine bound. Balas [1985] describes a first attempt to obtain bounds by polyhedral techniques.

The usual enumeration scheme is due to Giffler & Thompson [1960]. It generates all active schedules by constructing them from front to back. At each stage, the subset $\mathcal{O}'$ of operations $O_{ij}$ all of whose predecessors have been scheduled is determined and their earliest possible completion times $r_{ij} + p_{ij}$ are calculated. It suffices to consider only a machine on which the minimum value of $r_{ij} + p_{ij}$ is achieved and to branch by successively scheduling next on that machine all operations in $\mathcal{O}'$ for which the release time is strictly smaller than this minimum. In this scheme, several edges are oriented at each stage.

Lageweg, Lenstra & Rinnooy Kan [1977] and Carlier & Pinson [1988] describe alternative enumeration schemes whereby at each stage, a single edge is selected and oriented in either of two ways. Barker & McMahon [1985] branch by rearranging the operations in a critical block that occurs on the maximum weight path.

We briefly outline three of the many implemented branch and bound algorithms for job shop scheduling. McMahon & Florian [1975] combine the Giffler–Thompson enumeration scheme with the $1 \mid r_j \mid L_{\max}$ bound, which is computed for all machines by their own algorithm. Lageweg [1984] applies the same branching rule, computes the $1 \mid prec, r_j \mid L_{\max}$ bound only for a few

for other hard problems. However, the investigations in this direction are still at an initial stage.

Dispatch rules show an erratic behavior. The rule proposed by Lageweg, Lenstra & Rinnooy Kan [1977] constructs a schedule of length 1082, and most other priority functions do worse.

Adams, Balas & Zawack [1988] report that their sliding bottleneck heuristic obtains a schedule of length 1015 in ten CPU seconds, solving 249 single-machine problems on the way. Their partial enumeration procedure succeeds in finding the optimum, after 851 seconds and 270 runs of the first heuristic.

Five runs of the simulated annealing algorithm of Van Laarhoven, Aarts & Lenstra [1992], with a standard setting of the cooling parameters, take 6000 seconds on average and produce an average schedule length of 942.4, with a minimum of 937. If 6000 seconds are spent on deterministic neighborhood search, which accepts only true improvements, more than 9000 local optima are found, the best one of which has a value of 1006. Five runs with a much slower cooling schedule take about 16 hours each and produce solution values of 930 (twice), 934, 935 and 938. In comparison to other approximative approaches, simulated annealing requires unusual computation times, but it yields consistently good solutions with a modest amount of human implementation effort and relatively little insight into the combinatorial structure of the problem type under consideration.

## PART V. MORE SEQUENCING AND SCHEDULING

In the preceding sections, we have been exclusively concerned with the class of deterministic machine scheduling problems. Several extensions of this class are worthy of further investigation. A natural extension involves the presence of additional resources, where each resource has a limited size and each job requires the use of a part of each resource during its execution. The resulting *resource-constrained project scheduling* problems are considered in Section 15. We also may relax the assumption that all problem data are known in advance and investigate *stochastic machine scheduling* problems. This class is the subject of Section 16. We will not enter the area of *stochastic project scheduling*, which is surveyed by Möhring & Radermacher [1985b].

### 15. Resource-constrained project scheduling

#### *15.0. A matching formulation for $P2 \mid p_j = 1 \mid C_{\max}$ with resource constraints*

Consider a single-operation model, and suppose there are $l$ additional resources $R_h$ $(h = 1, \ldots, l)$. For each resource $R_h$, there is a *size* $s_h$, which is the amount of $R_h$ available at any time. For each resource $R_h$ and each job $J_j$, there is a *requirement* $r_{hj}$, which is the amount of $R_h$ required by $J_j$ at all times

– if $\sigma$ a positive integer, then all $s_h$ are constants, equal to $\sigma$; if $\sigma = \cdot$, then the $s_h$ are part of the input;

– if $\rho$ is a positive integer, then all $r_{hj}$ have a constant upper bound, equal to $\rho$; if $\rho = \cdot$, then no such bounds are specified.

Blazewicz, Lenstra & Rinnooy Kan investigate the computational complexity of $Q \mid res \cdots, prec, p_j = 1 \mid C_{\max}$ and its special cases. The resulting exhaustive complexity classification is presented in Figure 5. We have already seen in Section 15.0 that $P2 \mid res \cdots, p_j = 1 \mid C_{\max}$ is solvable by matching techniques. Also note that $P3 \mid res\, 1 \cdot \cdot, p_j = 1 \mid C_{\max}$ is an immediate generalization of the

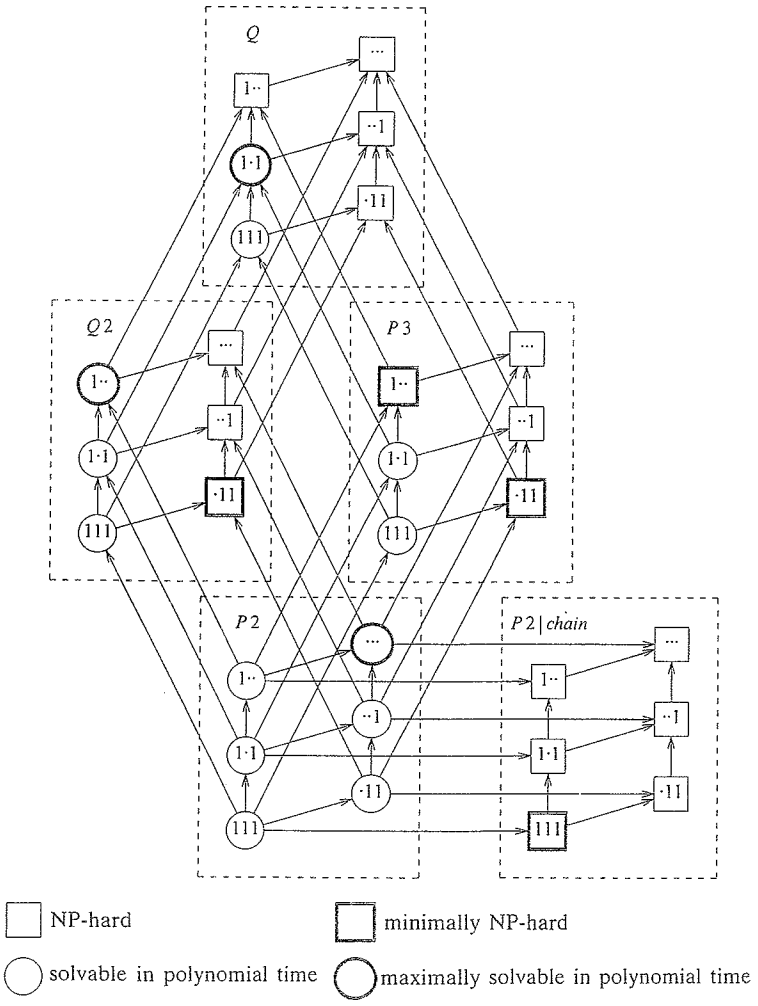



Fig. 5. Complexity of scheduling unit-time jobs on parallel machines subject to resource constraints.

presupposes constant resource availability over time. In addition, they generalize the traditional precedence constraints of the form

$$C_j + p_j \leq C_k \quad \text{whenever } J_j \rightarrow J_k$$

to *temporal constraints* of the form

$$C_j + d_{jk} \leq C_k \quad \text{for all } J_j, J_k ,$$

where $d_{jk}$ is a (possibly negative) *distance* from $J_j$ to $J_k$. The resulting model is quite general. It allows for the specification of job release dates and deadlines, of minimal and maximal time lags between jobs, and of time-dependent resource consumption per job.

The investigation of this model leads to structural insights as well as computational methods. This is also true for the related model involving traditional precedence constraints [Radermacher, 1985/6] and for the dual model in which resource consumption is to be minimized subject to a common job deadline [Möhring, 1984]. The approach leads to new classes of polynomially solvable problems that are characterized by the structure of the family of forbidden subsets [Möhring, 1983]. For the general model, it can be shown that for any optimality criterion that is nondecreasing in the job completion times, attention can be restricted to left-justified schedules. Enumerative methods can be designed that, as in the case of $J \mid \mid C_{\max}$, construct feasible schedules by adding at least one precedence constraint among the jobs in each forbidden subset.

In the case of job shop scheduling, the number of edges is $O(n^2)$. Similarly, the present model is only computationally feasible when the number of forbidden subsets is not too large. It is sufficient if $\mathcal{N}$ contains only those forbidden subsets that are minimal under set inclusion. A branch and bound method that branches by successively considering all possibilities to eliminate a particular forbidden subset and obtains lower bounds by simply computing a longest path with respect to the augmented temporal constraints, compares favorably with the integer programming algorithm of Talbot & Patterson [1978].

## 16. Stochastic machine scheduling

### *16.0. List scheduling for* $P \mid p_j \sim \exp(\lambda_j) \mid E C_{\max}, E \Sigma C_j$

Suppose that $m$ identical parallel machines have to process $n$ independent jobs. In contrast to what we have assumed so far, the processing times are not given beforehand but become known only after the jobs have been allocated to the machines. More specifically, each processing time $p_j$ follows an exponential distribution with parameter $\lambda_j$, for $j = 1, \ldots, n$. We want to minimize the

then $F_\pi(N) > F_{\pi'}(N)$. After at most $m$ interchanges, the policy applied at time 0 is the LEPT rule, and we have that $F_\pi(N) > F_{\mathrm{LEPT}}(N)$.

It is interesting to note that, while $P \mid\mid C_{\max}$ is NP-hard, a stochastic variant of the problem is solvable in polynomial time. As observed above, LEPT should be viewed as an algorithm for the preemptive problem, and preemptive scheduling in a deterministic setting is not hard either. Indeed, for the case of uniform machines, Weiss & Pinedo [1980] prove that a preemptive LEPT (SEPT) policy, which allows reallocation of jobs to machines at job completion times, solves $Q \mid pmtn, \boldsymbol{p}_j \sim \exp(\lambda_j) \mid \mathrm{E}\boldsymbol{C}_{\max}$ $(\mathrm{E}\Sigma\boldsymbol{C}_j)$.

### 16.1. Deterministic and stochastic data

The scheduling models discussed in the earlier sections are based on the assumption that all problem data are known in advance. This assumption is not always justified. Processing times may be subject to fluctuations, and job arrivals and machine breakdowns are often random events.

A substantial literature exists in which scheduling problems are considered from a probabilistic perspective. A deterministic scheduling model may give rise to various stochastic counterparts, as there is a choice in the parameters that are randomized, in their distributions, and in the classes of policies that can be applied. A characteristic feature of these models is that the stochastic parameters are regarded as independent random variables with a given distribution and that their realization occurs only after the scheduling decision has been made.

Surprisingly, there are many cases where a simple rule which is merely a heuristic for the deterministic model has a stochastic reformulation which solves the stochastic model to optimality; we have seen an example in the previous section. In the deterministic model, one has perfect information, and capitalizing on it in minimizing the realization of a performance measure may require exponential time. In the stochastic model, one has imperfect information, and the problem of minimizing the expectation of a performance measure may be computationally tractable. In such cases, the scheduling decision is based on distributional information such as first and second moments. In general, however, optimal policies may be dynamic and require information on the history up to the current point in time.

Results in this area are technically complicated; they rely on semi-Markovian decision theory and stochastic dynamic optimization. Within the scope of this section, it is not possible to do full justice to the literature. We present some typical results for the main types of machine environments below, concentrating on scheduling models with random processing times. We refer to Pinedo [1983] for scheduling with random release and due dates, to Pinedo & Rammouz [1988] and Birge, Frenk, Mittenthal & Rinnooy Kan [1990] for single-machine scheduling with random breakdowns, and to the surveys by Pinedo & Schrage [1982], Weiss [1982], Forst [1984], Pinedo [1984], Möhring,

mum completion time on two identical machines if all the jobs at the same level have the same parameter. Frostig [1988] extends this work.

Pinedo & Weiss [1987] investigate the case of identical expected processing times. Their result confirms the intuition that, at least for some simple distributions, the jobs with the largest variance should be scheduled first.

### 16.4. Multi-operation models

Pinedo's [1984] survey is a good source of information on stochastic shop scheduling. Most work has concentrated on flow shops; Pinedo & Weiss [1984] deal with some stochastic variants of the Gonzalez-Sahni [1976] algorithm for $O2 \mid \mid C_{\max}$ (see Section 12.0).

Brumelle & Sidney [1982] show that Johnson's [1954] algorithm for $F2 \mid \mid C_{\max}$ also applies to the exponential case. If $p_{1j} \sim \exp(\lambda_j)$ and $p_{2j} \sim \exp(\mu_j)$, then sequencing in order of nonincreasing $\lambda_j - \mu_j$ minimizes the expected maximum completion time.

For $F \mid \mid C_{\max}$, it is usually assumed that the $p_{ij}$ are independent random variables whose distributions do not depend on $i$. Weber [1979] shows that, in the exponential case, any sequence minimizes $EC_{\max}$. Pinedo [1982] observes that, under fairly general conditions, any sequence for which $Ep_{ij}$ is first nondecreasing and then nonincreasing is optimal; as a rule of thumb, jobs with smaller expected processing time and larger variance should come at the beginning or at the end of a schedule, with the others occupying the middle part. These observations carry over to the model in which no intermediate storage is available, so that a job can only leave a machine when its next machine is available. We refer to Foley & Suresh [1986] and Wie & Pinedo [1986] for more recent work on the latter model, and to Boxma & Forst [1986] for a result on a stochastic version of $F \mid \mid \Sigma U_j$.

Not surprisingly, job shops pose even greater challenges. The only successful analysis has been carried out by Pinedo [1981] for an exponential variant of $J2 \mid m_j \leq 2 \mid C_{\max}$ (see Section 14.1).

The results in stochastic scheduling are scattered, and they have been obtained through a considerable and sometimes disheartening effort. In the words of Coffman, Hofri & Weiss [1989], 'there is a great need for new mathematical techniques useful for simplifying the derivation of results'.

**Acknowledgements**

We gratefully acknowledge the contribution of Leen Stougie to Section 16.0, the comments and suggestions of Leslie Hall, Ben Lageweg, Michael Langston, Joseph Leung, Rolf Möhring, Michael Pinedo, Chris Potts, Steef van de Velde, and Gideon Weiss, and the help of Gerard Kindervater in preparing the figures. The research of the first author was partially supported by the National Science Foundation under grant CCR-8704184. This paper was written when

Bartusch, M., R.H. Möhring and F.J. Radermacher (1988b). Scheduling project networks with resource constraints and time windows. *Ann. Oper. Res.* 16, 201–240.

Belouadah, H., M.E. Posner and C.N. Potts (1989). A branch and bound algorithm for scheduling jobs with release dates on a single machine to minimize total weighted completion time, Preprint OR14, Faculty of Mathematical Studies, University of Southampton.

Belov, I.S., and J.N. Stolin (1974). An algorithm for the single-route scheduling problem, (in Russian), in: *Mathematical Economics and Functional Analysis*, Nauka, Moscow, pp. 248–257.

Bianco, L., and S. Ricciardelli (1982). Scheduling of a single machine to minimize total weighted completion time subject to release dates. *Naval Res. Logist. Quart.* 29, 151–167.

Birge, J., J.B.G. Frenk, J. Mittenthal and A.H.G. Rinnooy Kan (1990). Single machine scheduling subject to stochastic breakdowns. *Naval Res. Logist.* 37, 661–677.

Blau, R.A. (1973). *N*-job, one machine sequencing problems under uncertainty. *Management Sci.* 20, 101–109.

Blazewicz, J. (1987). Selected topics in scheduling theory. *Ann. Discrete Math.* 31, 1–60.

Blazewicz, J., G. Finke, R. Haupt and G. Schmidt (1988). New trends in machine scheduling. *European J. Oper. Res.* 37, 303–317.

Blazewicz, J., J.K. Lenstra and A.H.G. Rinnooy Kan (1983). Scheduling subject to resource constraints: classification and complexity. *Discrete Appl. Math.* 5, 11–24.

Boxma, O.J. (1984). Probabilistic analysis of the LPT scheduling rule, in: E. Gelenbe (ed.) *Performance '84*, North-Holland, Amsterdam, pp. 475–490.

Boxma, O.J., and F.G. Forst (1986). Minimizing the expected weighted number of tardy jobs in stochastic flow shops. *Oper. Res. Lett.* 5, 119–126.

Bratley, P., M. Florian and P. Robillard (1973). On sequencing with earliest starts and due dates with application to computing bounds for the $(n/m/G/F_{max})$ problem. *Naval Res. Logist. Quart.* 20, 57–67.

Bratley, P., M. Florian and P. Robillard (1975). Scheduling with earliest start and due date constraints on multiple machines. *Naval Res. Logist. Quart.* 22, 165–173.

Brucker, P. (1981). Minimizing maximum lateness in a two-machine unit-time job shop. *Computing* 27, 367–370.

Brucker, P. (1982). A linear time algorithm to minimize maximum lateness for the two-machine, unit-time, job-shop, scheduling problem, in: R.F. Drenick and F. Kozin (eds.), *System Modeling and Optimization*, Lecture Notes in Control and Information Sciences, Vol. 38, Springer, Berlin, pp. 566–571.

Brucker, P., M.R. Garey and D.S. Johnson (1977). Scheduling equal-length tasks under tree-like precedence constraints to minimize maximum lateness. *Math. Oper. Res.* 2, 275–284.

Brumelle, S.L., and J.B. Sidney (1982). The two machine makespan problem with stochastic flow times, Technical Report, Univ. of British Columbia, Vancouver.

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

Bruno, J.L., and P.J. Downey (1977). Sequencing tasks with exponential service times on two machines, Technical Report, Department of Electrical Engineering and Computer Science, Univ. of California, Santa Barbara.

Bruno, J.L., and P.J. Downey (1986). Probabilistic bounds on the performance of list scheduling. *SIAM J. Comput.* 15, 409–417.

Bruno, J.L., P.J. Downey and G.N. Frederickson (1981). Sequencing tasks with exponential service times to minimize the expected flowtime or makespan. *J. Assoc. Comput. Mach.* 28, 100–113.

Bruno, J.L., 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.

Buer, H., and R.H. Möhring (1983). A fast algorithm for the decomposition of graphs and posets. *Math. Oper. Res.* 8, 170–184.

Campbell, H.G., R.A. Dudek and M.L. Smith (1970). A heuristic algorithm for the $n$ job, $m$ machine sequencing problem. *Management Sci.* 16B, 630–637.

ı J. Oper. Res. 11, 42–47.
dentical machines to minimize

b-shop problem. *Management*

eduling algorithm. *Oper. Res.*

›rocessing computing systems,
r Science, Univ. of Illinois at

lgorithms for multiprocessors
Lecture Notes in Computer

research involving due date

low-shop schedules. *J. Assoc.*

processors. *SIAM J. Comput.*

ɜnt jobs with release and due

ject scheduling with resource
ɾes. 29, 262–273.
*Theory*, Wiley, New York.
(1987). Minimizing expected
. 19, 177–201.
ed makespans for largest-fit
, North-Holland, Amsterdam,

application of bin-packing to

relative performance of list

›r two-processor systems. *Acta*

chastic jobs with a two point
ɔi. 3, 89–116.
). Asymptotic methods in the
*gement Sci.* 34, 266–290.
*Scheduling*, Addison-Wesley,

res, *Proc. 3rd Annual ACM*

with random processing times

euristics. *Management Sci.* 23,

duling of tree structured tasks.

s on unrelated processors. *J.*

– A survey. *J. Indust. Engrg.*

Historical review and categori-

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

Dempster, M.A.H., J.K. Lenstra and A.H.G. Rinnooy Kan (eds.) (1982). *Deterministic and Stochastic Scheduling*, Reidel, Dordrecht.

Dessouky, M.I., and J.S. Deogun (1981). Sequencing jobs with unequal ready times to minimize mean flow time. *SIAM J. Comput.* 10, 192–202.

Dessouky, M.I., B.J. Lageweg, J.K. Lenstra and S.L. Van de Velde (1990). Scheduling identical jobs on uniform parallel machines. *Statist. Neerlandica* 44, 115–123.

Dileepan, P., and T. Sen (1988). Bicriterion static scheduling research for a single machine. *Omega* 16, 53–59.

Dobson, G. (1984). Scheduling independent tasks on uniform processors. *SIAM J. Comput.* 13, 705–716.

Dolev, D., and M.K. Warmuth (1984). Scheduling precedence graphs of bounded height. *J. Algorithms* 5, 48–59.

Dolev, D., and M.K. Warmuth (1985a). Scheduling flat graphs. *SIAM J. Comput.* 14, 638–657.

Dolev, D., and M.K. Warmuth (1985b). Profile scheduling of opposing forests and level orders. *SIAM J. Alg. Disc. Meth.* 6, 665–687.

Du, J., and J.Y.-T. Leung (1988a). Scheduling tree-structured tasks with restricted execution times. *Inform. Process. Lett.* 28, 183–188.

Du, J., and J.Y.-T. Leung (1988b). Minimizing mean flow time with release time and deadline constraints, Technical Report, Computer Science Program, Univ. of Texas, Dallas.

Du, J., and J.Y.-T. Leung (1989). Scheduling tree-structured tasks on two processors to minimize schedule length. *SIAM J. Discrete Math.* 2, 176–196.

Du, J., and J.Y.-T. Leung (1990). Minimizing total tardiness on one machine is NP-hard. *Math. Oper. Res.* 15, 483–495.

Du, J., J.Y.-T. Leung and C.S. Wong (1989). Minimizing the number of late jobs with release time constraints, Technical Report, Computer Science Program, Univ. of Texas, Dallas.

Du, J., J.Y.-T. Leung and G.H. Young (1988). Minimizing mean flow time with release time constraint, Technical Report, Computer Science Program, Univ. of Texas, Dallas.

Du, J., J.Y.-T. Leung and G.H. Young (1991). Scheduling chain-structured tasks to minimize makespan and mean flow time. *Inform. and Comput.* 92, 219–236.

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

Edmonds, J. (1965). Minimum partition of a matroid into independent subsets. *J. Res. Nat. Bur. Standards* 69B, 67–72.

Elmaghraby, S.E. (1968). The one-machine sequencing problem with delay costs. *J. Indust. Engrg.* 19, 105–108.

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

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

Erschler, J., G. Fontan, C. Merce and F. Roubellat (1982). Applying new dominance concepts to job schedule optimization. *European J. Oper. Res.* 11, 60–66.

Erschler, J., G. Fontan, C. Merce and F. Roubellat (1983). A new dominance concept in scheduling $n$ jobs on a single machine with ready times and due dates. *Oper. Res.* 31, 114–127.

Federgruen, A., and H. Groenevelt (1986). Preemptive scheduling of uniform machines by ordinary network flow techniques. *Management Sci.* 32, 341–349.

Fiala, T. (1983). An algorithm for the open-shop problem. *Math. Oper. Res.* 8, 100–109.

Fischetti, M., and S. Martello (1987). Worst-case analysis of the differencing method for the partition problem. *Math. Programming* 37, 117–120.

Fisher, H., and G.L. Thompson (1963). Probabilistic learning combinations of local job-shop scheduling rules, in: J.F. Muth and G.L. Thompson (eds.), *Industrial Scheduling*, Prentice-Hall, Englewood Cliffs, NJ, pp. 225–251.

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

Garey, M.R., D.S. Johnson, R.E. Tarjan and M. Yannakakis (1983). Scheduling opposing forests. *SIAM J. Alg. Disc. Meth.* 4, 72–93.
Garey, M.R., R.E. Tarjan and G.T. Wilfong (1988). One-processor scheduling with symmetric earliness and tardiness penalties. *Math. Oper. Res.* 13, 330–348.
Gazmuri, P.G. (1985). Probabilistic analysis of a machine scheduling problem. *Math. Oper. Res.* 10, 328–339.
Gelders, L., and P.R. Kleindorfer (1974). Coordinating aggregate and detailed scheduling decisions in the one-machine job shop: Part I. Theory. *Oper. Res.* 22, 46–60.
Gelders, L., and P.R. Kleindorfer (1975). Coordinating aggregate and detailed scheduling in the one-machine job shop: II – Computation and structure. *Oper. Res.* 23, 312–324.
Gens, G.V., and E.V. Levner (1978). Approximative algorithms for certain universal problems in scheduling theory. *Engrg. Cybernet.* 16(6), 31–36.
Gens, G.V., and E.V. Levner (1981). Fast approximation algorithm for job sequencing with deadlines. *Discrete Appl. Math.* 3, 313–318.
Gere, W.S. (1966). Heuristics in job shop scheduling. *Management Sci.* 13, 167–190.
Giffler, B., and G.L. Thompson (1960). Algorithms for solving production-scheduling problems. *Oper. Res.* 8, 487–503.
Gilmore, P.C., and R.E. Gomory (1964). Sequencing a one-state variable machine: A solvable case of the traveling salesman problem. *Oper. Res.* 12, 655–679.
Gilmore, P.C., E.L. Lawler and D.B. Shmoys (1985). Well-solvable cases, in: Lawler, Lenstra, Rinnooy Kan & Shmoys [1985], Chapter 4.
Gonzalez, T. (1977). Optimal mean finish time preemptive schedules, Technical Report 220, Computer Science Department, Pennsylvania State University.
Gonzalez, T. (1979). A note on open shop preemptive schedules. *IEEE Trans. Comput.* C-28, 782–786.
Gonzalez, T. (1982). Unit execution time shop problems. *Math. Oper. Res.* 7, 57–66.
Gonzalez, T., O.H. Ibarra, and S. Sahni (1977). Bounds for LPT schedules on uniform processors. *SIAM J. Comput.* 6, 155–166.
Gonzalez, T., and D.B. Johnson (1980). A new algorithm for preemptive scheduling of trees. *J. Assoc. Comput. Mach.* 27, 287–312.
Gonzalez, T., E.L. Lawler and S. Sahni (1990). Optimal preemptive scheduling of two unrelated processors. *ORSA J. Comput.* 2, 219–224.
Gonzalez, T., and S. Sahni (1976). Open shop scheduling to minimize finish time. *J. Assoc. Comput. Mach.* 23, 665–679.
Gonzalez, T., and S. Sahni (1978a). Flowshop and jobshop schedules: Complexity and approximation. *Oper. Res.* 26, 36–52.
Gonzalez, T., and S. Sahni (1978b). Preemptive scheduling of uniform processor systems. *J. Assoc. Comput. Mach.* 25, 92–101.
Goyal, D.K. (1977). Non-preemptive scheduling of unequal execution time tasks on two identical processors. Technical Report CS-77-039, Computer Science Department, Washington State University, Pullman.
Goyal, S.K., and C. Sriskandarajah (1988). No-wait shop scheduling: Computational complexity and approximate algorithms. *Opsearch* 25, 220–244.
Grabowski, J. (1980). On two-machine scheduling with release dates to minimize maximum lateness. *Opsearch* 17, 133–154.
Grabowski, J. (1982). A new algorithm of solving the flow-shop problem, in: G. Feichtinger and P. Kall (eds.), *Operations Research in Progress*, Reidel, Dordrecht, pp. 57–75.
Grabowski, J., E. Skubalska and C. Smutnicki (1983). On flow shop scheduling with release and due dates to minimize maximum lateness. *J. Oper. Res. Soc.* 34, 615–620.
Graham, R.L. (1966). Bounds for certain multiprocessing anomalies. *Bell System Tech. J.* 45, 1563–1581.
Graham, R.L. (1969). Bounds on multiprocessing timing anomalies. *SIAM J. Appl. Math.* 17, 416–429.
Graham, R.L. (-). Unpublished.

Johnson, D.S. (1983). The NP-completeness column: An ongoing guide. *J. Algorithms* 4, 189–203.
Johnson, S.M. (1954). Optimal two- and three-stage production schedules with setup times included. *Naval Res. Logist. Quart.* 1, 61–68.
Johnson, S.M. (1958). Discussion: Sequencing $n$ jobs on two machines with arbitrary time lags. *Management Sci.* 5, 299–303.
Kao, E.P.C., and M. Queyranne (1982). On dynamic programming methods for assembly line balancing. *Oper. Res.* 30, 375–390.
Karmarkar, N., and R.M. Karp (1982). The differencing method of set partitioning, Report UCB/CSD 82/113, Computer Science Division, Univ. of California, Berkeley.
Karp, R.M. (1972). Reducibility among combinatorial problems, in: R.E. Miller, and J.W. Thatcher (eds.) (1972). *Complexity of Computer Computations*, Plenum Press, New York, pp. 85–103.
Karp, R.M. (1975). On the computational complexity of combinatorial problems. *Networks* 5, 45–68.
Kaufman, M.T. (1974). An almost-optimal algorithm for the assembly line scheduling problem. *IEEE Trans. Comput.* C-23, 1169–1174.
Kawaguchi, T., and S. Kyan (1986). Worst case bound of an LRF schedule for the mean weighted flow-time problem. *SIAM J. Comput.* 15, 1119–1129.
Kawaguchi, T., and S. Kyan (1988). Deterministic scheduling in computer systems: A survey. *J. Oper. Res. Soc. Japan* 31, 190–217.
Khachiyan, L.G. (1979). A polynomial algorithm in linear programming. *Soviet Math. Dokl.* 20, 191–194.
Kise, H., T. Ibaraki, and H. Mine (1978). A solvable case of the one-machine scheduling problem with ready and due times. *Oper. Res.* 26, 121–126.
Kise, H., T. Ibaraki, and H. Mine (1979). Performance analysis of six approximation algorithms for the one-machine maximum lateness scheduling problem with ready times. *J. Oper. Res. Soc. Japan* 22, 205–224.
Kohler, W.H., and K. Steiglitz (1975). Exact, approximate and guaranteed accuracy algorithms for the flow-shop problem $n/2/F/\bar{F}$. *J. Assoc. Comput. Mach.* 22, 106–114.
Kumar, P.R., and J. Walrand (1985). Individually optimal routing in parallel systems. *J. Appl. Probab.* 22, 989–995.
Kunde, M. (1976). Beste Schranken beim LP-Scheduling, Bericht 7603, Institut für Informatik und Praktische Mathematik, Universität Kiel.
Kunde, M. (1981). Nonpreemptive LP-scheduling on homogeneous multiprocessor systems. *SIAM J. Comput.* 10, 151–173.
Labetoulle, J., E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan (1984). Preemptive scheduling of uniform machines subject to release dates, in: Pulleyblank [1984], pp. 245–261.
Lageweg, B.J. (1984). Private communication.
Lageweg, B.J. (-). Unpublished.
Lageweg, B.J., E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan (1981). Computer aided complexity classification of deterministic scheduling problems, Report BW 138, Centre for Mathematics and Computer Science, Amsterdam.
Lageweg, B.J., J.K. Lenstra, E.L. Lawler and A.H.G. Rinnooy Kan (1982). Computer-aided complexity classification of combinatorial problems. *Commun. ACM* 25, 817–822.
Lageweg, B.J., J.K. Lenstra and A.H.G. Rinnooy Kan (1976). Minimizing maximum lateness on one machine: computational experience and some applications. *Statist. Neerlandica* 30, 25–41.
Lageweg, B.J., J.K. Lenstra and A.H.G. Rinnooy Kan (1977). Job-shop scheduling by implicit enumeration. *Management Sci.* 24, 441–450.
Lageweg, B.J., J.K. Lenstra and A.H.G. Rinnooy Kan (1978). A general bounding scheme for the permutation flow-shop problem. *Oper. Res.* 26, 53–67.
Lam, S., and R. Sethi (1977). Worst case analysis of two scheduling algorithms. *SIAM J. Comput.* 6, 518–536.
Larson, R.E., M.I. Dessouky and R.E. Devor (1985). A forward-backward procedure for the single machine problem to minimize maximum lateness. *IIE Trans.* 17, 252–260.

Lenstra, J.K., and A.H.G. Rinnooy Kan (1979). Computational complexity of discrete optimization problems. *Ann. Discrete Math.* 4, 121–140.
Lenstra, J.K., and A.H.G. Rinnooy Kan (1980). Complexity results for scheduling chains on a single machine. *European J. Oper. Res.* 4, 270–275.
Lenstra, J.K., and A.H.G. Rinnooy Kan (1984). New directions in scheduling theory. *Oper. Res. Lett.* 2, 255–259.
Lenstra, J.K., and A.H.G. Rinnooy Kan (1985). Sequencing and scheduling, in: O'hEigeartaigh, Lenstra & Rinnooy Kan [1985], pp. 164–189.
Lenstra, J.K., A.H.G. Rinnooy Kan and P. Brucker (1977). Complexity of machine scheduling problems. *Ann. Discrete Math.* 1, 343–362.
Lenstra, J.K., D.B. Shmoys and E. Tardos (1990). Approximation algorithms for scheduling unrelated parallel machines. *Math. Programming* 46, 259–271.
Leung, J.Y.-T. (1991). Bin packing with restricted piece sizes. *Inform. Process. Lett.* 31, 145–149.
Leung, J.Y.-T., and G.H. Young (1989). Minimizing total tardiness on a single machine with precedence constraints, Technical Report, Computer Science Program, Univ. of Texas, Dallas.
Levin, L.A. (1973). Universal sequential search problems. *Problemy Peredachi Informatsii* 9, 115–116. English translation: *Problems Inform. Transmission* 9 (1975) 265–266.
Liu, C.Y., and R.L. Bulfin (1985). On the complexity of preemptive open-shop scheduling problems. *Oper. Res. Lett.* 4, 71–74.
Liu, J.W.S., and C.L. Liu (1974a). Bounds on scheduling algorithms for heterogeneous computing systems, in: J.L. Rosenfeld (ed.), *Information Processing 74*, North-Holland, Amsterdam, pp. 349–353.
Liu, J.W.S., and C.L. Liu (1974b). Bounds on scheduling algorithms for heterogeneous computing systems, Technical Report UIUCDCS-R-74-632, Department of Computer Science, Univ. of Illinois at Urbana-Champaign, 68 pp.
Liu, J.W.S., and C.L. Liu (1974c). Performance analysis of heterogeneous multi-processor computing systems, in: E. Gelenbe and R. Mahl (eds.), *Computer Architectures and Networks*, North-Holland, Amsterdam, pp. 331–343.
Loulou, R. (1984). Tight bounds and probabilistic analysis of two heuristics for parallel processor scheduling. *Math. Oper. Res.* 9, 142–150.
Marcotte, O., and L.E. Trotter Jr (1984). An application of matroid polyhedral theory to unit-execution time, tree-precedence constrained job scheduling, in: Pulleyblank [1984], pp. 263–271.
Martel, C.U. (1982). Preemptive scheduling with release times, deadlines and due times. *J. Assoc. Comput. Mach.* 29, 812–829.
Matsuo, H., C.J. Suh and R.S. Sullivan (1988). A controlled search simulated annealing method for the general jobshop scheduling problem, Working Paper 03-44-88, Graduate School of Business, Univ. of Texas, Austin.
Maxwell, W.L. (1970). On sequencing $n$ jobs on one machine to minimize the number of late jobs. *Management Sci.* 16, 295–297.
McCormick, S.T., and M.L. Pinedo (1989). Scheduling $n$ independent jobs on $m$ uniform machines with both flow time and makespan objectives: A parametric analysis, Department of Industrial Engineering and Operations Research, Columbia University, New York.
McMahon, G.B. (1969). Optimal production schedules for flow shops. *Canad. Oper. Res. Soc. J.* 7, 141–151.
McMahon, G.B. (1971). A study of algorithms for industrial scheduling problems, Ph.D. Thesis, Univ. of New South Wales, Kensington.
McMahon, G.B., and M. Florian (1975). On scheduling with ready times and due dates to minimize maximum lateness. *Oper. Res.* 23, 475–482.
McNaughton, R. (1959). Scheduling with deadlines and loss functions. *Management Sci.* 6, 1–12.
Meilijson, I., and A. Tamir (1984). Minimizing flow time on parallel identical processors with variable unit processing time. *Oper. Res.* 32, 440–446.
Mitten, L.G. (1958). Sequencing $n$ jobs on two machines with arbitrary time lags. *Management Sci.* 5, 293–298.

Palmer, D.S. (1965). Sequencing jobs through a multi-stage process in the minimum total time – A quick method of obtaining a near optimum. *Oper. Res. Quart.* 16, 101–107.
Panwalkar, S.S., and W. Iskander (1977). A survey of scheduling rules. *Oper. Res.* 25, 45–61.
Papadimitriou, C.H., and P.C. Kannelakis (1980). Flowshop scheduling with limited temporary storage. *J. Assoc. Comput. Mach.* 27, 533–549.
Papadimitriou, C.H., and M. Yannakakis (1979). Scheduling interval-ordered tasks. *SIAM J. Comput.* 8, 405–409.
Papadimitriou, C.H., and M. Yannakakis (1990). Towards an architecture-independent analysis of parallel algorithms. *SIAM J. Comput.* 19, 322–328.
Piehler, J. (1960). Ein Beitrag zum Reihenfolgeproblem. *Unternehmensforsch.* 4, 138–142.
Pinedo, M.L. (1981). A note on the two machine job shop with exponential processing times. *Naval Res. Logist. Quart.* 28, 693–696.
Pinedo, M.L. (1982). Minimizing the expected makespan in stochastic flow shops. *Oper. Res.* 30, 148–162.
Pinedo, M.L. (1983). Stochastic scheduling with release dates and due dates. *Oper. Res.* 31, 559–572.
Pinedo, M.L. (1984). Optimal policies in stochastic shop scheduling. *Ann. Oper. Res.* 1, 305–329.
Pinedo, M.L., and L. Schrage (1982). Stochastic shop scheduling: A survey, in: Dempster, Lenstra & Rinnooy Kan [1982], pp. 181–196.
Pinedo, M.L., and E. Rammouz (1988). A note on stochastic scheduling on a single machine subject to breakdown and repair. *Probab. Engrg. Inform. Sci.* 2, 41–49.
Pinedo, M.L., and G. Weiss (1984). Scheduling jobs with exponentially distributed processing times on two machines with resource constraints. *Management Sci.* 30, 883–889.
Pinedo, M.L., and G. Weiss (1985). Scheduling jobs with exponentially distributed processing times and intree precedence constraints on two parallel machines. *Oper. Res.* 33, 1381–1388.
Pinedo, M.L., and G. Weiss (1987). The 'largest variance first' policy in some stochastic scheduling problems. *Oper. Res.* 35, 884–891.
Posner, M.E. (1985). Minimizing weighted completion times with deadlines. *Oper. Res.* 33, 562–574.
Potts, C.N. (1980a). An adaptive branching rule for the permutation flow-shop problem. *European J. Oper. Res.* 5, 19–25.
Potts, C.N. (1980b). Analysis of a heuristic for one machine sequencing with release dates and delivery times. *Oper. Res.* 28, 1436–1441.
Potts, C.N. (1980c). An algorithm for the single machine sequencing problem with precedence constraints. *Math. Programming Study* 13, 78–87.
Potts, C.N. (1985a). Analysis of a linear programming heuristic for scheduling unrelated parallel machines. *Discrete Appl. Math.* 10, 155–164.
Potts, C.N. (1985b). Analysis of heuristics for two-machine flow-shop sequencing subject to release dates. *Math. Oper. Res.* 10, 576–584.
Potts, C.N. (1985c). A Lagrangean based branch and bound algorithm for single machine sequencing with precedence constraints to minimize total weighted completion time. *Management Sci.* 31, 1300–1311.
Potts, C.N., and L.N. Van Wassenhove (1982). A decomposition algorithm for the single machine total tardiness problem. *Oper. Res. Lett.* 1, 177–181.
Potts, C.N., and L.N. Van Wassenhove (1983). An algorithm for single machine sequencing with deadlines to minimize total weighted completion time. *European J. Oper. Res.* 12, 379–387.
Potts, C.N., and L.N. Van Wassenhove (1985). A branch and bound algorithm for the total weighted tardiness problem. *Oper. Res.* 33, 363–377.
Potts, C.N., and L.N. Van Wassenhove (1987). Dynamic programming and decomposition approaches for the single machine total tardiness problem. *European J. Oper. Res.* 32, 405–414.
Potts, C.N., and L.N. Van Wassenhove (1988). Algorithms for scheduling a single machine to minimize the weighted number of late jobs. *Management Sci.* 34, 843–858.
Pulleyblank W.R. (ed.) (1984). *Progress in Combinatorial Optimization*, Academic Press, New York.

Shmoys, D.B., and É. Tardos (1993). Computational complexity of combinatorial problems, in: R.L. Graham, M. Grötschel, and L. Lovász (eds.), *Handbook in Combinatorics*, North-Holland, Amsterdam.
Shwimer, J. (1972). On the $N$-jobs, one-machine, sequence-independent scheduling problem with tardiness penalties: A branch-and-bound solution. *Management Sci.* 18B, 301–313.
Sidney, J.B. (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, Vol. 86, Springer, Berlin, pp. 393–398.
Sidney, J.B. (1975). Decomposition algorithms for single-machine sequencing with precedence relations and deferral costs. *Oper. Res.* 23, 283–298.
Sidney, J.B. (1979). The two-machine maximum flow time problem with series parallel precedence relations. *Oper. Res.* 27, 782–791.
Sidney, J.B. (1981). A decomposition algorithm for sequencing with general precedence constraints. *Math. Oper. Res.* 6, 190–204.
Sidney, J.B., and G. Steiner (1986). Optimal sequencing by modular decomposition: Polynomial algorithms. *Oper. Res.* 34, 606–612.
Simons, B. (1978). A fast algorithm for single processor scheduling, *Proc. 19th Ann. Symp. Foundations of Computer Science*, pp. 246–252.
Simons, B. (1983). Multiprocessor scheduling of unit-time jobs with arbitrary release times and deadlines. *SIAM J. Comput.* 12, 294–299.
Simons, B., and M. Warmuth (1989). A fast algorithm for multiprocessor scheduling of unit-length jobs. *SIAM J. Comput.* 18, 690–710.
Smith, M.L., S.S. Panwalkar and R.A. Dudek (1975). Flow shop sequencing with ordered processing time matrices. *Management Sci.* 21, 544–549.
Smith, M.L., S.S. Panwalkar and R.A. Dudek (1976). Flow shop sequencing problem with ordered processing time matrices: A general case. *Naval Res. Logist. Quart.* 23, 481–486.
Smith, W.E. (1956). Various optimizers for single-stage production. *Naval Res. Logist. Quart.* 3, 59–66.
Stockmeyer, L.J. (1992). Computational complexity, in: E.G. Coffman Jr, J.K. Lenstra and A.H.G. Rinnooy Kan (eds.), *Handbooks in Operations Research and Management Science; Vol. 3: Computing*, North-Holland, Amsterdam, Chapter 9, pp. 455–517.
Szwarc, W. (1968). On some sequencing problems. *Naval Res. Logist. Quart.* 15, 127–155.
Szwarc, W. (1971). Elimination methods in the $m \times n$ sequencing problem. *Naval Res. Logist. Quart.* 18, 295–305.
Szwarc, W. (1973). Optimal elimination methods in the $m \times n$ sequencing problem. *Oper. Res.* 21, 1250–1259.
Szwarc, W. (1978). Dominance conditions for the three-machine flow-shop problem. *Oper. Res.* 26, 203–206.
Talbot, F.B., and J.H. Patterson (1978). An efficient integer programming algorithm with network cuts for solving resource-constrained scheduling problems. *Management Sci.* 24, 1163–1174.
Turner, S., and D. Booth (1987). Comparison of heuristics for flow shop sequencing. *Omega* 15, 75–78.
Ullman, J.D. (1975). NP-Complete scheduling problems. *J. Comput. System Sci.* 10, 384–393.
Ullman, J.D. (1976). Complexity of sequencing problems, in: Coffman [1976], pp. 139–164.
Van de Velde, S.L., (1990). Minimizing total completion time in the two-machine flow shop by Lagrangian relaxation. *Ann. Oper. Res.* 26, 257–268.
Van Laarhoven, P.J.M., E.H.L. Aarts and J.K. Lenstra (1992). Job shop scheduling by simulated annealing. *Oper. Res.* 40, 113–125.
Villarreal, F.J., and R.L. Bulfin (1983). Scheduling a single machine to minimize the weighted number of tardy jobs. *AIIE Trans.* 15, 337–343.
Weber, R.R. (1979). The interchangeability of $\cdot/M/1$ queues in series. *J. Appl. Probab.* 16, 690–695.
Weber, R.R., P. Varaiya and J. Walrand (1986). Scheduling jobs with stochastically ordered