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

types of sequencing and scheduling problems that have been investigated, and then describes the types of algorithms and the concepts of complexity theory that we will use throughout. Next, the class of deterministic machine scheduling problems that we will consider is introduced. Parts II, III and IV deal with the single machine, parallel machine and multi-operation problems in this class, respectively. Finally, Part V is devoted to the two generalizations of the deterministic machine scheduling model.

Each of the thirteen sections in Parts II–V starts with the full treatment of a relatively simple but crucial result. After this highlight, we review the other results that have been obtained for the subclass under consideration, in the style of two previous surveys by Graham, Lawler, Lenstra & Rinnooy Kan [1979] and Lawler, Lenstra & Rinnooy Kan [1982].

1. Sequencing and scheduling problems

The theory of sequencing and scheduling, more than any other area in operations research, is characterized by a virtually unlimited number of problem types. Most research has traditionally been focused on *deterministic machine scheduling*. Our presentation reflects this emphasis. It already allows for more than enough variety, as the reader will soon realize, but it is also based on some restrictive assumptions.

The first restriction concerns the type of resource. A *machine* is a resource that can perform at most one activity at any time. The activities are commonly referred to as *jobs*, and it is also assumed that a job is worked on by at most one machine at any time. It is not hard to think of more general scheduling situations in which, at one point in time, a resource serves several jobs and a job uses several resources. That leads us into the area of *resource-constrained project scheduling*, which is the subject of Section 15.

The second restriction concerns the *deterministic* nature of the problems. All the information that defines a problem instance is known with certainty in advance. Deterministic scheduling is part of combinatorial optimization. Indeed, all the techniques of combinatorial optimization have at some point been applied to scheduling problems. It is an obvious extension to assume that some of the problem data are subject to random fluctuations. The area of *stochastic machine scheduling* is briefly reviewed in Section 16.

In studying the allocation of machines to jobs, we are concerned with scheduling at the detailed, *operational* level. We will pay no attention to tactical decisions, such as the determination of due dates, or to strategical decisions, such as the acquisition of machines.

Further, we will restrict ourselves to the minimization of a *single optimality criterion* which is *nondecreasing in each of the job completion times*. This excludes nonregular criteria, which involve, e.g., the earliness of the jobs or the number of setups, and multicriteria scheduling, which is a relatively unexplored area.

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.

A computational problem can be viewed as a function f that maps each input x in some given domain to an output $f(x)$ in some given range. Although there may be many ways to represent the input domain for a particular problem, these specifics will be largely unimportant. We will be interested in studying the time required to compute $f(x)$ as a function of the length of the encoding of the input x, denoted $|x|$. For a more precise discussion, a mathematical model of an algorithm, a Turing machine, is commonly used, but it will suffice to think in terms of any standard programming language. In considering an algorithm that computes $f(x)$ on input x, we will measure its efficiency by an upper bound $T(n)$ on the number of steps that the algorithm takes on any input x with $|x| = n$. We will not be concerned with the precise form of the function T but rather with its asymptotic order. For this purpose, we say that $T(n) = O(g(n))$ if there exist constants c and n_0 such that $T(n) \leq cg(n)$ for all $n \geq n_0$. We will consider a problem 'easy' if there exists an algorithm for its solution which has running time $T(n) = O(n^k)$ for some constant k; that is, $T(n)$ is bounded by a polynomial function of n.

Most of the problems in which we are interested are optimization problems, where, for input x, the output $f(x)$ is the smallest value in a range of feasible integral values. It will be convenient to focus on *decision* problems, where the output range is {yes, no}. For any minimization problem f, there is an associated decision problem, the output of which answers the question 'Is $f(x) \leq k$?' for any given k. If the decision problem is easy, then one can typically apply binary search over k to obtain an algorithm for f with polynomially bounded running time. Let P denote the class of decision problems that can be solved in polynomial time.

Unfortunately, for a majority of the problems that we shall encounter, no polynomial-time algorithm is known. It is an important open question if any of these problems can be solved in polynomial time. Nonetheless, a beautiful theory developed by Cook [1971], Karp [1972] and Levin [1973] has provided a means of giving strong evidence that no such algorithm exists for a particular problem.

When a scheduling problem is formulated as a decision problem, e.g., 'Is there a feasible schedule that completes within the deadline d?', there is an important asymmetry between those inputs whose output is 'yes' and those whose output is 'no'. Note that a 'yes' answer can be certified by a small amount of information: the schedule that meets the deadline. Given this *certificate*, the 'yes' answer can be verified in polynomial time. Let NP denote the class of decision problems where each 'yes' input x has a certificate y, such that $|y|$ is bounded by a polynomial in $|x|$ and there is a polynomial-time algorithm to verify that y is a valid certificate for x. The class NP contains an enormous number of problems from a wide range of fields, including optimization, number theory, coding theory, and graph theory. Many of these problems are not known to be solvable in polynomial time. One of the major open problems of modern mathematics is whether P equals NP, and it is generally conjectured that this is not the case.

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 τ 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 *ρ-approximation algorithm* ($\rho > 1$) delivers a solution with value at most $\rho f(x)$ for each input x. Some NP-hard

problems have a *polynomial approximation scheme*, which is a family of algorithms $\{A_\epsilon\}$ such that, for each $\epsilon > 0$, A_ϵ is a polynomial-time $(1+\epsilon)$-approximation algorithm. The running time of A_ϵ may depend not only on the input size but also on the value of ϵ. If it is bounded by a polynomial in $|x|$ and $1/\epsilon$, then the family is called a *fully polynomial approximation scheme*.

The notions presented thus far have all been based on a *worst-case* analysis of the running time or the quality of the solution delivered. It would be desirable to understand the behavior for 'typical' inputs. To do this it appears necessary to assume a probability distribution over the inputs. We shall also discuss results that can be obtained through this sort of *probabilistic* analysis.

3. A class of deterministic machine scheduling problems

Suppose that m machines M_i $(i = 1, \ldots, m)$ have to process n jobs J_j $(j = 1, \ldots, n)$. A *schedule* is an allocation of one or more time intervals on one or more machines to each job. A schedule is *feasible* if no two time intervals on the same machine overlap, if no two time intervals allocated to the same job overlap, and if, in addition, it meets a number of specific requirements concerning the machine environment and the job characteristics. A schedule is *optimal* if it minimizes a given optimality criterion. The machine environment, the job characteristics and the optimality criterion that together define a problem type are specified in terms of a three-field classification $\alpha \mid \beta \mid \gamma$, which is introduced in this section.

3.1. Job data

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

– a *number of operations* m_j;

– a *processing requirement* p_j in the case of single-operation models, or a collection of *processing requirements* p_{ij} in the case of multi-operation models;

– a *release date* r_j, on which J_j becomes available for processing;

– a nondecreasing real *cost function* f_j, measuring the cost $f_j(t)$ incurred if J_j is completed at time t;

– a *due date* d_j and a *weight* w_j, that may be used in defining f_j.

In general, m_j, p_j, p_{ij}, r_j, d_j and w_j have integral values.

3.2. Machine environment

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

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

– $\alpha_1 = \circ$: *single machine*; $p_{1j} = p_j$;
– $\alpha_1 = P$: *identical parallel machines*; $p_{ij} = p_j$ 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 μ_{ij} during p_{ij} time units, with $\mu_{i,j} \neq \mu_{i+1,j}$ for $i = 1, \ldots, m_j - 1$.

If α_2 is a positive integer, then m is a constant, equal to α_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.

Occasionally, this field will contain additional characteristics such as $m_j \leq 2$ or $p_{ij} \in \{1, 2\}$. The interpretation of these should be obvious.

There are many more types of precedence relations than suggested above. We will encounter generalizations of a rooted tree, such as series–parallel constraints and opposing forests, special cases of a tree, such as intrees, outtrees and chains, and other types, such as interval orders and level orders.

3.4. Optimality criteria

The third field $\gamma \in \{f_{\max}, \Sigma f_j\}$ refers to the optimality criterion. Given a schedule, we can compute for each J_j:

- the *completion time* C_j;
- the *lateness* $L_j = C_j - d_j$;
- the *tardiness* $T_j = \max\{0, C_j - d_j\}$;
- the *unit penalty* $U_j = 0$ if $C_j \leq d_j$, $U_j = 1$ otherwise.

The optimality criteria most commonly chosen involve the minimization of

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

where $f_{\max} = \max_{1 \leq j \leq n} f_j(C_j)$ with $f_j(C_j) = C_j$, L_j, respectively, or of

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

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

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

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

3.5. Three examples

$1 \mid prec \mid L_{\max}$ is the problem of minimizing maximum lateness on a single machine subject to general precedence constraints. It can be solved in polynomial time (Section 4).

$R \mid pmtn \mid \Sigma C_j$ is the problem of minimizing total completion time on an arbitrary number of unrelated parallel machines, allowing preemption. Its complexity is unknown (Section 8).

$J3 \mid p_{ij} = 1 \mid C_{\max}$ is the problem of minimizing maximum completion time in a three-machine job shop with unit processing times. It is NP-hard (Section 14).

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 $L^*_{\max} \leqslant k$ 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 α_2 is restricted to values from $\{1, 2, 3, \circ\}$, $\beta_1 = pmtn$ excludes $\beta_4 = p_{(i)j} = 1$, and β 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

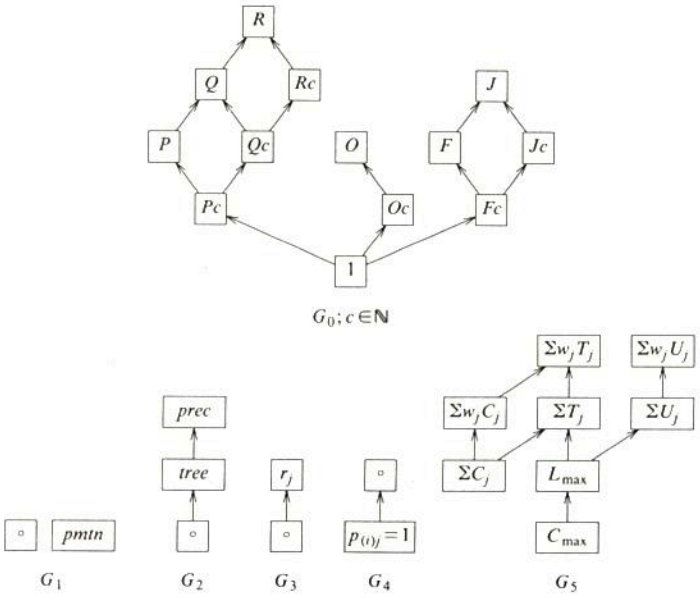


Fig. 1. Problem classification: the graphs G_i $(i = 0, \ldots, 5)$.

unary encoding, 464 are solvable in pseudopolynomial time, 3588 are strongly NP-hard, and 484 are open.

PART II. THE SINGLE MACHINE

The single machine case has been the object of extensive research ever since the seminal work by Jackson [1955] and Smith [1956]. We will survey the principal results, classifying them according to the optimality criterion in question. As a general result, we note that, if all $r_j = 0$, then only schedules without preemption and without machine idle time need be considered [Conway, Maxwell & Miller, 1967].

4. Minmax criteria

4.0. Lawler's algorithm for $1 \mid prec \mid f_{\max}$

The problem $1 \mid prec \mid f_{\max}$ has a particularly simple and elegant solution. Note that the cost functions of the jobs can be quite arbitrary and different from one another, provided only that they are nondecreasing.

Let $N = \{1, 2, \ldots, n\}$ be the index set of all jobs, and let $L \subseteq N$ be the index set of jobs without successors. For any subset $S \subseteq N$, let $p(S) = \Sigma_{j \in S}\, p_j$ and let $f^*_{\max}(S)$ denote the cost of an optimal schedule for the jobs indexed by S. Clearly, $f^*_{\max}(N)$ satisfies the following two inequalities:

$$f^*_{\max}(N) \geq \min_{j \in L} f_j(p(N)),$$

$$f^*_{\max}(N) \geq f^*_{\max}(N - \{j\}) \quad \text{for all } j \in N.$$

Now let J_l with $l \in L$ be such that

$$f_l(p(N)) = \min_{j \in L} f_j(p(N)).$$

We have

$$f^*_{\max}(N) \geq \max\{f_l(p(N)),\ f^*_{\max}(N - \{l\})\}.$$

But the right-hand side of this inequality is precisely the cost of an optimal schedule subject to the condition that J_l is processed last. It follows that there exists an optimal schedule in which J_l is in the last position. By repeated application of this rule, one obtains an optimal schedule in $O(n^2)$ time. This algorithm is due to Lawler [1973].

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, $\max_j f_{\pi(j)}(\sum_{i=1}^{j-1} c_{\pi(i)})$. 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

time of the form $r_j + kp$ $(1 \leq j, k \leq n)$ is excluded, at most n^2 iterations will occur and the feasibility question is answered in $O(n^3 \log n)$ time. Garey, Johnson, Simons & Tarjan [1981] give an improved implementation that requires only $O(n \log n)$ time. Bisection search over the possible $L_{\max}$ values leads to a polynomial-time algorithm for $1 \mid r_j, p_j = p \mid L_{\max}$.

These three special cases as well as the preemptive variant remain well solved in the presence of precedence constraints. It suffices to update release and due dates such that $r_j < r_k$ and $d_j < d_k$ whenever $J_j \rightarrow J_k$, as described by Lageweg, Lenstra & Rinnooy Kan [1976]. Monma [1982] gives a linear-time algorithm for $1 \mid prec, p_j = 1 \mid L_{\max}$.

Various elegant enumerative methods exist for solving $1 \mid prec, r_j \mid L_{\max}$. Baker & Su [1974] obtain a lower bound by allowing preemption; their enumeration scheme simply generates all active schedules, i.e., schedules in which one cannot decrease the starting time of a job without increasing the starting time of another one. McMahon & Florian [1975] propose a more ingenious approach. Lageweg, Lenstra & Rinnooy Kan [1976] slightly modify this method to obtain very fast solution of quite large problems. Their algorithm makes use of an equivalent formulation in which due dates are replaced by *delivery times* q_j, and if a job completes at time C_j, it is delivered at time $C_j + q_j$; the aim is to minimize $\max_j C_j + q_j$. The role of release times and delivery times is completely symmetric. One can take advantage of this fact and obtain superior performance by interchanging release times and delivery times under certain conditions. Carlier [1982] and Larson, Dessouky & Devor [1985] propose different branching rules, which yield more efficient algorithms for this relatively easy NP-hard problem. Nowicki & Zdrzalka [1986] observe that in the approach suggested by Carlier, the proof of optimality may be somewhat more elusive than originally believed. Nowicki & Smutnicki [1987] provide alternative lower bound procedures. Zdrzalka & Grabowski [1989] consider extensions of these methods to $1 \mid prec, r_j \mid f_{\max}$.

Dominance results among the schedules may be used in the obvious way to speed up enumerative procedures. Erschler, Fontan, Merce & Roubellat [1982, 1983] introduce dominance based on the (r_j, d_j) intervals, assuming that the objective is simply to meet all due dates.

Little work has been done on the worst-case analysis of approximation algorithms for single machine problems. For $1 \mid r_j \mid L_{\max}$, one must be careful in specifying the problem, in order to obtain reasonable approximation results. First, it is possible that $L^*_{\max} = 0$, and any algorithm that may err in such a case will have unbounded relative error. In fact, deciding if $L^*_{\max} \leq 0$ is NP-complete, and so it is probably impossible to completely remove this curious technicality. Note that by focusing on the special case that $r_j \geq 0$ and $d_j \leq 0$ for all j, this difficulty is avoided. This is identical to viewing the problem in the delivery time model, since if $q_j = -d_j$, then $C_j + q_j = C_j - d_j$. Kise, Ibaraki & Mine [1979] provide another justification for studying this case, by arguing that the guarantee should be invariant under certain simple transformations of the input data. Six approximation algorithms are considered, and the extended

Jackson's rule (EJ) is shown to guarantee

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

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

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

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

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

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

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

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* $1 \mid \mid \Sigma w_j C_j$

For the problem $1 \mid \mid \Sigma w_j C_j$, 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.

In the special case $1||\Sigma C_j$, any sequence is optimal that places the jobs in order of nondecreasing p_j. This *shortest processing time* or SPT rule is one of the most celebrated algorithms in sequencing and scheduling. It is often used for more complicated problems, sometimes without much theoretical support for its superior performance.

5.1. Decomposable precedence constraints

Smith's rule can be viewed as an instance of a more general phenomenon. Consider the following very general problem. Given a set of n jobs and a real-valued function f that assigns a value $f(\pi)$ to each permutation π of the job indices, find a permutation π^* such that

$$f(\pi^*) = \min_{\pi} f(\pi) .$$

If we know nothing about the structure of the function f, there is little that we can do, except to evaluate $f(\pi)$ for each of the $n!$ permutations π. However, it may be that we are able to establish that there is a transitive and complete relation $\preceq$, a *quasi-total order*, on the index set of the jobs, with the property that for any two jobs J_b, J_c and any permutation of the form $\alpha bc\delta$, we have

$$b \preceq c \Rightarrow f(\alpha bc\delta) \preceq f(\alpha cb\delta) .$$

If such a *job interchange relation* $\preceq$ exists, an optimal permutation π^* can be found by simply ordering the jobs according to $\preceq$, with $O(n \log n)$ comparisons of jobs with respect to $\preceq$. Smith's rule for $1||\Sigma w_j C_j$ and Jackson's rule for $1||L_{\max}$ can be seen to be special cases.

In fact, there has been a great deal of work in using this general framework to provide polynomial-time algorithms for special classes of precedence constraints. For tree-like precedence constraints, results of Horn [1972], Adolphson & Hu [1973] and Sidney [1975] give $O(n \log n)$ algorithms.

The decomposition approach of Sidney [1975] is applicable to a much broader setting. Most typical is the case of *series–parallel* precedence constraints, for which Lawler [1978a] gives an $O(n \log n)$ algorithm. The crucial observation for each of these cases is that the precedence graph can be broken down into modules, such that an optimal solution for each module can be extended to an optimal solution for the entire instance. (For example, a module can be defined as a set of jobs where each job in the module has the same relation to jobs outside it.) In order to handle precedence constraints, we introduce the notion of a *string interchange relation* that generalizes a job interchange relation by letting b and c, in the implication above, represent disjoint strings of job indices. We will focus on objective functions that admit of a string interchange relation; one such function is $\Sigma\, w_j C_j$.

Given a decomposition tree representing the way in which the modules of the precedence graph are composed, an ordered set of strings is computed for

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 σ_1 in the first set (with respect to $\leqslant$) is bigger than the smallest string σ_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

of Smith's rule [Baker, 1974], but, surprisingly, $1 \mid pmtn, r_j \mid \Sigma w_j C_j$ is strongly NP-hard [Labetoulle, Lawler, Lenstra & Rinnooy Kan, 1984].

If a deadline $\bar{d}_j$ on the completion of each job J_j is introduced, $1 \mid \bar{d}_j \mid \Sigma C_j$ can be solved by another simple extension of Smith's rule [Smith, 1956], but the weighted case $1 \mid \bar{d}_j \mid \Sigma w_j C_j$ is strongly NP-hard [Lenstra, Rinnooy Kan & Brucker, 1977]. Du & Leung [1988b] establish NP-hardness of $1 \mid pmtn, r_j, \bar{d}_j \mid \Sigma C_j$.

For $1 \mid \mid \Sigma w_j C_j$ with either release times or deadlines, several elimination criteria and branch and bound algorithms have been proposed. Potts & Van Wassenhove [1983] apply Lagrangean relaxation to the problem with deadlines, and dualize the constraints $C_j \leq \bar{d}_j$. The Lagrangean multipliers are adjusted so that a simple heuristic for the original problem provides an optimal solution to the relaxed problem. Hariri & Potts [1983] consider the variant with release times, and dualize the constraints $C_j \geq r_j + p_j$ instead. Rinaldi & Sassano [1977], Bianco & Ricciardelli [1982], and Dessouky & Deogun [1981] give other branch and bound procedures for this problem, based on a variety of lower bound methods and dominance relations. Posner [1985] and Bagchi & Ahmadi [1987] give improvements on the lower bound method of Potts & Van Wassenhove [1983], where in each case, the new heuristic is proved to dominate the previous methods. Belouadah, Posner & Potts [1989] extend this approach and use it within a branch and bound algorithm.

6. Weighted number of late jobs

6.0. *Karp, Lawler & Moore on* $1 \mid \mid \Sigma w_j U_j$

Karp [1972] included the decision version of $1 \mid \mid \Sigma w_j U_j$ in his list of 21 NP-complete problems. His proof is based on an idea that has been applied to many other scheduling problems.

Recall the NP-complete *partition* problem from Section 2: given n numbers $a_1, \ldots, a_n$ with $\Sigma_{j=1}^{n} a_j = 2b$, does there exist a set $S \subset \{1, \ldots, n\}$ such that $\Sigma_{j \in S} a_j = b$? For any instance of this problem, we define an instance of $1 \mid \mid \Sigma w_j U_j$ with n jobs and $p_j = w_j = a_j$, $d_j = b$ $(j = 1, \ldots, n)$. Consider any schedule, where we may assume that all the processing is done in the interval $[0, 2b]$. The jobs that are completed by time b are on time, the others are late, and the $\Sigma w_j U_j$ value of the schedule is equal to the total processing requirement of these late jobs. It follows that, for any schedule, $\Sigma w_j U_j \geq b$. Equality can be achieved if and only if there exists a set of jobs of total length b, i.e., if and only if the original instance of the partition problem is a 'yes' instance.

Given the complexity status of the partition problem, we know that $1 \mid \mid \Sigma w_j U_j$ is NP-hard in the ordinary sense, and not in the strong sense. In fact, the latter result is unlikely to hold, as the problem is solvable in pseudopolynomial time. This was proved by Lawler & Moore [1969], who proposed a dynamic programming approach.

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 $d_1 \leq \cdots \leq d_n$, 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

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

we have that

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

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., $p_j < p_k \Rightarrow w_j \geq w_k$).

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., $r_j < r_k \Rightarrow d_j \leq d_k$); 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.

Villarreal & Bulfin [1983] present a branch and bound procedure for $1||\Sigma w_j U_j$. Two lower bounds are obtained by applying the algorithms Moore–Hodgson and Lawler as if, respectively, the weights and processing times are identical. (Note that in the case of identical processing times, any set of weights is oppositely ordered.) Potts & Van Wassenhove [1988] give an $O(n \log n)$ algorithm to solve the linear relaxation of a natural integer programming formulation of $1||\Sigma w_j U_j$. Computational experiments confirm that this is an extremely effective lower bound.

Sahni [1976] gives a pseudopolynomial-time algorithm for $1||\Sigma w_j U_j$ that requires $O(n\Sigma w_j)$ time and uses this to derive an approximation algorithm A_k with $O(n^3 k)$ running time such that

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

where $\bar{U}_j = 1 - U_j$. For reasons similar to those discussed in Section 4.2 for $1|r_j|L_{\max}$, it is easier to design approximation algorithms with respect to this complementary objective. Unlike that case, however, it is possible to decide in polynomial time whether $\Sigma w_j U_j^* = 0$. Gens & Levner [1978] exploit this to give an algorithm B_k with running time $O(n^3 k)$ such that

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

By obtaining a preliminary upper bound on the optimum that is within a factor of 2, Gens & Levner [1981] improve the running time of a variant of B_k to $O(n^2 \log n + n^2 k)$. For $1|tree|\Sigma w_j U_j$, Ibarra & Kim [1978] give algorithms D_k of order $O(kn^{k+2})$ with the same worst-case error bound as the algorithm A_k due to Sahni [1976].

7. Total tardiness and beyond

7.0. A branch and bound algorithm for $1||\Sigma f_j$

Let us first consider the problem with unit processing times, $1|p_j = 1|\Sigma f_j$. In this case, the cost of scheduling J_j in position k is given by $f_j(k)$, irrespective of the ordering of the other jobs. The problem is therefore equivalent to finding a permutation π of $\{1, \ldots, n\}$ that minimizes $\Sigma_j f_j(\pi(j))$. This is a weighted bipartite matching problem, which can be solved in $O(n^3)$ time.

For the case of arbitrary processing times, Rinnooy Kan, Lageweg & Lenstra [1975] applied the same idea to compute a *lower bound* on the costs of an optimal schedule. Suppose that $p_1 \leqslant \cdots \leqslant p_n$ and define $t_k = p_1 + \cdots + p_k$ for $k = 1, \ldots, n$. Then $f_j(t_k)$ is a lower bound on the cost of scheduling J_j in position k, and an overall lower bound is obtained by solving the weighted bipartite matching problem with coefficients $f_j(t_k)$.

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 lth 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.

Algorithms based on straightforward but cleverly implemented dynamic programming offer a surprisingly good alternative. Baker & Schrage [1978] and Schrage & Baker [1978] suggest compact labeling schemes that can handle up to 50 jobs. Lawler [1979b] gives a more efficient implementation of this approach; Kao & Queyranne [1982] describe carefully designed experiments which confirm that this method is a practical improvement as well. Potts & Van Wassenhove [1982] consider the unweighted problem, and use a combination of the Baker–Schrage algorithm and a decomposition approach implied by the algorithm of Lawler [1977]. Potts & Van Wassenhove [1987] compare the dynamic programming algorithms of Schrage & Baker [1978] and Lawler [1979b], and then consider the relative merits of the decomposition approach when used in a dynamic programming framework or in an algorithm that, as in their previous work, resembles branch and bound.

Abdul-Razaq & Potts [1988] consider $1|\ |\Sigma f_j$ where the costs are no longer assumed to be nondecreasing functions of completion time; however, the constraint that a schedule may not contain idle time is added. Since the straightforward dynamic programming formulation has an unmanageable number of states, a lower bound is computed by recursively solving a formulation with a smaller state space, and then used within a branch and bound procedure.

Using his pseudopolynomial algorithm for $1|\ |\Sigma T_j$ mentioned above, Lawler [1982c] presents a fully polynomial approximation scheme, such that algorithm A_k runs in $O(n^7k)$ time and guarantees

$$\sum T_j(A_k) \Big/ \sum T_j^* \leq 1 + \frac{1}{k}.$$

Fisher & Krieger [1984] study the following general problem: let P_j be a nonincreasing and concave *profit function* of the starting time of J_j; maximize the total profit. They use a heuristic based on a generalization of Smith's rule (GS) to get provably good solutions:

$$\sum P_j(\mathrm{GS}) \Big/ \sum P_j^* \geq \tfrac{2}{3}.$$

PART III. PARALLEL MACHINES

Recall from Section 3 the definitions of *identical*, *uniform* and *unrelated* machines, denoted by P, Q and R, respectively.

Section 8 deals with minsum criteria. We will be able to review some interesting polynomial-time algorithms, especially for the minimization of ΣC_j. We then turn to minmax criteria. Section 9 considers the nonpreemptive case with general processing times. The simplest problem of this type, $P2|\ |C_{\max}$, is already NP-hard, and we will concentrate on the analysis of approximation algorithms. Section 10 considers the preemptive case. The situation is much

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, ΣC_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 kth 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

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

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

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

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, ΣC_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

of Conway, Maxwell & Miller [1967] accomplishes: schedule the jobs in order of nondecreasing processing times, and assign each job to the earliest available machine.

In the case of uniform machines, ΣC_j is a weighted sum of p_j values, where each weight is of the form k/s_i (k indicating the position and s_i the speed of M_i), and no weight may be used more than once. Once again, we want to select the n smallest of these mn weights and to match the smallest weights with the longest jobs. Horowitz & Sahni [1976] propose to maintain a priority queue of the smallest m unused weights and to build the schedule backwards by assigning the next longest job to the machine associated with the smallest available weight. This algorithm can be implemented to run in $O(n \log n)$ time.

8.1. *Unit-length jobs on uniform machines*

The problems $Q \mid p_j = 1 \mid \Sigma f_j$ and $Q \mid p_j = 1 \mid f_{\max}$ are easily solved in polynomial time. First, observe that there exists an optimal schedule in which the jobs are executed in the time periods with the n earliest possible completion times. These completion times can be generated in $O(n \log m)$ time: initialize a priority queue with completion times $1/s_i$ ($i = 1, \ldots, m$); at a general step, remove the smallest completion time from the queue and, if this time is k/s_i, insert $(k+1)/s_i$ into the queue. Let $t_1, \ldots, t_n$ denote the n smallest completion times, in nondecreasing order.

$Q \mid p_j = 1 \mid \Sigma f_j$ is now solved by finding an optimal assignment of the jobs to these completion times. This amounts to formulating and solving an $n \times n$ weighted bipartite matching problem with cost coefficients $c_{jk} = f_j(t_k)$; this requires $O(n^3)$ time. Various special cases can be solved more efficiently. Thus $Q \mid p_j = 1 \mid \Sigma w_j C_j$ is solved by assigning the job with the kth largest weight to t_k, and $Q \mid p_j = 1 \mid \Sigma T_j$ is solved by assigning the job with the kth smallest due date to t_k; the time required is $O(n \log n)$, the time needed to sort weights or due dates. $Q \mid p_j = 1 \mid \Sigma w_j U_j$ is solved by considering the completion times from largest to smallest and scheduling, from among all unassigned jobs that would be on time (if any), a job with maximal weight; with appropriate use of priority queues, this can again be done in $O(n \log n)$ time. In the presence of release dates, dynamic programming can be applied to solve $Q \mid r_j, p_j = 1 \mid \Sigma C_j$ in $O(m^2 n^{2m+1} \log n)$ time, which is polynomial only for fixed values of m.

$Q \mid p_j = 1 \mid f_{\max}$ is solved by a method that resembles Lawler's algorithm for $1 \mid \mid f_{\max}$ (see Section 4.0). Consider the completion times from largest to smallest and, at each successive completion time t, schedule a job J_j for which $f_j(t)$ is minimal: this yields an optimal schedule in $O(n^2)$ time. $Q \mid p_j = 1 \mid L_{\max}$ and $Q \mid r_j, p_j = 1 \mid C_{\max}$ can be solved in $O(n \log n)$ time be simply matching the kth smallest due date, or release date, with t_k.

These results are due to Lawler [-], and Dessouky, Lageweg, Lenstra and Van de Velde [1990]. Lawler [1976a] shows that the special case $P \mid p_j = 1 \mid \Sigma U_j$ can be solved in $O(n \log n)$ time.

Complexity results for the precedence-constrained problem $P \mid prec, p_j = 1 \mid \Sigma C_j$ and its special cases will be mentioned in Section 11.1.

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 ΣC_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

& Moore [1969] is applicable to special cases of $R\,|\,|\,\Sigma f_j$ and $R\,|\,|\,f_{\max}$ in which the following condition is satisfied: it is possible to index the jobs in such a way that the jobs assigned to a given machine can be assumed to be processed in order of their indices. For example, this condition holds in the case of $R\,|\,|\,C_{\max}$ (any indexing is satisfactory), $R\,|\,|\,\Sigma w_j U_j$ (index in order of due dates), and $Q\,|\,|\,\Sigma w_j C_j$ (index in order of the ratios p_j/w_j).

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

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

and in the case of Σf_j criteria,

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

In both cases, the initial conditions are

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

These equations can be solved in $O(mnC^m)$ time, where C is an upper bound on the completion time of any job in an optimal schedule. If the machines are uniform, then only $m-1$ of the values $t_1, \ldots, t_m$ in the equation for $F_j(t_1, \ldots, t_m)$ are independent. This means, for example, that the time bound for $Q\,|\,|\,\Sigma w_j C_j$ can be reduced by a factor of C to $O(mnC^{m-1})$.

One variation of the above technique solves $Q\,|\,r_j\,|\,C_{\max}$, and another variation solves $Q\,|\,|\,\Sigma w_j U_j$ in $O(mn(\max_j d_j)^m)$ time. Still other dynamic programming approaches can be used to solve $P\,|\,|\,\Sigma f_j$ and $P\,|\,|\,f_{\max}$ in $O(m \cdot \min\{3^n, n2^n C\})$ time.

8.3. *Minsum criteria with preemption*

A theorem of McNaughton [1959] states that for $P\,|\,pmtn\,|\,\Sigma w_j C_j$ there is no schedule with a finite number of preemptions which yields a smaller criterion value than an optimal nonpreemptive schedule. The finiteness restriction can be removed by appropriate application of results from open shop theory. It therefore follows that the procedure of Section 8.0 solves $P\,|\,pmtn\,|\,\Sigma C_i$ in $O(n \log n)$ time, and that $P2\,|\,pmtn\,|\,\Sigma w_j C_j$ is NP-hard. Du, Leung & Young [1991] extend McNaughton's theorem to the case of chain-like precedence constraints, which implies that $P2\,|\,pmtn,tree\,|\,\Sigma C_j$ is strongly NP-hard.

McNaughton's theorem does not apply to uniform machines, as can be demonstrated by a simple counterexample. There is, however, a polynomial algorithm for $Q\,|\,pmtn\,|\,\Sigma C_j$. Lawler & Labetoulle [1978] show that there exists an optimal preemptive schedule in which $C_j \leqslant C_k$ if $p_j < p_k$. This result is the

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 ΣC_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,

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

To see this, let J_l be the last job to be completed in a list schedule, and note that no machine can be idle before time $t = C_{\max}(\mathrm{LS}) - p_l$, when J_l starts processing. Intuitively, the performance guarantee follows from the observation that both t and p_l are lower bounds on the length of any schedule. More formally, we have $\Sigma_{j \neq l} p_j \geq mt$ and therefore

$$C_{\max}(\mathrm{LS}) = t + p_l \leq \frac{1}{m} \sum_{j \neq l} p_j + p_l = \frac{1}{m} \sum_j p_j + \frac{m-1}{m} p_l .$$

The observations that

$$C^*_{\max} \geq \frac{1}{m} \sum_j p_j , \qquad C^*_{\max} \geq p_l ,$$

now yield the desired result.

The bound is tight for any value of m, as is shown by the following class of instances. Let $n = m(m-1)+1$, $p_1 = \cdots = p_{n-1} = 1$, $p_n = m$, and consider the list $(J_1, J_2, \ldots, J_n)$. It is not hard to see that $C_{\max}(\mathrm{LS}) = 2m - 1$ and $C^*_{\max} = m$.

The worst-case analysis also gives insight into the average-case performance of list scheduling. We know that, for any instance,

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

In order to give a probabilistic analysis of list scheduling, we assume that the processing times $\boldsymbol{p}_j$ are selected from a given probability distribution, and we study the error term under this distribution. (Note that random variables are printed in boldface italic.) For the case that the $\boldsymbol{p}_j$ are independently and uniformly distributed over the interval $[0, 1]$, Bruno & Downey [1986] show that

$$\lim_{n \to \infty} \Pr\left[\max_j \boldsymbol{p}_j \Big/ \sum_j \boldsymbol{p}_j > 4/n \right] = 0 .$$

In other words, as long as n grows faster than m, list schedules are asymptotically optimal in probability.

9.1. *Identical machines*

By far the most studied scheduling model from the viewpoint of approximation algorithms is $P \mid \mid C_{\max}$. Garey, Graham & Johnson [1978] and Coffman, Lueker & Rinnooy Kan [1988] give easily readable introductions into the techniques involved in, respectively, the worst-case and probabilistic analysis of approximation algorithms.

In the previous section, we have seen that list scheduling is guaranteed to produce a schedule with maximum completion time less than twice the optimal.

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:

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

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 ith bin will be processed by M_i. Coffman, Garey & Johnsons show that, if k packing attempts are made, the algorithm (denoted by MF_k) runs in time $O(n \log n + kn \log m)$ and satisfies

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

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

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

Friesen & Langston [1986] refine the iterated approximation algorithm to provide algorithms 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

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

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

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

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

scheme. Sahni [1976] has improved this result, by devising algorithms A_k with $O(n(n^2k)^{m-1})$ running time which satisfy

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

For any fixed number of machines, these algorithms constitute a fully polynomial approximation scheme. For $m = 2$, algorithm A_k can be improved to run in time $O(n^2k)$. As in the cases of $1||\Sigma w_j U_j$ (Section 6.1) and $P||\Sigma w_j C_j$ (Section 8.2), the algorithms A_k are based on a clever combination of dynamic programming and rounding and are beyond the scope of the present discussion.

Hochbaum & Shmoys [1987] use a variation on the multifit approach to provide a polynomial approximation scheme for $P||C_{\max}$, which replaces a (traditional) approximation algorithm in the binary search with a *dual approximation algorithm*. Given a capacity d and a set of jobs to pack, a ρ-dual approximation algorithm ($\rho > 1$) produces a packing that uses *at most* the minimum number of bins of capacity d, but the packing may use bins of capacity ρd. Using a ρ-dual approximation algorithm within binary search for k iterations, one obtains a $(\rho + 2^{-k})$-approximation algorithm for $P||C_{\max}$. Hochbaum and Shmoys further provide a family of algorithms D_k, such that D_k is a $(1 + 1/k)$-dual approximation algorithm and has running time $O((kn)^{k^2})$; Leung [1991] improves the running time to $O((kn)^{k \log k})$. For $k = 5$ and $k = 6$, Hochbaum & Shmoys refine their approach to obtain algorithms with $O(n \log n)$ and $O(n(m^4 + \log n))$ running times, respectively. Since $P||C_{\max}$ is strongly NP-hard, there is no fully polynomial approximation scheme for it unless P = NP.

Several bounds are available which take into account the processing times of the jobs. Recall that the probabilistic analysis discussed in Section 9.0 relies on such a (worst-case) bound for list scheduling. Achugbue & Chin [1981] prove two results relating the performance ratio of list scheduling to the value of $\pi = \max_j p_j / \min_j p_j$. If $\pi \leq 3$, then

$$C_{\max}(\mathrm{LS})/C^*_{\max} \leq \begin{cases} \frac{5}{3} & \text{if } m = 3, 4, \\ \frac{17}{10} & \text{if } m = 5, \\ 2 - \dfrac{1}{3\lfloor m/3 \rfloor} & \text{if } m \geq 6, \end{cases} \qquad (\dagger)$$

and if $\pi \leq 2$,

$$C_{\max}(\mathrm{LS})/C^*_{\max} \leq \begin{cases} \frac{3}{2} & \text{if } m = 2, 3, \\ \frac{5}{3} - \dfrac{1}{3\lfloor m/2 \rfloor} & \text{if } m \geq 4. \end{cases} \qquad (\dagger)$$

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

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

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),

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

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 $\mathrm{E}[I(\mathrm{LPT})] \leq c_m m^2/(n+1)$, 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 $m \geq 2$, 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, $\Pr[\lim_{n\to\infty} \boldsymbol{C}_{\max}(\mathrm{LPT}) - \boldsymbol{C}^*_{\max} = 0] = 1$.

Given that the absolute error of the LPT rule approaches 0, a further issue is the rate at which the error converges to 0. Boxma [1984] and Frenk & Rinnooy Kan [1986] show that under a broad range of distributions, the expected absolute error is $O(n^{-c})$ for some positive constant c. Karmarkar & Karp [1982] suggest an entirely different approach, the *differencing method*, and prove that with probability approaching 1, the difference between the completion times of the last and first machines is $O(n^{-c \log n})$ for some positive c. Fischetti & Martello [1987] give a worst-case analysis of this heuristic for $P2||C_{\max}$ and prove that it is a $\frac{7}{6}$-approximation algorithm.

9.2. *Uniform machines*

Many of the results in the previous section can be generalized to the uniform machine model. The initial work in this area is due to Liu & Liu [1974a,b,c], who consider arbitrary list scheduling as well as a natural extension of the scheme of Graham that optimally schedules the k longest jobs and then uses list scheduling on the remaining jobs. The performance of these algorithms on uniform machines is significantly worse; for example,

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

The most natural way to implement list scheduling on uniform machines is to assign the next job on the list to any machine that becomes idle. However, this produces schedules without *unforced idleness*, and the optimal schedule might require such idle time. Another implementation LS′ is studied by Cho & Sahni [1980], where the next job in the list is scheduled on the machine on which it will finish earliest. They prove that

$$C_{\max}(\mathrm{LS}')/C^*_{\max} \leq \begin{cases} (1+\sqrt{5})/2 & \text{for } m = 2\,, \\ (1+(\sqrt{2m-2}))/2 & \text{for } m > 2\,. \end{cases}$$

The bound is tight for $m \leq 6$, but in general, the worst known examples have a performance ratio of $\lfloor(\log_2(3m-1)+1)/2\rfloor$. This approach followed the work of Gonzalez, Ibarra & Sahni [1977], who consider the analogous generalization LPT′ of LPT and show that

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

Dobson [1984] and Friesen [1987] improve this analysis to obtain an upper bound of $\frac{19}{12}$, and also provide examples that have performance ratio 1.52. Morrison [1988] shows that LPT is better than LS, in that

$$C_{\max}(\mathrm{LPT})/C^*_{\max} \leq \max\{\max_i s_i/(2\min_i s_i), 2\}\,. \qquad (\dagger)$$

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 ρ-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

bound is due to Davis & Jaffe [1981], who give a variant of a list scheduling algorithm for which

$$C_{\max}(\mathrm{LS}')/C^*_{\max} \leq 2.5\sqrt{m} + 1 + \frac{1}{2\sqrt{m}} ,$$

and also provide examples that show that this analysis is tight up to a constant factor.

Potts [1985a] proposes an algorithm based on linear programming (LP), the running time of which is polynomial only for fixed m. He proves

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

In contrast to the scheme of Horowitz & Sahni, this is a practical algorithm for a modest number of machines, since the space requirements do not grow exponentially in the number of machines. Lenstra, Shmoys & Tardos [1990] extend this approach in two ways. First, they give a modified algorithm LP′ that runs in polynomial time and still satisfies

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

Second, for a fixed number of machines, they give a polynomial approximation scheme, based on a combination of enumeration of partial schedules and linear programming, which has only modest space requirements.

10. Minmax criteria with preemption

10.0. McNaughton's wrap-around rule for $P \mid pmtn \mid C_{\max}$

McNaughton's [1959] solution of $P \mid pmtn \mid C_{\max}$ is probably the simplest and earliest instance of an approach that has been successfully applied to other preemptive scheduling problems: we first provide an obvious lower bound on the value of an optimal schedule and then construct a schedule that matches this bound.

In this case, we see that the maximum completion time of any schedule is at least

$$\max\left\{\max_j p_j , \left(\sum_j p_j\right)\Big/ m\right\} .$$

A schedule meeting this bound can be constructed in $O(n)$ time: just fill the machines successively, scheduling the jobs in any order and splitting a job whenever the above time bound is met. The number of preemptions occurring in this schedule is at most $m - 1$, and it is possible to design a class of problems

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

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

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

time. Certainly, Gonzalez, Lawler & Sahni [1990] show how to solve the special case $R2 \mid pmtn \mid C_{\max}$ in $O(n)$ time.

10.2. *Release dates, due dates, and other complications*

Horn [1974] gives a procedure to solve $P \mid pmtn \mid L_{\max}$ and $P \mid pmtn, r_j \mid C_{\max}$ in $O(n^2)$ time. Gonzalez & Johnson [1980] give a more efficient algorithm that uses only $O(mn)$ time.

More generally, Horn [1974] shows that the existence of a feasible preemptive schedule with given release dates and deadlines can be tested by means of a network flow model in $O(n^3)$ time. A binary search can then be conducted on the optimal value of $L_{\max}$, with each trial value of $L_{\max}$ inducing deadlines that are checked for feasibility by means of the network computation. Labetoulle, Lawler, Lenstra & Rinnooy Kan [1984] show that this yields an $O(n^3 \min\{n^2, \log n + \log \max_j p_j\})$ algorithm.

Other restrictions on allowable preemptive schedules have been investigated. Schmidt [1983] considers the case where the *machines* are only available in certain given time intervals, and shows that the existence of a feasible preemptive schedule can be tested in polynomial time. Rayward-Smith [1987b] studies the situation where a delay of k time units is incurred when a job is preempted from one machine to another. He observes that imposing such delays on identical machines increases $C^*_{\max}$ by at most $k-1$. Thus, for $k=1$, the problem is solvable in polynomial time by McNaughton's rule. Surprisingly, for any fixed $k \geq 2$, the problem is NP-hard.

In the case of uniform machines, Sahni & Cho [1980] show how to test the existence of a feasible preemptive schedule with given release dates and a common deadline in $O(n \log n + mn)$ time; the algorithm generates $O(mn)$ preemptions in the worst case. More generally, Sahni & Cho [1979b] and Labetoulle, Lawler, Lenstra & Rinnooy Kan [1984] show that $Q \mid pmtn, r_j \mid C_{\max}$ and, by symmetry, $Q \mid pmtn \mid L_{\max}$ are solvable in $O(n \log n + mn)$ time, where the number of preemptions generated is $O(mn)$.

The feasibility test of Horn mentioned above has been adapted by Bruno & Gonzalez [1976] to the case of two uniform machines and extended to a polynomial-time algorithm for $Q2 \mid pmtn, r_j \mid L_{\max}$ by Labetoulle, Lawler, Lenstra & Rinnooy Kan [1984].

Martel [1982] presents a polynomial-time algorithm for $Q \mid pmtn, r_j \mid L_{\max}$. His method is in fact a special case of a more general algorithm of Lawler & Martel [1982] for computing maximal *polymatroidal* network flows. Federgruen & Groenevelt [1986] give an improved algorithm for the problem by reducing it to the ordinary maximum flow problem; if there are machines of t distinct speeds (and so $t \leq m$), their algorithm runs in $O(tn^3)$ time.

The technique of Lawler & Labetoulle [1978] also yields a polynomial-time algorithm based on linear programming for $R \mid pmtn, r_j \mid L_{\max}$.

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 ρ-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.

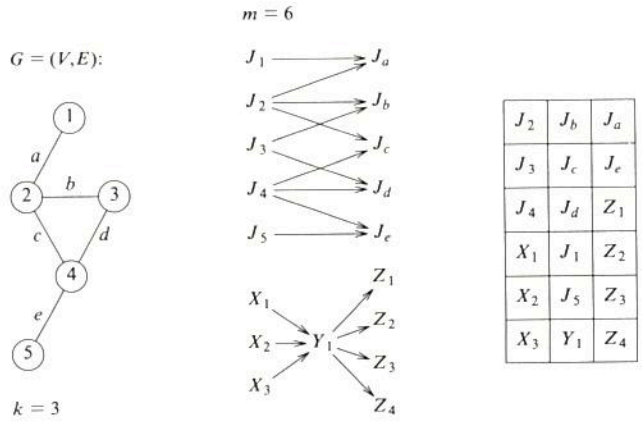


Fig. 2. The clique problem reduces to $P \mid prec, p_j = 1 \mid C_{\max}$. (a) Instance of the clique problem. (b) Corresponding instance of $P \mid prec, p_j = 1 \mid C_{\max}$. (c) Feasible schedule for $P \mid prec, p_j = 1 \mid C_{\max}$.

More precisely, suppose that a clique on k vertices exists. We then schedule the k jobs corresponding to the clique vertices and the $m - k$ jobs X_x in the first time slot. In view of the precedence constraints, we can schedule the l jobs corresponding to the clique edges and the $m - l - k'$ jobs Y_y in the second time slot; we also schedule the k' remaining vertex jobs there. We finally schedule the l' remaining edge jobs and the $m - l'$ jobs Z_z in the third time slot. This is a feasible schedule of length 3.

Conversely, suppose that no clique of size k exists. In any schedule of length 3, exactly k vertex jobs are processed in the first time slot. However, any set of k vertex jobs releases at most $l - 1$ edge jobs for processing in the second time slot. Since at that point only $m - l$ other jobs are available for processing, the schedule cannot be feasible.

11.1. Unit-length jobs on identical machines

We have seen that $P \mid prec, p_j = 1 \mid C_{\max}$ is NP-hard. It is an important open question whether this remains true for any constant value of $m \geq 3$. The problem is well solved, however, if the precedence relation is of the *tree* type or if $m = 2$.

Hu [1961] gives a polynomial-time algorithm to solve $P \mid tree, p_j = 1 \mid C_{\max}$. Hsu [1966] and Sethi [1976a] give improvements that lead to an $O(n)$ time procedure. We will describe a procedure for the case of an *intree* (each job has at most one successor); an alternative algorithm for the case of an *outtree* (each job has at most one predecessor) is given by Davida & Linton [1976]. The *level* of a job is defined as the number of jobs in the unique path to the root of the precedence tree. At the beginning of each time unit, as many available jobs as possible are scheduled on the m machines, where highest priority is granted to the jobs with the largest levels. Thus, Hu's algorithm is a nonpreemptive *list scheduling* algorithm (cf. Section 9.0). It can also be viewed as a *critical path* scheduling algorithm: the next job chosen is the one which heads the longest current chain of unexecuted jobs. Marcotte & Trotter [1984] show that Hu's algorithm can also be derived from a minmax result of Edmonds [1965] on covering the elements of a matroid by its bases; in this application, the elements correspond to jobs, and a transversal matroid is obtained with bases corresponding to feasible machine histories.

Brucker, Garey & Johnson [1977] show that, if the precedence constraints are in the form of an intree, then Hu's algorithm can be adapted to minimize $L_{\max}$; on the other hand, if the precedence constraints form an outtree, then the $L_{\max}$ problem turns out to be NP-hard. Monma [1982] improves the former result by giving a linear-time algorithm.

Garey, Johnson, Tarjan & Yannakakis [1983] consider the case in which the precedence graph is an *opposing forest*, that is, the disjoint union of an inforest and an outforest. They show that if m is arbitrary, then minimizing $C_{\max}$ is NP-hard, but if m is fixed, then the problem can be solved in polynomial time. Papadimitriou & Yannakakis [1979] consider the case in which the precedence

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 $J_k \in S_i$, $J_l \in S_{i+1}$,

J_k precedes J_l; then $\lceil |S_1|/2 \rceil + \cdots + \lceil |S_t|/2 \rceil$ is clearly a lower bound. Gabow's proof implies the duality result that the maximum lower bound that can be obtained in this way is equal to $C^*_{\max}$.

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

The reduction given in Section 11.0 also implies that $P \mid prec, p_j = 1 \mid \Sigma C_j$ is NP-hard. Hu's algorithm does not yield an optimal ΣC_j schedule in the case of intrees, but in the case of outtrees, Rosenfeld [-] has observed that critical path scheduling minimizes both $C_{\max}$ and ΣC_j. Similarly, Garey [-] has shown that the Coffman–Graham algorithm minimizes ΣC_j as well.

As far as approximation algorithms for $P \mid prec, p_j = 1 \mid C_{\max}$ are concerned, we have already noted in Section 11.0 that, unless P = NP, the best possible worst-case bound for a polynomial-time algorithm would be $\frac{4}{3}$. The performance of both Hu's algorithm and the Coffman–Graham algorithm has been analyzed.

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

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

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

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

If MS denotes the algorithm which schedules as the next job the one having the greatest number of successors, then Ibarra & Kim [1976] prove that

$$C_{\max}(\mathrm{MS})/C^*_{\max} \leqslant \tfrac{4}{3} \quad \text{for } m = 2\,. \qquad (\dagger)$$

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

Finally, we mention some results for related models.

Ullman [1975] and Lenstra & Rinnooy Kan [1978] show that both $P2 \mid prec, p_j \in \{1, 2\} \mid C_{\max}$ and $P2 \mid prec, p_j \in \{1, 2\} \mid \Sigma C_j$ are NP-hard. Nakajima, Leung & Hakimi [1981] give a complicated $O(n \log n)$ algorithm to find the optimal solution for $P2 \mid tree, p_j \in \{1, 2\} \mid C_{\max}$; for practical purposes, a heuristic due to Kaufman [1974] which has a worst-case *absolute* error of 1, may be more attractive. Du & Leung [1989] give an $O(n^2 \log n)$ algorithm to solve $P2 \mid tree, p_j \in \{1, 3\} \mid C_{\max}$ to optimality. On the other hand, Du & Leung

[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

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

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

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

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,

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

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,

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

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

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

Kunde [1981] shows that for tree-type and chain-type precedence constraints, there are slightly improved upper bounds for CP of $2-2/(m+1)$ and $\frac{5}{3}$, respectively. For now, let $C^*_{\max}(pmtn)$ denote the optimal value of $C_{\max}$ if preemption is allowed. Kaufman [1974] shows that for tree-type precedence constraints,

$$C_{\max}(\mathrm{CP}) \leqslant C^*_{\max}(pmtn) + \left(1 - \frac{1}{m}\right) \max_j p_j / \min_j p_j \,. \qquad (\dagger)$$

Du, Leung & Young [1991] prove that $P2 \mid tree \mid C_{\max}$ is strongly NP-hard, even for chains. Graham [-] shows that for general precedence constraints

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

For $P \mid prec, r_j \mid L_{\max}$, Hall & Shmoys [1989] observe that in the delivery time model, the same proof technique again yields

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

As remarked above, it is an open question whether $Pm \mid prec, p_j = 1 \mid C_{\max}$ (i.e., with m fixed) is solvable in polynomial time. In fact, it is a challenging problem even to approximate an optimal solution appreciably better than a factor of 2 in polynomial time for fixed values of m.

Even less is known about approximation algorithms for uniform machines. Liu & Liu [1974b] also consider $Q \mid prec \mid C_{\max}$ and show that

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

Note that this yields the result of Graham [1966] when all speeds are equal. As above, similar bounds can be proved relative to the preemptive optimum, or relative to an altered problem.

Jaffe [1980a] shows that using all of the machines in list scheduling may be wasteful in the worst case. The arguments of Liu & Liu are generalized to show that by list scheduling on the fastest l machines (LS_l), if $s_1 \geqslant \cdots \geqslant s_m$,

$$C_{\max}(\mathrm{LS}_l)/C^*_{\max} \leqslant \sum_{i=1}^{m} s_i \Big/ \sum_{i=1}^{l} s_i + s_1/s_l - s_1 \Big/ \sum_{i=1}^{l} s_i \,. \qquad (\dagger)$$

By minimizing this quantity, Jaffe derives an algorithm LS* for which

$$C_{\max}(\mathrm{LS}^*)/C^*_{\max} \leqslant \sqrt{m} + \mathrm{O}(m^{1/4}) \,.$$

This bound is tight up to a constant factor. The surprising aspect of this algorithm is that the decision about the number of machines to be used is made without the knowledge of the processing requirements.

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.

Lawler [1982a] shows that some well-solvable problems involving the non-preemptive scheduling of unit-time jobs turn out to have well-solvable counterparts involving the preemptive scheduling of jobs with arbitrary processing times. The algorithms of Brucker, Garey & Johnson [1977] for $P \mid intree, p_j = 1 \mid L_{\max}$ and of Garey & Johnson [1976, 1977] for $P2 \mid prec, p_j = 1 \mid L_{\max}$ and $P2 \mid prec, r_j, p_j = 1 \mid L_{\max}$ (see Section 11.1) all have preemptive counterparts. For example, $P \mid pmtn, intree \mid L_{\max}$ can be solved in $O(n^2)$ time. For uniform machines, Lawler shows that $Q2 \mid pmtn, prec \mid L_{\max}$ and $Q2 \mid pmtn, prec, r_j \mid L_{\max}$ can be solved in $O(n^2)$ and $O(n^6)$ time, respectively. These results suggest a strong relationship between the two models.

It is not hard to see that $R2 \mid pmtn, tree \mid C_{\max}$ is NP-hard in the strong sense, even for chains [Lenstra, -].

As to approximation algorithms, Lam & Sethi [1977], much in the same spirit as their work mentioned in Section 11.1, analyze the performance of the Muntz–Coffman (MC) algorithm for $P \mid pmtn, prec \mid C_{\max}$. They show

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

For $Q \mid pmtn, prec \mid C_{\max}$, Horvath, Lam & Sethi [1977] prove that the Muntz–Coffman algorithm guarantees

$$C_{\max}(\mathrm{MC})/C^*_{\max} \leq \sqrt{3m/2} ,$$

and examples are given to prove that this bound is tight within a constant factor. Jaffe [1980b] studies the performance of *maximal usage schedules* (MUS) for $Q \mid pmtn, prec \mid C_{\max}$, i.e., schedules without unforced idleness in which at any time the jobs being processed are assigned to the fastest machines. It is shown that

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

and examples are given for which the bound $\sqrt{m-1}$ is approached arbitrarily closely. A slightly weaker bound on these schedules can also be proved using the techniques of Jaffe [1980a].

PART IV. MULTI-OPERATION MODELS

We now pass on to problems in which each job requires execution on more than one machine. Recall from Section 3 that in an *open shop* (denoted by O) the order in which a job passes through the machines is immaterial, whereas in a *flow shop* (F) each job has the same machine ordering $(M_1, \ldots, M_m)$ and in a *job shop* (J) the jobs may have different machine orderings. We survey these problem classes in Sections 12, 13 and 14, respectively. Our presentation

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.

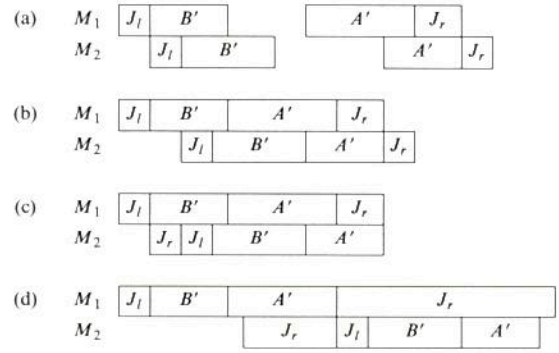


Fig. 3. Solving the two-machine open shop scheduling problem.

12.1. *The nonpreemptive open shop*

There is little hope of finding polynomial-time algorithms for nonpreemptive open shop scheduling problems beyond $O2|\ |C_{\max}$. Gonzalez & Sahni [1976] show that $O3|\ |C_{\max}$ is NP-hard in the ordinary sense. NP-hardness in the strong sense has been established for $O2|\ |L_{\max}$ and $O2|r_j|C_{\max}$ [Lawler, Lenstra & Rinnooy Kan, 1981], $O2|\ |\Sigma C_j$ [Achugbue & Chin, 1982a], $O2|tree|C_{\max}$ and $O|\ |C_{\max}$ [Lenstra, -], and for a number of m-machine multi-operation problems with 0–1 processing times [Gonzalez, 1982].

We mention a few positive results. Adiri & Aizikowitz [1989] investigate machine dominance, which occurs if $\min_j p_{hj} \geqslant \max_j p_{ij}$ for some M_h and M_i with $h \neq i$; under this condition, $O3|\ |C_{\max}$ is well solvable. Fiala [1983] uses results from graph theory to develop an $O(m^3n^2)$ algorithm for $O|\ |C_{\max}$ if $\max_i \Sigma_j p_{ij} \geqslant (16m' \log m' + 5m') \max_{i,j} p_{ij}$, where m' is the roundup of m to the closest power of 2. As to approximation algorithms, Achugbue & Chin [1982a] derive tight bounds on the length of arbitrary schedules and SPT schedules for $O|\ |\Sigma C_j$.

12.2. *The preemptive open shop*

The result on $O2|\ |C_{\max}$ presented in Section 12.0 also applies to the preemptive case. The lower bound on the schedule length remains valid if preemption is allowed. Hence, there is no advantage to preemption for $m = 2$, and $O2|pmtn|C_{\max}$ can be solved in $O(n)$ time.

More generally, $O|pmtn|C_{\max}$ is solvable in polynomial time as well [Gonzalez & Sahni, 1976]. We had already occasion to refer to this result in Section 10.1. An outline of the algorithm, adapted from Lawler & Labetoulle [1978], follows below.

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 $C^*_{\max} \geqslant C$. It is possible to construct a feasible schedule for which $C_{\max} = C$; hence, this schedule will be optimal.

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

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

For a given decrementing set S, let δ 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.

13. Flow shops

13.0. Johnson's algorithm for $F2|\ |C_{\max}$

In one of the first papers on deterministic machine scheduling, Johnson [1954] gives an $O(n \log n)$ algorithm to solve $F2|\ |C_{\max}$. The algorithm is surprisingly simple: first arrange the jobs with $p_{1j} \leq p_{2j}$ in order of nondecreasing p_{1j}, and then arrange the remaining jobs in order of nonincreasing p_{2j}.

The correctness proof of this algorithm is also straightforward. Notice that the algorithm produces a *permutation schedule*, in which each machine processes the jobs in the same order. An easy interchange argument shows that there exists an optimal schedule that is a permutation schedule. We now make three observations. For a permutation schedule, $C_{\max}$ is determined by the processing time of some k jobs on M_1, followed by the processing time of $n+1-k$ jobs on M_2. This implies that, if all p_{ij} are decreased by the same value p, then for each permutation schedule, $C_{\max}$ decreases by $(n+1)p$. Finally, if $p_{1j}=0$, then J_j is scheduled first in some optimal schedule, and similarly, if $p_{2j}=0$, then J_j is scheduled last in some optimal schedule. Putting these pieces together, we see that an optimal schedule can be constructed by repeatedly finding the minimum p_{ij} value among the unscheduled jobs, subtracting this value from all processing times, and scheduling the job with a zero processing time. This algorithm is clearly equivalent to the one given above.

13.1. Two or three machines

As a general result, Conway, Maxwell & Miller [1967] observe that there exists an optimal $F|\ |C_{\max}$ schedule with the same processing order on M_1 and M_2 and the same processing order on M_{m-1} and M_m. Hence, if there are no more than three machines, we can restrict our attention to permutation schedules. The reader is invited to construct a four-machine instance in which a job necessarily 'passes' another one between M_2 and M_3 in the optimal schedule.

$F3|\ |C_{\max}$ is strongly NP-hard [Garey, Johnson & Sethi, 1976]. A fair amount of effort has been devoted to the identification of special cases and variants that are solvable in polynomial time. For example, Johnson [1954] already shows that the case in which $\max_j p_{2j} \leq \max\{\min_j p_{1j}, \min_j p_{3j}\}$ is solved by applying his algorithm to processing times $(p_{1j}+p_{2j},\ p_{2j}+p_{3j})$. Conway, Maxwell & Miller [1967] show that the same rule works if M_2 is a *nonbottleneck machine*, i.e., is a machine that can process any number of jobs at the same time. A two-machine variant involves *time lags* l_j, which are minimum time intervals between the completion time of J_j on M_1 and its starting time on M_2 [Mitten, 1958; Johnson, 1958; Nabeshima, 1963; Szwarc, 1968]; these lags can be viewed as processing times on a nonbottleneck machine in between M_1 and M_2, so one has to apply Johnson's algorithm to

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 $F2 \mid \mid \Sigma C_i$, 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.

13.2. Flow shop scheduling

We know from Section 13.1 that, for the general $F\,|\,|\,C_{\max}$ problem, permutation schedules are not necessarily optimal. Nevertheless, in the literature on *enumerative optimization methods* for flow shop scheduling it has become a tradition to assume identical processing orders on all machines and to look for the best permutation schedule.

The usual enumeration scheme generates schedules by building them from front to back. That is, at a node at the lth level of the search tree, a partial schedule $(J_{\sigma(1)}, \ldots, J_{\sigma(l)})$ has been formed and the jobs with index set $S = \{1, \ldots, n\} - \{\sigma(1), \ldots, \sigma(l)\}$ are candidates for the $(l+1)$th position. One then needs to find a lower bound on the length of all possible completions of the partial schedule. Almost all lower bounds developed so far are captured by the following bounding scheme due to Lageweg, Lenstra & Rinnooy Kan [1978].

Let us relax the constraint that each machine can process at most one job at a time, for all machines but at most two, say, M_u and M_v $(1 \leqslant u \leqslant v \leqslant m)$. We then obtain the following problem. Each job J_j $(j \in S)$ has to be processed on five machines N_{*u}, M_u, N_{uv}, M_v, N_{v*} in that order. N_{*u}, N_{uv} and N_{v*} are nonbottleneck machines, of infinite capacity; if $C(\sigma, i)$ denotes the completion time of $J_{\sigma(l)}$ on M_i, then the processing times of J_j $(j \in S)$ on N_{*u}, N_{uv} and N_{v*} are defined by

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

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

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

respectively. M_u and M_v are ordinary machines of unit capacity, with processing times p_{uj} and p_{vj}, respectively. We wish to find a permutation schedule that minimizes $C_{\max}$. We can interpret N_{*u} as yielding release dates q_{*j} on M_u and N_{v*} as setting due dates $-q_{v*j}$ on M_v, with respect to which $L_{\max}$ is to be minimized. Note that we can remove any of the nonbottleneck machines from the problem by underestimating its contribution to the lower bound to be its minimum processing time; valid lower bounds are obtained by adding these contributions to the optimal solution value of the remaining problem.

If we choose $u = v$ and remove both N_{*u} and N_{u*} from the problem, we obtain the *machine-based bound* proposed by Ignall & Schrage [1965]:

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

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

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

This bound is tight even for LPT schedules, in which the jobs are ordered according to nonincreasing sums of processing times. They also give an $O(mn \log n)$ heuristic H based on $\lfloor m/2 \rfloor$ applications of Johnson's algorithm, with

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

Röck and Schmidt [1983] use an aggregation heuristic that first constructs a two-machine instance by combining the first $\lceil m/2 \rceil$ machines to get M_1 and the rest to get M_2, and then applies Johnson's algorithm; the resulting permutation schedule also has a performance ratio of $\lceil m/2 \rceil$. Belov & Stolin [1974] and Sevastyanov [1975] use geometric tools to obtain permutation schedules that are quite close to the overall optimum. The best result along these lines is due to Sevastyanov [1980], who gives an $O(m^2n^2)$ algorithm S to find a schedule which has an absolute error bound that is independent of n:

$$C_{\max}(S) - C^*_{\max} \leq m(m-1) \max_{i,j} p_{ij} .$$

For the formulation and empirical evaluation of various rules for the construction and iterative improvement of flow shop schedules, we refer to Palmer [1965], Campbell, Dudek & Smith [1970], Dannenbring [1977], Nawaz, Enscore & Ham [1983], Turner & Booth [1987], and Osman & Potts [1989]. The current champions are the fast *insertion* method of Nawaz, Enscore & Ham and the less efficient but more effective *simulated annealing* algorithm of Osman & Potts. Simulated annealing is a randomized variant of iterative improvement, which accepts deteriorations with a small and decreasing probability in an attempt to avoid bad local optima and to get settled in a global optimum. In the experiments of Osman & Potts, the neighborhood of a permutation schedule contains all schedules that can be obtained by moving a single job to another position.

13.3. No wait in process

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

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

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

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

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

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

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.

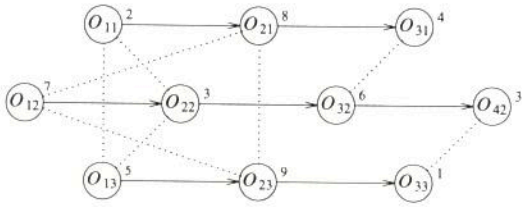
The job shop scheduling problem has now been formulated as the problem of finding an orientation of the edges of a disjunctive graph that minimizes the maximum path weight. We refer to Figure 4 for an example.

14.1. Two or three machines

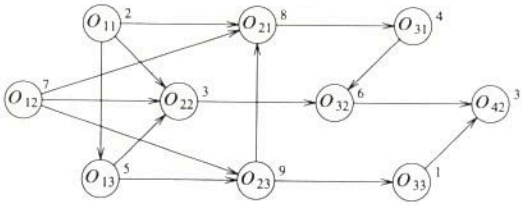
A simple extension of Johnson's algorithm for $F2 \mid \mid C_{max}$ allows solution of $J2 \mid m_j \leqslant 2 \mid C_{max}$ in $O(n \log n)$ time [Jackson, 1956]. Let $\mathcal{J}_i$ be the set of jobs with operations on M_i only $(i = 1, 2)$, and let $\mathcal{J}_{hi}$ be the set of jobs that go 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

J_j	m_j	μ_{1j}	μ_{2j}	μ_{3j}	μ_{4j}	p_{1j}	p_{2j}	p_{3j}	p_{4j}
J_1	3	M_1	M_2	M_3	—	2	8	4	—
J_2	4	M_2	M_1	M_3	M_4	7	3	6	3
J_3	3	M_1	M_2	M_4	—	5	9	1	—

(a)



(b)



(c)

Fig. 4. A job shop scheduling problem. (a) Instance. (b) Instance, represented as a disjunctive graph. (c) Feasible schedule, represented as an acyclic directed graph.

schedule by executing the jobs on M_1 in the order $(\mathcal{J}_{12}, \mathcal{J}_1, \mathcal{J}_{21})$ and on M_2 in the order $(\mathcal{J}_{21}, \mathcal{J}_2, \mathcal{J}_{12})$.

Hefetz & Adiri [1982] solve another special case, $J2 \mid p_{ij} = 1 \mid C_{\max}$, in time linear in the total number of operations, through a rule that gives priority to the longest remaining job. Brucker [1981, 1982] extends this result to $J2 \mid p_{ij} = 1 \mid L_{\max}$.

This, however, is probably as far as we can get. $J2 \mid m_j \leq 3 \mid C_{\max}$ and $J3 \mid m_j \leq 2 \mid C_{\max}$ are NP-hard [Lenstra, Rinnooy Kan & Brucker, 1977; Gonzalez & Sahni, 1978a], $J2 \mid p_{ij} \in \{1, 2\} \mid C_{\max}$ and $J3 \mid p_{ij} = 1 \mid C_{\max}$ are strongly NP-hard [Lenstra & Rinnooy Kan, 1979], and these results still hold if preemption is allowed. Also, $J2 \mid pmtn \mid \Sigma C_j$ is strongly NP-hard [Lenstra, -]; recall that the corresponding open shop and flow shop problems are open.

14.2. General job shop scheduling

Optimization algorithms for the $J \mid \mid C_{\max}$ problem proceed by branch and bound. We will describe methods of that type in terms of the disjunctive graph $(\mathcal{O}, A, E)$, where $\mathcal{O}$ is the set of operations, A the arc set, and E the edge set.

A node in the search tree is usually characterized by an orientation of each

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

promising machines using Carlier's [1982] algorithm, and obtains upper bounds with a heuristic due to Lageweg, Lenstra & Rinnooy Kan [1977]. Carlier & Pinson [1988] implement their novel enumeration schemes, the $1 \mid pmtn, prec, r_j \mid L_{\max}$ bound (which can be computed in polynomial time), and a collection of powerful elimination rules for which we refer to their paper.

Most *approximation algorithms* for job shop scheduling use a dispatch rule, which schedules the operations according to some priority function. Gonzalez & Sahni [1978a] observe that the performance guarantees for the flow shop algorithms AS and LPT (see Section 13.2) also apply to the case of a job shop. A considerable effort has been invested in the empirical testing of rules of this type [Gere, 1966; Conway, Maxwell & Miller, 1967; Day & Hottenstein, 1970; Panwalkar & Iskander, 1977; Haupt, 1989].

Adams, Balas & Zawack [1988] develop a *sliding bottleneck heuristic*, which employs an ingenious combination of schedule construction and iterative improvement, guided by solutions to single-machine problems of the type described above. They also embed this method in a second heuristic that proceeds by partial enumeration of the solution space.

Matsuo, Suh & Sullivan [1988] and Van Laarhoven, Aarts & Lenstra [1992] apply the principle of *simulated annealing* (see Section 13.2) to the job shop scheduling problem. In the latter paper, the neighborhood of a schedule contains all schedules that can be obtained by interchanging two operations O_{ij} and $O_{i'j'}$ on the same machine such that the arc $(O_{ij}, O_{i'j'})$ is on a maximum weight path. In the former paper, the neighborhood structure is more complex.

14.3. $10 \times 10 = 930$

The computational merits of all these algorithms are accurately reflected by their performance on the notorious 10-job 10-machine problem instance due to Fisher & Thompson [1963].

The single-machine bound, maximized over all machines, has a value of 808. McMahon & Florian [1975] found a schedule of length 972. Fisher, Lageweg, Lenstra & Rinnooy Kan [1983] applied surrogate duality relaxation of the capacity constraints and of the precedence constraints to find lower bounds of 813 and 808, respectively; the computational effort involved did not encourage them to carry on the search beyond the root of the tree. Lageweg [1984] found a schedule of length 930, without proving optimality; he also computed a number of multi-machine lower bounds, ranging from a three-machine bound of 874 to a six-machine bound of 907. Carlier & Pinson [1988] were the first to prove optimality of the value 930, after generating 22 021 nodes and five hours of computing. The main drawback of all these enumerative methods, besides the limited problem sizes that can be handled, is their sensitivity towards particular problem instances and also towards the initial value of the upper bound.

The computational experience with polyhedral techniques that has been reported until now is slightly disappointing in view of what has been achieved

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

during its execution. A schedule is *feasible* with respect to the resources if at any time t the index set I_t of jobs being executed at time t satisfies $\Sigma_{j \in I_t} r_{hj} \leq s_h$, for $h = 1, \ldots, l$.

In the case $P2 \mid p_j = 1 \mid C_{\max}$, Garey & Johnson [1975] propose to represent the resource constraints by a graph with vertex set $\{1, \ldots, n\}$ and an edge $\{j, k\}$ whenever $r_{hj} + r_{hk} \leq s_h$ for $h = 1, \ldots, l$. That is, vertices j and k are adjacent if and only if J_j and J_k can be processed simultaneously. A matching M in the graph obviously corresponds to a schedule of length $n - |M|$, and an optimal schedule is obtained by computing a maximum cardinality matching.

15.1. Machines and resources

Sequencing and scheduling is concerned with the optimal allocation of scarce resources to activities over time. So far, the resources and the activities have been of a relatively simple nature. It was assumed that an activity, or job, requires at most one resource, or machine, at a time. Also, a machine is able to process at most one job at a time. This unit-capacity is constant, and not affected by its use.

It is obvious that scheduling situations of a more general nature exist. Certain types of resources are depleted by use (e.g., money or energy) or are available in amounts that vary over time, in a predictable manner (e.g., seasonal labor) or in an unpredictable manner (e.g., vulnerable equipment). At one point in time, a resource may be shared among several jobs, and a job may need several resources. The resource amounts required by a job may vary during its processing and, indeed, the processing time itself could depend on the amount or type of resource allocated, as in the case of uniform or unrelated machines.

Through these generalizations, the domain of deterministic scheduling theory is considerably extended. Usually referred to as *resource-constrained project scheduling*, the area covers a tremendous variety of problem types.

15.2. Classification and complexity

To approach this area in the best tradition of deterministic scheduling theory would require the development of a detailed problem classification, followed by a complexity analysis involving polynomial-time algorithms and NP-hardness proofs.

A modest attempt along these lines was made by Blazewicz, Lenstra & Rinnooy Kan [1983]. They consider resource constraints of the type defined in the first paragraph of Section 15.0, and propose to include these in the second field of the problem classification through a parameter $res\lambda\sigma\rho$, where λ, σ, and ρ specify the number of resources, their sizes, and the amounts required. More precisely,

– if λ is a positive integer, then l is a constant, equal to λ; if $\lambda = \cdot$, then l is specified as part of the input;

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

– if ρ is a positive integer, then all r_{hj} have a constant upper bound, equal to ρ; 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 $P3 \mid res\, 1 \cdots, p_j = 1 \mid C_{\max}$ is an immediate generalization of the

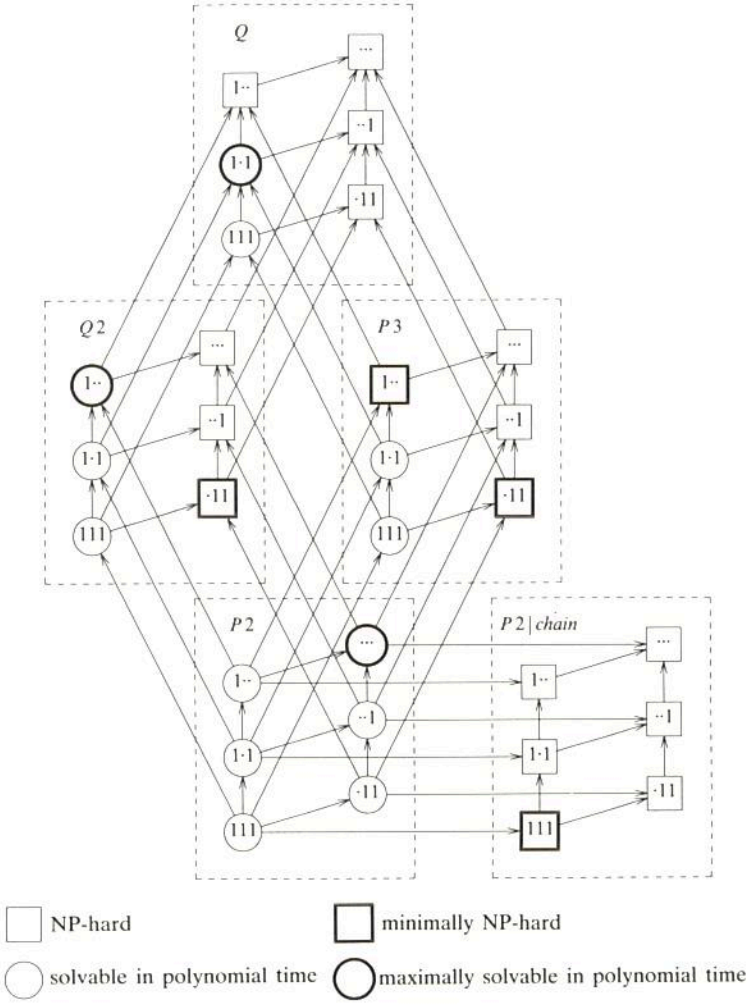


Fig. 5. Complexity of scheduling unit-time jobs on parallel machines subject to resource constraints.

3-partition problem and thereby strongly NP-hard; see Section 2 and Garey & Johnson [1975].

These results are not encouraging, in that virtually all except the simplest resource-constrained project scheduling problems turn out to be NP-hard. In the next section, we abolish the search for special structure and review two optimization models of a fairly general nature.

15.3. Two optimization models

The early literature on optimization and approximation in resource-constrained project scheduling is reviewed by Davis [1966, 1973]. Optimization models are traditionally cast in terms of integer programming. We start by presenting one such formulation, due to Talbot & Patterson [1978] and Christofides, Alvarez-Valdes & Tamarit [1987].

For simplicity, we consider the $P \mid res \cdots, prec \mid C_{\max}$ problem, i.e., $P \mid prec \mid C_{\max}$ with resource constraints as described in the first paragraph of Section 15.0. We also suppose that $m \geqslant n$ and that one job, J_n, succeeds all others. The problem is then to find nonnegative job completion times C_j, which define index sets I_t of jobs executed at time t, such that C_n is minimized subject to precedence constraints and resource constraints:

$$C_j + p_j \leqslant C_k \text{ whenever } J_j \rightarrow J_k ,$$

$$\sum_{j \in I_t} r_{hj} \leqslant s_h \quad \text{for all } R_h \text{ and all } t .$$

To convert the latter set of constraints into linear form, we introduce 0–1 variables y_{jt}, with $y_{jt} = 1$ if and only if $C_j = t$. Obviously, $C_j = \Sigma_t \, t y_{jt}$, and the resource constraints can be rewritten as

$$\sum_{j=1}^{n} r_{hj} \sum_{u=t}^{t+p_j-1} y_{jt} \leqslant s_h \quad \text{for all } h \text{ and } t .$$

Branch and bound algorithms using bounds based on the linear relaxation, cutting planes, and Lagrangean relaxation of the resource constraints are reasonably effective for problems with up to three resources and 25 jobs.

An entirely different approach was taken by Bartusch, Möhring & Radermacher [1988b]. Recall the formulation of the $J \mid \mid C_{\max}$ problem in terms of a disjunctive graph, where each *edge* corresponds to a pair of operations that cannot be processed simultaneously since they require the same machine. Following earlier work by Balas [1970], Bartusch, Möhring & Radermacher generalize this idea, by defining resource constraints in the form of a family $\mathcal{N} = \{N_1, \ldots, N_l\}$ of *forbidden subsets*. Each N_h is a subset of jobs that cannot be executed simultaneously because of its collective resource requirements; this

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 $P \mid p_j \sim \exp(\lambda_j) \mid \mathrm{E}C_{\max}, \mathrm{E}\Sigma C_j$

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 λ_j, for $j = 1, \ldots, n$. We want to minimize the

expected maximum completion time $E\boldsymbol{C}_{\max}$ or the expected total completion time $E\Sigma\boldsymbol{C}_j$. (As noted before, random variables are printed in boldface italic.)

Results of Bruno & Downey [1977] for $m = 2$ and of Bruno, Downey & Frederickson [1981] for arbitrary m state that these problems are solved by simple list scheduling policies. The *longest expected processing time* (LEPT) rule, which schedules the jobs in order of nonincreasing values $1/\lambda_j$, minimizes $E\boldsymbol{C}_{\max}$; the *shortest expected processing time* (SEPT) rule, which schedules the jobs in the reverse order, minimizes $E\Sigma\boldsymbol{C}_j$.

We will sketch a proof of the optimality of the LEPT rule for minimizing $E\boldsymbol{C}_{\max}$. This proof, which is due to Weiss & Pinedo [1980], relies on the formulation of the preemptive version of the problem in terms of a semi-Markov decision process. Note, however, that the LEPT rule will never preempt a job, because of the memoryless property of the exponential distribution.

Let $N = \{1, \ldots, n\}$ be the index set of all jobs, and let $F_\pi(S)$ denote the minimum expected maximum completion time for the jobs indexed by $S \subseteq N$ under a scheduling policy π. Consider a policy π that at time 0 selects a set $S_\pi \subseteq N$ to be processed, preempts the schedule at time $t > 0$, and applies the LEPT rule from time t onwards. By time t, a job J_j is completed with probability $\lambda_j t + o(t)$, and two or more jobs are completed with probability $o(t)$, for $t \to 0$. It now follows from Markov decision theory that

$$F_\pi(N) = t + \sum_{j \in S_\pi} \lambda_j t F_{\mathrm{LEPT}}(N - \{j\}) + \Big(1 - \sum_{j \in S_\pi} \lambda_j t\Big) F_{\mathrm{LEPT}}(N) + o(t), \quad t \to 0.$$

Without loss of generality, we assume that $|S_\pi| = m < n$. If π is not the LEPT policy, then there exist jobs J_k, J_l with $\lambda_k < \lambda_l$ such that $k \notin S_\pi$, $l \in S_\pi$. Now define another policy π' that at time 0 selects a set $S_{\pi'} = S_\pi \cup \{k\} - \{l\}$ and applies LEPT from time t onwards. We have that

$$\begin{aligned} F_\pi(N) - F_{\pi'}(N) = t[&\lambda_k(F_{\mathrm{LEPT}}(N) - F_{\mathrm{LEPT}}(N - \{k\})) \\ &- \lambda_l(F_{\mathrm{LEPT}}(N) - F_{\mathrm{LEPT}}(N - \{l\}))] \\ &+ o(t), \quad t \to 0. \end{aligned}$$

Lengthy but rather straightforward calculations, which are not given here, show that the expression within square brackets is positive. The argument is by induction on n and uses the following simple recursion:

$$F_{\mathrm{LEPT}}(N) = \Big(1 + \sum_{j \in S_{\mathrm{LEPT}}} \lambda_j F_{\mathrm{LEPT}}(N - \{j\})\Big) \Big/ \sum_{j \in S_{\mathrm{LEPT}}} \lambda_j,$$

where S_{LEPT} contains the smallest m λ's. It follows that, if t is small enough,

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 $F_\pi(N) > F_{\text{LEPT}}(N)$.

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,

Radermacher & Weiss [1984, 1985], Möhring & Radermacher [1985b] and Frenk [1988] for further information.

16.2. The single machine

In stochastic single-machine scheduling, Gittins' work on *dynamic allocation indices* initiated an important line of research. A prototypical result is the following. One machine has to process n jobs. The job processing times $\boldsymbol{p}_j$ are independent, nonnegative and identically distributed random variables, whose distribution function F has an increasing completion rate $(\mathrm{d}F(t)/\mathrm{d}t)/(1-F(t))$. If job J_j completes at time C_j, then a reward $\alpha_j \mathrm{e}^{-\beta C_j}$ is incurred. The objective is to maximize the *total* expected reward. It is achieved by scheduling the jobs in order of nonincreasing ratios $\alpha_j \mathrm{E}\mathrm{e}^{-\beta p_j}/(1-\mathrm{E}\mathrm{e}^{-\beta p_j})$. This ratio can be interpreted as the expected reward for J_j per unit of expected discounted processing time. The increasing completion rate of F ensures that there is no advantage to preemption.

This result follows from the mathematical theory of *bandit processes*. Subsequent work by Gittins & Glazebrook has led to many extensions. Forst [1984] present a survey of this part of the literature.

Another class of results concerns the situation in which the $\boldsymbol{p}_j$ are independent, nonnegative random variables, and the objective is to minimize the expected *maximum* job completion cost subject to precedence constraints. Hodgson [1977] generalizes the algorithm of Lawler [1973] for $1 \mid prec \mid f_{\max}$ (see Section 4.0) to solve this problem. The result subsumes earlier work involving deterministic due dates, such as the minimization of the maximum probability of lateness [Banerjee, 1965], the maximization of the probability that every job is on time [Crabill & Maxwell, 1969], and the minimization of the maximum expected weighted tardiness [Blau, 1973].

16.3. Parallel machines

Research in stochastic parallel machine scheduling has focused on extending the results quoted in Section 16.0 beyond the realm of exponential distributions. Weber has shown that, as a necessary condition, the processing time distributions have to be consistent in terms of completion rates (i.e., either all decreasing or all increasing) or in terms of likelihood rates (i.e., the $\log \mathrm{d}F_j/\mathrm{d}t$ either all convex or all concave). Weiss [1982] reviews this work. Weber, Varaiya & Walrand [1986] show that SEPT minimizes the expected total completion time on identical machines if the processing times are stochastically comparable.

The extension to uniform machines has been explored by Agrawala, Coffman, Garey & Tripathi [1984], Kumar & Walrand [1985], Coffman, Flatto, Garey & Weber [1987] and Righter [1988].

For the case of intree precedence constraints and exponential processing times, Pinedo & Weiss [1985] prove that HLF minimizes the expected maxi-

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 $\boldsymbol{p}_{1j} \sim \exp(\lambda_j)$ and $\boldsymbol{p}_{2j} \sim \exp(\mu_j)$, 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 $\boldsymbol{p}_{ij}$ are independent random variables whose distributions do not depend on i. Weber [1979] shows that, in the exponential case, any sequence minimizes $\mathrm{E}\boldsymbol{C}_{\max}$. Pinedo [1982] observes that, under fairly general conditions, any sequence for which $\mathrm{E}\boldsymbol{p}_{ij}$ 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

the second author was visiting the Sloan School of Management of the Massachusetts Institute of Technology. The research of the second and fourth author was partially supported by the Presidential Young Investigator Award of the fourth author, with matching support from IBM, Sun Microsystems, and UPS. The research of the fourth author was also supported by Air Force contract AFOSR-86-0078.

References

Abdul-Razaq, T.S., and C.N. Potts (1988). Dynamic programming state-space relaxation for single-machine scheduling. *J. Oper. Res. Soc.* 39, 141–152.

Achugbue, J.O., and F.Y. Chin (1981). Bounds on schedules for independent tasks with similar execution times. *J. Assoc. Comput. Mach.* 28, 81–99.

Achugbue, J.O., and F.Y. Chin (1982a). Scheduling the open shop to minimize mean flow time. *SIAM J. Comput.* 11, 709–720.

Achugbue, J.O., and F.Y. Chin (1982b). Complexity and solution of some three-stage flow shop scheduling problems. *Math. Oper. Res.* 7, 532–544.

Adams, J., E. Balas and D. Zawack (1988). The shifting bottleneck procedure for job shop scheduling. *Management Sci.* 34, 391–401.

Adiri, I., and N. Aizikowitz (1989). Openshop scheduling problems with dominated machines. *Naval Res. Logist.* 36, 273–281.

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

Agrawala, A.K., E.G. Coffman Jr, M.R. Garey and S.K. Tripathi (1984). A stochastic optimization algorithm minimizing expected flow times on uniform processors. *IEEE Trans. Comput.* 33, 351–356.

Bagchi, U., and R.H. Ahmadi (1987). An improved lower bound for minimizing weighted completion times with deadlines. *Oper. Res.* 35, 311–313.

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

Baker, K.R., E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan (1983). Preemptive scheduling of a single machine to minimize maximum cost subject to release dates and precedence constraints. *Oper. Res.* 31, 381–386.

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

Baker, K.R., and G.D. Scudder (1990). Sequencing with earliness and tardiness penalties: A review. *Oper. Res.* 38, 22–36.

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

Balas, E. (1970). Project scheduling with resource constraints, in: E.M.L. Beale (ed.), *Applications of Mathematical Programming Techniques*, English Univ. Press, London, pp. 187–200.

Balas, E. (1985). On the facial structure of scheduling polyhedra. *Math. Programming Stud.* 24, 179–218.

Banerjee, B.P. (1965). Single facility sequencing with random execution times. *Oper. Res.* 13, 358–364.

Barker, J.R., and G.B. McMahon (1985). Scheduling the general job-shop. *Management Sci.* 31, 594–598.

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

Bartusch, M., R.H. Möhring and F.J. Radermacher (1988a). *M*-machine unit time scheduling: A report of ongoing research, in: A. Kurzhanski, K. Neumann and D. Pallaschke (eds.), *Optimization, Parallel Processing, and Applications*, Lecture Notes in Economics and Mathematical Systems, Vol. 304, Springer, Berlin, pp. 165–212.

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.

Carlier, J. (1982). The one-machine sequencing problem. *European J. Oper. Res.* 11, 42–47.
Carlier, J. (1987). Scheduling jobs with release dates and tails on identical machines to minimize makespan. *European J. Oper. Res.* 29, 298–306.
Carlier, J., and E. Pinson (1988). An algorithm for solving the job-shop problem. *Management Sci.* 35, 164–176.
Charlton, J.M., and C.C. Death (1970). A generalized machine scheduling algorithm. *Oper. Res. Quart.* 21, 127–134.
Chen, N.-F. (1975). An analysis of scheduling algorithms in multiprocessing computing systems, Technical Report UIUCDCS-R-75-724, Department of Computer Science, Univ. of Illinois at Urbana-Champaign.
Chen, N.-F., and C.L. Liu (1975). On a class of scheduling algorithms for multiprocessors computing systems, in: T.-Y. Feng (ed.), *Parallel Processing*, Lecture Notes in Computer Science, Vol. 24, Springer, Berlin, pp. 1–16.
Cheng, T.C.E., and M.C. Gupta (1989). Survey of scheduling research involving due date determination decisions. *European J. Oper. Res.* 38, 156–166.
Chin, F.Y., and L.-L. Tsai (1981). On *J*-maximal and *J*-minimal flow-shop schedules. *J. Assoc. Comput. Mach.* 28, 462–476.
Cho, Y., and S. Sahni (1980). Bounds for list schedules on uniform processors. *SIAM J. Comput.* 9, 91–103.
Cho, Y., and S. Sahni (1981). Preemptive scheduling of independent jobs with release and due times on open, flow and job shops. *Oper. Res.* 29, 511–522.
Christofides, N., R. Alvarez-Valdes and J.M. Tamarit (1987). Project scheduling with resource constraints: A branch and bound approach. *European J. Oper. Res.* 29, 262–273.
Coffman Jr, E.G. (ed.) (1976). *Computer & Job/Shop Scheduling Theory*, Wiley, New York.
Coffman Jr, E.G., L. Flatto, M.R. Garey and R.R. Weber (1987). Minimizing expected makespans on uniform processor systems. *Adv. in Appl. Probab.* 19, 177–201.
Coffman Jr, E.G., L. Flatto and G.S. Lueker (1984). Expected makespans for largest-fit multiprocessor scheduling, in: E. Gelenbe (ed.), *Performance '84*, North-Holland, Amsterdam, pp. 491–506.
Coffman Jr, E.G., M.R. Garey and D.S. Johnson (1978). An application of bin-packing to multiprocessor scheduling. *SIAM J. Comput.* 7, 1–17.
Coffman Jr, E.G., and E.N. Gilbert (1985). On the expected relative performance of list scheduling. *Oper. Res.* 33, 548–561.
Coffman Jr, E.G., and R.L. Graham (1972). Optimal scheduling for two-processor systems. *Acta Inform.* 1, 200–213.
Coffman Jr, E.G., M. Hofri and G. Weiss (1989). Scheduling stochastic jobs with a two point distribution on two parallel machines. *Probab. Engrg. Inform. Sci.* 3, 89–116.
Coffman Jr, E.G., G.S. Lueker and A.H.G. Rinnooy Kan (1988). Asymptotic methods in the probabilistic analysis of sequencing and packing heuristics. *Management Sci.* 34, 266–290.
Conway, R.W., W.L. Maxwell and L.W. Miller (1967). *Theory of Scheduling*, Addison-Wesley, Reading, MA.
Cook, S.A. (1971). The complexity of theorem-proving procedures, *Proc. 3rd Annual ACM Symp. Theory of Computing*, pp. 151–158.
Crabill, T.B., and W.L. Maxwell (1969). Single machine sequencing with random processing times and random due-dates. *Naval Res. Logist. Quart.* 16, 549–554.
Dannenbring, D.G. (1977). An evaluation of flow shop sequencing heuristics. *Management Sci.* 23, 1174–1182.
Davida, G.I., and D.J. Linton (1976). A new algorithm for the scheduling of tree structured tasks. *Proc. Conf. Inform. Sci and Syst.*, Baltimore, MD, pp. 543–548.
Davis, E., and J.M. Jaffe (1981). Algorithms for scheduling tasks on unrelated processors. *J. Assoc. Comput. Mach.* 28, 721–736.
Davis, E.W. (1966). Resource allocation in project network models – A survey. *J. Indust. Engrg.* 17, 177–188.
Davis, E.W. (1973). Project scheduling under resource constraints – Historical review and categorization of procedures. *AIIE Trans.* 5, 297–313.

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.

Fisher, M.L., and A.M. Krieger (1984). Analysis of a linearization heuristic for single-machine scheduling to maximize profit. *Math. Programming* 28, 218–225.

Fisher, M.L., B.J. Lageweg, J.K. Lenstra and A.H.G. Rinnooy Kan (1983). Surrogate duality relaxation for job scheduling. *Discrete Appl. Math.* 5, 65–75.

Foley, R.D., and S. Suresh (1986). Scheduling n nonoverlapping jobs and two stochastic jobs in a flow shop. *Naval Res. Logist. Quart.* 33, 123–128.

Forst, F.G. (1984). A review of the static, stochastic job sequencing literature. *Opsearch* 21, 127–144.

Frederickson, G.N. (1983). Scheduling unit-time tasks with integer release times and deadlines. *Inform. Process. Lett.* 16, 171–173.

French, S. (1982). *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop*, Horwood, Chichester.

Frenk, J.B.G. (1988). A general framework for stochastic one-machine scheduling problems with zero release times and no partial ordering, Report 8819, Econometric Institute, Erasmus University, Rotterdam.

Frenk, J.B.G., and A.H.G. Rinnooy Kan (1986). The rate of convergence to optimality of the LPT rule. *Discrete Appl. Math.* 14, 187–197.

Frenk, J.B.G., and A.H.G. Rinnooy Kan (1987). The asymptotic optimality of the LPT rule. *Math. Oper. Res.* 12, 241–254.

Friesen, D.K. (1984). Tighter bounds for the multifit processor scheduling algorithm. *SIAM J. Comput.* 13, 170–181.

Friesen, D.K. (1987). Tighter bounds for LPT scheduling on uniform processors. *SIAM J. Comput.* 16, 554–660.

Friesen, D.K., and M.A. Langston (1983). Bounds for multifit scheduling on uniform processors. *SIAM J. Comput.* 12, 60–70.

Friesen, D.K., and M.A. Langston (1986). Evaluation of a MULTIFIT-based scheduling algorithm. *J. Algorithms* 7, 35–59.

Frostig, E. (1988). A stochastic scheduling problem with intree precedence constraints. *Oper. Res.* 36, 937–943.

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

Gabow, H.N. (1982). An almost linear-time algorithm for two-processor scheduling. *J. Assoc. Comput. Mach.* 29, 766–780.

Gabow, H.N. (1988). Scheduling UET systems on two uniform processors and length two pipelines. *SIAM J. Comput.* 17, 810–829.

Gabow, H.N., and R.E. Tarjan (1985). A linear-time algorithm for a special case of disjoint set union. *J. Comput. System Sci.* 30, 209–221.

Garey, M.R. (-). Unpublished.

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

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

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

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

Garey, M.R., and D.S. Johnson (1978). Strong NP-completeness results: Motivation, examples and implications. *J. Assoc. Comput. Mach.* 25, 499–508.

Garey, M.R., and D.S. Johnson (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco.

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

Garey, M.R., D.S. Johnson, B.B. Simons and R.E. Tarjan (1981). Scheduling unit-time tasks with arbitrary release times and deadlines. *SIAM J. Comput.* 10, 256–269.

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.

Graham, R.L., E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. *Ann. Discrete Math.* 5, 287–326.

Graves, S.C. (1981). A review of production scheduling. *Oper. Res.* 29, 646–675.

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

Gupta, S.K., and J. Kyparisis (1987). Single machine scheduling research. *Omega* 15, 207–227.

Gusfield, D. (1984). Bounds for naive multiple machine scheduling with release times and deadlines. *J. Algorithms* 5, 1–6.

Hall, L.A., and D.B. Shmoys (1989). Approximation schemes for constrained scheduling problems, *Proc. 30th IEEE Symp. Foundations of Computer Science*, pp. 134–139.

Hall, L.A., and D.B. Shmoys (1992). Jackson's rule for one-machine scheduling: Making a good heuristic better. *Math. Oper. Res.* 17, 22–35.

Hariri, A.M.A., and C.N. Potts (1983). An algorithm for single machine sequencing with release dates to minimize total weighted completion time. *Discrete Appl. Math.* 5, 99–109.

Hariri, A.M.A., and C.N. Potts (1984). Algorithms for two-machine flow-shop sequencing with precedence constraints. *European J. Oper. Res.* 17, 238–248.

Haupt, R. (1989). A survey of priority rule-based scheduling. *OR Spektrum* 11, 3–16.

Hefetz, N., and I. Adiri (1982). An efficient optimal algorithm for the two-machines unit-time jobshop schedule-length problem. *Math. Oper. Res.* 7, 354–360.

Hochbaum, D.S., and D.B. Shmoys (1987). Using dual approximation algorithms for scheduling problems: Theoretical and practical results. *J. Assoc. Comput. Mach.* 34, 144–162.

Hochbaum, D.S., and D.B. Shmoys (1988). A polynomial approximation scheme for machine scheduling on uniform processors: Using the dual approximation approach. *SIAM J. Comput.* 17, 539–551.

Hodgson, S.M. (1977). A note on single machine sequencing with random processing times. *Management Sci.* 23, 1144–1146.

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

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

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

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

Horvath, E.C., S. Lam and R. Sethi (1977). A level algorithm for preemptive scheduling. *J. Assoc. Comput. Mach.* 24, 32–43.

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

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

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

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

Ibarra, O.H., and C.E. Kim (1978). Approximation algorithms for certain scheduling problems. *Math. Oper. Res.* 3, 197–204.

Ignall, E., and L. Schrage (1965). Application of the branch and bound technique to some flow-shop scheduling problems. *Oper. Res.* 13, 400–412.

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

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

Jaffe, J.M. (1980a). Efficient scheduling of tasks without full use of processor resources. *Theoret. Comput. Sci.* 12, 1–17.

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

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.

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

Lawler, E.L. (1976a). Sequencing to minimize the weighted number of tardy jobs. *RAIRO Rech. Opér.* 10(5) Suppl., 27–33.

Lawler, E.L. (1976b). *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, New York.

Lawler, E.L. (1977). A 'pseudopolynomial' algorithm for sequencing jobs to minimize total tardiness. *Ann. Discrete Math.* 1, 331–342.

Lawler, E.L. (1978a). Sequencing jobs to minimize total weighted completion time subject to precedence constraints. *Ann. Discrete Math.* 2, 75–90.

Lawler, E.L. (1978b). Sequencing problems with series parallel precedence constraints, Unpublished manuscript.

Lawler, E.L. (1979a). Preemptive scheduling of uniform parallel machines to minimize the weighted number of late jobs, Report BW 105, Centre for Mathematics and Computer Science, Amsterdam.

Lawler, E.L. (1979b). Efficient implementation of dynamic programming algorithms for sequencing problems, Report BW 106, Centre for Mathematics and Computer Science, Amsterdam.

Lawler, E.L. (1982a). Preemptive scheduling of precedence-constrained jobs on parallel machines, in: Dempster, Lenstra & Rinnooy Kan [1982], pp. 101–123.

Lawler, E.L. (1982b). Scheduling a single machine to minimize the number of late jobs, Preprint, Computer Science Division, Univ. of California, Berkeley.

Lawler, E.L. (1982c). A fully polynomial approximation scheme for the total tardiness problem. *Oper. Res. Lett.* 1, 207–208.

Lawler, E.L. (1983). Recent results in the theory of machine scheduling, in: A. Bachem, M. Grötschel and B. Korte (eds.), *Mathematical Programming: The State of the Art – Bonn 1982*, Springer, Berlin, 202–234.

Lawler, E.L. (-). Unpublished.

Lawler, E.L., and J. Labetoulle (1978). On preemptive scheduling of unrelated parallel processors by linear programming. *J. Assoc. Comput. Mach.* 25, 612–619.

Lawler, E.L., and J.K. Lenstra (1982). Machine scheduling with precedence constraints, in: I. Rival (ed.), *Ordered Sets*, Reidel, Dordrecht, pp. 655–675.

Lawler, E.L., J.K. Lenstra and A.H.G. Rinnooy Kan (1981). Minimizing maximum lateness in a two-machine open shop. *Math. Oper. Res.* 6, 153–158; Erratum. *Math. Oper. Res.* 7 (1982) 635.

Lawler, E.L., J.K. Lenstra and A.H.G. Rinnooy Kan (1982). Recent developments in deterministic sequencing and scheduling: A survey, in: Dempster, Lenstra & Rinnooy Kan [1982], pp. 35–73.

Lawler, E.L., J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys (eds.) (1985). *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Wiley, Chichester.

Lawler, E.L., J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys (1989). Sequencing and scheduling: Algorithms and complexity, Report BS-R8909, Centre for Mathematics and Computer Science, Amsterdam.

Lawler, E.L., and C.U. Martel (1982). Computing maximal 'polymatroidal' network flows. *Math. Oper. Res.* 7, 334–347.

Lawler, E.L., and C.U. Martel (1989). Preemptive scheduling of two uniform machines to minimize the number of late jobs. *Oper. Res.* 37, 314–318.

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

Lenstra, J.K. (1977). Sequencing by enumerative methods, Mathematical Centre Tracts 69, Centre for Mathematics and Computer Science, Amsterdam.

Lenstra, J.K. (-). Unpublished.

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

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.

Möhring, R.H. (1983). Scheduling problems with a singular solution. *Ann. Discrete Math.* 16, 225–239.

Möhring, R.H. (1984). Minimizing costs of resource requirements in project networks subject to a fixed completion time. *Oper. Res.* 32, 89–120.

Möhring, R.H. (1989). Computationally tractable classes of ordered sets, in: I. Rival (ed.), *Algorithms and Order*, Kluwer Academic, Dordrecht, pp. 105–193.

Möhring, R.H., and F.J. Radermacher (1985a). Generalized results on the polynomiality of certain weighted sum scheduling problems. *Methods Oper. Res.* 49, 405–417.

Möhring, R.H., and F.J. Radermacher (1985b). An introduction to stochastic scheduling problems, in: K. Neumann and D. Pallaschke (eds.), *Contributions to Operations Research*, Lecture Notes in Economics and Mathematical Systems, Vol. 240, Springer, Berlin, pp. 72–130.

Möhring, R.H., F.J. Radermacher and G. Weiss (1984). Stochastic scheduling problems I: General strategies. *Z. Oper. Res.* 28, 193–260.

Möhring, R.H., F.J. Radermacher and G. Weiss (1985). Stochastic scheduling problems II: Set strategies. *Z. Oper. Res.* 29, 65–104.

Monma, C.L. (1979). The two-machine maximum flow-time problem with series-parallel precedence constraints: An algorithm and extensions. *Oper. Res.* 27, 792–798.

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

Monma, C.L. (1981). Sequencing with general precedence constraints. *Discrete Appl. Math.* 3, 137–150.

Monma, C.L. (1982). Linear-time algorithms for scheduling on parallel processors. *Oper. Res.* 30, 116–124.

Monma, C.L., and A.H.G. Rinnooy Kan (1983). A concise survey of efficiently solvable special cases of the permutation flow-shop problem. *RAIRO Rech. Opér.* 17, 105–119.

Monma, C.L., and J.B. Sidney (1979). Sequencing with series-parallel precedence constraints. *Math. Oper. Res.* 4, 215–224.

Monma, C.L., and J.B. Sidney (1987). Optimal sequencing via modular decomposition: Characterizations of sequencing functions. *Math. Oper. Res.* 12, 22–31.

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

Morrison, J.F. (1988). A note on LPT scheduling. *Oper. Res. Lett.* 7, 77–79.

Muller, J.H., and J. Spinrad (1989). Incremental modular decomposition. *J. Assoc. Comput. Mach.* 36, 1–19.

Muntz, R.R., and E.G. Coffman Jr (1969). Optimal preemptive scheduling on two-processor systems. *IEEE Trans. Comput.* C-18, 1014–1020.

Muntz, R.R., and E.G. Coffman Jr (1970). Preemptive scheduling of real time tasks on multiprocessor systems. *J. Assoc. Comput. Mach.* 17, 324–338.

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

Nakajima, K., J.Y.-T. Leung and S.L. Hakimi (1981). Optimal two processor scheduling of tree precedence constrained tasks with two execution times. *Performance Evaluation* 1, 320–330.

Nawaz, M., E.E. Enscore Jr and I. Ham (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega* 11, 91–95.

Németi, L. (1964). Das Reihenfolgeproblem in der Fertigungsprogrammierung und Linearplanung mit logischen Bedingungen. *Mathematica (Cluj)* 6, 87–99.

Nowicki, E., and C. Smutnicki (1987). On lower bounds on the minimum maximum lateness on one machine subject to release date. *Opsearch* 24, 106–110.

Nowicki, E., and S. Zdrzalka (1986). A note on minimizing maximum lateness in a one-machine sequencing problem with release dates. *European J. Oper. Res.* 23, 266–267.

O'hEigeartaigh, M., J.K. Lenstra and A.H.G. Rinnooy Kan (eds.) (1985). *Combinatorial Optimization: Annotated Bibliographies*, Wiley, Chichester.

Osman, I.H., and C.N. Potts (1989). Simulated annealing for permutation flow-shop scheduling. *Omega* 17, 551–557.

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.

Rachamadugu, R.M.V. (1987). A note on the weighted tardiness problem. *Oper. Res.* 35, 450–452.
Radermacher, F.J. (1985/6). Scheduling of project networks. *Ann. Oper. Res.* 4, 227–252.
Raghavachari, M. (1988). Scheduling problems with non-regular penalty functions: A review. *Opsearch* 25, 144–164.
Rayward-Smith, V.J. (1987a). UET scheduling with unit interprocessor communication delays. *Discrete Appl. Math.* 18, 55–71.
Rayward-Smith, V.J. (1987b). The complexity of preemptive scheduling given interprocessor communication delays. *Inform. Process. Lett.* 25, 123–125.
Reddi, S.S., and C.V. Ramamoorthy (1972). On the flow-shop sequencing problem with no wait in process. *Oper. Res. Quart.* 23, 323–331.
Righter, R., (1988). Job scheduling to minimize expected weighted flowtime on uniform processors. *Syst. and Control Lett.* 10, 211–216.
Rinaldi, G., and A. Sassano (1977). On a job scheduling problem with different ready times: Some properties and a new algorithm to determine the optimal solution, Report R.77-24, Istituto di Automatica, Univ. di Roma.
Rinnooy Kan, A.H.G. (1976). *Machine Scheduling Problems: Classification, Complexity and Computations*, Nijhoff, The Hague.
Rinnooy Kan, A.H.G., B.J. Lageweg and J.K. Lenstra (1975). Minimizing total costs in one-machine scheduling. *Oper. Res.* 23, 908–927.
Röck, H. (1984a). The three-machine no-wait flow shop problem is NP-complete. *J. Assoc. Comput. Mach.* 31, 336–345.
Röck, H. (1984b). Some new results in flow shop scheduling. *Z. Oper. Res.* 28, 1–16.
Röck, H., and G. Schmidt (1983). Machine aggregation heuristics in shop scheduling. *Methods Oper. Res.* 45, 303–314.
Rosenfeld, P. (-). Unpublished.
Rothkopf, M.H. (1966). Scheduling independent tasks on parallel processors. *Management Sci.* 12, 437–447.
Roy, B., and B. Sussmann (1964). Les problèmes d'ordonnancement avec contraintes disjonctives, Note DS no. 9 bis, SEMA, Montrouge.
Sahni, S. (1976). Algorithms for scheduling independent tasks. *J. Assoc. Comput. Mach.* 23, 116–127.
Sahni, S., and Y. Cho (1979a). Complexity of scheduling jobs with no wait in process. *Math. Oper. Res.* 4, 448–457.
Sahni, S., and Y. Cho (1979b). Nearly on line scheduling of a uniform processor system with release times. *SIAM J. Comput.* 8, 275–285.
Sahni, S., and Y. Cho (1980). Scheduling independent tasks with due times on a uniform processor system. *J. Assoc. Comput. Mach.* 27, 550–563.
Sarin, S.C., S. Ahn and A.B. Bishop (1988). An improved branching scheme for the branch and bound procedure of scheduling n jobs on m machines to minimize total weighted flowtime. *Internat. J. Prod. Res.* 26, 1183–1191.
Schmidt, G. (1983). Preemptive scheduling on identical processors with time dependent availabilities, Bericht 83-4, Fachbereich 20 Informatik, Technische Universität Berlin.
Schrage, L. (1970). Solving resource-constrained network problems by implicit enumeration – nonpreemptive case. *Oper. Res.* 18, 263–278.
Schrage, L., and K.R. Baker (1978). Dynamic programming solution of sequencing problems with precedence constraints. *Oper. Res.* 26, 444–449.
Sethi, R. (1976a). Algorithms for minimal-length schedules, in: Coffman [1976], pp. 51–99.
Sethi, R. (1976b). Scheduling graphs on two processors. *SIAM J. Comput.* 5, 73–82.
Sethi, R. (1977). On the complexity of mean flow time scheduling, *Math. Oper. Res.* 2, 320–330.
Sevastyanov, S.V. (1975). On an asymptotic approach to certain problems of scheduling theory (in Russian). *Upravlyaemye Systemy* 14, 40–51.
Sevastyanov, S.V. (1980). Approximation algorithms for Johnson's problem and for the summation of vectors (in Russian), *Upravlyaemye Systemy* 20, 64–73.

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

processing times on parallel machines to minimize expected flowtime. *J. Appl. Probab.* 23, 841–847.

Weiss, G. (1982). Multiserver stochastic scheduling, in: Dempster, Lenstra & Rinnooy Kan [1982], pp. 157–179.

Weiss, G., and M.L. Pinedo (1980). Scheduling tasks with exponential service times on non-identical processors to minimize various cost functions. *J. Appl. Probab.* 17, 187–202.

Wie, S.-H., and M.L. Pinedo (1986). On minimizing the expected makespan and flow time in stochastic flow shops with blocking. *Math. Oper. Res.* 11, 336–342.

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

Yue, M. (1990). On the exact upper bound for the multifit processor scheduling algorithm. *Ann. Oper. Res.* 24, 233–259.

Zdrzalka, S., and J. Grabowski (1989). An algorithm for single machine sequencing with release dates to minimize maximum cost. *Discrete Appl. Math.* 23, 73–89.