Handbook of Combinatorial Optimization
D.-Z. Du and P.M. Pardalos (Eds.) pp. 21-169

A Review of Machine Scheduling: Complexity, Algorithms and Approximability

Bo Chen
Warwick Business School
University of Warwick
Coventry CV4 7AL, U.K.
E-mail: B.Chen@warwick.ac.uk

Chris N. Potts
Faculty of Mathematical Studies
University of Southampton
Southampton SO17 1BJ, U.K.
E-mail: cnp@maths.soton.ac.uk

Gerhard J. Woeginger
Institut für Mathematik
Graz University of Technology
Steyrergasse 30
A-8010 Graz, Austria
E-mail: gwoegi@opt.math.tu-graz.ac.at

Contents

1 Introduction

The scheduling of computer and manufacturing systems has been the subject of extensive research for over forty years. In addition to computers and manufacturing, scheduling theory can be applied to many areas including agriculture, hospitals and transport. The main focus is on the efficient allocation of one or more resources to activities over time. Adopting manufacturing terminology, a *job* consists of one or more activities, and a *machine* is a resource that can perform at most one activity at a time. We concentrate on *deterministic machine scheduling* for which it is assumed that all data that define a problem instance are known with certainty.

Much of the early work on scheduling was concerned with deriving rules that find optimal schedules for some simple models. Examples include problems of scheduling a single machine to minimize the maximum lateness of the jobs, for which an optimal solution is obtained by sequencing the jobs according to the earliest due date (EDD) rule of Jackson [281], and of scheduling a

single machine to minimize the sum of weighted completion times of the jobs, for which an optimal solution is obtained by sequencing the jobs according to the shortest weighted processing time (SWPT) rule of Smith [495]. These simple problems arise infrequently in practice. Nevertheless, the analysis of simple models provides valuable insights that enable more complex systems to be scheduled. Moreover, the performance of many production systems is often dictated by the quality of the schedules for a bottleneck machine. Again, simple models can be useful for scheduling such a machine.

A significant research topic in scheduling is the use of complexity theory to classify scheduling problems as polynomially solvable or $\mathcal{NP}$-hard. For details of the theory, we refer the reader to the books by Garey & Johnson [189] and Papadimitriou [413]. Many fundamental results in the area of scheduling are derived by Lenstra, Rinnooy Kan & Brucker [349]. The $\mathcal{NP}$-hardness of a problem suggests that there are instances for which the computation time required to find an optimal solution increases exponentially with problem size. If large computation times for such problems are unacceptable, then a *heuristic* (method) or an *approximation algorithm* is used to give an approximate solution. There is clearly a trade-off between the investment in computation time for obtaining a solution and the quality of that solution.

To obtain exact solutions of $\mathcal{NP}$-hard scheduling problems, an enumerative algorithm is usually applied. The main types of enumerative algorithms are *branch and bound* and *dynamic programming*, and both may benefit from dominance rules which help to restrict the search. Moreover, the ability of a branch and bound algorithm to generate an optimal solution at reasonable computational expense usually depends on the quality of the bounding scheme that is used to eliminate partial solutions. Dominance rules and bounding schemes are usually based on problem-specific features.

The performance of heuristics is often evaluated empirically. However, it is sometimes possible to carry out a theoretical analysis of heuristic performance. Following the pioneering work of Graham [225, 226] on list scheduling heuristics for parallel machines, research on obtaining performance guarantees through *worst-case analysis* has developed significantly. Generally, heuristics with stronger guarantees are preferred.

For problems where enumerative algorithms are unable to solve problems with more than a handful of jobs, and the solutions generated by simple heuristic methods may be far from the optimum, *local search* methods are extremely useful. These methods are described by Aarts & Lenstra [1], and their application to scheduling problems is reviewed by Anderson, Glass &

Potts [23]. For many scheduling problems, local search methods including simulated annealing, tabu search and genetic algorithms are very successful in generating high-quality solutions.

This chapter surveys research on deterministic machine scheduling. Our survey includes statements of complexity results, descriptions of enumerative algorithms together with an indication of the size of instances that they might reasonably be expected to solve, the main features of local search methods including an indication of their ability to generate near-optimal solutions, and details of performance guarantees for approximation algorithms. As a complement to the present chapter, we refer the reader to an excellent survey on machine scheduling by Lawler, Lenstra, Rinnooy Kan & Shmoys [335], with additional material in the books by Tanaev, Gordon & Shafransky [524] and by Tanaev, Sotskov & Strusevich [525].

The rest of this chapter is organized as follows. Section 2 provides a description of scheduling problems, and then presents a classical representation scheme that is based on the physical environment and the performance criterion for the problem. The various methodologies that are used in scheduling research are described in Section 3. Sections 4, 5, 6 and 7 contain our survey of results for classical problems involving a single machine, parallel machines with preemption, parallels machine without preemption, and multi-stage systems, respectively. In Section 8, we give results for a selection of non-classical models. Finally, some concluding remarks are contained in Section 9.

2 Scheduling Models

The machine scheduling problems that we consider can be described as follows. There are m machines that are used to process n jobs. A *schedule* specifies, for each machine i ($i = 1, \ldots, m$) and each job j ($j = 1, \ldots, n$), one or more time intervals throughout which processing is performed on j by i. A schedule is *feasible* if there is no overlapping of time intervals corresponding to the same job (so that a job cannot be processed by two machines at once), or of time intervals corresponding to the same machine (so that a machine cannot process two jobs at the same time), and also if it satisfies various requirements relating to the specific problem type. The problem type is specified by the machine environment, the job characteristics and an optimality criterion.

2.1 Machine Environment

Different configurations of machines are possible. An *operation* refers to a specified period of processing by some machine type. We assume that all machines become available to process jobs at time zero.

In a *single-stage* production system, each job requires one operation, whereas in multi-stage systems the jobs require operations at different stages. Single-stage systems involve either a single machine, or m machines operating in parallel. In the case of parallel machines, each machine has the same function. We consider three cases: *identical parallel machines* in which each processing time is independent of the machine performing the job; *uniform parallel machines* in which the machines operate at different speeds but are otherwise identical; and *unrelated parallel machines* in which the processing time of a job depends on the machine assignment.

There are three main types of *multi-stage* systems. All such systems that we consider comprise s stages, each having a different function. In a *flow shop* with s stages, the processing of each job goes through the stages $1, \ldots, s$ in that order. In an *open shop*, the processing of each job also goes once through each stage, but the routing (that specifies the sequence of stages through which a job must pass) can differ between jobs and forms part of the decision process. In a *job shop*, each job has a prescribed routing through the stages, and the routing may differ from job to job. There are also multiprocessor variants of multi-stage systems, where each stage comprises several (usually identical) parallel machines.

2.2 Job Characteristics

The processing requirements of each job j are given: for the case of a single machine and identical parallel machines, p_j is the processing time; for uniform parallel machines, the processing time on machine i may be expressed as p_j/s_i, where s_i is the speed of machine i; for the case of unrelated parallel machines, a flow shop and an open shop, p_{ij} is the processing time on machine/stage i; and for a job shop, p_{ij} denotes the processing time of the ith operation (which is not necessarily performed at stage i). We assume that all p_j and p_{ij} are non-negative integers. Given an instance, denote by $p_{\max}$ the maximum value of all p_j or all p_{ij}.

In addition to its processing requirements, a job is characterized by its availability for processing, any dependence on other jobs, and whether interruptions in the processing of its operations are allowed. The availability

of each job j may be restricted by its integer *release date* r_j that defines when it becomes available for processing, and/or by its integer *deadline* $\overline{d}_j$ that specifies the time by which it must be completed.

Job dependence arises when there are *precedence constraints* on the jobs. If job j has precedence over job k, then k cannot start its processing until j is completed. Precedence constraints are usually specified by a directed acyclic precedence graph G with vertices $1, \ldots, n$. There is a directed path from vertex j to vertex k if and only if job j has precedence over job k.

Some scheduling models allow *preemption*: the processing of any operation may be interrupted and resumed at a later time on the same or on a different machine.

In classical scheduling models, it is assumed that the scheduler has full information of the problem instance, such as total number of jobs to be scheduled, their release dates and their processing times, before the process of scheduling actually starts. Such models assume that scheduling is *off-line*. By contrast, if information about the problem instance is made available to the scheduler job by job during the course of scheduling, then the model requires scheduling to be *on-line*. In these models, we assume that, unless stated otherwise, the scheduler's decision to assign and schedule a job or operation is *irrevocable*. According to the way information on job characteristics is released to the scheduler, we classify on-line scheduling problems into the following three basic categories.

- In the model of *scheduling over list*, the scheduler is confronted with the jobs one-by-one as they appear in a list. The existence of a job is not known until all its predecessors in the list have already been scheduled.

- In the model of *scheduling over time*, all jobs arrive at their release dates. The jobs are scheduled with the passage of time and, at any point of time, the scheduler only has knowledge of those jobs that have already arrived.

In the above two on-line paradigms, we assume that once a job is known to the scheduler, its processing requirement is also known. Therefore, we call these two basic paradigms *clairvoyant*.

- The third basic on-line paradigm is the *non-clairvoyant scheduling* model. In this model, the processing requirement of a job is unknown until its processing is completed.

In scheduling over time, *restarts* are sometimes allowed in which the processing of a job is interrupted to allow another job to start, and at some later time the interrupted job is started again from scratch. In *nearly on-line* scheduling over time, the release date of next job is always known to the scheduler. This small amount of additional information makes certain on-line scheduling problems much easier to tackle. A good source of information on all kinds of on-line scheduling models is the survey article by Sgall [481].

2.3 Optimality Criteria

For each job j, an integer *due date* d_j and a positive integer *weight* w_j may be specified. Given a schedule σ, we can compute for job j: the *completion time* $C_j(\sigma)$; the *flow time* $F_j(\sigma) = C_j(\sigma) - r_j$; the *lateness* $L_j(\sigma) = C_j(\sigma) - d_j$; the *earliness* $E_j(\sigma) = \max\{d_j - C_j(\sigma), 0\}$; the *tardiness* $T_j(\sigma) = \max\{C_j(\sigma) - d_j, 0\}$; and the *unit penalty* $U_j(\sigma) = 1$ if $C_j(\sigma) > d_j$, and $U_j(\sigma) = 0$ otherwise. Moreover, if f_j is a *regular* objective function, i.e., a non-decreasing cost function, then the cost of job j is $f_j(\sigma) = f_j(C_j(\sigma))$. If there is no ambiguity about the schedule under consideration, we write C_j, F_j, L_j, E_j, T_j, U_j, and f_j, respectively.

Some commonly used optimality criteria involve the minimization of: the maximum completion time, or makespan, $C_{\max} = \max_j C_j$; the maximum lateness $L_{\max} = \max_j L_j$; the maximum cost $f_{\max} = \max_j f_j$; the maximum earliness $E_{\max} = \max_j E_j$; the total (weighted) completion time $\sum_j (w_j)C_j$; the total (weighted) flow time $\sum_j (w_j)F_j$; the total (weighted) earliness $\sum_j (w_j)E_j$; the total (weighted) tardiness $\sum_j (w_j)T_j$; the (weighted) number of late jobs $\sum_j (w_j)U_j$; or the total cost $\sum f_j$, where each maximization and each summation is taken over all jobs j. Also, some situations require more than one of these criteria to be considered.

2.4 Three-Field Representation

It is convenient to adopt the representation scheme of Graham, Lawler, Lenstra & Rinnooy Kan [228]. This is a three-field descriptor $\alpha|\beta|\gamma$ which indicates problem type: α represents the machine environment, β defines the job characteristics, and γ is the optimality criterion.

Let $\circ$ denote the empty symbol. The first field takes the form $\alpha = \alpha_1\alpha_2\alpha_3$, where α_1, α_2 and α_3 are interpreted as follows.

- $\alpha_1 \in \{\circ, P, Q, R, F, O, J\}$:

- $\alpha_1 = \circ$: a single machine;
- $\alpha_1 = P$: identical parallel machines;
- $\alpha_1 = Q$: uniform parallel machines;
- $\alpha_1 = R$: unrelated parallel machines;
- $\alpha_1 = O$: an open shop;
- $\alpha_1 = F$: a flow shop;
- $\alpha_1 = J$: a job shop.

- $\alpha_2 \in \{\circ, m, s\}$:

  - $\alpha_2 = \circ$: the number of machines/stages is arbitrary;
  - $\alpha_2 = m$: there is a fixed number m of machines;
  - $\alpha_2 = s$: there is a fixed number s of stages.

- $\alpha_3 \in \{\circ, (P\overline{m}), (P\overline{m}_1, \ldots, P\overline{m}_s), (P)\}$:

  - $\alpha_3 = \circ$: a single stage, or several stages each with a single machine;
  - $\alpha_3 = (P\overline{m})$: multi-stage with $\overline{m}$ identical parallel machines at each stage;
  - $\alpha_3 = (P\overline{m}_1, \ldots, P\overline{m}_s)$: multi-stage with $\overline{m}_k$ identical parallel machines at stage k;
  - $\alpha_3 = (P)$: multi-stage with an arbitrary number of identical parallel machines at each stage.

We note that for a single machine problem $\alpha_1 = \circ$, $\alpha_2 = 1$ and $\alpha_3 = \circ$, whereas $\alpha_1 \neq \circ$ and $\alpha_2 \neq 1$ for other problem types. Further, $\alpha_3 = \circ$ if $\alpha_1 \in \{\circ, P, Q, R\}$.

The second field $\beta \subseteq \{\beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6\}$ indicates job characteristics as follows.

- $\beta_1 \in \{\circ, \textit{on-line-list}, \textit{on-line}, \textit{on-line-list-nclv}, \textit{on-line-nclv}\}$:

  - $\beta_1 = \circ$: off-line scheduling;
  - $\beta_1 = \textit{on-line-list}$: on-line scheduling over list;
  - $\beta_1 = \textit{on-line}$: together with $\beta_2 = r_j$, it indicates on-line scheduling over time;
  - $\beta_1 = \textit{on-line-list-nclv}$: on-line non-clairvoyant scheduling over list;
  - $\beta_1 = \textit{on-line-nclv}$: together with $\beta_2 = r_j$, it indicates on-line non-clairvoyant scheduling over time.

- $\beta_2 \in \{\circ, r_j\}$:
  - $\beta_2 = \circ$: no release dates are specified;
  - $\beta_2 = r_j$: jobs have release dates.
- $\beta_3 \in \{\circ, \overline{d}_j\}$:
  - $\beta_3 = \circ$: no deadlines are specified;
  - $\beta_3 = \overline{d}_j$: jobs have deadlines.
- $\beta_4 \in \{\circ, pmtn\}$:
  - $\beta_4 = \circ$: no preemption is allowed;
  - $\beta_4 = pmtn$: operations of jobs may be preempted.
- $\beta_5 \in \{\circ, intree, outtree, tree, chain, prec\}$:
  - $\beta_5 = \circ$: no precedence constraints are specified;
  - $\beta_5 = chain$: precedence constraints on jobs are defined where each vertex has outdegree and indegree at most one;
  - $\beta_5 = intree$: precedence constraints on jobs are defined by a rooted tree which has indegree at most one for each vertex;
  - $\beta_5 = outtree$: precedence constraints on jobs are defined by a rooted tree which has outdegree at most one for each vertex;
  - $\beta_5 = tree$: precedence constraints on jobs are defined by a rooted intree or outtree;
  - $\beta_5 = prec$: jobs have arbitrary precedence constraints.
- $\beta_6 \in \{\circ, p_j = 1, p_{ij} = 1\}$:
  - $\beta_6 = \circ$: processing times are arbitrary;
  - $\beta_6 = p_j = 1$: all jobs in a single-stage system have unit processing times;
  - $\beta_6 = p_{ij} = 1$: all operations in a multi-stage system have unit processing times.

Lastly, the third field defines the optimality criterion, which involves the minimization of

$$\gamma \in \{C_{\max},\ L_{\max},\ E_{\max}\ , T_{\max},\ f_{\max},\ \sum(w_j)C_j,\ \sum(w_j)F_j\ \sum(w_j)E_j,\ \sum(w_j)T_j,\ \sum(w_j)U_j,\ \sum f_j\}.$$

Furthermore, as indicated in Section 2.3, it is sometimes appropriate to consider several of these criteria.

To illustrate the three-field descriptor, we present four examples.

$1|r_j, prec|\sum w_j C_j$ is the problem of scheduling jobs with release dates and precedence constraints on a single machine to minimize the total weighted completion time.

$R|pmtn|L_{\max}$ is the problem of preemptively scheduling jobs on an arbitrary number of unrelated parallel machines to minimize the maximum lateness.

$O3|p_{ij} = 1|\sum U_j$ is the problem of scheduling jobs in a three-machine open shop to minimize the number of late jobs, where the processing time of each operation is one unit.

$F2(P4, P3)||C_{\max}$ is the problem of scheduling jobs in a two-stage flow shop to minimize the makespan, where the two stages comprise four and three identical parallel machines, respectively.

3 Methodology

In this section, we outline the methods and techniques that are used to analyze and solve scheduling problems. A scheduling problem is a special type of combinatorial optimization problem. Thus, we can use the methodology that is used for combinatorial optimization. For example, the main tools for providing *negative* results (non-existence of fast algorithms for a specific problem) come from computational complexity theory. Also, the main tools for providing *positive* results are enumerative algorithms for finding exact solutions, local search methods for finding approximate solutions, and polynomial time approximation algorithms with guarantees on their worst-case performance.

The $\mathcal{NP}$-hardness of an optimization problem suggests that it is not always possible to find an optimal solution quickly. Therefore, instead of searching for an optimal solution with enormous computational effort, we may instead use a local search method or an approximation algorithm to generate approximate solutions that are close to the optimum with considerably less investment in computational resources.

3.1 Computational Complexity

Practical experience has shown that some scheduling problems are easier to solve than others. For example, computers of today can solve instances of

problem $1||\sum w_jC_j$ with several thousands of jobs within seconds, whereas it takes at least several hours to solve some even moderately sized instances of problem $J||C_{\max}$ with, for example, 30 jobs and 30 machines. *Computational complexity theory* provides a mathematical framework that is able to explain these 'observations from practice' and that yields a classification of problems into easy and hard ones. In this section, we briefly sketch some of the main points of this theory. For more information, the reader is referred to the books by Garey & Johnson [189] and Papadimitriou [413].

A *computational problem* can be viewed as a function f that maps every *input* x to an *output* $f(x)$. In complexity theory, we usually deal with so-called *decision* problems, where the output $f(x)$ can only take the two values of YES and NO. Since by means of binary search one can represent every optimization problem as a 'short' sequence of decision problems, this restriction is not really essential. We distinguish between two basic schemes of *encoding* the numbers in an input. One scheme is the *unary encoding*, where any integer k is encoded by k bits (e.g., $6 \doteq 111111$). Another scheme is the standard *binary encoding* that is used in every standard computer (e.g., $6 \doteq 110$). Any other encoding scheme uses roughly (i.e., up to a constant factor) the same bits as the binary encoding scheme. The *size* $|x|$ of an input x is the overall number of bits used for encoding x under some given encoding scheme.

An *algorithm* is a step-by-step procedure for solving a computational problem. For a given input x, it generates the correct output $f(x)$ after a finite number of steps. The *time complexity* of an algorithm expresses its worst-case time requirements, i.e., the total number of elementary operations, such as additions, multiplications and comparisons, for each possible problem instance as a function of the size of the instance. An algorithm is said to be *polynomial* or a polynomial (time) algorithm if its time complexity is bounded by a polynomial in input size.

In complexity theory, an algorithm is considered to be 'good' if it is polynomial; otherwise, it is considered to be 'bad'. Similarly, a computational problem is considered easy if it possesses a polynomial algorithm, in which case, it is said to be *polynomially solvable*. Problem $1||\sum w_jC_j$ that is mentioned in the first paragraph of this section is polynomially solvable, and thus easy. The class $\mathcal{P}$ is the set of all polynomially solvable problems.

When an optimization problem is formulated as a decision problem, one can usually certify those instances x for which $f(x)$=YES by a short *certificate*. For example, any YES-instance of "is there a feasible schedule for which all jobs are completed by their due dates" has as a short certificate a

corresponding schedule in which all jobs are on time. Given the certificate, a YES-answer can be verified in polynomial time. The class $\mathcal{NP}$ contains all problems that fulfill the following two conditions.

(i) For every YES-instance x there exists a certificate y such that $|y|$ is polynomially bounded in $|x|$.

(ii) There exists a polynomial algorithm that verifies whether a given y is a valid certificate for a given instance x.

Even though it is not known whether $\mathcal{P}$=$\mathcal{NP}$, there is a general belief that this is not the case. In fact, answering this question is one of the hardest and most important open problems in computer science.

The most difficult problems in $\mathcal{NP}$ are the *$\mathcal{NP}$-complete* problems. Unless $\mathcal{P}$=$\mathcal{NP}$, which is considered unlikely, an $\mathcal{NP}$-complete problem does not possess a polynomial algorithm. Intuitively speaking, a problem X in $\mathcal{NP}$ is $\mathcal{NP}$-complete if any other problem in $\mathcal{NP}$ can be solved in polynomial time by an algorithm that makes a polynomial number of calls to a subroutine that solves problem X. Note that this implies that, if an $\mathcal{NP}$-complete problem allowed a polynomial algorithm, then *all* problems in $\mathcal{NP}$ would allow polynomial algorithms. Researchers on complexity theory have identified a huge body of $\mathcal{NP}$-complete problems. The problem $J||C_{\max}$ that is mentioned in the first paragraph of this section is $\mathcal{NP}$-complete, and thus is difficult.

Note that the concepts of polynomial solvability and $\mathcal{NP}$-completeness crucially depend on the encoding scheme used. If one changes the encoding scheme from binary to unary, the problem may become easier, as the input becomes longer and hence the restrictions on the running time of a polynomial algorithm are less stringent. A problem that is $\mathcal{NP}$-complete under the unary encoding scheme is called *strongly* $\mathcal{NP}$-complete. An optimization problem is (strongly) *$\mathcal{NP}$-hard* if its decision version is (strongly) $\mathcal{NP}$-complete. When we state that a problem is $\mathcal{NP}$-hard, it should be assumed that the problem is not known to be strongly $\mathcal{NP}$-hard. A problem that can be solved in polynomial time under the unary encoding scheme is said to be *pseudo-polynomially* solvable. The corresponding algorithm is called a *pseudo-polynomial (time)* algorithm.

3.2 Enumerative Algorithms

3.2.1 Branch and Bound

The process of solving a problem using a branch and bound algorithm can be conveniently represented by a *search tree*. Each node of the search tree corresponds to a subset of feasible solutions to a problem. A *branching rule* specifies how the feasible solutions at a node are partitioned into subsets, each corresponding to a descendant node of the search tree. For many single machine scheduling problems, a solution is represented by a sequence of jobs. For such problems, commonly used branching rules involve forward sequencing and backward sequencing. In a *forward sequencing branching rule*, each node corresponds to an initial partial sequence in which the jobs in the first positions are fixed. The first branching creates nodes corresponding to every job that can appear in the first position in the sequence, the second branching creates nodes that fix the job in the second position of the sequence, and so on. A *backward sequencing branching rule* is similar except that the sequence is built from the end: each node corresponds to a final partial sequence in which the jobs in the last positions are fixed.

The scheduling problems that we consider require an objective function to be minimized. A *lower bounding scheme* associates a lower bound with each node of the search tree. The idea is to eliminate any node for which the lower bound is greater than or equal to the value of the best known feasible solution, since any further exploration of such a node cannot lead to an improved solution. Lower bounds are usually obtained by solving a relaxed problem. For example, lower bounds for non-preemptive scheduling problems can be obtained by solving the corresponding preemptive version of the problem. One widely used general-purpose technique for obtaining lower bounds is *Lagrangean relaxation*. Having obtained a suitable formulation of the problem, Lagrangean relaxation removes some of the constraints and incorporates them into the objective function using Lagrange multipliers. An iterative method known as *subgradient optimization* can be used to find values of the multipliers that yield the best bounds, although a constructive method for finding multiplier values that is based on problem-specific features sometimes provides good quality bounds at low computational effort. Solving the *linear programming relaxation* of an integer programming formulation offers an alternative method of computing lower bounds. The tightness of the resulting lower bounds is often improved through a polyhedral approach that provides additional *valid inequalities*.

Another noteworthy feature of branch and bound algorithms is the use of *dominance rules* that attempt to eliminate nodes prior to the computation of lower bounds. Also, a heuristic method is often used at certain nodes (especially the root node) to generate feasible solutions, the values of which provide *upper bounds*. A *search strategy* defines the order in which the nodes of the tree are selected for branching. One commonly used strategy is to branch from a node with the smallest lower bound. However, it is easier to implement a depth-first search plus backtracking strategy that branches from a node in the most recently created subset.

3.2.2 Dynamic Programming

A dynamic programming algorithm has the property that each partial solution is represented by a *state* to which a value (or cost) is assigned. When two or more partial solutions achieve the same state, one with the lowest value is retained, while the others are eliminated. It is necessary to define the states in such a way that this type of elimination is valid. For example, consider the single machine sequencing problem $1||\sum f_j$ in which partial solutions are initial partial sequences. A state can be represented by the jobs that are sequenced and the time that the machine becomes available to process the unsequenced jobs.

A decision transforms a partial solution from one state to another. All possible decisions are considered. The elimination of partial solutions is usually implemented by means of a *recursion*, which compares the value of all possible ways of entering a state and chooses the one which has minimum value. Applying the recursion determines the optimal solution value, and backtracking is then used to find the corresponding decisions that define the solution.

3.3 Local Search

3.3.1 Neighborhood Search

In our discussion of local search heuristics, we sometimes mention some constructive heuristics which are useful for obtaining a starting solution in neighborhood search. Such a heuristic typically generates a single solution, and its running time is typically a low-order polynomial.

In neighborhood search, a current solution is transformed into a new solution according to some neighborhood structure. An acceptance rule

decides whether the move from the current solution to the transformed solution should be accepted, although the decision is sometimes delayed until the complete neighborhood (or a subset of it) is explored. If a move is accepted, then the transformed solution replaces the previous solution and becomes the current solution; otherwise, the move is rejected and the current solution is retained. This process is repeated until some termination criterion is satisfied. The acceptance rule is usually based on the objective function values of the current solution and its neighbor. Examples of neighborhoods in single machine sequencing are the *transpose neighborhood* in which two jobs occupying adjacent positions in the sequence are interchanged, the *swap neighborhood* in which any two jobs in the sequence are interchanged, and the *insert neighborhood* in which one job is removed from its current position in the sequence and inserted elsewhere.

The simplest type of neighborhood search method is *descent*, which is sometimes known as iterative local improvement. In this method, only moves that result in an improvement in the objective function value are accepted. Under a *first improve* search, the first move that improves the objective function value is accepted. On the other hand, *best improve* selects a move that yields the best objective function value among all neighbors. When no further improvement can be achieved, a descent method terminates with a solution that is a local optimum. The local optimum is not necessarily the true global optimum. A widely used remedy for this drawback is to use *multi-start descent* in which multiple runs of descent from different starting solutions are performed, and the best overall solution is selected. Simulated annealing and tabu search are other neighborhood search methods which allow the search to escape from local optima.

In *simulated annealing*, a probabilistic acceptance rule is used. More precisely, any move that results in an improvement in the objective function value, or leaves the value unchanged, is accepted. On the other hand, a move that increases the objective function value by δ is accepted with probability $\exp(-\delta/t)$, where t is a parameter known as the *temperature*. The value of t changes during the course of the search; typically t starts at a relatively high value and then gradually decreases. Simulated annealing has a variant called *threshold accepting* in which a move is accepted if and only if $\delta \leq T$, where T is a parameter that plays the same role as temperature in simulated annealing.

In contrast to simulated annealing, *tabu search* is a deterministic neighborhood search method. A move is made to the best solution in the neighborhood of the current solution (even if this increases the objective function

value), subject to certain forbidden moves that are defined by a *tabu list*. The tabu list stores attributes of the previous few moves, and moves that reverse these attributes are not allowed (they are tabu). The tabu list is primarily aimed at preventing the method from cycling, although it also serves the function of driving the search into unexplored regions of the solution space. To prevent the occasional loss of a high quality solution, a tabu move is sometimes allowed if it satisfies an *aspiration* condition.

3.3.2 Genetic Algorithms

In contrast to the single current solution in neighborhood search, a genetic algorithm works with a population of solutions, where each solution is represented as a string. A *mating pool* of solutions is formed from the current population, where some solutions may not be selected, and others may appear several times. Solutions are chosen for the mating pool according to fitness values, where better quality solutions are assigned a higher fitness. A new population is formed by applying two genetic operators. First, pairs of solutions in the mating pool undergo *crossover*: sections of the strings for the two solutions are interchanged, thereby giving two new solutions. The aim is that one of the new solutions will inherit the desirable features of both parents. Second, *mutation* changes some elements of the string with the aim of maintaining diversity in the new population. The process is repeated until some termination condition is satisfied.

3.4 Approximation Algorithms

Approximation algorithms are used to generate approximate solutions with modest computational effort. For a scheduling problem with the objective of minimizing a cost function $F(\cdot) \geq 0$, an algorithm H is called a *ρ-approximation algorithm* ($\rho \geq 1$) if $F(H(I)) \leq \rho F(S^*(I))$ for all problem instances I, where $H(I)$ and $S^*(I)$ denote the schedule found by algorithm H and an optimal schedule, respectively. We refer to ρ as a *ratio guarantee* for algorithm H since it provides a bound on the performance under any circumstances. If such ρ is the smallest possible value, then it is called the *worst-case (performance) ratio* of algorithm H. An approximation algorithm is often called a *heuristic*. A family of algorithms $\{H_\varepsilon\}$ for a problem is called a *polynomial (time) approximation scheme* (PTAS, for short) if, for every $\varepsilon > 0$, H_ε is a $(1+\varepsilon)$-approximation algorithm whose running time is polynomial in the input size. Furthermore, if the running time of every

H_ε is bounded by a polynomial in the input size and $1/\varepsilon$, then the family is called a *fully polynomial (time) approximation scheme* (FPTAS). If a scheduling problem is strongly $\mathcal{NP}$-hard, then it allows neither an FPTAS nor a pseudo-polynomial algorithm unless $\mathcal{P}=\mathcal{NP}$ (see, for example, Garey & Johnson [189]).

There are other methods than the aforementioned *worst-case* analysis for evaluating the performance of an approximation algorithm, which include *empirical* analysis and *probabilistic* analysis. Empirical analysis of an algorithm basically involves running the algorithm on a large number of test problem instances, preferably with known or estimated optimal solutions. The instances should have a variety of different characteristics including those that are likely to occur in practice. The performance of the algorithm is then evaluated statistically by comparing the solution values that are generated with the optimal or estimated values. Since such an evaluation depends on the characteristics of the generated sample instances, this approach is mainly used for comparing the empirical performance of alternative algorithms. Probabilistic analysis of an algorithm can be regarded as the analytical counterpart of empirical analysis. The main goal of the analysis is to provide a probabilistic characterization for the average-case performance of the heuristics, under some assumptions on the probability distribution of the problem parameters.

In this chapter, we are mainly concerned with worst-case analysis of algorithms. We distinguish between on-line and off-line performance measures by using different terminology. If in the above definition for the worst-case performance, H is an on-line ρ-approximation algorithm, then we say that it is a ρ-*competitive* algorithm or H is ρ-*competitive*. The worst case ratio of an on-line algorithm is referred to as its *competitive ratio*. Note that the competitiveness of an on-line algorithm is evaluated with respect to an off-line optimal algorithm, and hence it indicates the loss associated with not having complete information of the problem instances. In our review, both ρ-approximation algorithms and on-line algorithms are assumed to be polynomial, unless otherwise stated. When giving statements about the running time of a PTAS or an FPTAS, we refer to the time requirement to obtain a ratio guarantee of $1+\varepsilon$, for any $\varepsilon > 0$. If not explicitly stated otherwise, the running times of approximation algorithms are low-order polynomials.

4 Single Machine Problems

4.1 Maximum Lateness

4.1.1 Complexity

For problem $1||L_{\max}$, Jackson [281] shows that an optimal solution is obtained in $O(n \log n)$ time by sequencing jobs in non-decreasing order of their due dates. This method is known as the earliest due date or EDD rule. To justify the use of the EDD rule for this problem, we use the following job reinsertion argument. Consider any optimal sequence σ. Suppose that some job k is sequenced before another job j, where $d_k > d_j$. Consider the transformed sequence σ' in which job k is removed from its original position and inserted immediately after job j. All jobs sequenced before k and after j in σ have the same completion time in both sequences, and job j together with all jobs sequenced between k and j in σ are completed p_k units earlier in σ'. Moreover, $L_k(\sigma') = C_k(\sigma') - d_k = C_j(\sigma) - d_k < C_j(\sigma) - d_j = L_j(\sigma)$. Thus, $L_{\max}(\sigma') \leq L_{\max}(\sigma)$, which implies that σ' is also an optimal sequence. Repetition of this job reinsertion argument yields an optimal sequence in which jobs appear in EDD order. Note that there may be optimal sequences in which jobs are not sequenced in non-decreasing order of their due dates.

In the presence of deadlines, the problem remains polynomially solvable. For this problem $1|\bar{d}_j|L_{\max}$, a bisection search over all possible values of $L_{\max}$ is used. If L is a trial value of $L_{\max}$, then each job is assigned a deadline $\min\{\bar{d}_j, d_j + L\}$. To test whether this value of L can be achieved, the jobs are sequenced in EDD order with respect to the assigned deadlines.

Polynomial solvability is also achievable when there are precedence constraints, since Lawler [321] proposes an $O(n^2)$ algorithm for the more general problem $1|prec|f_{\max}$. The key observation, which is proved using a job reinsertion argument of the type described above, is that some job j, which has no successors and is chosen so that $f_j(\sum_{k=1}^{n} p_k)$ is as small is possible, appears in the last position of some optimal sequence. Repeated application of this result provides an optimal sequence. Baker, Lawler, Lenstra & Rinnooy Kan [32] and Gordon & Tanaev [218] independently generalize Lawler's approach to problem $1|r_j, pmtn, prec|f_{\max}$. Both algorithms also require $O(n^2)$ time.

Release dates add a substantial degree of difficulty, since problem $1|r_j|L_{\max}$ is shown by Lenstra, Rinnooy Kan & Brucker [349] to be strongly $\mathcal{NP}$-hard. However, an optimal solution for problem $1|r_j, pmtn|L_{\max}$ is constructed in $O(n \log n)$ time using a generalized EDD algorithm of Horn [272].

Starting at the smallest release date, this algorithm always schedules an available job with the smallest due date. A preemption occurs at a release date if the newly released job has a smaller due date than that of the job currently being processed. The resulting schedule has at most $n-1$ preemptions. Also, problem $1|r_j, p_j = 1|L_{\max}$ is solvable in $O(n)$ by an algorithm of Frederickson [175].

Simons [493] develops an $O(n^3 \log n)$ algorithm for the feasibility problem $1|r_j, \bar{d}_j, p_j = p|\cdot$, where p is an arbitrary positive integer. Using a bisection search over all possible values of $L_{\max}$, she uses this feasibility algorithm to develop a polynomial time algorithm for problem $1|r_j, p_j = p, \bar{d}_j|L_{\max}$. Using an improved implementation, Garey, Johnson, Simons & Tarjan [191] reduce the time complexity of the feasibility algorithm to $O(n \log n)$.

An observation of Lageweg, Lenstra & Rinnooy Kan [316] sometimes allows precedence constraints to be handled by simply adjusting release and due dates and then applying the appropriate algorithm for the problem without precedence constraints. If job j is required to precede job k, then the release date of job k can be reset to $\max\{r_k, r_j+p_j\}$, and the due date of job j can be reset to $\min\{d_j, d_k-p_k\}$. Such a resetting requires $O(h)$ time, where h is the number of precedences. Thus, an $O(n \log n + h)$ algorithm is obtained for problems $1|r_j, pmtn, prec|L_{\max}$. Further, Monma [381] uses due date resetting to obtain an $O(n+h)$ algorithm for problem $1|prec, p_j = 1|L_{\max}$.

Problem $1|r_j|L_{\max}$ is of special importance since it is a subproblem of some multi-stage problems which have makespan or maximum lateness as the optimality criterion. An important structural feature, as observed by Lageweg, Lenstra & Rinnooy Kan [316], is that of *reversibility*. If the due date of each job j is replaced by a *delivery time* q_j, where $q_j = -d_j$, and it is required to minimize $\max_j\{C_j + q_j\}$, then an equivalent problem results. Release dates and delivery times play a symmetric role, and can be interchanged for each job to yield an equivalent problem.

4.1.2 Enumerative Algorithms

A key feature of branch and bound, and approximation algorithms for problem $1|r_j|L_{\max}$ is the $O(n \log n)$ heuristic of Schrage [466] which schedules jobs using the generalized EDD rule. Specifically, this heuristic selects, from the available jobs, one with the smallest due date to schedule next. Suppose that the jobs are reindexed so that the heuristic yields the sequence $(1, \ldots, n)$. Then the maximum lateness is given by

$$L_{\max} = r_i + \sum_{j=i}^{l} p_j - d_l,$$

where l is the *critical end job*, $(i, \ldots, l)$ form a *block* of jobs, and $i \leq l$. The heuristic ensures that $r_i \leq r_j$ for $j = i, \ldots, n$. If job l has the largest due date among jobs $i, \ldots, l$, then the heuristic provides an optimal solution. Otherwise, we can choose an *interference job* k, which is the last job before l in the schedule such that $d_k > d_l$. None of the jobs $k+1, \ldots, l$ is available when job k starts processing. Also,

$$\mathrm{LB}(S) = \min_{j \in A} r_j + \sum_{j \in S} p_j - \max_{j \in B} d_j$$

where $A \subseteq S$ and $B \subseteq S$ are the sets of jobs that are candidates to be sequenced before and after the other jobs in S, respectively, is a lower bound for any subset S of jobs. It is easily verified by considering $S = \{k, \ldots, l\}$ that if some jobs in $\{k+1, \ldots, l\}$ are sequenced before and after job k, then the $\mathrm{LB}(S)$ exceeds the maximum lateness of the heuristic. Thus, job k is either sequenced before or after all jobs in $\{k+1, \ldots, l\}$ in each optimal schedule.

For problem $1|r_j|L_{\max}$, various branch and bound algorithms are available which extend to problem $1|r_j, prec|L_{\max}$ by using the release date and due date adjustments that are described in Section 4.1.1. The algorithm of Baker & Su [34] employs a forward sequencing branching rule and uses a lower bound that is obtained by allowing preemption. More sophisticated approaches of McMahon & Florian [377], Lageweg, Lenstra & Rinnooy Kan [316], Carlier [79] and Larson, Dessouky & Devor [320] use special branching rules that are obtained from the generalized EDD heuristic. Carlier's algorithm appears to be the most efficient. It uses a binary branching rule that fixes the interference job k to be sequenced either before or after all jobs in $\{k+1, \ldots, l\}$ by resetting the due date and release date of job k, as described above, and it employs the lower bounds $\mathrm{LB}(\{k+1, \ldots, l\})$ and $\mathrm{LB}(\{k, \ldots, l\})$ which require less computation time than the preemptive lower bound. Computational results for instances with up to 10 000 jobs are reported. Carlier [79], Nowicki & Zdrzałka [403], and Nowicki & Smutnicki [398] prove that the preemptive lower bound dominates many of its various competitors. Grabowski, Nowicki & Zdrzałka [223] make the observation that moving a job from one position to another within a block does not result in an improved schedule if these positions are not the first or last in the block. They use this observation to develop a branching rule,

although the resulting branch and bound algorithm is less efficient than Carlier's algorithm. Zdrzałka & Grabowski [574] develop a branch and bound algorithm for problem $1|r_j, prec|f_{\max}$ using some of the ideas in algorithms for problem $1|r_j, prec|L_{\max}$.

Some work on the job shop problem $J||C_{\max}$ provides results relevant to problem $1|r_j|L_{\max}$. For example, Balas, Lenstra & Vazacopoulos [37] generalize Carlier's algorithm to a version of problem $1|r_j, prec|L_{\max}$ with delayed precedence constraints, where each precedence between a pair of jobs imposes a delay between the completion time of the first and the start of the second. Branch and bound algorithms that rely heavily on analysis of single machine subproblems of the form $1|r_j|L_{\max}$ are discussed in Section 7.3.2.

4.1.3 Approximation

Worst-case analysis of approximation algorithms for problems $1|r_j|L_{\max}$ and $1|r_j, prec|L_{\max}$ assumes that all due dates are non-positive. Equivalently, the delivery time model is assumed with non-negative delivery times. Without this restriction, results are likely to be elusive, since the problem of determining whether $L_{\max} \leq 0$ is $\mathcal{NP}$-complete. Kise, Ibaraki & Mine [299] analyze six approximation algorithms, including the generalized EDD heuristic of Schrage [466], and show that each has a worst-case ratio of 2. Potts [424] proposes an $O(n^2 \log n)$ algorithm in which the generalized EDD heuristic is applied up to n times, and the best solution is selected. In each application, a constraint is added that the interference job is scheduled after the critical end job. This algorithm has a worst-case ratio of 3/2. Nowicki & Smutnicki [400] show that the computational effort to achieve a worst-case ratio of 3/2 can be reduced to $O(n \log n)$. Their algorithm applies the generalized EDD heuristic and then constructs one other sequence, finally selecting the better of the two schedules. An improvement to the worst-case ratio is obtained by Hall & Shmoys [235] in an algorithm that relies on applying the generalized EDD heuristic to both the original and reverse problems. Their $O(n^2 \log n)$ algorithm has a worst-case ratio of 4/3. Hall & Shmoys also propose two PTAS's for problem $1|r_j|L_{\max}$, the running times of which are $O(2^{4\bar{\varepsilon}}(n\bar{\varepsilon})^{4\bar{\varepsilon}+3})$ and $O(n \log n + n(4\bar{\varepsilon})^{8\bar{\varepsilon}^2+8\bar{\varepsilon}+2})$, where $\bar{\varepsilon} = 1/\varepsilon$.

The on-line problem $1|on\text{-}line, r_j|L_{\max}$ is analyzed by Hoogeveen & Vestjens [270, 547]. They show that no on-line algorithm can be better than $(1+\sqrt{5})/2$-competitive. Note that if we use the EDD rule to schedule a job *whenever* the machine becomes idle, then the maximum lateness of the resulting schedule can be as large as twice the optimal value. Taking this

into account, Hoogeveen & Vestjens introduce a clever waiting strategy in applying the EDD rule. They prove that their modified EDD algorithm is $(1+\sqrt{5})/2$-competitive, and therefore is the best possible.

4.2 Total Weighted Completion Time

4.2.1 Complexity

For problem $1||\sum w_jC_j$, Smith [495] shows that an optimal solution is obtained in $O(n\log n)$ time by sequencing jobs in non-decreasing order of p_j/w_j. This method is known as the shortest weighted processing time or SWPT rule; it becomes the SPT rule in the case of unit weights. To show that any optimal sequence is obtained from the SWPT rule, we use an adjacent job interchange argument. Consider any sequence σ in which the jobs are not sequenced in SWPT order. Then σ contains a job k that is sequenced immediately before another job j, where $p_k/w_k > p_j/w_j$. Consider the transformed sequence σ' in which jobs j and k are transposed. All jobs apart from j and k have the same completion time in both sequences. Therefore,

$$\begin{aligned}\sum w_hC_h(\sigma') - \sum w_hC_h(\sigma) & \\ &= w_k(C_k(\sigma') - C_k(\sigma)) - w_j(C_j(\sigma) - C_j(\sigma') \\ &= w_kp_j - w_jp_k \\ &= w_jw_k(p_j/w_j - p_k/w_k) < 0,\end{aligned}$$

which shows that σ cannot be an optimal sequence.

In the presence of precedence constraints, the polynomial solvability of problem $1|prec|\sum w_jC_j$ depends on the structure of the precedence constraints. Horn [272], Adolphson & Hu [10] and Sidney [491] each develop $O(n\log n)$ algorithms for precedence constraints in the form of a tree, while Lawler [324] considers *series-parallel* precedence constraints. A series-parallel graph is one that is built by performing a sequence of series and parallel operations to subgraphs, where the initial subgraph comprises n single vertices that correspond to jobs. A series operation adds a precedence arc from each vertex of the first subgraph to each vertex of the second, whereas a parallel operation combines two subgraphs without adding precedence arcs. The composition process can be represented by a tree, which can be found in $O(n^2)$ time from the precedence graph using an algorithm of Muller & Spinrad [388] that improves upon a previous $O(n^3)$ algorithm

of Buer & Möhring [75]. After obtaining this tree, Lawler's algorithm requires $O(n \log n)$ time. For general precedence constraints, Lawler [324] and Lenstra & Rinnooy Kan [346] show that problems $1|prec|\sum C_j$ and $1|prec, p_j = 1|\sum w_j C_j$ are strongly $\mathcal{NP}$-hard.

Release dates also add to the difficulty, since problem $1|r_j|\sum C_j$ is shown by Lenstra, Rinnooy Kan & Brucker [349] to be strongly $\mathcal{NP}$-hard. However, problem $1|r_j, pmtn|\sum C_j$ is solvable by the shortest remaining processing time (SRPT) rule of Schrage [464]: each time that a job is completed, or at the next release date, the job to be processed next has the smallest remaining processing time among the available jobs. This algorithm does not extend to minimizing total weighted completion time, since Labetoulle, Lawler, Lenstra & Rinnooy Kan [315] show that problem $1|r_j, pmtn|\sum w_j C_j$ is strongly $\mathcal{NP}$-hard.

In the case of deadlines, problem $1|\overline{d}_j|\sum C_j$ is solvable in $O(n \log n)$ time by a backward scheduling algorithm of Smith [495] that repeatedly selects the job with the largest processing time among feasible candidates to be sequenced in the last unfilled position. However, Lenstra, Rinnooy Kan & Brucker prove that $1|\overline{d}_j|\sum w_j C_j$ is strongly $\mathcal{NP}$-hard. Du & Leung [149] show $\mathcal{NP}$-hardness for problem $1|r_j, pmtn, \overline{d}_j|\sum C_j$.

4.2.2 Enumerative Algorithms

Potts [422, 425] develops a 'linear-ordering' formulation for $1|prec|\sum w_j C_j$ which uses zero-one variables x_{jk} to indicate whether of not job j is sequenced before job k. In [425], he performs a Lagrangean relaxation of the constraints $x_{jk} + x_{kj} = 1$ and $x_{jk} + x_{kl} + x_{lj} \geq 1$, selecting each multiplier as large as possible subject to retaining the non-negativity of the objective function coefficients. The resulting branch and bound algorithm is effective in solving problems with up to 100 jobs. Queyranne & Wang [441] develop an alternative 'completion time' formulation that uses variables C_j and valid inequalities of the form

$$\sum_{j \in S} p_j C_j \geq \frac{1}{2}\Big(\sum_{j \in S} p_j^2 + (\sum_{j \in S} p_j)^2\Big),$$

for any $S \subseteq \{1, \ldots, n\}$, which represent the machine capacity constraint. They also derive valid inequalities that use the precedence constraints. Together, these inequalities give a complete polyhedral characterization when the precedence constraints are series-parallel. Queyranne & Wang [442]

evaluate the quality of the lower bounds provided by this polyhedral approach. Van de Velde [538] proposes lower bounds that are derived through a Lagrangean relaxation of the precedence constraints in a completion time formulation formulation. He develops a dual ascent method for finding the multipliers. Also, these multipliers are used in a heuristic that is based on decomposition. Hoogeveen & Van de Velde [266] improve on the Lagrangean lower bound of Velde by a new technique that introduces slack variables in Lagrangean problem. Although the resulting bound can never be tighter than the polyhedral bound of Queyranne & Wang, its computational requirements are significantly less.

Branch and bound algorithms for problems $1|r_j|\sum C_j$ and $1|r_j|\sum w_jC_j$ rely extensively on the dominance rules derived by Dessouky & Deogun [138], Bianco & Ricciardelli [51], Deogun [137], Hariri & Potts [240], Belouadah, Posner & Potts [47] and Chu [109]. For problem $1|r_j|\sum C_j$, Ahmadi & Bagchi [11] show that the lower bound obtained by solving the preemptive problem $1|r_j, pmtn|\sum C_j$ using the SRPT rule dominates all of the other known bounds. Chu uses this lower bound in a branch and bound algorithm that is effective in solving problems with up to 100 jobs.

For problem $1|r_j|\sum w_jC_j$, allowing preemption does not lead to a useful lower bounding scheme since the corresponding preemptive problem is $\mathcal{NP}$-hard (see Section 4.2.1. Hariri & Potts [240] derive a lower bound by performing a Lagrangean relaxation of the constraints $C_j \geq r_j + p_j$. They use a 'multiplier adjustment' method to construct values of the multipliers. In this method, multiplier values are chosen so that the solution of the Lagrangean problem corresponds to a schedule that is previously obtained by a heuristic method. An improved lower bounding scheme is developed by Belouadah, Posner & Potts [47], which is based on the principle of job splitting. If job j is split into two pieces j_1 and j_2, then the pieces are regarded as jobs with processing times and weights chosen so that $p_{j_1} + p_{j_2} = p_j$ and $w_{j_1} + w_{j_2} = w_j$. If the two pieces are scheduled contiguously, then this splitting reduces the total weighted completion time by the 'breaking cost' $w_{j_1}p_{j_2}$. A forward procedure constructs a schedule and splits jobs so that, at any time, the machine processes an available job or piece j for which p_j/w_j is as small as possible. The lower bound is the sum of the total weighted completion time for this schedule and the breaking cost. Computational results show that the branch and bound algorithm is effective in solving instances with up to 50 jobs. Dyer & Wolsey [154], Sousa & Wolsey [499] and Van den Akker [533] study integer programming formulations and derive valid inequalities.

For problem $1|\overline{d}_j|\sum w_jC_j$, branch and bound algorithms are similar in spirit to those for $1|r_j|\sum w_jC_j$. In particular, Potts & Van Wassenhove [431] develop a lower bound based on the Lagrangean relaxation of constraints $C_j \leq \overline{d}_j$, where multipliers are computed using a multiplier adjustment method, and Posner [420] proposes an improved algorithm which uses a lower bounding procedure that is based on job splitting.

4.2.3 Approximation

There are various studies on approximation algorithms for both the off-line and on-line versions of problems $1|r_j|\sum C_j$ and $1|r_j|\sum w_jC_j$. We first discuss off-line results.

For the unweighted problem $1|r_j|\sum C_j$, Phillips, Stein & Wein [417] propose a 2-approximation algorithm in which the jobs are sequenced in the order σ that they complete in the solution of the corresponding preemptive problem $1|r_j, pmtn|\sum C_j$. If the jobs are renumbered so that $\sigma = (1, \ldots, n)$, then the completion time of each job k in the schedule defined by σ is

$$C_k(\sigma) = r_i + \sum_{j=i}^{k} p_j,$$

for some job i, where $i \leq k$. Since $r_i \leq C_k(pmtn)$ and $\sum_{j=i}^{k} p_j \leq C_k(pmtn)$, where $C_k(pmtn)$ is the completion time of job k in the preemptive schedule, we obtain $\sum C_j(\sigma) \leq 2\sum C_j(pmtn)$. The observation that $\sum C_j(pmtn)$ is a lower bound on the optimal value of the total completion time for the non-preemptive problem provides the desired ratio guarantee of 2.

Chekuri, Motwani, Natarajan & Stein [89] give an improved polynomial time approximation algorithm for problem $1|r_j|\sum C_j$ that has a ratio guarantee of $e/(e-1) \approx 1.58$ (where e denotes the base of the natural logarithm). Their algorithm is based on a clever rounding technique of the preemptive relaxation of this problem. Kellerer, Tautenhahn & Woeginger [297] provide an approximation algorithm for the corresponding problem $1|r_j|\sum F_j$ of minimizing total flow time, where the flow time of job j is defined as $F_j = C_j - r_j$, with a ratio guarantee of $O(\sqrt{n})$. They also prove that, unless $\mathcal{P}$=$\mathcal{NP}$, up to a constant factor no better guarantee can be achieved.

Hall, Schulz, Shmoys & Wein [233] propose a general methodology for developing approximation algorithms based on sequencing jobs according to their completion times in the solution of a linear program. This technique yields a 3-approximation algorithm for problem $1|r_j, prec|\sum w_jC_j$.

Using a similar approach, Goemans [206] develops a 2-approximation algorithm for problem $1|r_j|\sum w_jC_j$ that runs in $O(n^2)$ time. He simultaneously considers two equivalent linear programming relaxations for the problem: one involves completion time variables and the other one involves preemptive time-indexed variables. While the analysis is based on the first relaxation, the approximation algorithm is essentially based on the second. Hall, Schulz, Shmoys & Wein [233] also use their general method to establish 2-approximation algorithms for the three problems $1|prec|\sum w_jC_j$, $1|r_j, prec, p_j = 1|\sum w_jC_j$ and $1|r_j, pmtn, prec|\sum w_jC_j$.

We now review results for on-line scheduling. Fiat & Woeginger [169] discuss problem $1|on\text{-}line\text{-}list|\sum C_j$. They give a $(\log n)^{1+\varepsilon}$-competitive on-line algorithm, and they show that no on-line algorithm can be better than $\log n$-competitive, even if preemption is allowed. The closely related problem $1|on\text{-}line\text{-}list|\sum F_j$ does not possess an on-line algorithm whose competitive ratio is bounded by a function that solely depends on n; the competitive ratio of any on-line algorithm for this problem must depend on the job processing times.

For problem $1|on\text{-}line, r_j|\sum C_j$, Hoogeveen & Vestjens [270, 547] show that no on-line algorithm can be better than 2-competitive. Three different 2-competitive algorithms are reported independently by Hoogeveen & Vestjens [270, 547], by Phillips, Stein & Wein [417], and by Stougie [506]. Recall that in the case of zero release dates, it is optimal to schedule jobs according to the SPT rule. Each of the three on-line algorithms is a variant of the SPT algorithm. Hoogeveen & Vestjens apply SPT with modified job release dates, and Phillips, Stein & Wein use an optimal preemptive schedule as a guide. Note that SPT in its original form is only n-competitive for minimizing either $\sum C_j$ or $\sum F_j$ (Mao, Kincaid & Rifkin [365]); up to a constant factor, this rather disappointing bound is best possible for problem $1|on\text{-}line, r_j|\sum F_j$. The results of Chekuri, Motwani, Natarajan & Stein [89] yield a *randomized* on-line approximation algorithm for problem $1|on\text{-}line, r_j|\sum C_j$ with a ratio guarantee of $e/(e-1) \approx 1.58$. Vestjens [547] shows that this is the best possible ratio that can be guaranteed by a randomized algorithm in this on-line model.

If job weights are imposed, a competitive on-line algorithm has to be more involved. We will see in Section 6.4.2 that by using the on-line framework of Hall, Shmoys & Wein [236] to *assign* jobs into sub-intervals of time, a $(4+\varepsilon)$-competitive algorithm for problem $1|on\text{-}line, r_j|\sum w_jC_j$ can be derived. However, it is possible to do better here. Recall that, for the off-line problem $1||\sum w_jC_j$, the SWPT rule generates an optimal sched-

ule. By applying the SWPT rule to sequence the jobs assigned to each sub-interval of time by the on-line framework, Hall, Schulz, Shmoys & Wein [233] show that a $(3+\varepsilon)$-competitive algorithm results. Through the use of randomization, Chakrabarti et al. [84] improve the on-line framework to obtain an algorithm for problem $1|\textit{on-line}, r_j|\sum w_jC_j$ that is $(2.89+\varepsilon)$-competitive. Goemans [206] uses his analysis for problem $1|r_j|\sum w_jC_j$ to develop a $(1+\sqrt{2})$-competitive algorithm for problem $1|\textit{on-line}, r_j|\sum w_jC_j$ that has a time requirement of $O(n\log n)$.

For non-clairvoyant scheduling, all results that are presented in Sections 5.4.2 and Sections 6.4.3 for parallel machines naturally also apply to the special case of a single machine.

4.3 Total Weighted Tardiness

4.3.1 Complexity

For problem $1||\sum T_j$, Lawler [323] presents a pseudo-polynomial dynamic programming algorithm, which relies on the following *decomposition* of the problem. Suppose that the jobs are indexed so that $d_1 \leq \cdots \leq d_n$, where $p_j \leq p_{j+1}$ if $d_j = d_{j+1}$, and let job j have the largest processing time. Then there is some index k, where $j \leq k \leq n$, such that jobs $1,\ldots,j-1,j+1,\ldots,k$ are sequenced before job j and jobs $k+1,\ldots,n$ are sequenced after job k. Lawler observes that each subproblem is represented by the time at which processing starts, and by three job indices which identify the jobs in the subproblem. The resulting dynamic programming algorithm requires $O(n^4P)$ time, where $P=\sum_{j=1}^{n} p_j$. The complexity status of $1||\sum T_j$ is resolved by Du & Leung [148], who show that the problem is $\mathcal{NP}$-hard. Koulamas [302] provides a review of algorithms for problem $1||\sum T_j$ and some of its generalizations.

Problem $1||\sum w_jT_j$ is shown by Lawler [323] and Lenstra, Rinnooy Kan & Brucker [349] to be strongly $\mathcal{NP}$-hard. Since $1|r_j,p_j=1|\sum T_j$ can be formulated as a linear assignment problem, it is solvable in $O(n^3)$ time. Lenstra & Rinnooy Kan [346] show that problem $1|\textit{prec}, p_j=1|\sum T_j$ is strongly $\mathcal{NP}$-hard, while strong $\mathcal{NP}$-hardness of problem $1|r_j|\sum T_j$ follows from the corresponding result for problem $1|r_j|L_{\max}$.

4.3.2 Enumerative Algorithms

Most enumerative algorithms for problems $1||\sum T_j$ and $1||\sum w_jT_j$ rely heavily on dominance rules to restrict the search. A typical example is

of the form: there exists an optimal sequence in which job j precedes job k if $p_j \leq p_k$, $w_j \geq w_k$ and $d_j \leq d_k$. Although the main rules are due to Emmons [160], others are derived by Elmaghraby [158] and Shwimer [489], while further development is provided by Rinnooy Kan, Lageweg & Lenstra [447]. A framework to prove the validity of these results is provided by Yu [568]. Tansel & Sabuncuoglu [526] provide some insights into problem 'hardness' using geometric observations on the structure of Emmons' dominance rules.

A wide variety of dynamic programming, branch and bound, and hybrid algorithms for problems $1||\sum T_j$ and $1||\sum w_j T_j$ appear in the literature. Schrage & Baker [467] and Lawler [326] propose alternative implementations of a dynamic programming algorithm in which the precedence relations obtained from Emmons' dominance rules are regarded as precedence constraints.

The most effective enumerative algorithms for problem $1||\sum T_j$ employ Lawler's decomposition ideas. Potts & Van Wassenhove [430], and later Szwarc [514], develop conditions, in addition to those originally given by Lawler [323], under which some positions need not be considered in the search for the optimal position of the job with the largest processing time. Potts & Van Wassenhove [430] propose an algorithm that successively decomposes subproblems until they are small enough to be solved by the dynamic programming algorithm of Schrage & Baker [467]. Using this approach, they are able to solve problems with up to 100 jobs. In another study, Potts & Van Wassenhove [433] provide a computational review of different implementations of dynamic programming and decomposition algorithms. A more effective branch and bound algorithm is developed by Szwarc & Mukhopadhyay [517]. They obtain a partial decomposition of the problem from Emmons' dominance rules and from analysis of adjacent job orderings between each pair of jobs. Their algorithm uses a decomposition-based branching rule that includes the conditions developed by Szwarc [514], and computes a lower bound from SPT and EDD orderings based on a relaxation that disassociates due dates from processing times. They provide computational results for instances with up to 150 jobs. Della Croce, Tadei, Baracco & Grosso [132] develop an alternative type of decomposition and an improvement to the lower bound of Szwarc & Mukhopadhyay. Their branch and bound algorithm exhibits similar computational behavior to that of Szwarc & Mukhopadhyay. An alternative decomposition approach with the potential to provided improved algorithms is developed by Chang, Lu, Tang & Yu [86].

For problem $1||\sum w_j T_j$, various relaxations provide lower bounding sche-

mes for inclusion in branch and bound algorithms. The transportation relaxation of Gelders & Kleindorfer [194, 195] uses a zero-one integer programming formulation with variables that indicate whether a job j is processed in a unit-time interval $[t-1, t]$. The linear assignment relaxation of Rinnooy Kan, Lageweg & Lenstra [447] computes an under-estimate of the total weighted tardiness of assigning job j to position k in the sequence. Fisher [172] proposes a Lagrangean relaxation of the machine capacity constraints, which produces fairly tight lower bounds but at considerable computational expense. The Lagrange multipliers can be regarded as a price for using the machine during the relevant unit-time interval. Subgradient optimization is used to find values of the multipliers. Potts & Van Wassenhove [432] adopt the approach of using quickly computed but weaker lower bounds. They use a Lagrangean relaxation of the constraints $T_j \geq C_j - d_j$, but use a 'multiplier adjustment' method to find values of the multipliers, as described in Section 4.2.2. Hoogeveen & Van de Velde [266] suggest a slight strengthening of this lower bound using their slack variable technique (see Section 4.2.2). Abdul-Razaq, Potts & Van Wassenhove [4] review various dynamic programming and branch and bound algorithms. Included in their review is a lower bound based on dynamic programming state-space relaxation. This relaxation maps the state-space of a dynamic programming formulation onto a smaller state space, which has the effect of allowing a 'sequence' in which some jobs appear more than once and others do not appear. Penalties, which assign a cost each time a job is sequenced, are used to improve the lower bound. This state-space relaxation bound requires pseudo-polynomial time. In their computational experiments, Abdul-Razaq, Potts & Van Wassenhove find that the branch and bound algorithm of Potts & Van Wassenhove [432] is the most effective, and it can solve instances with up to 50 jobs.

For problem $1|r_j|\sum T_j$, Chu [108] proposes a branch and bound algorithm. His lower bound is computed from a relaxation that disassociates due dates from release dates and processing times: the preemptive problem $1|r_j, pmtn|\sum C_j$ is solved first using the shortest remaining processing time rule, and then due dates are assigned to the respective completion times in EDD order. Various dominance rules are employed, and the branching rule first uses forward sequencing but switches to backward sequencing when the completion time of the partial sequence exceeds the largest due date. Computational results show that the algorithm is effective in solving instances with up to 30 jobs.

4.3.3 Local Search

Numerous heuristic methods that have modest computational requirements are available for problem $1||\sum T_j$. Wilkerson & Irwin [558] propose a heuristic that resembles a descent algorithm that applies the transpose neighborhood to the EDD sequence. Fry, Vicens, MacLeod & Fernandez [182] also use a descent method with the transpose neighborhood, but incorporating three alternative initial sequences and three alternative search strategies. Panwalkar, Smith & Koulamas [411] propose an $O(n^2)$ heuristic which adds jobs to an initial partial sequence according to tests based on pairs of jobs. Cheng [103] and Alidaee & Gopalan [19] point out that the heuristics of Panwalkar, Smith & Koulamas, and Wilkerson & Irwin are closely related to a modified due date heuristic of Baker & Bertrand [31], which successively schedules a job j for which $\max\{t+p_j, d_j\}$ is as small as possible when the machine becomes available at time t. Holsenback & Russell [253] and Russell & Holsenback [456] develop an $O(n^2)$ heuristic in which the EDD sequence is repeatedly improved by moving a job to a later position in the sequence. Based on results in [253, 456], the heuristic of Russell & Holsenback appears to be the most effective, and generates solutions to 100-job instances with total tardiness that exceeds the optimal value by just over 1%, on average.

Yu & Yu [570] extend an earlier decomposition heuristic approach of Potts & Van Wassenhove [435]. Their $O(n^2)$ heuristic starts with an EDD sequence, and then searches for the best position of a job with the largest processing time. This job is then fixed in the selected position, thereby generating two subproblems. Each subproblem is decomposed in a similar way until a complete sequence of jobs is obtained. Initial computational results with this decomposition approach are encouraging.

There have been several studies on local search algorithms for problems $1||\sum T_j$ and $1||\sum w_jT_j$. Initial work of Matsuo, Suh & Sullivan [372] studies simulated annealing based on the transpose neighborhood. Potts & Van Wassenhove [435] use the swap neighborhood in descent and simulated annealing algorithms, which they compare with special-purpose heuristics. For problem $1||\sum T_j$, they find that a special-purpose decomposition heuristic is slightly superior to the local search methods. On the other hand, a finely tuned simulated annealing algorithm that employs a descent routine gives the best results for $1||\sum w_jT_j$.

Crauwels, Potts & Van Wassenhove [118] compare multi-start descent, simulated annealing, threshold accepting, tabu search and genetic algorithms for problem $1||\sum w_jT_j$. For the natural sequence representation of

solutions, the swap neighborhood is used in multi-start descent, simulated annealing, threshold accepting and tabu search methods. They also propose a novel binary string representation, where each element indicates whether the corresponding job is on time or late; a decoding heuristic constructs a sequence of jobs from the binary string. A neighborhood move reverses one element of the binary string. Computational results for instances with up to 100 jobs indicated that a genetic algorithm that uses the binary representation of solutions and incorporates a descent routine based on the transpose neighborhood, and the two tabu search algorithms that use the sequence and binary representations provide the best quality solutions.

Congram, Potts and Van de Velde [113] propose a new neighborhood technique, which they call *dynasearch*, for problem $1||\sum w_j T_j$. This technique uses dynamic programming to search an exponential sized neighborhood comprising combinations of swaps in $O(n^2)$ time. They propose a multi-start descent algorithm, where the new starting solution is obtained from the previous local optimum by performing a small number of random swap moves. Computational results for the same test instances show the superiority of the dynasearch method over the local search algorithms of Crauwels, Potts & Van Wassenhove.

4.3.4 Approximation

Developing approximation algorithms with good performance guarantees for problem $1||\sum T_j$ is difficult. Chang, Matsuo & Tang [87] analyze several neighborhood search heuristics, Yu & Liu [569] analyze the heuristic of Wilkerson & Irwin [558], Yu [567] analyzes a backward version of Wilkerson & Irwin's heuristic, and Yu & Yu [570] analyze their decomposition heuristic. However, the best ratio guarantee for any of these heuristics in $n/2$.

Lawler [327] presents an FPTAS for problem $1||\sum T_j$ which is based on the rounding of state variables in his decomposition-based pseudo-polynomial dynamic programming algorithm. The approximation algorithm has a running time of $O(n^7/\varepsilon)$. Based on the ideas of Gens & Levner [197], Kovalyov [303] develops a procedure to find an upper bound on the optimal objective function value that is within a factor of three of a lower bound; this reduces the running time to $O(n^6 \log n + n^6/\varepsilon)$.

4.4 Weighted Number of Late Jobs

4.4.1 Complexity

For problem $1||\sum w_jU_j$, a solution is specified by a partition of the jobs into two subsets; those which are on time and those which are late. A schedule is constructed from the partition by sequencing first the on-time jobs in EDD order, while the late jobs are sequenced arbitrarily after all on-time jobs. For problem $1||\sum U_j$, an algorithm of Moore [385], which is sometimes known as the Moore-Hodgson algorithm, solves the problem in $O(n \log n)$ time. This algorithm repeatedly adds jobs in EDD order to the end of a partial schedule of on-time jobs. If the addition of job j results in this job being completed after time d_j, then a job in the partial schedule with the largest processing time is removed and declared late. Discarding a job with the largest processing time increases the opportunity for subsequent jobs to be completed by their due dates. Sidney [490] generalizes this algorithm to handle the case that a specified subset of jobs must be on time. He observes that jobs of the specified subset are not considered when discarding jobs, and that it may be necessary to discard more than one job to ensure that the last job in the current partial schedule is on time. An adaptation of Moore's algorithm to problem $1||\sum w_jU_j$ for the case of reverse agreeability of processing times with weights (where $p_j < p_k$ implies that $w_j \geq w_k$) is proposed by Lawler [322]. Thus, problem $1|p_j = 1|\sum w_jU_j$ is solvable in $O(n \log n)$ time. For problem $1|p_j = 1|\sum U_j$, Monma [381] observes that, by creating sets $S_k = \{j|k \leq d_j < k+1\}$ for $k = 1, \ldots, n-1$ and $S_n = \{j|d_j \geq n\}$, and considering jobs in the order $S_1, \ldots, S_n$, discarding a job when it is late, an optimal solution is obtained in $O(n)$ time.

An instance of the decision version of problem $1||\sum w_jU_j$ with $w_j = p_j$ and $d_j = \sum_{h=1}^{n} p_h/2$ for each job j is equivalent to the problem *Partition*, which is $\mathcal{NP}$-complete, as observed by Karp [294], without being strongly $\mathcal{NP}$-complete. Thus, problem $1||\sum w_jU_j$ is $\mathcal{NP}$-hard. However, the problem is pseudo-polynomially solvable by the following dynamic programming algorithm of Lawler & Moore [338]. Suppose that the jobs are indexed so that $d_1 \leq \cdots \leq d_n$, and let $F_j(t)$ be the minimum weighted number of late jobs for the subproblem involving jobs $1, \ldots, j$, where the last on-time job completes at time t. Using initialization of

$$F_0(t) = \begin{cases} 0 & \text{for } t = 0, \\ \infty & \text{for } t \neq 0, \end{cases}$$

we have the following recursion: for $j = 1, \ldots, n$,

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

where $T = \min\{d_n, \sum_{j=1}^{n} p_j\}$ is an upper bound on the completion time of the last on-time job. The minimum weighted number of late jobs is

$$\min_{t=0,\ldots,T}\{F_n(t)\},$$

and an optimal schedule is found by backtracking. The algorithm requires $O(nT)$ time. Note that an alternative recursion can be defined in which the state variable t is interchanged with the value of $F_j(t)$, so that the weighted number of late jobs becomes a state variable, and the minimum completion time of the last on-time job is a function value. This alternative dynamic programming algorithm requires $O(nW)$ time, where $W = \sum_{j=1}^{n} w_j$.

Polynomial solvability of minimizing the number of late jobs is not maintained in the presence of deadlines or precedence constraints. Lawler [330] shows that $1|\bar{d}_j|\sum U_j$ is $\mathcal{NP}$-hard, although this problem is open with respect to pseudo-polynomial solvability. Strong $\mathcal{NP}$-hardness of $1|prec, p_j = 1|\sum U_j$ is established by Garey & Johnson [186], a result which remains valid for the case of precedence constraints in the form of a set of chains (Lenstra & Rinnooy Kan [348]), and when each due date is equal to the earliest job completion time as defined by the precedence constraints, plus some constant k (Steiner [502]). Polynomial algorithms for some special cases of the latter model are developed by Sharary & Zaguia [485] and Steiner.

Release dates also introduce a substantial degree of difficulty, since Lenstra, Rinnooy Kan & Brucker [349] show that problem $1|r_j|\sum U_j$ is strongly $\mathcal{NP}$-hard. However, if preemption is allowed, the problem is solvable by a dynamic programming algorithm of Lawler [331]: he presents an $O(n^3W^2)$ algorithm for problem $1|r_j, pmtn|\sum w_jU_j$, the time complexity of which is $O(n^5)$ for problem $1|r_j, pmtn|\sum U_j$. Lawler [332] generalizes Moore's algorithm to solve the following special cases with release dates in $O(n\log n)$ time: problem $1|r_j|\sum U_j$ (and $1|r_j, pmtn|\sum U_j$ where preemption is not necessary) for the case of agreeability of release dates with processing times; and problem $1|r_j, pmtn|\sum w_jU_j$ for the case of nested $[r_j, d_j]$ intervals for the jobs, and for the case of agreeability of release dates with processing times and reverse agreeability of release dates and processing times with weights.

4.4.2 Enumerative Algorithms

Branch and bound algorithms for problem $1||\sum w_jU_j$ are proposed by Villarreal & Bulfin [550] and Potts & Van Wassenhove [434], each of which use a dominance rule to restrict the size of the search tree, and a binary branching rule that fixes a job to be on time or late at each branching. Villarreal & Bulfin use two fairly weak lower bounds that are based on solving a relaxation of the original problem using Moore's algorithm. Potts & Van Wassenhove develop an improved Moore-based bound that dominates Villarreal & Bulfin's bound. They also propose a lower bound that solves a scaled version of the problem using Lawler & Moore's dynamic programming [338]. Further, they propose a lower bounding procedure that solves the following linear programming relaxation in $O(n \log n)$ time:

$$\begin{array}{lll} \text{minimize} & \sum_{j=1}^{n} w_j u_j & \\ \text{subject to} & \sum_{h=1}^{j} p_h(1-u_h) \le d_j & \text{for } j = 1, \ldots, n, \\ & 0 \le u_j \le 1 & \text{for } j = 1, \ldots, n, \end{array}$$

where $u_j = 1$ if job j is late, and $u_j = 0$ if job j is on time. Computational results show that the linear programming based bound is preferred to the other lower bounds. Another significant contribution of Potts & Van Wassenhove is a reduction procedure which uses bounds in an attempt to establish that certain jobs are on time and certain jobs are late in an optimal schedule. If the jobs are indexed so that $d_1 \le \cdots \le d_n$, and job j is on-time, then by resetting due dates so that

$$\begin{array}{ll} d_h = \min\{d_h, d_j - p_j\} & \text{for } h = 1, \ldots, j-1, \\ d_h = d_h - p_j & \text{for } h = j+1, \ldots, n, \end{array}$$

job j can be eliminated from the problem. Computational results obtained by first applying this reduction procedure, and then solving the resulting reduced problem, either with the Lawler-Moore dynamic programming algorithm or with the branch and bound algorithm that uses the linear programming based bound, shows that both methods are successful in solving instances with up to 1000 jobs. Preference between dynamic programming and branch and bound depends the characteristics of the data for the particular problem instance that is to be solved.

For problem $1|r_j|\sum U_j$, Dauzère-Pérès [122] develops procedures for obtaining lower and upper bounds. The lower bound is obtained by solving

a sequence of linear programming problems. An $O(n^2)$ heuristic that uses the key ideas in Moore's algorithm is proposed to generate an upper bound. Computational results for instances with up to 50 jobs show that differences between the lower and upper bounds are small (at most two) for most problem classes.

A branch and bound algorithm for problem $1|\bar{d}_j|\sum U_j$ is proposed by Hariri & Potts [244]. They use a pseudo-polynomial lower bounding procedure that is developed using dynamic programming state-space relaxation, as described in Section 4.3.2, and a binary branching rule that fixes a job to be on time or late at each branching. Computational results for instances with up to 300 jobs show that the lower bound is sufficiently tight to restrict the size of the search tree.

4.4.3 Approximation

Sahni [457] and Gens & Levner [196, 197] propose FPTAS's for problem $1||\sum w_jU_j$. Sahni's scheme applies to the problem of maximizing the weighted number of on-time jobs, and Gens & Levner's scheme applies to the problem of minimizing the weighted number of late jobs. The scheme of Gens & Levner [196] has a running time of $O(n^3/\varepsilon)$. By developing a procedure to find an upper bound on the optimal objective function value that is within a factor of two of a lower bound, Gens & Levner [197] reduce the running time to $O(n^2\log n+n^2/\varepsilon)$. Ibarra & Kim [279] describe a PTAS for the maximization version of problem $1|tree|\sum U_j$.

4.5 Total Weighted Earliness and Tardiness

4.5.1 Complexity

Models with total (weighted) earliness plus total (weighted) tardiness as the optimality criterion are widely studied. Baker & Scudder [33] survey the main results in this area of scheduling. One variant that has received much attention is problem $1|d_j=d|\sum(w_jE_j+w'_jT_j)$ in which jobs have a common due date d. In some studies the common due date is *unrestricted*, which is the case for $d\geq\sum_{j=1}^{n}p_j$; otherwise, the common due date is *restricted*.

For problem $1|d_j=d|\sum(E_j+T_j)$ with a restricted common due date, Hall, Kubiak & Sethi [237] and Hoogeveen & Van de Velde [264] establish $\mathcal{NP}$-hardness. Hall, Kubiak & Sethi also derive a pseudo-polynomial dynamic programming algorithm that requires $O(nP)$ time, where $P=\sum_{j=1}^{n}p_j$. Kanet [292] derives the following properties of an optimal solution

for the unrestricted common due date version of this problem, from which he develops an $O(n \log n)$ algorithm. First, there is no idle time between jobs. Second, one of the jobs is completed exactly at time d. Third, the schedule is a V-shaped, which means that jobs completed after time d are sequenced in SPT order, while jobs completed at or before time d are sequenced in LPT order (non-increasing order of processing times). Let a and b denote the numbers of jobs that complete after time d, and before or at time d, respectively, where $a + b = n$. If jobs $\alpha(1), \ldots, \alpha(a)$, respectively, complete after time d, then their objective function contribution is

$$ap_{\alpha(1)} + (a-1)p_{\alpha(2)} + \cdots + 2p_{\alpha(a-1)} + 1p_{\alpha(a)}.$$

Similarly, the objective function contribution is

$$0p_{\beta(1)} + 1p_{\beta(2)} + \cdots + (b-2)p_{\beta(b-1)} + (b-1)p_{\beta(b)}$$

if jobs $\beta(1), \ldots, \beta(b-1)$, respectively, complete before time d and job $\beta(b)$ completes exactly at time d. Thus, $a, a-1, \ldots, 2, 1$ and $0, 1, \ldots, b-2, b-1$ are regarded as *positional weights* for the processing times. The positional weights are minimized by choosing $b = \lceil n/2 \rceil$ and $a = \lfloor n/2 \rfloor$. Further, the objective function is minimized by assigning a longest job to a position with the smallest weight, a second longest job to a position with a second smallest weight, and so on.

Hall & Posner [238] prove that problem $1|d_j = d| \sum w_j(E_j + T_j)$ is $\mathcal{NP}$-hard. For this problem, pseudo-polynomial dynamic programming algorithms are available. In particular, Hoogeveen & Van de Velde [264] and Hall & Posner [238] present $O(n^2 d)$ and $O(nP)$ algorithms for the case of a restricted and unrestricted common due date, respectively. Problem $1|d_j = d| \sum (w_j E_j + w'_j T_j)$ is open with respect to pseudo-polynomial solvability.

The main properties that are used in the derivation of Kanet's algorithm for problem $1|d_j = d| \sum (E_j + T_j)$ with an unrestricted common due date also hold for various generalizations of this model, thereby providing $O(n \log n)$ algorithms. Specifically, Panwalkar, Smith & Seidmann [412] consider the minimization of $\sum (wE_j + w'T_j + w'C_j) + d$, where the common due date d is a decision variable. Cheng [102] argues that there should be no penalty for completion times close to the due date, and considers the minimization of $\sum (wE'_j + w'T'_j)$, where $E'_j = 0$ for $E_j \leq \tau$ and $E'_j = E_j$ otherwise, and $T'_j = 0$ for $T_j \leq \tau$ and $T'_j = T_j$ otherwise, where $\tau < \min_j p_j/2$ is a given tolerance. More natural models involving the minimization of $\sum (w \max\{E_j - \tau_j, 0\} +$

$w' \max\{T_j - \tau'_j, 0\}$), where $p_j > \tau_j + \tau'_j$, and $\sum(\max\{E_j - \tau, 0\} + w' \max\{T_j - \tau, 0\})$, for any non-negative τ, are analyzed by Baker & Scudder [33], and Weng & Ventura [555], respectively.

Hoogeveen & Van de Velde [268] consider the almost common due date problem in which the due date of each job j lies in the interval $[D, D+p_j]$, for some constant D. For the unrestricted problem $1|d_j \in [D, D+p_j]|\sum(wE_j + w'T_j)$ with large D, they derive an $O(n^2)$ dynamic programming algorithm.

For unrestricted due dates, Verma & Dessouky [546] consider problem $1|p_j = 1|\sum w_j(E_j + T_j)$ for the case of unit processing times and due dates that are not assumed to be integers. They show that this problem is polynomially solvable by presenting an integer programming formulation, and then showing that there exists an optimal integer solution of the linear programming relaxation. Since problem $1||\sum(E_j + T_j)$ is $\mathcal{NP}$-hard when there is no restriction on processing times and due dates, the problem of determining a schedule by inserting idle time between jobs, for a given job sequence, is of interest. For problems $1||\sum w_j(E_j + T_j)$ and $1||\sum(w_jE_j + w'_jT_j)$, Garey, Tarjan & Wilfong [193] and Davis & Kanet [127] present $O(n \log n)$ and $O(n^2)$ algorithms, respectively, for idle time insertion. An alternative algorithm of Swzarc & Mukhopadhyay [516] requires $O(qn)$, where q is the number of clusters of jobs. In computational results for randomly generated instances, Swzarc & Mukhopadhyay show that their algorithm requires substantially less computation time than that of Davis & Kanet.

Closely related to these earliness-tardiness models is the problem of minimizing the completion time variance, which we denote by $1||\sum(C_j - \overline{C})^2$, where $\overline{C}$ is the average completion time. Eilon & Chowdhury [157] prove that the V-shaped property holds for this problem. Cai [76] discusses the weighted problem $1||\sum w_j(C_j - \overline{C})^2$: in the case of reverse agreeability of processing times with weights (where $p_j < p_k$ implies that $w_j \geq w_k$), the V-shaped property still holds; however, this result does not extend to the general case with arbitrary weights. Kubiak [311] shows that all of these problems are $\mathcal{NP}$-hard. Kahlbacher [290] and De, Ghosh & Wells [128] develop dynamic programming algorithms for computing the best V-shaped schedule in pseudo-polynomial time. Hence, problem $1||\sum(C_j - \overline{C})^2$ and problem $1||\sum w_j(C_j - \overline{C})^2$ with agreeable weights are both pseudo-polynomially solvable. Further, problem $1||\sum w_j(C_j - \overline{C})^2$ with arbitrary weights is open with respect to pseudo-polynomial solvability.

4.5.2 Enumerative Algorithms

For problem $1|d_j = d|\sum(E_j + T_j)$ with a restricted due date, Szwarc [512] develops a branch and bound algorithm for the case that the start time of the schedule is given. Computational results for instances with up to 25 jobs are given. Hoogeveen, Oosterhout & Van de Velde [262] propose an improved branch and bound algorithm. Their lower bounds are obtained by performing a Lagrangean relaxation of the constraint that the total processing of early jobs cannot exceed d. This produces a Lagrangean problem in which the positional weight of each early job increases by the value of the multiplier from its natural value. An analytical method for determining an optimal value of the Lagrange multiplier allows the lower bound to be computed in $O(n \log n)$ time. The branching rule fixes jobs in non-increasing order of processing times to be early or late. Computational results for instances with up to 1000 jobs show that search trees are small, and all of the instances with more than 30 jobs are solved at the root node of the search tree. For problem $1|d_j = d|\sum(w_j E_j + w'_j T_j)$, Van den Akker, Hoogeveen & Van de Velde [535] propose a column generation algorithm (details of the column generation approach are given in Section 6.2.2). Computational results for the case of an unrestricted common due date show that this algorithm is effective in solving instances with up to 60 jobs.

For problem $1||\sum w_j(C_j - \overline{C})^2$, Bagchi, Sullivan & Chang [29] derive various dominance rules. By using a tree search approach (without lower bounds), they solve instances with up to 20 jobs. Ventura & Weng [545] present a quadratic integer programming formulation from which they derive a lower bound using Lagrangean relaxation. The lower bound is not tested in a branch and bound algorithm. However, computational results for instances with up to 500 jobs show that the lower bound is fairly tight.

Without a common due date, problems with total earliness and tardiness criteria present a much greater challenge. Using similar analysis to that of Szwarc & Mukhopadhyay [517] for problem $1||\sum T_j$ (see Section 4.3.2), Szwarc [515] considers adjacent job orderings between each pair of jobs for problem $1||\sum(wE_j + w'T_j)$ to obtain a partial decomposition. By incorporating a branching rule, but no lower bounds, he gives computational results for instances with 10 jobs. Abdul-Razaq & Potts [3] develop a branch and bound algorithm for a constrained version of problem $1||\sum(w_j E_j + w'_j T_j)$ which forces processing to start at time zero, and forbids machine idle time is between jobs. Their lower bounding scheme, which is based on dynamic programming state-space relaxation, is similar to that of Abdul-Razaq, Potts

& Van Wassenhove [4] for problem $1||\sum w_jT_j$ (see Section 4.3.2). Although the lower bounds are fairly tight, large search trees are generated for instances with 25 jobs. Hoogeveen & Van de Velde [267] propose a branch and bound algorithm for the general version of problem $1||\sum(wE_j + w'T_j)$. They use a backward branching rule for sequencing, which is combined with an $O(n^2)$ algorithm for inserting idle time between jobs. Their algorithm includes a variety of dominance rules, and use five different lower bounds that are developed from relaxations of the objective function, the machine capacity, the due dates, and the processing times, and a Lagrangean relaxation of the non-negativity of earliness. Computational results show that search trees become large when there are more than about 15 jobs.

4.5.3 Local Search

Any problem for which some V-shaped property can be established essentially requires the jobs to be partitioned into two subsets. In this case, the problem can be treated as one of partitioning, since sequencing is straightforward once the partition of jobs is known. Mittenthal, Raghavachari & Rana [379] propose a simulated annealing algorithm which is applicable when a V-shaped property holds. They use a neighborhood that either reassigns a job from one subset to the other, or swaps two jobs in different subsets provided that these jobs are adjacent pairs in an SPT ordering. Their algorithm first applies two descent procedures, the first of which uses the reassign neighborhood and the second uses the swap neighborhood, and then applies simulated annealing with the swap neighborhood. Computational results for problem $1||\sum(C_j - \overline{C})^2$ with instances containing up to 20 jobs show that the algorithm generates an optimal solution for each test problem. Genetic algorithms for problems $1|d_j = d|\sum(w_jE_j + w'_jT_j)$ and $1||\sum w_j(C_j - \overline{C})^2$ are proposed by Lee & Kim [341] and Gupta, Gupta & Kumar [229], respectively.

For problem $1||\sum(w_jE_j + w'_jT_j)$, there are efficient algorithms, as indicated in Section 4.5.1, for inserting idle time optimally, provided that the sequence of jobs is given. With this approach, local search can be used for finding a sequence of jobs. Yano & Kim [566] propose a descent method which uses the transpose neighborhood. Computational results for instances with up to 20 jobs in which the weights are proportional to the processing times of the respective jobs show that their descent method consistently generates an optimal solution.

4.5.4 Approximation

For problem $1|d_j = d|\sum(E_j + T_j)$ with a restricted due date, Hoogeveen, Oosterhout & Van de Velde [262] present a method that transforms the solution of the Lagrangean problem in their lower bounding scheme into a feasible schedule. They show that this approximation algorithm, which requires $O(n \log n)$ time, has a worst-case ratio of 4/3.

For problem $1|d_j = d|\sum w_j(E_j + T_j)$, Hall & Posner [238] use their dynamic programming algorithm to derive an FPTAS for the special case that each weight is bounded from above by a polynomial function of n. Jurisch, Kubiak & Józefowska [289] establish that this special case in fact is solvable in polynomial time. Kovalyov & Kubiak [304] and Woeginger [562] each present an FPTAS for the general case without imposing any restriction on the total weight of the jobs. Woeginger [561] gives a FPTAS for problem $1||\sum w_j(C_j - \overline{C})^2$ with agreeable weights. For problem $1||\sum(C_j - \overline{C})^2$, De, Ghosh & Wells [128] and Cai [76] independently develop an FPTAS, each of which has a running time of $O(n^3\varepsilon)$.

4.6 Other Criteria

4.6.1 Single Criteria

Problem $1||\sum w_j e^{\alpha(C_j - d_j)}$, where α is a positive constant, is solvable in $O(n \log n)$ time by a rule of Rothkopf [452] in which jobs are sequenced in non-decreasing order of $e^{\alpha d_j}(1 - e^{-\alpha p_j})/w_j$. Rothkopf & Smith [453] show that, among a certain class of objective functions that are closed under scalar multiplication, the only problems of minimizing total cost that can be solved by a priority rule are $1||\sum w_j C_j$ and $1||\sum w_j e^{\alpha(C_j - d_j)}$.

Problem $1||\sum w_j C_j^2$ is studied by Townsend [528]. His analysis shows that the SPT rule solves problem $1||\sum C_j^2$ in $O(n \log n)$ time. However, the complexity status of problem $1||\sum w_j C_j^2$ is open. Townsend proposes a branch and bound algorithm, but does not perform computational tests to assess its effectiveness. Szwarc, Posner & Liu [518] analyze adjacent job orderings to decompose the problem, as Section 4.3.2 They report that 191 of 200 instances with up to 100 jobs are solved with this approach. Szwarc [513] and Della Croce et al. [131] develop this decomposition approach further by using a branch and bound algorithm to solve any subproblems that cannot be decomposed, and by developing stronger decomposition rules. In their computational tests, Della Croce et al. solve instances with up to 400 jobs. However, a greater challenge results if a strong positive correlation between

processing times and weights is introduced; in this case, their results are limited to instances with up to 100 jobs. Szwarc & Mukhopadhyay [516] develop a similar type of algorithm for problem $1||\sum w_j(C_j - p_j)^2$

The *late work* for job j is the amount of processing performed after its due date, and is denoted by V_j. Problems $1|pmtn|\sum V_j$ and $1||\sum V_j$ are analyzed by Potts & Van Wassenhove [437]. For problem $1|pmtn|\sum V_j$, they show that the minimum total late work is equal to the maximum tardiness $T_{\max}$ for the EDD sequence. Further, if $T_{\max} > 0$, an optimal solution is obtained by removing the first $T_{\max}$ units of processing from the EDD schedule, and rescheduling it after the last job. By contrast, problem $1||\sum V_j$ is shown to be $\mathcal{NP}$-hard, but is pseudo-polynomially solvable in $O(n(T_{\max}+p_{\max}))$ time by dynamic programming, where $p_{\max}$ denotes the largest processing time. The algorithm relies on the property that on-time jobs (those which are started before their due dates) are sequenced in EDD order. Computational results for instances with up to 10 000 jobs show the effectiveness of the dynamic programming algorithm. Potts & Van Wassenhove [438] derive two FPTAS's for problem $1||\sum V_j$, based on the rounding of state variables in two alternative dynamic programming formulations. The more efficient scheme has a running time of (n^2/ε).

Problems of minimizing total weighted late work are studied by Hariri, Potts & Van Wassenhove [246]. They present an $O(n \log n)$ algorithm for problem $1|pmtn|\sum w_j V_j$. For problem $1||\sum w_j V_j$, although the result for the unweighted problem that on-time jobs are sequenced in EDD order no longer holds, it is shown that at most one on-time job violates the EDD order at any time. This property allows the development of a pseudo-polynomial algorithm with a time requirement of $O(n^2 P)$, where $P = \sum_{j=1}^{n} p_j$. A branch and bound algorithm is proposed which uses dynamic programming state-space relaxation method to obtain a lower bounds. Computational results show that the algorithm is effective in solving instances with up to 700 jobs. By rounding state variables in a dynamic programming formulation, Kovalyov, Potts & Van Wassenhove [305] derive an FPTAS, which has a running time of $O(n^3 \log n + n^3/\varepsilon)$.

4.6.2 Multiple Criteria

When there are multiple criteria, either a *hierarchical* or a *simultaneous* approach can be adopted. Under a hierarchical approach, the criteria are ranked in order of importance; the first criterion is optimized first, the second criterion is then optimized, subject to achieving the optimum with respect

to the first criterion, and so on. For simultaneous optimization, there are two approaches. First, all 'efficient' (or Pareto optimal) schedules can be generated, where an efficient schedule is one in which any improvement to the performance for one of the criteria causes a deterioration with respect to one of the other criteria. Second, a single objective function can be constructed, for example by forming a linear combination of the various criteria, which is then optimized.

There is an extensive literature dealing with many aspects of multi-criteria scheduling. Surveys of algorithms and complexity results in this area are given by Dipeepan & Sen [140], Fry, Armstrong & Lewis [181], Hoogeveen [256], Lee & Vairaktarakis [342] and Chen & Bulfin [99]. We only provide the most significant complexity results below.

Van Wassenhove & Gelders [541] propose a pseudo-polynomial algorithm for finding all efficient schedules with respect to $\sum C_j$ and $L_{\max}$. Their algorithm searches all possible $L_{\max}$ values. Since a given $L_{\max}$ value imposes job deadlines $\overline{d}_j$, the algorithm of Smith [495] is used to solve the corresponding $1|\overline{d}_j|\sum C_j$ problem. By showing that the maximum number of efficient points is $O(n^2)$, Hoogeveen & Van de Velde [265] provide an $O(n^3 \log n)$ algorithm for finding all efficient schedules. They also provide a generalization to the case that the two criteria are $\sum C_j$ and $f_{\max}$, which increases the time complexity to $O(n^3 \min\{n, \log P\})$, where $P = \sum_{j=1}^{n} p_j$. Hoogeveen [256] proves strong $\mathcal{NP}$-hardness of bicriteria problems involving $\sum w_j C_j$ and $L_{\max}$ (unless the primary criterion is $\sum w_j C_j$ in a hierarchical approach).

The complexity status of several problems in which one of the criteria is $\sum U_j$ remains open. For example, if the criteria are $\sum U_j$ and $L_{\max}$, the hierarchical problem is open, irrespective of the selection of the primary criterion. Also, open problems occur for the criteria $\sum U_j$ and $\sum C_j$ (unless the primary criterion is $\sum C_j$ in a hierarchical approach).

There are a number of significant results for the case that one of the criteria is maximum earliness. Garey, Tarjan & Wilfong [193] develop an $O(n(\log n + \log p_{\max}))$ algorithm for the problem of scheduling a single machine to minimize the maximum absolute deviation of job completion times from due dates, or equivalently problem $1||\max\{E_{\max}, T_{\max}\}$. Li, Chen & Cheng [354] show that an instance of problem $1||\max\{wE_{\max}, w'T_{\max}\}$, for arbitrary non-negative weights w and w', can be polynomially transformed into an instance of problem $1||\max\{E_{\max}, T_{\max}\}$, thereby allowing algorithms for unweighted problem to be used. Hoogeveen [258] shows that all efficient schedules for minimizing a function of two or three maximum cost

criteria can be found in $O(n^4)$ and $O(n^8)$ time, respectively. The *promptness* of a job is defined as its given target start time minus its actual start time. For the case that the target start time of each job j lies in the interval $[d_j - p_j - A, d_j - A]$, for some constant A, Hoogeveen [257] presents an $O(n^2 \log n)$ algorithm for finding all efficient schedules.

5 Parallel Machine Problems: Unit-Length Jobs and Preemption

Intuitively, scheduling jobs with preemption to some extent can be regarded as similar to scheduling unit jobs without preemption. Therefore, we discuss these two models together. Allowing preemption greatly mitigates the difficulty of solving scheduling problems, although each preemption may incur a cost in practice. If we consider a scheduling problem as a zero-one integer program with decision variables indicating the assignment of jobs (or their operations) to machines and time slots, then preemption allows a variable to take a value between zero and one, thereby indicating the proportion of the job or its operation to be assigned. From this perspective, polynomial solvability of preemptive scheduling problems is normally expected. Indeed, for many scheduling problems, preemption is a vital element for polynomial solvability.

It is quite simple to schedule jobs of equal lengths without any complicating constraints, such as release dates or precedence constraints. Just beyond the trivial case of scheduling identical machines, consider the problems $Q|p_j = 1|\sum f_j$ and $Q|p_j = 1|f_{\max}$. It is easily observed that there always exists an optimal schedule which processes the jobs in n time periods that have the earliest completion times. Once these n time slots have been identified, the next step is to allocate the jobs to the time slots according to the objective function. In the case of minimizing $\sum f_j$, this allocation amounts to solving an $n \times n$ weighted bipartite matching problem. In the case of minimizing $f_{\max}$, this involves the following iterated greedy allocation. Consider the last time slot t among the time slots that have not been considered yet, allocate such a job j that minimizes $f_j(t)$. When release dates and deadlines are introduced, similar approaches can be used. Further details are given by Dessouky, Lageweg, Lenstra & Van de Velde [139] and Lawler, Lenstra, Rinnooy Kan & Shmoys [335].

A survey of results for parallel machine scheduling, including both preemptive and non-preemptive models, is given by Cheng & Sin [104].

5.1 Minmax Criteria

It is not surprising that the first paper in parallel machine scheduling deals with preemption. In his paper, McNaughton [378] uses a simple wrap-around procedure, which runs in $O(n)$ time and generates at most $m-1$ preemptions, to solve problem $P|pmtn|C_{\max}$ to optimality. The basic idea behind it is quite simple: first calculate a lower bound on the value of an optimal schedule and then construct a schedule that matches the bound. Note that in this approach, the lower bound is used as a deadline for constructing a feasible schedule. Once a feasible schedule is found, it is also optimal. The lower bounds is the maximum of the average machine workload and the processing time of the longest job.

McNaughton's approach has been successfully applied to other preemptive scheduling problems. Following a generalization of McNaughton's lower bound to the case of uniform machines by Liu & Yang [361], Horvath, Lam & Sethi [276] suggest the *longest remaining processing time on fastest machines* (LRPT-FM) rule: at each time point, the job with the most remaining processing time is scheduled on the fastest available machine. Thereby they solve problem $Q|pmtn|C_{\max}$ in $O(mn^2)$ time. Note that the LRPT-FM rule is an adaptation of the *critical path* method of Muntz & Coffman [393], which we discuss later in Section 5.3. A less intuitive and more case-based algorithm is developed by Gonzalez & Sahni [217], which significantly reduces both the running time to $O(n+m\log m)$ and the number of preemptions.

At the cost of solving a linear program, in which a decision variable represents the total time spent by each job on each machine, Lawler & Labetoulle [333] provide a general lower bound on the optimal makespan for problem $R|pmtn|C_{\max}$, and then solve it to optimality as an $O|pmtn|C_{\max}$ problem (see Section 7). For the case of two machines, Gonzalez, Lawler & Sahni [214] develop a linear time algorithm for problem $R2|pmtn|C_{\max}$.

Horn [274] generalizes McNaughton's approach to the case where jobs have deadlines. By considering the amount of processing that has to be performed on each job by its deadline, Horn provides a necessary and sufficient analytic condition for a feasible schedule to exist. This condition leads directly an $O(n^2)$ algorithm for problem $P|pmtn|L_{\max}$, and for problem $P|r_j, pmtn|C_{\max}$ which is its symmetric counterpart.

More generally, if both release dates and deadlines are present, Horn [274] formulates the problem of testing the existence of a feasible preemptive schedule as a network flow model, which can be solved in $O(n^3)$ time. Note that, if all jobs are released at the same time, such a test can be performed

in $O(n(\log m + \log n))$ time by an algorithm of Sahni [458]. Combining Horn's network flow formulation with a binary search, Labetoulle, Lawler, Lenstra & Rinnooy Kan [315] propose an $O(n^3 \min\{n^2, \log n + \log \max_j p_j\})$ algorithm for problem $P|r_j, pmtn|L_{\max}$.

In the case of uniform machines, Sahni & Cho [461] and Labetoulle, Lawler, Lenstra & Rinnooy Kan [315] show that problem $Q|r_j, pmtn|C_{\max}$ is solvable in $O(n \log n + mn)$ time; the resulting schedule has at most $O(mn)$ preemptions.

Martel [366] provides an algorithm for problem $Q|r_j, pmtn|L_{\max}$. By computing a maximal network flow, Federgruen & Groenevelt [164] develop an improved algorithm that runs in $O(kn^3)$ time, where k is the total number of distinct speeds. The linear programming technique of Lawler & Labetoulle [333] can be used to provide a polynomial time algorithm for problem $R|r_j, pmtn|L_{\max}$.

5.2 Minsum Criteria

McNaughton [378] shows that for problem $P||\sum w_j C_j$, the total weighted completion time cannot be reduced by allowing preemption. This property is extended by Du, Leung & Young [153] to the case that jobs have precedence constraints in the form of a set of chains. Therefore, we discuss these models in Section 6 which considers non-preemptive parallel machine scheduling. However, for other models, preemption may help. Actually, simple examples show that, if the machines have different speeds, or the jobs are subject to precedence constraints in the form of intrees or outtrees, preemption can reduce the total weighted completion time.

Gonzalez [209] solves $Q|pmtn|\sum C_j$ in $O(n \log n + mn)$ time. His algorithm is based on the *shortest remaining processing time on fastest machines* (SRPT-FM) rule: at each time point, schedule the job with the shortest remaining processing time on the fastest available machine. More generally, by combining the SRPT-FM rule with the LRPT-FM rule mentioned in the previous section, McCormick & Pinedo [374] compute in $O(m^3 n)$ time the entire tradeoff curve of schedules that *simultaneously* minimize both the total weighted completion time and the makespan.

If release dates are considered, problems become $\mathcal{NP}$-hard as is the case for problem $P2|r_j, pmtn|\sum C_j$ (Du, Leung & Young [152]). Until very recently, little was known about approximation algorithms for problems involving minsum criteria. Phillips, Stein & Wein [417] introduce an integer program to formulate the non-preemptive scheduling of jobs

that are preempted into unit-length pieces. Then they round the solution of the corresponding linear programming relaxation to a feasible preemptive schedule. Their techniques result in a $(16+\varepsilon)$-, $(24+\varepsilon)$- and $(32+\varepsilon)$-approximation algorithm for problems $P|r_j, pmtn|\sum w_jC_j$, $R|r_j, pmtn|\sum C_j$ and $R|r_j, pmtn|\sum w_jC_j$, respectively. The constant-approximation algorithm for $R|pmtn|\sum C_j$ is particularly interesting, since the complexity status of the problem is still open, and is vexing issue in the area of preemptive scheduling, given that its non-preemptive counterpart is polynomially solvable (see Section 6.2). The ratio guarantee for problem $P|r_j, pmtn|\sum w_jC_j$ has been significantly improved, even with precedence constraints.

We now discuss other minsum criteria. Lawler [329] shows that problem $P|pmtn|\sum U_j$ is $\mathcal{NP}$-hard. However, when the number of machines is fixed, Lawler [325] shows that problems $Qm|pmtn|\sum w_jU_j$ and $Qm|pmtn|\sum w_jU_j$ are solvable in pseudo-polynomial and polynomial time, respectively. Moreover, Lawler & Martel [337] provide an FPTAS for problem $Q2|pmtn|\sum w_jU_j$.

All problems of minimizing total tardiness are hard, since Section 4.3.1 points out that problems $1||\sum T_j$ and $1||\sum w_jT_j$ are $\mathcal{NP}$-hard and strongly $\mathcal{NP}$-hard, respectively, and there is no advantage in allowing preemption for these single machine problems.

5.3 Precedence Constraints

5.3.1 Unit-Length Jobs

To appreciate the effect of precedence constraints, it is helpful to start with the scheduling of jobs with equal lengths. There has been an intensive research in this area.

The presence of general precedence constraints makes all problems $\mathcal{NP}$-hard if the number of machines is part of input, since this is the case for problems $P|prec, p_j = 1|C_{\max}$ and $P|prec, p_j = 1|\sum C_j$, as shown by Ullman [530] and Lenstra & Rinnooy Kan [346], respectively. Actually, Lenstra & Rinnooy Kan show that it is $\mathcal{NP}$-complete even to approximate the solution of problem $P|prec, p_j = 1|C_{\max}$ within a factor better than $4/3$.

However, this stagnant situation changes immediately if the form of the precedence relations is relaxed into that of a *tree*, or if the number of machines is fixed.

Hu [277] gives an algorithm for problems $P|tree, p_j = 1|C_{\max}$ and

$P|outtree, p_j = 1|\sum C_j$ using a non-preemptive *critical path* scheduling algorithm, which always schedules the job that heads the longest current chain of unscheduled jobs. Various algorithms have been designed for a number of special cases of problem $Pm|prec, p_j = 1|C_{\max}$, which include the case where the precedence graph is an *opposing forest*, that is, the disjoint union of an in-forest where each job has at most one immediate successor and an out-forest where each job has at most one immediate predecessor (Garey, Johnson, Tarjan & Yannakakis [192]).

Similarly, the formulation by Fujii, Kasami & Ninomiya [183] of problem $P2|prec, p_j = 1|C_{\max}$ as a maximum cardinality matching problem, which is solvable in $O(n^3)$ time, provides the groundwork for the development of a series of improved algorithms. Moreover, polynomial solvability has been extended to minimizing $\sum C_j$ and to including job release dates and due dates (Garey & Johnson [186, 187]).

Some of the simple algorithms for problem $P2|prec, p_j = 1|C_{\max}$ are adapted to the more general problem $P|prec, p_j = 1|C_{\max}$, and are shown to have some good worst-case guarantees. On the other hand, various minimal $\mathcal{NP}$-hard problems have been identified (see Lawler, Lenstra, Rinnooy Kan & Shmoys [335] for more details).

Polyhedral analysis for minimizing total weighted completion times yields strong lower bounds on optimal solution values; Queyranne & Schulz [440] give a comprehensive survey of the main results in this area. Motivated by the results of Van den Akker [533], Hall, Schulz, Shmoys & Wein [233] show that if list scheduling (which always schedules the first in a prespecified list of unscheduled jobs whenever a machine becomes available) is guided by an optimal solution to a linear programming relaxation, then it is guaranteed to produce high quality approximation algorithms, both off-line and on-line, for minimizing total weighted completion time. As a result, significant improvements to the previous approximation results are obtained. In particular, a 3-approximation algorithm is given for problem $P|r_j, prec, p_j = 1|\sum w_j C_j$, where this bound improves to $3 - 1/m$ if there are no (non-trivial) release dates.

5.3.2 General-Length Jobs

The polynomial solvability picture for scheduling jobs with arbitrary processing times under preemption is very similar to that for the non-preemptive scheduling of unit-length jobs. This similarity suggests a close relationship between these two models, as Lawler [328] observes as a result of deriving

polynomial algorithms for a series of counterparts in the former model of well-solvable problems in the latter model.

In parallel to the complexity results for scheduling jobs of unit-length, problem $P|pmtn,prec|C_{\max}$ is shown to be $\mathcal{NP}$-hard by Ullman [531], whereas problems $P|pmtn,tree|C_{\max}$, $P2|pmtn,prec|C_{\max}$ and even $Q2|pmtn,prec|C_{\max}$ are all polynomially solvable by the algorithms of Muntz & Coffman [392, 393] and Horvath, Lam & Sethi [276], respectively.

Interestingly, the algorithm of Muntz & Coffman [392, 393] also focuses on the critical path of the precedence graph, as is the case in Hu's algorithm for scheduling unit-length jobs. However, Gonzalez & Johnson [213] use a totally different list scheduling approach for solving problem $P|pmtn, tree|C_{\max}$. Their algorithm segregates the jobs into two classes. In one class there is what can be termed the 'backbone' of the problem, a superset of those jobs whose start and finish times are fixed in any optimal schedule. The other jobs can in general be scheduled with some freedom. Their algorithm runs in $O(n \log m)$ time. In the same spirit as their LRPT-FM algorithm for problem $Q|pmtn|C_{\max}$ (see Section 5.1), Horvath, Lam & Sethi [276] solve problem $Q2|pmtn, prec|C_{\max}$ in $O(mn^2)$ time.

Lam & Sethi [319] adapt the Muntz-Coffman algorithm for problem $P|pmtn, prec|C_{\max}$ and show that it has a worst-case ratio of $2 - 2/m$. Similarly, in the case of uniform machines, Horvath, Lam & Sethi [276] prove that this algorithm has a ratio guarantee of $\sqrt{3m/2}$, which is tight up to a constant factor. To improve the Muntz-Coffman algorithm, Jaffe [284] suggests scheduling jobs without unforced idleness by always using fastest machine available. He proves that this improves the ratio guarantee to $\sqrt{m} + 1/2$.

In parallel to their approximation results for problem $P|r_j, prec, p_j = 1|\sum w_j C_j$, Hall, Schulz, Shmoys & Wein [233] show that for problem $P|r_j, prec, pmtn|\sum w_j C_j$ a linear programming guided list scheduling algorithm has a ratio guarantee of 3; this bound is improved to $3 - 1/m$ if there are no (non-trivial) release dates.

5.4 On-Line Algorithms

5.4.1 Clairvoyant Scheduling

Preemption makes on-line scheduling quite successful. Observe that, for problem $P|on\text{-}line\text{-}list, pmtn|C_{\max}$, the main additional difficulty is to reserve appropriate spaces in anticipation of future jobs. In their algorithm for scheduling over a list, Chen, Van Vliet & Woeginger [95] maintain a certain

pattern of machine loads during the whole process of scheduling, and they prove that if this pattern is a geometric sequence of the ratio $m : (m-1)$, then the algorithm has a competitive ratio of $1/(1-(1-1/m)^m)$. This competitive ratio approaches $e/(e-1) \approx 1.582$ as m becomes large. They also show that such a competitive ratio is the best possible for every $m \geq 2$.

On-line scheduling becomes much easier if jobs arrive over time: For the nearly on-line variant of problem $P|\textit{on-line}, r_j, \textit{pmtn}|C_{\max}$ in which the release date of the next job is always known, Gonzalez & Johnson [213] give a 1-competitive (i.e., optimal) algorithm. Sahni & Cho [459] and Labetoulle, Lawler, Lenstra & Rinnooy Kan [315] extend this result to the nearly on-line variant of problem $Q|\textit{on-line}, r_j, \textit{pmtn}|C_{\max}$ on uniform machines. Hong & Leung [255] show that an optimal, 1-competitive (fully) on-line algorithm for $P|\textit{on-line}, r_j, \textit{pmtn}|C_{\max}$ is actually quite easy to derive: whenever a new job arrives, all the *active* jobs which have arrived but have not yet been finished are rescheduled according to a slightly modified McNaughton's wrap-around rule. For the (fully) on-line version of $Q|\textit{on-line}, r_j, \textit{pmtn}|C_{\max}$, Vestjens [547] shows that there exists a 1-competitive on-line algorithm if and only if the speeds satisfy $s_{i-1}/s_i \leq s_i/s_{i+1}$ for $i = 2, \ldots, m-1$, where s_i is the speed of ith fastest machine.

In contrast to the single machine case (see Section 4.1), it is still unknown whether preemption is beneficial for the parallel machine problem. Vestjens [547] shows that the competitive ratio of any on-line algorithm for $P|\textit{on-line}, r_j, \textit{pmtn}|L_{\max}$ is at least $12/11 \approx 1.090$.

In Section 4.2, we have seen that the shortest remaining processing time rule is 1-competitive for problem $1|\textit{on-line}, r_j, \textit{pmtn}|\sum C_j$. This result does not extend to the case of more than one machine. In fact, Vestjens [547] shows that the competitive ratio of any on-line algorithm for problem $P|\textit{on-line}, r_j, \textit{pmtn}|\sum C_j$ is at least $22/21 \approx 1.047$. Phillips, Stein & Wein [417] describe a variation of SRPT and prove that it is 2-competitive.

For problem $P|\textit{on-line}, r_j, \textit{pmtn}|\sum F_j$, Leonardi & Raz [352] prove that the SRPT algorithm has a competitive ratio of $O(\log \min\{n/m, \pi\})$, where $\pi = \max_j p_j / \min_j p_j$; this ratio is best possible up to a constant factor.

5.4.2 Non-Clairvoyant Scheduling

Due to lack of information, it is natural to adopt a strategy which at all times ensures that every uncompleted job has received an equal amount of processing. Not surprisingly, this *round-robin* strategy is actually very competitive for problem $P|\textit{on-line-nclv}, r_j, \textit{pmtn}|\sum F_j$. Motwani, Phillips

& Torng [387] prove that if all jobs are released at the same time, then the round-robin strategy has a competitive ratio of $2 - 2m/(n+m)$, for $n \geq m$. Moreover, no on-line algorithm can have a better competitive ratio. Under a dynamic environment where jobs arrive over time, they show that round-robin is $O(n/\log n)$-competitive, and no non-clairvoyant on-line algorithm is better than $\Omega(n^{1/3})$-competitive. As might be expected, some knowledge of job sizes does help. Motwani, Phillips & Torng prove that if the ratio π of the largest to smallest job processing time is a constant, then a trivial algorithm, *run-to-completion*, which runs each job to completion for any given job order, is π-competitive; this is the best possible competitive ratio for non-clairvoyant on-line algorithms for this variant.

For preemptive scheduling where the processing of a job can be stopped and restarted later, but any previous processing is lost after a preemption, Shmoys, Wein & Williamson [488] give a $(4\log m + 6)$-competitive algorithm, which is the best possible up to a constant factor, for problem $Qm|$*on-line-nclv*$, r_j,$ *pmtn*$|C_{\max}$, and a $(4\log n + 5)$-competitive algorithm for problem $Rm|$*on-line-nclv*$, r_j,$ *pmtn*$|C_{\max}$.

6 Parallel Machine Problems: No Preemption

6.1 Minmax Criteria

6.1.1 Complexity

In contrast to the wide availability of efficient algorithms for preemptive scheduling and for scheduling unit-length jobs, polynomial solvability is rare in cases where preemption is forbidden. This can easily be seen from the fact that problem $P2||C_{\max}$ is $\mathcal{NP}$-hard (Garey & Johnson [189]) and problem $P||C_{\max}$ is strongly $\mathcal{NP}$-hard (Garey & Johnson [188]). However, pseudo-polynomial algorithms can be derived for several problems with parallel machines if the number of machines is fixed. Using the approach of Rothkopf [452] and Lawler & Moore [338], a state variable is associated with each machine to represent its total workload (although one of these state variables can be eliminated for the case of identical and uniform machines). This allows pseudo-polynomial algorithms to be developed for problems $Rm||C_{\max}$ and $Rm||L_{\max}$, which are also applicable to the case of identical and uniform machines.

6.1.2 Enumerative Algorithms

For problem $P||C_{\max}$, Dell'Amico & Martello [133] propose a branch and bound algorithm. Assuming that the jobs are indexed so that $p_1 \geq \cdots \geq p_n$, a trivial lower bound is given by $\max\{\sum_{j=1}^{n} p_j/m, p_1, p_m + p_{m+1}\}$. Using arguments from bin packing, a procedure is developed to improve upon this lower bound. The branching rule assigns a longest unscheduled job to one of the machines. Computational results show that the algorithm can solve instances with up to 10 000 jobs, and is far more efficient than dynamic programming.

Carlier [80] develops a branch and bound algorithm for problem $P|\ r_j|\ L_{\max}$. He adapts the generalized EDD heuristic for problem $1|r_j|L_{\max}$ to the corresponding parallel machine problem, and uses the heuristic solution to identify a critical set of jobs which then is used to compute a lower bound. Each job has an interval of availability, and a binary branching rule fixes a job to be either in the first half of the interval by assigning it a smaller deadline, or in the second half of the interval by assigning it a larger release date. Computational results for instances with 2, 3, 5 and 8 machines show that the algorithm is reasonably effective in solving instances with up to 100 jobs.

For problem $R||C_{\max}$, Van de Velde [537] and Martello, Soumis & Toth [367] propose branch and bound algorithms and heuristics. Van de Velde computes lower bounds using a surrogate duality relaxation, where multipliers are obtained using an ascent procedure. On the other hand, Martello, Soumis & Toth derive various bounds using Lagrangean relaxation, and compute multipliers using subgradient optimization. Computational results with instances having up to 200 jobs and 20 machines show that problem difficulty increases as the number of machines increases. The results also demonstrate the superiority of the algorithm of Martello, Soumis & Toth over Van de Velde's algorithm.

6.1.3 Local Search

Brucker, Hurink & Werner [63, 64] propose the use of a primary and secondary neighborhood for problem $P||C_{\max}$. For the primary neighborhood that reassigns a job on a most heavily loaded machine to a most lightly loaded machine, they show that a local optimum is obtained in $O(n^2)$ time. Their secondary neighborhood first makes an arbitrary reassignment of a job to another machine, and then applies the primary neighborhood to find a lo-

cal optimum. Their computational results for instances with up to 5000 jobs show that a simulated annealing method that uses the secondary neighborhood yields better quality solutions than simulated annealing based solely on the primary neighborhood.

Hariri & Potts [243] propose a descent algorithm for problem $R||C_{\max}$ which uses a neighborhood that either reassigns a job on a most heavily loaded machine to another machine, or interchanges a job on a most heavily loaded machine with a job on another machine. Computational results indicate that this descent algorithm generates better quality solutions than various two-phase heuristics (Potts [426], Lenstra, Shmoys & Tardos [350]) which use linear programming in their first phase to schedule most of the jobs. Hariri & Potts also report on initial experiments which indicate that a more complicated neighborhood structure has little effect on solution quality.

Glass, Potts & Shade [204] use the same reassign and swap neighborhood as Hariri & Potts in simulated annealing and tabu search algorithms for problem $R||C_{\max}$. They also describe a genetic algorithm in which a solution is represented by a list of machines to which the respective jobs are assigned, and a corresponding genetic descent algorithm in which the descent algorithm of Hariri & Potts is applied to each solution in every population. Computational results show that the simulated annealing, tabu search and genetic descent algorithms are roughly comparable, but the performance of the standard genetic algorithm is poor.

6.1.4 Approximation using List Scheduling

In a *list scheduling* algorithm, jobs are placed into a list, often in an arbitrary order. The algorithm always schedules the first *available* job on the list of unscheduled jobs whenever a machine becomes idle. More precisely, the availability of a job means that the job has been released and/or all its predecessors in the precedence graph have already been scheduled. In its simplicity and the fact that any optimal schedule can be constructed by a list scheduling algorithm, list scheduling is by far the most popular approach for scheduling parallel machines. If a list scheduling algorithm operates with an arbitrary list of jobs, then we denote the algorithm by LS. Since LS requires no knowledge of active and future jobs, its power will be mainly discussed in the section dealing with on-line scheduling.

In the first paper on the worst-case analysis of scheduling heuristics, Graham [225] shows that algorithm LS for problem $P||C_{\max}$ has a worst-case

ratio of $2-1/m$. If the job list is sorted in order of non-increasing processing times, then this algorithm is known as LPT, and is shown by Graham [226] to have an improved worst-case ratio of $4/3 - 1/(3m)$. Another variant of LS, as suggested by Graham, is to schedule the k longest jobs optimally, and then apply LS to the list of remaining jobs: this gives a ratio guarantee of $1 + (1 - 1/m)/(1 + \lfloor k/m \rfloor)$. Therefore, for fixed m, a family of these algorithms for different k's provides a PTAS, although the running time of $O(n^{km})$ is huge. Ibarra & Kim [278] show that LPT is no more than $1+2(m-1)/n$ away from the optimum makespan, if $n \geq 2(m-1)\pi$, where (as in Section 5.4) $\pi = \max_j p_j / \min_j p_j$.

If machines have different speeds, then Morrison [386] shows that LPT for problem $Q||C_{\max}$ has a worst-case ratio of $\max\{\sigma/2, 2\}$, where $\sigma = \max_i s_i / \min_i s_i$. A modified LPT algorithm, which assigns the current job to the machine on which it will finish first, is shown by Gonzales, Ibarra & Sahni [212] to improve the ratio guarantee to $2 - 2/(m+1)$. Subsequently, this guarantee is further improved to 19/12 by Dobson [141] and Friesen [178].

6.1.5 Bin-Packing Based Approximation

A second main approximation approach to problem $P||C_{\max}$ is to consider the dual problem, which is one of *bin-packing*. In a bin-packing problem, a number of items of various sizes are to be packed into a minimum number of bins of a certain given capacity. Note that problems of scheduling to minimize the makespan and of bin-packing share the same decision version. Naturally there is some common territory to explore.

Based on this principle of duality, Coffman, Garey & Johnson [111] propose a heuristic for problem $P||C_{\max}$, called *Multifit* (MF), to find by binary search the minimum capacity of the m bins into which the n items can be packed by a packing heuristic, called *first-fit decreasing* (FFD). In the FFD heuristic, each iteration packs the largest remaining item into the first bin into which it fits. They prove that MF has a ratio guarantee of $\rho + 2^{-k}$, where $\rho \leq 1.22$ and k denotes the number of binary search iterations. Later, Friesen [177] improves the bound to 1.2, and the minimum value of ρ is proved by Yue [571] to be 13/11. At the expense of a larger running time, the MF algorithm is refined by Friesen & Langston [180] to achieve a slightly better worst-case ratio of $72/61 + 2^{-k}$.

A similar principle of duality between scheduling and bin-packing is considered by Hochbaum & Shmoys [251]. Given the machine 'capacity' d, a

ρ-dual approximation algorithms ($\rho > 1$) produces a job 'packing' that uses at most the minimum number of machines of capacity d at the expense of possible capacity violation by no more than $(\rho - 1)d$. Using a family of dual approximation algorithms, Hochbaum & Shmoys provide a PTAS for problem $P||C_{\max}$.

The multifit approach and dual approximation approach are extended to uniform machines. Friesen & Langston [179] shows that the ρ in the worst-case ratio of MF is between 1.341 and 1.40, which is later improved to 1.38 by Chen [90]. Extension of the dual approximation approach by Hochbaum & Shmoys has finally lead to a PTAS for $Q||C_{\max}$ [252].

6.1.6 Approximation using Linear Programming

Extensive research has appeared in the literature on computing near-optimal solutions for scheduling models by rounding optimal solutions to linear programming relaxations. Many such applications are discussed throughout this review. There are two general approaches for exploiting the solution to a linear programming relaxation. The linear program is used either to guide the assignment of jobs to machines, or to derive job priorities that are used in constructing the schedule.

Extending the optimal solution to a linear programming relaxation by an enumerative process, Potts [426] obtains a 2-approximation algorithm for problem $R||C_{\max}$ when m is fixed. Lenstra, Shmoys & Tardos [350] extend Potts' approach by first establishing that the fractional solution to the linear programming relaxation can be rounded to a good integral approximation in polynomial time, thereby obviating the need for enumeration and removing the exponential dependence on m, and then deriving a 2-approximation algorithm even when m forms part of the input.

This approach is further extended to accommodate a more general objective criterion. Shmoys & Tardos [487] introduce a stronger rounding technique than the one of Lenstra, Shmoys & Tardos to develop a polynomial algorithm that can find a schedule with mean job completion time M and makespan at most $2T$, if a schedule with mean job completion time at most M and makespan at most T exists.

6.1.7 Other Approaches for Approximation

Based on a dynamic programming algorithm for problem $Pm||C_{\max}$ in which the state variables at each stage i form a set $S^{(i)}$ of $(m-1)$-tuples, repre-

senting the total workloads of the machines, Sahni [457] uses an *interval partitioning* method to restrict $|S^{(i)}|$, and hence provides an $O(n(n^2/\varepsilon)^{m-1})$ time FPTAS. In a similar vein, Horowitz & Sahni [275] derive an FPTAS for problems $Qm||C_{\max}$ and $Rm||C_{\max}$, each with a similar running time.

It is interesting to note that when the number of machines is part of the input these problems are strongly $\mathcal{NP}$-hard, as mentioned previously in this section. In fact, Lenstra, Shmoys & Tardos [350] prove that, unless $\mathcal{P}$=$\mathcal{NP}$, it is impossible to derive a polynomial time approximation algorithm for problem $R||C_{\max}$ with a worst-case ratio that is strictly better than 3/2.

Using a completely different approach, Hall & Shmoys [234] introduce the notion of an *outline*, which is a partial characterization of a schedule. From the outline, it is possible to compute relatively simply and quickly an optimal or near-optimal solution. Based on this notion they construct a PTAS for problem $P|r_j|L_{\max}$ (assuming that all due dates are non-positive).

6.2 Minsum Criteria

6.2.1 Complexity

The best known non-trivial problem that is polynomially solvable is $R||\sum C_j$. Horn [273] and Bruno, Coffman & Sethi [71] formulate this problem as a zero-one integer program that has a bipartite matching structure, and hence is solvable in $O(n^3)$ time. The formulation is based on the observation that the sum of completion times of jobs on the same machine is simply a weighted sum of their processing requirements on that machine, where the weight of any particular job processing requirement is the number of times it contributes to the individual completion times. This observation leads to simplifications for the case of identical or uniform machines.

It seems that this is the best we can do in terms of polynomial solvability, since any extension to a more general criterion (even if there are only two identical machines) or to including additional job characteristics results in $\mathcal{NP}$-hardness. In particular, problem $P2||\sum w_jC_j$ is $\mathcal{NP}$-hard, as shown by Bruno, Coffman & Sethi [71], and Lenstra, Rinnooy Kan & Brucker [349], and strong $\mathcal{NP}$-hardness of problem $P2|r_j|\sum C_j$ can be derived from the corresponding result for problem $1|r_j|\sum C_j$ (see Section 4.2.1). For complexity results on minsum problems with precedence constraints, see Section 6.3.

6.2.2 Enumerative Algorithms

Various branch and bound algorithms for problem $P||\sum w_jC_j$ appear in the literature. The earlier algorithms of Barnes & Brennan [42], Elmaghraby & Park [159], and Sarin, Ahn & Bishop [462] are based on a lower bound of Eastman, Even & Isaacs [155] that exploits the relationship with problem $1||\sum w_jC_j$. Elmaghraby & Park also derive a useful dominance rule: if $p_j \leq p_k$ and $w_j \geq w_k$, then there exists an optimal schedule in which job j is started before or at the same time as job k. Webster [551, 552] derives lower bounds based on job splitting (see Section 4.2). Computational results show that his bounds are tighter than the lower bound of Eastman, Even & Isaacs, although no branch and bound algorithm is developed.

Using a time-indexed formulation with variables that indicate whether job j is processed in a unit-time interval $[t-1, t]$, Belouadah & Potts [48] derive a lower bound for problem $P||\sum w_jC_j$ by performing a Lagrangean relaxation of the machine capacity constraints. They compute values of the multipliers by a simple constructive procedure, which allows the lower bound to be computed in polynomial time, even though the number of multipliers is pseudo-polynomial. Computational results show that the algorithm is effective in solving instances with up to 8 machines and 40 jobs, although problem difficulty increases as the number of machines increases.

Van den Akker, Hoogeveen & Van de Velde [534] propose an algorithm for problem $P||\sum w_jC_j$ that employs column generation. This approach uses a set partitioning formulation in which a zero-one variable for each single machine schedule indicates whether it is included in the solution of the parallel machine problem. Since the number of single machine schedules is large, only a limited number are included in the formulation at the outset. Using the dual variables in the solution of the linear programming relaxation of the set partitioning problem, new single machine schedules are generated by an $O(nP)$ dynamic programming pricing algorithm, where $P = \sum_{j=1}^{n} p_j$. Computational results for instances with up to 10 machines and 100 jobs show that that the column generation approach is more effective than the branch and bound algorithm of Belouadah & Potts, especially when the number of machines increases. An alternative column generation algorithm is developed independently by Chen & Powell [101], although it appears less effective than that of Van den Akker, Hoogeveen & Van de Velde. In another study, Chan, Kaminsky, Muriel & Simchi-Levi [85] show that the optimal value of the total weighted completion time is at most $(\sqrt{2}+1)/2$ times the value of the linear programming relaxation.

6.2.3 Approximation

As observed in Section 4.2, the SWPT rule guarantees optimality for problem $1||\sum w_jC_j$. If we apply list scheduling for problem $P||\sum w_jC_j$, where the list is constructed according to the SWPT rule, then the resulting heuristic has a worst-case ratio of $(1+\sqrt{2})/2$, which is proved by Kawaguchi & Kyan [296].

In the same spirit as his approach that combines dynamic programming and interval partitioning for problem $Pm||C_{\max}$, Sahni [457] constructs an $O(n(n^2/\varepsilon)^{m-1})$ time FPTAS for problem $Pm||\sum w_jC_j$. Sahni also develops a similar scheme for the following problem, which is shown by Bruno, Coffman & Sethi [72]) to be $\mathcal{NP}$-hard: among all possible $(m!)^{\lceil n/m\rceil}$ optimal schedules for $Pm||\sum C_j$, one with smallest makespan is required.

With respect to approximation algorithms for problem $P|r_j|\sum w_jC_j$, significant progress has been made recently. Phillips, Stein & Wein [417] derive a $(24+\varepsilon)$-approximation algorithm, which is based on some algorithmic and structural relationships between preemptive and non-preemptive schedules and linear programming relaxations of both. More recently, the ratio guarantee is improved to $4-1/m$ by Hall, Schulz, Shmoys & Wein [233] and by Queyranne [439].

For a much more general model $R|r_{ij}|\sum w_jC_j$, where the release date of each job j depends on the machine to which it is assigned, Hall, Schulz, Shmoys & Wein [233] present a 16/3-approximation algorithm. Inspired by time-indexed integer programming formulations, they introduce the notion of an *interval-indexed* formulation, in which the decision variables merely indicate in which time-interval a given job completes. Their algorithm combines techniques of rescaling and rounding the data, and builds on earlier research of Lenstra, Shmoys & Tardos [350] and Shmoys & Tardos [487] for constructing near-optimal solutions

Further exploitation of the relationship between preemptive and non-preemptive schedules by Chakrabarti et al. [84] leads an algorithm that converts a preemptive schedule with total completion time equal to C into a non-preemptive schedule with total completion time at most $\frac{7}{3}C$. When applied to the solution of the linear programming relaxation considered by Hall, Schulz, Shmoys & Wein [233] that is mentioned above, or to a dual ρ-approximation algorithm, this technique yields 3.5- and $(2.89+\varepsilon)$-approximation algorithms, respectively, for problem $P|r_j|\sum C_j$. These approximation results have been further improved, even for the on-line environment (see Section 6.4.2).

Alon, Azar, Woeginger & Yadid [20] derive a PTAS for the problem of minimizing the sum of the *machine* completion times taken to the power α, where $\alpha > 0$. Alon, Azar, Woeginger & Yadid [21] discuss the more general problem $P||\sum f(C_i)$; here the C_i are the machine completion times and f is a function that maps the non-negative reals into the non-negative reals. Assume that f is non-decreasing, convex, and fulfills

$$\forall \varepsilon > 0\ \exists \delta > 0\ \forall x, y \geq 0 :\ |x - y| \leq \delta y \Longrightarrow |f(x) - f(y)| \leq \varepsilon f(y).$$

This condition essentially states that f grows in a reasonably moderate way. They show that under these assumptions, problem $P||\sum f(C_i)$ possesses a PTAS.

Hoogeveen, Schuurman & Woeginger [263] establish the MAX SNP-hardness of problem $R|r_j|\sum C_j$; hence, unless $\mathcal{P}$=$\mathcal{NP}$, this problem cannot possess a PTAS. Bartal et al. [45] discuss a non-standard scheduling problem on m identical machines: The scheduler may *reject* jobs from processing; in this case, a job dependent penalty has to be paid. The goal is to minimize the total penalty incurred plus the makespan of the processed jobs. They give an FPTAS for the case where m is not part of the input, and a PTAS for the case where m is part of the input.

6.3 Precedence Constraints

In the presence of precedence constraints, all problems of interest become $\mathcal{NP}$-hard. In particular, Sethi [472] proves that problem $P2|tree|\sum C_j$ is $\mathcal{NP}$-hard, and Du, Leung & Young [153] prove that problem $P2|tree|C_{\max}$ is strongly $\mathcal{NP}$-hard. These complexity results remain true even for precedence constraints in the form of chains [153].

Approximation algorithms mainly involve list scheduling. For problem $P|prec|C_{\max}$, the worst-case ratio of LS is not affected by the presence of arbitrary precedence constraints, and remains as $2 - 1/m$. Graham [227] shows it is even the case that

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

where $C^*_{\max}(pmtn)$ denotes the optimal schedule length if preemption is allowed, which is a lower bound on $C^*_{\max}$. Surprisingly, Graham [227] observes that this ratio does not improve if the priority list is based on the critical path (CP) rule, i.e., if one always schedules the job with the longest outgoing chain of successors. However, if the precedence constraints take the

form of a tree or of chains, then Kunde [314] shows that performance of CP improves slightly since the worst-case ratios for these cases are $2-1/(m-1)$ and $5/3$, respectively.

If processing times are taken into account, then Kaufman [295] shows that for tree-type constraints,

$$C_{\max}(\mathrm{CP}) \leq C^*_{\max}(pmtn) + (1-1/m)\pi,$$

where, as before, $\pi = \max_j p_j / \min_j p_j$. By improving the way a list scheduling rule is guided by an optimal solution to a linear programming relaxation (cf. Section 5.3), Hall, Schulz, Shmoys & Wein [233] provide a 7-approximation algorithm for problem $P|r_j, prec|\sum w_j C_j$. This performance guarantee is improved to $5.33+\varepsilon$ by Chakrabarti et al. [84] after they have incorporated the randomization technique of Goemans & Kleinberg [207].

Building on the work of Lenstra & Rinnooy Kan [346], Hoogeveen, Schuurman & Woeginger [263] show that, unless $\mathcal{P}$=$\mathcal{NP}$, problem $P|prec|\sum C_j$ does not possess a polynomial time approximation algorithm with a better ratio guarantee than $8/7$.

6.4 On-Line Algorithms

6.4.1 Scheduling Over a List

The performance of on-line algorithms without preemption is less understood than that of on-line algorithms with preemption. We start with Graham's LS algorithm [225] for problem $P|$*on-line-list*$|C_{\max}$, which is $(2-1/m)$-competitive. Since LS can be regarded as a non-clairvoyant on-line algorithm, we discuss it in more detail in a later subsection. Faigle, Kern & Turan [163] observe that for $m=2$ or $m=3$, the competitiveness of LS is the best possible. Their work triggered progress on $P|$*on-line-list*$|C_{\max}$ for $m \geq 4$. From the worst-case examples for LS, the worst scenario is the arrival of a very long job at the time when the machine loads are evenly balanced (ironically, this is precisely what a good schedule should look like). This observation suggests that a pure 'greedy' approach to the allocation of jobs is not ideal, and that one machine should be lightly loaded to accommodate any possible long jobs. Using this philosophy, Galambos & Woeginger [184] and Chen, Van Vliet & Woeginger [94] provide algorithms with better competitive ratios than LS for all $m \geq 4$. For small numbers of machines ($m \leq 20$), the competitive ratios of the algorithms of Chen, Van Vliet & Woeginger are currently the best available.

For large m, the competitive ratios of the algorithms of Galambos & Woeginger, and Chen, Van Vliet & Woeginger for problem $P|$ *on-line-list*$|C_{\max}$ still approach 2. The first successful attack against the barrier of 2 is due to Bartal, Fiat, Karloff & Vohra [44]. Instead of leaving only a constant number of machines lightly loaded, they leave a constant *fraction* of machines lightly loaded. The competitive ratio of the resulting algorithm is 1.986. Later, this is improved to 1.945 by Karger, Phillips & Torng [293] and to 1.923 by Albers [16]. Albers also proves a lower bound of 1.852 on the competitive ratio for large numbers of machines ($m \geq 80$).

Note that in the case of identical machines, assigning a job to the first available machine is equivalent to assigning the job to the machine on which it finishes first. Extending LS in this way to an on-line algorithm LS$'$ for problem $Q|$*on-line-list*$|C_{\max}$, Cho & Sahni [105] prove that the competitive ratio of LS$'$ is $(1+\sqrt{5})/2$ for $m = 2$ and $1+\sqrt{(m-1)/2}$ for $3 \leq m \leq 6$. It is easy to check that for $m = 2$ and $m = 3$, these ratios are the best possible. For other values of m, Aspnes et al. [25] prove that LS$'$ is $\Theta(\log m)$-competitive. Results of Cho & Sahni [105] show that this bound is tight up to a constant factor. Aspnes et al. [25] improve the LS$'$ rule by a guessing and doubling strategy that leads to an 8-competitive algorithm. By a refinement of the doubling strategy, Berman, Charikar & Karpinski [50] improve the competitive ratio to $3+\sqrt{8} \approx 5.828$, and they derive a lower bound of 2.4380.

Both algorithms LS and LS$'$ are analyzed in the case of only two speeds 1 and s for problem $Q|$*on-line-list*$|C_{\max}$. For $s < 1$, Liu & Liu [358] show that LS is $(s+(m-1)/(s+m-1))$-competitive. For $s > 1$, Cho & Sahni [105] prove that the competitive ratio of LS$'$ is $3-4/(m+1)$ for all $m \geq 3$. LS$'$ is improved recently by Li & Shi [355] with a reduction of its competitive ratio by a constant independent of m. Their method is similar to those used by Galambos & Woeginger [184] and Chen, Van Vliet & Woeginger [94] for improving LS in the case of identical machines.

Epstein et al. [161] elaborate on problem $Q2|$*on-line-list*$|C_{\max}$. Suppose that the speeds of the two machines are 1 and $s \geq 1$. It is easy to see that list scheduling is the best deterministic on-line algorithm for any choice of s: Denote by $\phi = (\sqrt{5}+1)/2 \approx 1.6180$ the golden ratio. Then for $s \leq \phi$, the best possible competitive ratio is $1+s/(s+1)$, increasing from $3/2$ to ϕ. For $s \geq \phi$, the best possible competitive ratio is $1+1/s$, decreasing from ϕ to 1; this is the same ratio as for the trivial algorithm which puts *all* jobs on the faster machine. It turns out that list scheduling is also the best possible randomized algorithm for all speeds $s \geq 2$. On the other hand, for any speed $s < 2$, randomized algorithms are provably better than deterministic ones.

For problem $R|$*on-line-list*$|C_{\max}$, Aspnes et al. [25] first show that natural greedy approaches are far from optimally competitive by proving that the competitive ratio of LS$'$ is n. Then they describe an $O(\log m)$-competitive algorithm. Results of Azar, Naor & Rom [28] show that this algorithm is the best possible result up to a constant factor.

Awerbuch et al. [27] consider the on-line scheduling of unrelated machines so as to minimize the L_p norm ($p \geq 1$) of the machine completion times. They show that a simple algorithm, which always assigns the current job to the machine with minimum increase to the machine finish time, is $(1+\sqrt{2})$-competitive for the Euclidean norm and is $O(p)$-competitive in other L_p norms, and it is the best possible up to a constant factor. In the special case of identical machines with the Euclidean norm, Avidor, Azar & Sgall [26] prove that assigning the current job to the least loaded machine is the best strategy with a competitive ratio of $\sqrt{4/3}$. For the case that m is known to be sufficiently large, they provide an improved algorithm that is $(\sqrt{4/3}-\varepsilon)$-competitive for some constant $\varepsilon > 0$. Moreover, for any L_p norm and for any sufficiently large m, they give an algorithm that beats list scheduling.

Bartal et al. consider on-line scheduling over a list on parallel machines with job penalties (cf. Section 6.2.3). For $m = 2$, they derive a best possible competitive ratio of $(\sqrt{5}+1)/2 \approx 1.6180$. For an unbounded number of machines, they derive a best possible competitive ratio of $(\sqrt{5}+3)/2 \approx 2.6180$. The best possible competitive ratio for fixed $m \geq 3$ is unknown.

6.4.2 Scheduling Over Time

For the objective of minimizing the makespan, Shmoys, Wein & Williamson [488] describe a general technique to convert a non-clairvoyant scheduling algorithm for a problem with all jobs released at the same time to a non-clairvoyant on-line algorithm that can handle unknown release dates. They show that the quality of the schedule thus constructed is within a factor of 2 of the quality of schedules constructed by the off-line algorithm in the simpler environment. This technique does not only apply to parallel machine scheduling, but also to the entire class of shop scheduling problems. However, since any on-line algorithm that is designed by application of this technique cannot be guaranteed to be better than 2-competitive, we may expect to find better on-line algorithms for specific problems.

For problem $P|$*on-line*$, r_j|C_{\max}$, Chen & Vestjens [96] prove that an adapted LPT algorithm, which always schedules an available job with

the largest processing time once a machine becomes available, is (3/2)-competitive. They also show that no on-line algorithm can be better than 1.347-competitive for arbitrary m, or better than 1.382-competitive for $m = 2$.

We next consider the more general problem $P|\textit{on-line}, r_j|L_{\max}$, where all due dates are assumed to be non-positive when discussing competitive ratios. Gusfield [230] proves that the *difference* between the maximum lateness produced by an on-line heuristic and the minimum possible maximum lateness is at most $(2 - 1/m)p_{\max}$. This directly implies a weak bound of $3 - 1/m$ on the competitive ratio of the algorithm. Hall & Shmoys [234] observe that LS is 2-competitive even in the presence of precedence constraints. Vestjens [547] shows that no on-line algorithm can be better than (3/2)-competitive.

For problem $P|\textit{on-line}, r_j|\sum C_j$, Phillips, Stein & Wein [417] give a 3-competitive algorithm, which repeatedly converts some partial *pseudo-schedules* for problem $P|\textit{on-line}, r_j, \textit{pmtn}|\sum C_j$ into non-preemptive partial schedules by list scheduling jobs non-preemptively in the order of their completion times. Vestjens [547] shows that 1.309 is a lower bound on the competitive ratio of any on-line algorithm. For specific values of m, he gives improved lower bounds.

Using a dual ρ-approximation algorithm (see Section 6.1.5) as a subroutine, Hall, Shmoys & Wein [236] construct a so-called *greedy-interval* framework, which yields a 4ρ-competitive algorithm for minimizing the total weighted completion time. In particular, this results in $(4 + \varepsilon)$- and 8-competitive algorithms for problems $P|\textit{on-line}, r_j|\sum w_j C_j$ and $R|\textit{on-line}, r_j|\sum w_j C_j$, respectively. Improving and extending this on-line framework, Chakrabarti et al. [84] give a randomized on-line algorithm for bicriteria scheduling of identical machines. They show that, given a dual ρ-approximation algorithm for problem $P||\sum w_j C_j$, their algorithm yields a schedule that is simultaneously within a factor of 4ρ and 2.89ρ of the minimum makespan and total weighted completion time, respectively. For a very general class of scheduling models, Stein & Wein [500] provide a structural proof of the existence of schedules that are simultaneously within a factor of 2 of the minimum total weighted completion time and of the minimum makespan.

By using a preemptive one-machine relaxation, Chekuri, Motwani, Natarajan & Stein [89] provide a $(3 - 1/m)$-competitive algorithm for problem $P|\textit{on-line}, r_j|\sum C_j$. Further, they show that the algorithm can be improved to be 2.85-competitive by incorporating a modified greedy rule.

For uniform machines, Jaffe [283] presents an $O(\sqrt{m})$-competitive algorithm for problem $Q|\textit{on-line}, r_j, \textit{prec}|C_{\max}$. Since this algorithm is a variant of LS, it is non-clairvoyant and discussed later in Section 6.4.3. Based on a linear programming technique for estimating the speed at which each job should be run and another variant of the LS algorithm that can exploit this additional information, Chudak & Shmoys [110] give an $O(\log m)$-competitive algorithm. They also extend this result to an $O(\log m)$-competitive algorithm for problem $Q|\textit{on-line}, r_j, \textit{prec}|\sum w_j C_j$.

6.4.3 Non-Clairvoyant Scheduling

Observe that LS is a non-clairvoyant on-line scheduling algorithm. A classical result of Graham [225] yields that LS is $(2 - 1/m)$-competitive for $Pm|\textit{on-line-list-nclv}|C_{\max}$; LS is the best possible non-clairvoyant algorithm for this problem. Shmoys, Wein & Williamson [488] prove that even for the preemptive problem $Pm|\textit{on-line-list-nclv}, \textit{pmtn}|C_{\max}$, no better competitive ratio can be reached. LS also is $(2 - 1/m)$-competitive for all kinds of other variants with release dates, precedence constraints and preemption. As an interesting consequence, the usual fundamental difference between the preemptive and the non-preemptive models disappears when jobs are scheduled non-clairvoyantly on-line.

In terms of $\pi = \max_j p_j / \min_j p_j$, the competitive ratio of algorithm LS for problem $Pm|\textit{on-line-list-nclv}|C_{\max}$, which we denote by $R_m(\mathrm{LS})$, is derived by Achugbue & Chin [5]. If $\pi \leq 3$, then

$$R_m(\mathrm{LS}) = \begin{cases} 5/3 & \text{if } m = 3, 4, \\ 17/10 & \text{if } m = 5, \\ 2 - 1/(3\lfloor m/3 \rfloor) & \text{if } m \geq 6. \end{cases}$$

If $\pi \leq 2$, then

$$R_m(\mathrm{LS}) = \begin{cases} 3/2 & m = 2, 3, \\ 5/3 - 1/(3\lfloor m/2 \rfloor) & \text{if } m \geq 4. \end{cases}$$

The performance of LS can be significantly worse if the machines have different speeds. For problem $Q|\textit{on-line-list-nclv}|C_{\max}$, Liu & Liu [358, 359, 360] show that $R_m(\mathrm{LS}) = 1 + \sigma - \max_i s_i / \sum_i s_i$, where $\sigma = \max_i s_i / \min_i s_i$. This bound also holds in the presence of precedence constraints. Note that when all the speeds are equal, then this competitive ratio precisely reduces to Graham's result [225].

For problem $Q|$*on-line-list-nclv, prec*$|C_{\max}$, Jaffe [283] demonstrates that it is beneficial to selectively disregard the slow machines and to apply algorithm LS only to those machines whose speeds are within a factor of $\sqrt{m}$ of the fastest machine. Jaffe proves that this variant of LS is $(\sqrt{m}+O(m^{1/4}))$-competitive. This algorithm is shown to be asymptotically optimal by Davis & Jaffe [126], since a non-clairvoyant algorithm can never be better than $(\sqrt{m})$-competitive. In a similar vein, Davis & Jaffe develop a non-clairvoyant variant of LS for $R|$*on-line-list-nclv*$|C_{\max}$ and prove that it has a competitive ratio of $2.5\sqrt{m}+1+1/(2\sqrt{m})$.

Note that all the above non-clairvoyant scheduling results are applicable in the case that all jobs are released at the same time. Hence, we may invoke the general technique of Shmoys, Wein & Williamson [488] to convert a non-clairvoyant algorithm for scheduling jobs that arrive at the same time into a non-clairvoyant algorithm for scheduling jobs that arrive over time (see Section 6.4.2). This implies that all of these results can be carried over to the situation where jobs arrive over time, while losing a factor of at most 2 in the competitive ratios.

Shmoys, Wein & Williamson [488] discuss a variant of non-clairvoyant scheduling where job restarts are allowed. For problem $Q|$ *on-line-nclv*, $r_j|$ $C_{\max}$, they describe a $(4\log m+6)$-competitive on-line algorithm, which is the best possible up to a constant factor. For problem $R|$ *on-line-nclv*, $r_j|$ $C_{\max}$ (where for each job the relative speeds of the machines are known), they obtain an algorithm whose competitive ratio $8\log n+17$ depends on the number of jobs.

7 Multi-Stage Problems

In a *multi-stage* scheduling problem, the processing of each job is split into several *operations*, and each operation is assigned to one of s stages. For every stage, there is a corresponding machine type; operations in this stage can only be processed by machines of this type.

The three basic models of multi-stage scheduling (open shop, flow shop, and job shop) are introduced in Section 2.1. In the basic model, there is exactly *one* machine available for each stage. In an open shop and in a flow shop, each job j comprises s operations $O_{1j},\ldots,O_{sj}$, where operation O_{ij} can only be processed at stage i by machine type i. The processing time of operation O_{ij} is denoted by p_{ij}. In a job shop, each job j consists of a chain of operations. Every operation is assigned to a stage; the same stage may

occur several times in the chain. In the multiprocessor variants of a shop problem, for every stage i there are $\overline{m}_i$ identical machines that operate in parallel. In these variants, an operation at stage i may be executed by any of these $\overline{m}_i$ machines.

With respect to job precedence constraints, recall that that job j precedes job k demands that job k cannot start before job j has finished. In the context of multi-stage problems, this job dependence means that *none* of operations of job k can start before *all* operations of job j have finished. We note that there are different interpretations of precedence constraints for multi-stage problems, the main one of which is as follows: that job j has precedence over job k only requires that, on any machine, operations of job j have to finish before those of job k can start. In this chapter, we are only concerned with precedence constraints of the former type. For other types of precedence constraints, we refer the reader to Strusevich [508] as a starting point.

By an entry "$op = k$" or "$op \leq k$" in the second field of the scheduling notation, we denote the situation where every job consists of precisely k operations or at most k operations, respectively. Moreover, let $P_j = \sum_i p_{ij}$ be the total processing time or length of job j, and let $P_{\max} = \max_j P_j$ be the length of a longest job. Further, for the open shop and flow shop, Π_i denotes the total processing time at stage i, where $\Pi_i = \sum_j p_{ij}$, and $\Pi_{\max} = \max_i \Pi_i$.

7.1 The Open Shop

7.1.1 Complexity

Gonzalez & Sahni [215] show that $O2||C_{\max}$ allows a very simple polynomial time solution. An obvious lower bound on the makespan is

$$\max\{\Pi_1, \Pi_2, \max_j P_j\}.$$

They prove that there always exists a schedule with makespan that is equal to this lower bound, and that this schedule can be found in linear time. Since this lower bound is also a lower bound for the corresponding preemptive problem, the same algorithm also solves problem $O2|pmtn|C_{\max}$ in $O(n)$ time. Based on similar arguments and on extensive case distinctions, Shaklevich & Strusevich [484] develop linear time algorithms for the preemptive and non-preemptive scheduling of a two-machine open shop to minimize an arbitrary non-decreasing objective function of the two machine completion

times. In contrast, Sahni & Cho [460] prove strong $\mathcal{NP}$-hardness of the no-wait problem $O2|\textit{no-wait}|C_{\max}$ in which the second operation of each job must start immediately after the completion of its first operation.

Most remaining non-preemptive open shop problems are $\mathcal{NP}$-hard. For example, Gonzalez & Sahni [215] prove that problem $O3||C_{\max}$ is $\mathcal{NP}$-hard. However, deciding whether $O3||C_{\max}$ is strongly $\mathcal{NP}$-hard is an outstanding open problem. Open shop problems that are known to be strongly $\mathcal{NP}$-hard include $O||(C_{\max} \leq 4?)$ and hence $O||C_{\max}$ (Williamson et al. [559]), $O2|r_j|C_{\max}$ (Lawler, Lenstra & Rinnooy Kan [334]), $O2|\textit{tree}|C_{\max}$ (Lenstra [345]), $O2||L_{\max}$ (Lawler, Lenstra & Rinnooy Kan [334]), and $O2||\sum C_j$ (Achugbue & Chin [6]). Adiri & Aizikowitz [9] show that if some machine h dominates another machine i in the sense that $\min_j p_{kj} \geq \max_j p_{ij}$, then $O3||C_{\max}$ is solvable in polynomial time.

Many non-preemptive open shop problems are $\mathcal{NP}$-hard, even if every job comprises only two operations (with the other operations missing). Gonzalez & Sahni [215] prove that $O4|op = 2|C_{\max}$ is $\mathcal{NP}$-hard. The computational complexity of $O3|op = 2|C_{\max}$ is unknown. Note that the jobs in this problem are of three types (jobs that require stages 1 and 2, stages 1 and 3, and stages 2 and 3, respectively). Drobouchevitch & Strusevich [144] prove that if there are only jobs of two such types, then $O3|op \leq 2|C_{\max}$ is polynomially solvable.

Allowing preemption makes some open shop problems easier. For example, problem $O|\textit{pmtn}|C_{\max}$ can be solved in polynomial time by applying techniques from matching theory (Gonzalez & Sahni [215]). Lawler & Labetoulle [333] and Gonzalez [210] develop other network formulations and speed up the running time for solving this problem. Similar techniques are used by Lawler, Lenstra & Rinnooy Kan [334] to obtain linear time algorithms for problems $O2|\textit{pmtn}|L_{\max}$ and $O2|r_j, \textit{pmtn}|C_{\max}$. By combining a linear programming formulation with binary search, Cho & Sahni [106] derive polynomial time algorithms for problems $O|\textit{pmtn}|L_{\max}$ and $O|r_j, \textit{pmtn}|C_{\max}$. The algorithm of Cho & Sahni yields preemptive schedules that often *mix* the operations of jobs, i.e., one operation is preempted, and before this operation is resumed and completed, another operation that belongs to the same job is started and preempted, and so on. Interestingly, Cho & Sahni [106] prove that problem $O3|r_j, \textit{pmtn}|C_{\max}$ becomes $\mathcal{NP}$-hard if the mixing of operations is forbidden. Du & Leung [150] show that problem $O2|\textit{pmtn}|\sum C_j$ is $\mathcal{NP}$-hard, and Liu & Bulfin [356] establish that problem $O3|\textit{pmtn}|\sum C_j$ is strongly $\mathcal{NP}$-hard.

A number of polynomially solvable cases of open shop problems have

been identified. For problem $Om||C_{\max}$, it turns out that if the ratio $\Pi_{\max}/p_{\max}$ (i.e., the ratio between the largest machine load and the longest operation) is sufficiently large, then the problem becomes easy. More precisely, Fiala [168] applies deep results from graph theory together with so-called integer making techniques to prove that if $\Pi_{\max} \geq (16m' \log m' + 5m')p_{\max}$, where m' is the smallest power of 2 larger than m, then the optimal makespan is equal to $\Pi_{\max}$. Moreover, Fiala's result also yields a polynomial time algorithm for constructing an optimal schedule for this special case. By applying geometric methods (so-called *compact vector summation techniques*) for problem $O3||C_{\max}$, Sevastianov [478] proves that if $\Pi_{\max} \geq 7p_{\max}$, then the optimal makespan is equal to $\Pi_{\max}$. His results can be also translated into a polynomial time algorithm for this special case. We refer to Sevastianov [473, 476, 477] for several other results in the same spirit and for the corresponding references in the Soviet literature. We note that these techniques have been successfully applied to the approximation of flow shop problems to get absolute guarantees (see Section 7.2.4).

The open shop in which all operations have *unit processing times* tend to be much easier to solve than their unrestricted counterparts. Unit processing time problems are closely related to Latin squares and to Latin rectangles; various techniques such as those from edge coloring in bipartite graphs and from matching theory are useful. For example, Liu & Bulfin [357] show that problems $O|r_j, p_{ij} = 1|C_{\max}$, $O|p_{ij} = 1|L_{\max}$, $O|p_{ij} = 1|\sum T_j$ and $O|p_{ij} = 1|\sum U_j$ are solvable in polynomial time. For more information and numerous references, we refer the reader to the survey of Kubiak, Sriskandarajah & Zaras [312], and to the articles of Gonzalez [211], Liu & Bulfin [357], and Brucker, Jurisch & Jurisch [66].

We now consider multiprocessor variants. Problem $O2(P2, P1)||C_{\max}$ (and by symmetry $O2(P1, P2)||C_{\max}$) is easily seen to be $\mathcal{NP}$-hard, since it contains $P2||C_{\max}$ as a subproblem. It is unknown whether these two open shop problems are strongly $\mathcal{NP}$-hard. Lawler, Luby & Vazirani [336] prove that $O(P)|pmtn|C_{\max}$ is polynomially solvable. They also derive similar polynomial time algorithms for generalizations with uniform machines and with unrelated machines at each stage.

7.1.2 Enumerative Algorithms

In the design of enumerative and heuristic methods for problem $O||C_{\max}$, the following *disjunctive graph* formulation is useful. For each operation O_{ij}, there is a vertex with weight p_{ij}. There is an undirected edge correspond-

ing to each pair of operations that require the same job, and to each pair of operations that require the same machine. Choosing the order in which each job's operations are performed and the processing order on every machine corresponds to orienting the edges to produce a directed acyclic graph. Clearly, it is required to find an orientation that minimizes the length of a longest or *critical path*, where the length is defined as the sum of weights of vertices which lie on the path.

During the course of an algorithm, some of the disjunctive edges are oriented. For any operation O_{ij}, its *head* represents an earliest start time. For example, the length of a longest path in the graph containing oriented edges to a predecessor of the vertex corresponding to O_{ij} is a possible value for the head. Similarly, the *tail* for O_{ij} is the minimum time that must elapse between the completion of O_{ij} and the completion of the last operation. A possible value for the tail is the length of a longest path from a successor of the vertex corresponding to O_{ij}. Heads and tails are useful in the computation of lower bounds.

Brucker, Hurink, Jurish & Wöstmann [62] propose a branch and bound algorithm that relies heavily on the disjunctive graph formulation. They use a heuristic method to construct a feasible solution at each node of the search tree, and their branching rules is based on the following observations as to how the heuristic schedule can be improved. A critical path for the heuristic schedule contains *blocks*, where a block is a maximal set of vertices that each correspond to operations of the same job, or to operations that each require the same machine. To obtain an improved schedule, the jobs in some block must be reordered so that a different operation is sequenced either before all the other operations in the block or after all the other operations in the block. Thus, each branch of the search tree introduces precedence constraints to enforce such a reordering within some block. Lower bounds are computed by considering a subproblem for each of the machines. For a given machine, only the operations on that machine, together with their heads and tails are considered. The solution of problem $1|r_j, pmtn|L_{\max}$, where the release date is the head of the operation and the due date is minus the tail, provides a lower bound. Additional lower bounds are computed in a similar way for each job by considering all of its operations. Together with their heads and tails, scheduling these operations is equivalent to solving problem $1|r_j, pmtn|L_{\max}$. Computational results show that the algorithm is effective in solving instances with up to 10 jobs and 10 machines.

7.1.3 Local Search

There are relatively few studies on local search and other heuristics for problem $O||C_{\max}$. Taillard [521] presents detailed computational results for a tabu search algorithm, but does not provide details of his method. Bräsel, Tautenhahn & Werner [57] develop two types of heuristics. The first heuristic solves a sequence of bipartite matching problems, each of which is used to schedule $\min\{m, n\}$ operations, each belonging to a different job and requiring a different machine. Even though different objective functions for the matching problem are tested, the solution quality is poor. The second heuristic uses insertion. Specifically, all possible positions to insert the next operation are considered, and a position that yields the smallest makespan among the resulting partial schedules is selected. In some variants of the algorithm, several of the best positions are retained for further consideration. The operations are inserted in non-increasing order of their processing times, but with the added constraint that the first $\min\{m, n\}$ operations must be for different jobs and require different machines. Computational results for instances with $m = n$ show that the insertion heuristic often generates superior solutions to Taillard's tabu search method, and is therefore preferred.

7.1.4 Approximation: Ratio Guarantees

A feasible schedule for the open shop problem is called *dense* when any machine is idle if and only if there is no job which currently could be processed on that machine. This concept is introduced by Anná Racsmány (see Bárány & Fiala [40]) who observes for problem $O||C_{\max}$ that the makespan of any dense schedule is at most twice the optimal makespan. This result can also be derived as a corollary from a more general result of Aksjonov [15]. It is conjectured that, for every $m \geq 2$, the makespan of a dense schedule for problem $Om||C_{\max}$ is at most $2 - 1/m$ times the optimal makespan. Chen & Strusevich [93] prove this conjecture for $m \leq 3$.

Sevastianov & Woeginger [480] present a PTAS for problem $Om||C_{\max}$. It is unknown whether there exists an FPTAS for $Om||C_{\max}$. For the general problem $O||C_{\max}$, where the number of machines is part of the input, Williamson et al. [559] prove that the existence of a polynomial time approximation algorithm with worst-case ratio strictly less than $5/4$ would imply $\mathcal{P}$=$\mathcal{NP}$. This is a consequence of the fact that, unless $\mathcal{P}$=$\mathcal{NP}$, it is impossible to verify in polynomial time whether there exists a schedule of length 4

for an input where all processing times are integer. Williamson et al. also show there is a polynomial algorithm for determining whether there exists a schedule of length 3. Hoogeveen, Schuurman & Woeginger [263] prove that, unless $\mathcal{P}$=$\mathcal{NP}$, problem $O|\,|\sum C_j$ does not possess a PTAS.

For problem $O(P)|\,|C_{\max}$, Schuurman & Woeginger [471] give a simple 2-approximation algorithm that carries over the concept of dense schedules of Racsmány to the *multiprocessor* case. For problem $O2(P)|\,|C_{\max}$ with two stages, they give an improved approximation algorithm with a ratio guarantee of $3/2 + \varepsilon$. The PTAS of Sevastianov & Woeginger [480] can be generalized to problem $Os(Pm)|\,|C_{\max}$ in which there are a constant number of stages and a constant number of machines per stage.

Chen, Vestjens & Woeginger [97] consider on-line open shop problems. They provide approximation algorithms for problems $O2|$*on-line*$, r_j|C_{\max}$ and $O2|$*on-line*$, r_j,$ *pmtn*$|C_{\max}$ with competitive ratios of 3/2 and 5/4, respectively; both results are the best possible. They also show that, for problems $O2|$*on-line-nclv*$, r_j|C_{\max}$ and $O2|$*on-line-nclv*$, r_j,$ *pmtn*$|C_{\max}$, a greedy algorithm provides the best possible competitive ratio of 3/2 in each case. Chen and Woeginger [98] give a 1.875-competitive algorithm for problem $O2|$*on-line-list*$|C_{\max}$, and they show a lower bound of $(\sqrt{5}+1)/2 \approx 1.618$ on the competitive ratio of any algorithm for this problem. For problem $O2|$*on-line-list*$,$ *pmtn*$|C_{\max}$, they design a best possible (4/3)-competitive algorithm.

7.2 The Flow Shop

7.2.1 Complexity

A *permutation schedule* for a flow shop instance is a schedule in which each machine processes the jobs in the same order. Conway, Maxwell & Miller [114] show that, for any instance of problem $F|\,|C_{\max}$, there always exists an optimal schedule with the same processing order on the first two machines and with the same processing order on the last two machines. Consequently, if there are only two or three machines, then problem $F|\,|C_{\max}$ has an optimal solution that is a permutation schedule. An analogous statement does not hold for four machines: for two jobs with processing times $(4, 1, 1, 4)$ and $(1, 4, 4, 1)$, the optimal schedule has a makespan of 12 whereas the best permutation schedule has a makespan of 14. More generally, let $\phi(m)$ denote the least upper bound on the ratio between the makespan of the best permutation schedule and the makespan of the best unrestricted

schedule, where the ratio is taken over all instances of problem $Fm|\,|C_{\max}$. Röck & Schmidt [450] provide an algorithm that yields $\phi(m) \leq \lceil m/2 \rceil$. Potts, Shmoys & Williamson [429] construct a family of instances of problem $Fm|\,|C_{\max}$ for which $\phi(m) \geq \lceil \sqrt{m} + 1/2 \rceil / 2$. The exact growth rate of $\phi(m)$ is unknown.

In one of the first papers in the theory of scheduling, Johnson [286] demonstrates that problem $F2|\,|C_{\max}$ can be solved in $O(n \log n)$ time by the following sequencing rule: first schedule the jobs with $p_{1j} \leq p_{2j}$ in order of nondecreasing p_{1j}, and then schedule the remaining jobs in order of nonincreasing p_{2j}. Note that this rule produces a permutation schedule. Gonzalez & Sahni [215] observe that in the preemptive flow shop, preemptions on machine 1 and on machine m can be removed without increasing the makespan. Hence, Johnson's algorithm also solves problem $F2|pmtn|C_{\max}$ in $O(n \log n)$ time. On the other hand, Garey, Johnson & Sethi [190] show that problem $F3|\,|C_{\max}$ is strongly $\mathcal{NP}$-hard. As an immediate consequence, finding the best permutation schedule for problem $Fm|\,|C_{\max}$, for $m \geq 3$, is also strongly $\mathcal{NP}$-hard. There are several polynomially solvable special cases of problem $Fm|\,|C_{\max}$ that result from imposing certain inequalities on the processing times. For example, Johnson [286] observes that if $\max_j p_{2j} \leq \max\{\min_j p_{1j}, \min_j p_{3j}\}$ holds in an instance of problem $F3|\,|C_{\max}$, then the second machine is non-bottleneck, and the optimal algorithm for problem $F2|\,|C_{\max}$ can be suitably adapted. Monma & Rinnooy Kan [384] provide a survey of these types of results.

The following non-preemptive flow-shop problems are all strongly $\mathcal{NP}$-hard: $F2|r_j|C_{\max}$, $F2|\,|L_{\max}$, and $F2|tree|C_{\max}$ (Lenstra, Rinnooy Kan & Brucker [349]); and $F2|\,|\sum C_j$ (Garey, Johnson & Sethi [190]). In fact, $F2|\,|\sum C_j$ is strongly $\mathcal{NP}$-hard even if all operations on the first machine have the same processing time (Hoogeveen & Kawaguchi [259]). Also, the following preemptive flow-shop problems are all strongly $\mathcal{NP}$-hard: $F2|r_j, pmtn|C_{\max}$ and $F2|pmtn|L_{\max}$ (Cho & Sahni [106]); $F3|pmtn|C_{\max}$ (Gonzalez & Sahni [215]); and $F2|pmtn|\sum C_j$ (Du & Leung [150]).

Shaklevich, Hoogeveen & Pinedo [482] consider *proportionate* flow shops, where for every job j all of its operations O_{ij} have the same processing time p_j. For regular objective functions in proportionate flow shops, they show that an optimal solution can always be found among permutation schedules. This property yields, for many objective functions, polynomial time algorithms that are based on sorting. For example, problem $F|\,|C_{\max}$ can be solved in $O(n \log n)$ time in a proportionate flow shop. As a not-so-straightforward result, they show that, in a proportionate flow shop, problem

$F||\sum w_jC_j$ is solvable in $O(n^2)$ time by a greedy approach.

A prominent variant of the flow shop imposes a *no-wait* constraint. In this variant, once a job has started, it has to be processed without interruption, operation by operation, until it is completed. This variant has many applications. For example, in systems without intermediate storage between the machines, and in the chemical industry where jobs have to be processed at a continuously high temperature. Hall & Sriskandarajah [239] provide a thorough survey of complexity and algorithms for no-wait scheduling. For problem $F|$*no-wait*$|C_{\max}$, minimizing makespan can be modeled as a special case of the traveling salesman problem (see, for example, Wismer [560]). Since the distance matrix of the resulting traveling salesman problem has a special combinatorial structure, the famous subtour patching technique of Gilmore & Gomory [201] yields an $O(n\log n)$ time algorithm for problem $F2|$*no-wait*$|C_{\max}$. Rote & Woeginger [451] show that the time complexity of $O(n\log n)$ is best possible. With a trade-off between running time and optimality, they also develop an $O(n\log(1/\varepsilon))$ time FPTAS. The formulation of $F2|$*no-wait*$|C_{\max}$ as a traveling salesman problem heavily exploits the property that every job consists of two operations. If some of the jobs only have to be processed on the first machine, this formulation breaks down. In fact, Sahni & Cho [460] show that this variant with missing operations is strongly $\mathcal{NP}$-hard. Papadimitriou & Kannelakis [414] prove that problem $F4|$*no-wait*$|C_{\max}$ is strongly $\mathcal{NP}$-hard. Röck establishes that problem $F3|$*no-wait*$|C_{\max}$ is also strongly $\mathcal{NP}$-hard [448], and so are problems $F2|$*no-wait*$|L_{\max}$ and $F2|$*no-wait*$|\sum C_j$ (Röck [449]).

We now consider multiprocessor variants. Problems $F2(P2,P1)||C_{\max}$ and $F2(P1,P2)||C_{\max}$ are easily seen to be $\mathcal{NP}$-hard, since they contain $P2||C_{\max}$ as a subproblem. Hoogeveen, Lenstra & Veltman [261] show that these problems are even strongly $\mathcal{NP}$-hard, and so are the preemptive problems $F2(P2,P1)|$*pmtn*$|C_{\max}$ and $F2(P1,P2)|$*pmtn*$|C_{\max}$.

7.2.2 Enumerative Algorithms

Most of the literature on branch and bound for flow shops concentrates of finding the best permutation schedule. The traditional approach is to employ a forward sequencing branching rule, and to obtain lower bounds by solving subproblems that are often obtained through a relaxation of capacity constraints on selected machines. Typically, a single- or two-machine subproblem is selected, since larger subproblems are $\mathcal{NP}$-hard in general.

For the permutation flow shop $F||C_{\max}$, Ignall & Schrage [280] propose a

machine-based bound. Let T_i denotes the time that machine i $(i = 1, \ldots, m)$ completes processing an initial partial sequence of jobs that are fixed. If S is the set of unsequenced jobs, then the lower bound for machine i is equal to the earliest time that machine i can complete all of its processing plus the minimum time to process the last job on machines $i+1, \ldots, m$:

$$T_i + \sum_{j \in S} p_{ij} + \min_{j \in S} \sum_{h=i+1}^{m} p_{hj}.$$

McMahon & Burton [376] develop a *job-based bound* that can be used in combination with the machine-based bound. However, each of these bounds is dominated by the *two-machine bounds* that are developed independently by Lageweg, Lenstra & Rinnooy Kan [318] and Potts [421]. For machines h and i, where $h < i$, the two-machine bound is given by

$$T_h + M_{h,i}(S) + \min_{j \in S} \sum_{h=i+1}^{m} p_{hj},$$

where $M_{h,i}(S)$ is the makespan for the problem in which jobs of the set S are processed on machines $h, \ldots, i$, and the capacity constraints on machines $h+1, \ldots, i-1$ are relaxed so that they become non-bottleneck machines. Recall that $M_{h,i}(S)$ is computed by applying an adapted version of the algorithm of Johnson [286] to sequence the jobs.

Branch and bound algorithms for the permutation flow shop $F|\,|C_{\max}$ with a forward sequencing branching rule benefit from dominance rules to prune the search tree. These rules typically eliminate some job j from the list of candidates to be appended to the current initial partial sequence if certain conditions are satisfied. The most useful of these rules are developed by Gupta & Reddi [205], McMahon [375] and Szwarc [509, 510, 511].

Problem $F|\,|C_{\max}$ is *reversible*, which means that an equivalent problem results if the jobs pass through the machines in the order $m, \ldots, 1$. This property leads Potts [422] to propose a branching rule for the permutation flow shop $F|\,|C_{\max}$ that allows both forward and backward sequencing, so that each node of the search tree defines an initial and a final partial sequence. Grabowski [221] adopts a branching rule that uses the block approach (see Section 7.1.2).

In spite of the various enhancements to the basic branch and bound algorithms for the permutation flow shop $F|\,|C_{\max}$, the performance remains unsatisfactory. The algorithm of Potts [422] appears to be the most effective,

although it experiences difficulty in solving moderately sized instances with 15 jobs and 4 machines.

Hariri & Potts [241] develop a branch and bound algorithms for problem $F2|prec|C_{\max}$, where job j having precedence over job k is interpreted as constraining j to be sequenced before k on both machines. They use Lagrangean relaxation to derive a generalized job-based bound, and test three branching rules. Computational results indicate that their best algorithm is reasonably effective in solving instances with up to 80 jobs.

For minimizing maximum lateness in a permutation flow shop, branch and bound algorithms are proposed for problem $F2||L_{\max}$ by Townsend [527], for problem $F2|\ r_j|\ L_{\max}$ by Grabowski [220], and for problem $F|\ r_j|\ L_{\max}$ by Grabowski, Skubalska & Smutnicki [224], where the two latter algorithms are based on the block approach. Tadei, Gupta, Dela Croce & Cortesi [519] develop branch and bound algorithms for problem $F2|r_j|C_{\max}$, which are also applicable to the reverse problem $F2||L_{\max}$. Their algorithms include improved lower bounds and several dominance rules. In a computational comparison between a forward branching rule and a block based rule, they find that the forward branching rule is preferred and allows instances with up to 80 jobs to be solved effectively.

For problem $F2||\sum C_j$, branch and bound algorithms are developed by Ignall & Schrage [280], Ahmadi & Bagchi [12], Van de Velde [536] and Della Croce, Narayan & Tadei [129]. Ignall & Schrage use machine-based bounds, each of which is computed by sequencing the jobs in SPT order of their processing times on the relevant machine. Ahmadi & Bagchi develop a stronger lower bound for the second machine by considering a subproblem of the type $1|r_j|\sum C_j$, where the release dates are equal to the processing times on the first machine. They obtain their bound by solving the relaxed problem $1|r_j, pmtn|\sum C_j$ using the SRPT rule. Van de Velde's bound is obtained by performing a Lagrangean relaxation of the constraints that specify that a job cannot start on the second machine until its processing is completed on the first machine. He also derives an improvement to this lower bound. Hoogeveen & Van de Velde [266] show how Van de Velde's original lower bound can be strengthened using their slack variable technique (see Section 4.2.2). Della Croce, Narayan & Tadei derive improvements to Ignall & Schrage's lower bound for the first machine, and to Ahmadi & Bagchi's bound for the second machine. They also perform computational tests to assess the performance of the various lower bounds in a branch and bound algorithm. The algorithm uses a forward sequencing branching rule, includes dominance rules and uses a descent heuristic to obtain an initial

upper bound. They find that Van de Velde's bound is the most effective, and the algorithm is successful in solving instances with up to 25 jobs.

Hariri & Potts [241] develop a branch and bound algorithm for the permutation flow shop problem $F||\sum U_j$. They first observe that single machine subproblems of the type $1||\sum U_j$ can be formed and then solved using the algorithm of Moore [385] (see Section 4.4.1). Two procedures for improving these initial lower bounds are then proposed. The branch and bound algorithm, which uses a forward sequencing branching rule, is reasonably effective in solving instances with up to 15 jobs, or up to 20 jobs if there are only two or three machines.

7.2.3 Local Search

Research on heuristics and local search methods for the flow shop has focused mainly on finding permutation schedules. For the permutation flow shop $F||C_{\max}$, Campbell, Dudek & Smith [78] suggest aggregating machines to produce an instance of problem $F2||C_{\max}$, which is solved by the algorithm of Johnson [286]. However, among the heuristics for the permutation flow shop $F||C_{\max}$ that do not employ local search, an $O(mn^2)$ insertion method of Nawaz, Enscore & Ham [395] is the best (see Taillard [520] for a derivation of its time complexity). This heuristic builds a sequence by repeatedly inserting an unscheduled job with the largest total processing time into the best position of the current partial sequence. Rather surprisingly, it provides better quality solutions than those given by a descent method that uses the transpose neighborhood, as proposed by Dannenbring [120].

Various neighborhood search methods are available for the permutation flow shop $F||C_{\max}$. In independent studies on simulated annealing, Osman & Potts [409] and Ogbu & Smith [407] both find that the insert neighborhood is preferred to the swap neighborhood. The same conclusion is also reached for tabu search by Taillard [520] who uses the insert neighborhood to improve upon a previous result of Widmer & Hertz [557] that employs the swap neighborhood.

The more recently developed neighborhood search algorithms for the permutation flow shop $F||C_{\max}$ include special devices to improve performance. For example, Reeves [445] develops a tabu search algorithm that uses randomly generated subsets of the insert neighborhood, rather than the complete neighborhood. Also, Nowicki & Smutnicki [401] use a restricted version of the insert neighborhood. More precisely, the block structure of some critical path (as defined in Section 7.1.2) is used to remove from con-

sideration any neighbors that cannot improve upon the makespan of the current sequence. Another special feature of their algorithm is a backtracking procedure which allows the search to return to a previously generated best solution, but proceeds to a different neighborhood move to that made previously. Further, Werner [556] proposes a class of 'path' algorithms for the permutation flow shop $F||C_{\max}$. Each algorithm can be viewed as descent with the insert neighborhood, combined with exploration and backtracking.

Reeves [446] proposes a genetic algorithm which uses the *reorder crossover* in which jobs in the section of the string between the crossover points are reordered according to their relative positions in the other solution.

Some comparative results between local search algorithms for the permutation flow shop $F||C_{\max}$ are contained in the respective papers. Further evaluations can be made since several of these algorithms are tested on the instances generated by Taillard [521]. The tabu search algorithms of Reeves [445] and Nowicki & Smutnicki [402] generate better quality solutions than the other methods. However, the current champion is the algorithm of Nowicki & Smutnicki.

We now discuss some studies with different optimality criteria. Kohler & Steiglitz [301] compare different neighborhoods in descent algorithms for problem $F2||\sum C_j$, while Krone & Steiglitz [309] propose a descent algorithm for problem $F||\sum C_j$ in which permutation schedules are not assumed. For the permutation flow shop $F||\sum w_j C_j$, Glass & Potts [203] perform a computational comparison of multi-start descent, simulated annealing, threshold accepting, tabu search, and two genetic algorithms, one of which applies descent to each solution in every population. The neighborhood search algorithms each use the swap neighborhood which performs marginally better than insert in initial experiments. Simulated annealing and the genetic algorithm that incorporates descent generate the best quality solutions, and the latter method is slightly superior.

Kim [298] and Adenso-Días [8] propose tabu search algorithms for $F||\sum T_j$ and $F||\sum w_j T_j$, respectively. Kim uses the swap neighborhood, whereas Adenso-Días uses a combination of a restricted swap and a restricted insert neighborhood.

7.2.4 Approximation: Absolute Guarantees

In this section, we only deal with makespan minimization in a flow shop. One approach to problem $Fm||C_{\max}$ is based on a result of 1913 by Steinitz [503],

which was discovered independently by at least three groups of researchers in the 1970s: by Belov & Stolin in Kharkov [49], by Sevastianov in Novosibirsk [473], and by Bárány & Fiala in Budapest [39, 40]. Of these, only the work of Bárány & Fiala [39] is accessible to Western researchers. We refer the reader to a comprehensive survey article of Sevastianov [476]. All of these results show, with different precisions, that the optimum makespan cannot be too far away from the lower bound $\Pi_{\max}$.

Typical of the absolute performance guarantees is the following result of Sevastianov [473] and Bárány & Fiala [40]. "For any instance of $Fm||C_{\max}$, there exists a permutation schedule with makespan at most $\Pi_{\max} + m(m-1)p_{\max}$. Moreover, such a permutation schedule can be found in polynomial time." The critical parameter in this statement is the term $m(m-1)$. Belov & Stolin [49] proved a weaker term of order $\Theta(m^{5/2})$. Sevastianov [477] provides a small improvement to $(m-1)(m-2+1/(m-2))$. For the special case $F3||C_{\max}$, Sevastianov [475] shows that there always exists a permutation schedule with makespan bounded by $\Pi_{\max} + 3p_{\max}$, and that the factor of 3 is best possible. Similar results are also known for the job shop problem. We do not discuss these results, but instead refer the reader to the survey of Sevastianov [476].

7.2.5 Approximation: Ratio Guarantees

A feasible schedule is *active* when any machine is idle if and only if there is no job which currently could be processed on that machine without delaying another operation. Gonzalez & Sahni [216] show that for the permutation flow shop, the makespan of any active schedule is at most a factor of m away from the optimal makespan. Gonzalez & Sahni also present an $O(mn \log n)$ time approximation algorithm by solving $\lceil m/2 \rceil$ two-machine flow shop subproblems optimally. An alternative machine aggregation approach is described by Nowicki & Smutnicki [399] and by Röck & Schmidt [450] that reduces the original problem to an artificial two-machine flow shop problem for which an optimal permutation defines an approximate permutation schedule. Such a schedule can be found in $O(mn + n \log n)$ time. These approximation algorithms have a worst-case ratio of $\lceil m/2 \rceil$.

Potts [427] investigates the performance of five polynomial time approximation algorithms for problem $F2|r_j|C_{\max}$. The best one of these involves the repeated application of a dynamic variant of Johnson's algorithm [286] to modified versions of the problem, and has a worst-case ratio of 5/3. Hall [231] derives a PTAS for problem $F2|r_j|C_{\max}$. The strongest known

result for makespan minimization in flow shops is a PTAS for $Fm|r_j|C_{\max}$ (Hall [232]). Hall's PTAS can be modified to handle the case of finding the best permutation schedule for problem $Fm|r_j|C_{\max}$.

For the general problem $F||C_{\max}$ (where the number of machines is part of the input), Williamson et al. [559] prove that the existence of a polynomial time approximation algorithm with worst-case ratio strictly less than $5/4$ would imply $\mathcal{P}=\mathcal{NP}$. Shmoys, Stein & Wein [486] construct randomized approximation algorithms with a worst-case ratio of $O(\log^2 m/\log\log m)$ for problem $F||C_{\max}$. Schmidt, Siegel & Srinivasan [463] show how the same approximation ratio can be achieved with a deterministic algorithm. Deciding whether there exists an approximation algorithm with constant worst-case ratio for problem $F||C_{\max}$ is an outstanding open problem. The approximability behaviour of finding the best permutation schedule for problem $F||C_{\max}$ is completely unclear. We do not even know how to exclude the existence of a PTAS.

Neumytov & Sevastianov [397] investigate the special case $F|op = 2|C_{\max}$, where every job goes through only two stages, one of which is stage 1. They prove that this restricted problem is $\mathcal{NP}$-hard for $s \geq 3$ stages. Drobouchevitch & Strusevich [143] derive a polynomial time approximation algorithm with a worst-case guarantee of $3/2$ for this special case.

For the *multiprocessor* problem $F2(P)||C_{\max}$, Schuurman & Woeginger [470] design a PTAS. The PTAS of Hall [232] can also be generalized to problem $Fs(Pm)||C_{\max}$, i.e., to the variant with a constant number of stages and a constant number of machines per stage. The approximability status of $F3(P)||C_{\max}$ is unknown. The known approximability results for makespan minimization in a multiprocessor flow shop are summarized in Table 1.

Goyal & Sriskandarajah [219] survey approximation results for the *no-wait* flow shop. Glass, Gupta & Potts [202] investigate the $\mathcal{NP}$-hard version of problem $F2|\textit{no-wait}|C_{\max}$ with missing operations on the second machine, and derive an approximation algorithm with a worst-case ratio of $4/3$. Schulz [468] describes a polynomial time approximation algorithm with a worst-case ratio of $2m+1$ for problem $Fm|r_j, \textit{prec}|\sum w_jC_j$. This result is based on an LP-formulation and can also be generalized to problem $F|r_j, \textit{prec}, \textit{no-wait}|\sum w_jC_j$. Hoogeveen, Schuurman & Woeginger [263] prove that, unless $\mathcal{P}=\mathcal{NP}$, problem $F||\sum C_j$ does not possess a PTAS.

		Number of machines per stage		
		=1	constant	arbitrary
Number of stages	=2	poly-time	PTAS	PTAS
	const ≥ 3	PTAS	PTAS	open
	arbitrary	$\nexists$ PTAS	$\nexists$ PTAS	$\nexists$ PTAS

Table 1: The approximability of makespan minimization in a *multiprocessor flow shop*. The entry "poly-time" stands for polynomially solvable, an entry "PTAS" stands for the existence of a PTAS, and an entry "open" means that the approximability is unknown.

7.3 The Job Shop

A job shop in which every job is processed at most once on any machine is called *acyclic*. In literature of job shop scheduling, *machine repetition* is usually not allowed, i.e., consecutive operations of the same job must always be assigned to different machines. We follow this convention unless stated otherwise.

In this subsection we use the following additional job characteristics in the second field.

- $\{\circ, n = k, n \leq k, op \leq \ell, rep, acyc\}$:
  - $\circ$: no special multi-stage restrictions;
  - $n = k$: there are exactly k jobs;
  - $n \leq k$: there are at most k jobs;
  - $op \leq \ell$: there are at most ℓ operations per job;
  - *rep*: machine repetition is allowed;
  - *acyc*: every job is processed at most once on any machine.

7.3.1 Complexity

Apparently the job shop problem was formulated and investigated for the first time by Akers & Friedman [14]. In fact, in the Soviet literature, the job shop problem is usually called the *Akers-Friedman-problem* or *AF-problem* for short. Since the job shop is a generalization of the flow shop, all negative complexity results that are stated in Section 7.2.1 for the flow shop also apply to the job shop. In fact, we only know of two polynomially solvable cases

		Number m of machines			
		=2	=3	constant	arbitrary
Number n of jobs	=2	$\mathcal{P}$	$\mathcal{P}$	$\mathcal{P}$	$\mathcal{P}$
	=3	$\mathcal{P}$	$\mathcal{NP}$-hard	$\mathcal{NP}$-hard	$\mathcal{NP}$-hard
	constant	$\mathcal{P}$	$\mathcal{NP}$-hard	$\mathcal{NP}$-hard	$\mathcal{NP}$-hard
	arbitrary	$\mathcal{NP}$-hard	$\mathcal{NP}$-hard	$\mathcal{NP}$-hard	$\mathcal{NP}$-hard

Table 2: The computational complexity of makespan minimization in a *non-preemptive* job shop with m stages and n jobs, where an entry "$\mathcal{P}$" stands for polynomially solvable.

of the job shop problem with an unrestricted number of jobs. First, problem $J2|op \leq 2|C_{\max}$ is solvable in $O(n \log n)$ time by a simple extension by Jackson [282] of Johnson's algorithm for problem $F2||C_{\max}$. Second, problem $J2|p_{ij} = 1|C_{\max}$ can be solved in $O(n)$ time by an algorithm of Hefetz & Adiri [248]. However, if we deviate slightly from these two special cases, then we immediately encounter $\mathcal{NP}$-hard problems: $J2|op \leq 3|C_{\max}$ and $J3|op \leq 2|C_{\max}$ are $\mathcal{NP}$-hard (Lenstra, Rinnooy Kan & Brucker [349]); and problems $J2|p_{ij} \in \{1,2\}|C_{\max}$, $J3|p_{ij} = 1|C_{\max}$ and $J2|p_{ij} = 1, rep|C_{\max}$ are all strongly $\mathcal{NP}$-hard (Lenstra & Rinnooy Kan [347]). Finally, Sahni & Cho [460] show that problem $J2|op \leq 2, no\text{-}wait|C_{\max}$ is strongly $\mathcal{NP}$-hard, i.e., the variant where immediately after the completion of the first operation of a job the processing of the second operation of this job must start.

Another set of polynomially solvable cases of the job shop problem arises from restricting the number of jobs in the instance. The classical result in this area is due to Akers [13]: he shows that problem $J|n = 2|C_{\max}$ can be formulated as a shortest path problem among rectilinear obstacles in the plane, and hence derives a polynomial time algorithm. The approach of Akers also applies to problem $J|n = 2, rep|C_{\max}$. Based on ideas of Kravchenko & Sotskov [307], Brucker [61] derives a polynomial time algorithm for $J2|n = k|C_{\max}$. All other variants of makespan minimization problem are $\mathcal{NP}$-hard due to a breakthrough result of Sotskov & Shaklevich [497], which states that even $J3|n = 3|C_{\max}$ is $\mathcal{NP}$-hard. These complexity results are summarized in Table 2.

Somewhat surprisingly, the preemptive version of the job shop prob-

lem is even harder than its non-preemptive version. Essentially, the only polynomial solvability result known in this area comes from carrying over the shortest path formulation of Akers [13] to $J|n = 2, pmtn|C_{\max}$ and to $J|n = 2, pmtn, rep|C_{\max}$. For more jobs, problem $J2|n = 3, pmtn|C_{\max}$ is already $\mathcal{NP}$-hard (Brucker, Kravchenko & Sotskov [70]).

For objective functions other than the makespan, the only known polynomial solvability results are for problems $J2|n \leq k|\sum f(C_j)$ and $J2|n \leq k|\max f(C_j)$, where f is a regular function of the job completion times (Brucker, Kravchenko & Sotskov [69]).

7.3.2 Enumerative Algorithms

Most of the research on branch and bound algorithms for the job shop is for minimizing the makespan. As in Section 7.1.2 for the open shop, the disjunctive graph model is again useful for enumerative and heuristic methods for the job shop problem $J||C_{\max}$. In this case, edges between operations of the same job are directed, thereby reflecting the order of these operations. It is required to orient the undirected edges are for each pair of operations that require the same machines.

In early approaches by Németi [396], Charlton & Death [88] and Schrage [465] for solving problem $J||C_{\max}$ by branch and bound, a lower bound is obtained by disregarding edges that are not oriented in the disjunctive graph formulation, and then finding the length of a longest path. However, a much stronger lower bound is obtained by solving single machine subproblems, as observed by Bratley, Florian & Robillard [58] and McMahon & Florian [377]. For a given machine, the corresponding operations, together with their heads and tails, define an instance of problem $1|r_j|L_{\max}$, where the release date is the head of the operation and the due date is minus the tail. The solution value of this problem provides a lower bound. Thus, the results of Section 4.1 are useful. In particular, the single machine problem can be solved by the efficient branch and bound algorithm of Carlier [79], or alternatively the preemptive relaxation, namely problem $1|r_j, pmtn|L_{\max}$, can be solved using the generalized EDD algorithm. Lageweg, Lenstra & Rinnooy Kan [317] observe that a slight strengthening of the lower bound is possible by considering problem $1|r_j, prec|L_{\max}$, where the precedence constraints between operations on the selected machine are obtained from the oriented edges in the disjunctive graph. Brucker & Jurisch [65] develop an alternative lower bound using a two-job relaxation. It appears to be useful for problems in which the number of machines exceeds the number of jobs.

Other approaches for obtaining lower bounds have not led to improved branch and bound algorithms, generally because the quality of the lower bound is not sufficient to justify a very high investment in computation time. Examples include the surrogate duality relaxations of Fisher, Lageweg, Lenstra & Rinnooy Kan [173], and the linear programming relaxations of Balas [36], Applegate & Cook [24] and Martin & Shmoys [368].

Three main types of branching rules are used in branch and bound algorithms for problem $J||C_{\max}$. First, a forward branching rule can be used to generate active schedules, as suggested by Giffler & Thompson [200]. Second, a binary branching rule can be used to orient an edge one way or the other in the disjunctive graph formulation, as proposed by Lageweg, Lenstra & Rinnooy Kan [317]. There are various methods for selecting the edge that is used for branching. Third, using a block approach (see Sections 7.1.2 and 7.2.2), Barker & McMahon [41] suggest a branching rule that introduces precedence constraints that either force one operation of a block to be a predecessor or a successor of all other operations in this block.

Dominance rules are frequently applied to orient some of the disjunctive arcs. Any feasible solution provides an upper bound on the makespan, from which deadlines for individual operations can be computed. If it can be shown that orienting a disjunctive edge in a particular direction leads to a deadline violation, then for subsequent computations this arc can be oriented in the reverse direction. Such an orientation is known as *immediate selection*. Orienting disjunctive arcs may lead to increased values for heads and tails, which may in turn provide better lower bounds.

We now describe the features of some of the more successful branch and bound algorithms for problem $J||C_{\max}$. McMahon & Florian [377] use a forward active schedule branching rule, and solve problems $1|r_j|L_{\max}$ (with their own algorithm) to obtain lower bounds. Using the same lower bound, Barker & McMahon [41] design an algorithm that uses a block based branching rule. Lageweg, Lenstra & Rinnooy Kan [317] use a binary edge orientation branching rule, and obtain lower bounds by solving problems $1|r_j, prec|L_{\max}$. Carlier & Pinson [81] also use a binary edge orientation branching rule (but with a different method for selecting edges), and compute lower bounds by solving problems $1|r_j, pmtn, prec|L_{\max}$, after first applying immediate selection rules. In follow-up studies, Carlier & Pinson [82, 83], Applegate & Cook [24], and Brucker, Jurish & Krämer [67] develop more effective procedures for immediate selection, and adjusting heads and tails, thereby yielding improved branch and bound algorithms. Brucker, Jurish & Sievers [68] develop a branch and bound algorithm that uses a

block based branching rule, applies immediate selection, and computes lower bounds by solving problems $1|r_j, pmtn, prec|L_{\max}$. Martin & Shmoys [368] introduce more sophisticated techniques for adjusting heads and tails that lead to tighter lower bounds, and they propose two new branching rules.

For problem $J||C_{\max}$, there are several test instances in the literature including those of Fisher & Thompson [171], Lawrence [339], Adams, Balas & Zawack [7] and Taillard [521]. The most famous has 10 jobs and 10 machines, and remained unsolved for many years until Carier & Pinson [81] obtained an optimal solution. Even though the most recent algorithms now solve this problem without much difficulty, there are instances with 15 jobs and 15 machines that cannot be solved by currently available algorithms using reasonable amounts of computation time.

7.3.3 Local Search

Many heuristics are based on the use of priority rules, which are surveyed by Haupt [247]. These approaches use a priority rule to select an operation from a set of candidates to be sequenced next. The candidates may be chosen to create a non-delay schedule (no machine idle time is allowed if operations are available to be processed), an active schedule, or a limited delay schedule. Although priority rule heuristics are undemanding in their computational requirements, the quality of schedules that are generated tends to be erratic.

An effective heuristic approach is the *shifting bottleneck* procedure of Adams, Balas & Zawack [7]. This procedure constructs a schedule by selecting each machine in turn and orienting all of the corresponding edges in the disjunctive graph formulation. To orient these edges, a problem $1|r_j|L_{\max}$ for a selected (bottleneck) machine is solved by the branch and bound algorithm of Carlier [79], where previously oriented edges are used in the computation of release dates and due dates. Applegate & Cook [24] propose some improvements to the original shifting bottleneck procedure. Dauzère-Pérès & Lasserre [121] observe that the current orientation of some edges in the shifting bottleneck procedure may create a path between two operations which require the same machine. In this case, the minimum time delay between the start times of these operations can be computed. Thus, the $1|r_j|L_{\max}$ problems that are considered within the shifting bottleneck procedure should ideally incorporate *delayed precedence constraints* to account for these delays. Dauzère-Pérès & Lasserre use a heuristic approach for these single machine problems with delayed precedence constraints, whereas Balas, Lenstra & Vazacopoulos [37] obtain an exact solution by designing a

generalized version of Carlier's algorithm.

Various local search algorithms for $J||C_{\max}$ have been proposed, and they are reviewed by Vaessens, Aarts & Lenstra [532]. For neighborhood search, the block structure corresponding to some critical path again plays a key role. An improved schedule can only be obtained if, for at least one block, a different operation is sequenced either before all the other operations in the block or after all the other operations in the block. For the *critical transpose* neighborhood, two adjacent operations in a block are transposed; however, *critical end transpose* restricts the transpositions to either the first pair or the last pair of operations in a block. Similarly, for the *critical end insert* neighborhood, an operation in a block is inserted in the first or in the last position of the block.

Simulated annealing algorithms for problem $J||C_{\max}$ are proposed by Matsuo, Suh & Sullivan [373], Van Laarhoven, Aarts & Lenstra [539] and Yamada, Rosen & Nakano [565]. Matsuo, Suh & Sullivan use an extension of the critical end transpose neighborhood in which transposes of the predecessors and successors of one of the originally transposed operations are explored. The larger critical transpose neighborhood is adopted by Van Laarhoven, Aarts & Lenstra. Yamada, Rosen & Nakano [565] use the critical end insert neighborhood, and allow the possibility of backtracking to the best schedule that is currently generated.

Tabu search provides an attractive alternative to simulated annealing for problem $J||C_{\max}$. Taillard [522] use the critical transpose neighborhood. Special features of his method are the replacement of exact evaluations of the makespan of each neighbor by quickly computed lower bound estimates, and a randomly changing length of the tabu list. Barnes & Chambers [43] build on Taillard's method by including a backtracking device. Dell'Amico & Trubian [134] use a composite neighborhood consisting of generalized critical end transpose, which allows the reordering of three critical operations (one of which must start or end a block), and of critical end insert. Their algorithm adopts Tailard's ideas of selecting moves according to quickly computed lower bounds for the makespan, and of using a variable length tabu list. Nowicki & Smutnicki [400] use the critical end transpose neighborhood, and allow backtracking, as in their algorithm for the permutation flow shop $F||C_{\max}$.

Balas & Vazacopoulos [38] propose a *guided local search* procedure, which resembles tabu search based on the critical end insert neighborhood with a backtracking procedure. They also suggest several hybrids in which guided local search is embedded in the shifting bottleneck procedure.

Genetic algorithms for problem $J||C_{\max}$ often have special representations of solutions which require some type of heuristic to covert the representation into a schedule. Storer, Wu & Vaccari [505] propose a data perturbation representation, and a schedule is constructed from the perturbed data by using SPT as a priority rule to construct a limited delay schedule. They also suggest a heuristic set representation in which each string entry defines a priority rule that is used within a particular time period when limited delay scheduling is applied. Similar heuristic set approaches are used by Dorndorf & Pesch [142] and Smith [494]. Della Croce, Tadei & Volta [130] propose a priority representation in which each machine has a sequence that defines a priority order that is used in limited delay scheduling. A similar approach is used by Yamada & Nakano [564] who use completion times to define priorities. Pesch [416] uses a representation that defines the order in which two-job subproblems are solved when building a schedule. He also considers representations that give upper bounds on the solution of two-job or single-machine subproblems that allow some edges in the disjunctive graph to be oriented prior to applying a heuristic method. Nakano & Yamada [394] use an ordered pair representation that defines the relative order of each pair of jobs on each machine. Using a similar approach, Aarts, Van Laarhoven, Lenstra & Ulder [2] use a representation based on the relative order of adjacent jobs in a block.

Several of the algorithms for the job shop problem $J||C_{\max}$ are tested on the problems generated by Adams, Balas & Zawack [7], Fisher & Thompson [171], Lawrence [339] and Taillard [521]. Vaessens, Aarts & Lenstra [532] collate the objective function values generated by the various algorithms and provide standardized computation times. These results are extended by Balas & Vazacopoulos [38]. The tabu search algorithm of Nowicki & Smutnicki and the hybrid shifting bottleneck/guided local search algorithms of Balas & Vazacopoulos are generally preferred to the other approaches.

7.3.4 Approximation: Absolute Guarantees

Recall that two trivial lower bounds on the optimum makespan in an instance of problem $J||C_{\max}$ are the maximum machine load $\Pi_{\max}$, and the length of the longest job $P_{\max}$. In this subsection, let $Z = \max\{\Pi_{\max}, P_{\max}\}$.

In a celebrated paper in theoretical computer science, Leighton, Maggs & Rao [343] show that, for the acyclic job shop problem $J|p_{ij} = 1, acyc|C_{\max}$, the optimal makespan is at most a constant factor away from the lower bound Z. We would emphasize that this constant factor does *not* depend

on the number of machines, on the number of jobs, nor on other parameters. The proof given by Leighton, Maggs & Rao is nonconstructive and makes repeated use of the Lovász local lemma [162], a famous result in combinatorial theory. An efficient polynomial time algorithm for finding a schedule with makespan $O(Z)$ is provided by Leighton, Maggs & Richa [344].

For the general job shop problem $J||C_{\max}$, the best upper bound known for the optimal makespan is $O(Z \log^2 Z/(\log\log Z)^2)$ by Goldberg, Paterson, Srinivasan & Sweedyk [208], which improves upon earlier work of Shmoys, Stein & Wein [486]. Feige & Scheideler [165] show that the optimal makespan for problem $J|acyc|C_{\max}$ is $O(Z \log Z \log\log Z)$, and they construct instances of $J|acyc|C_{\max}$ for which the optimal makespan is $\Omega(Z \log Z/\log\log Z)$. Hence, the $O(Z)$ upper bound of Leighton, Maggs & Rao [343] cannot be carried over to arbitrary processing times. Moreover, Feige & Scheideler show that preemption may help in job shop scheduling: the optimal makespan for problem $J|acyc, pmtn|C_{\max}$ is $O(Z \log\log Z)$. However, the question of whether the upper bound can be improved to $O(Z)$ for problem $J|acyc, pmtn|C_{\max}$ remains open.

7.3.5 Approximation: Ratio Guarantees

The best positive approximation result for $J||C_{\max}$ is due to Shmoys, Stein & Wein [486]. They construct randomized approximation algorithms with worst-case ratios of $O(\log^2(m\mu)/\log\log(m\mu))$, where μ is the maximum number of operations per job. Schmidt, Siegel & Srinivasan [463] achieve the same approximation bound with a deterministic algorithm.

For the job shop problem $Jm||C_{\max}$ with a constant number m of machines, Shmoys, Stein & Wein [486] provide approximation algorithms with worst-case ratios of $(2+\varepsilon)$, where $\varepsilon > 0$ can be made arbitrarily close to 0. It is easily verified that their analysis also applies to the preemptive problem $Jm|pmtn|C_{\max}$, for which it yields the same worst-case guarantee of $(2+\varepsilon)$. Improving these approximation results to a bound smaller than 2 and deciding the existence of a PTAS are open problems. Sevastianov & Woeginger [479] derive an approximation algorithm for problem $J2|pmtn|C_{\max}$ with a worst-case ratio of 3/2. For the multiprocessor problem $J2(P)|op \leq 2|C_{\max}$, Schuurman & Woeginger [470] give a PTAS.

7.4 Other Multi-Stage Problems

Assembly lines

The assembly line problem is a multi-stage scheduling problem with s stages and a single machine per stage. The last operation O_{sj} of job j (the assembly operation) can only be started when its first $s-1$ operations $O_{1j}, \ldots, O_{s-1,j}$ all have already been completed. The first $s-1$ operations, however, may be run in parallel and may overlap in time. The assembly line problem is denoted by A; for example, makespan minimization in an assembly line with three stages is denoted by $A3||C_{\max}$.

The assembly line problem is introduced by Lee, Cheng & Lin [340], and is then studied by Potts et al. [428]. Problem $A2||C_{\max}$ is equivalent to problem $F2||C_{\max}$, and hence solvable in polynomial time. Problem $A3||C_{\max}$ is shown by Potts et al. to be strongly $\mathcal{NP}$-hard. They also provide a polynomial time approximation algorithm for problem $A||C_{\max}$ with a worst-case ratio of $2-1/(s-1)$. It is not known whether there there is a better polynomial time approximation algorithm for this general case where the number of stages is part of the input. However, for problem $As||C_{\max}$ with a fixed number of stages, the technique of Hall [232] can be adapted to yield a PTAS. Finally, for $A3||C_{\max}$ the optimal makespan is shown by Potts et al. to be at most $\Pi_{\max} + 1.25p_{\max}$.

The mixed shop

The mixed shop (denoted by JO) is a combination of the job shop and the open shop. Thus, we have job shop jobs and open shop jobs. Due to the complexity of the job shop (see Section 7.3.1), only problems with two machines can be expected to be solvable in polynomial time. Strusevich [507] shows that problems $JO2|op \leq 2|C_{\max}$ and $JO2|pmtn, op \leq 2|C_{\max}$ can be solved in $O(n \log n)$ time. Shaklevich & Sotskov [483] prove that problems $JO|n=2|C_{\max}$ and $JO|n=2|\sum C_j$ are $\mathcal{NP}$-hard. They also derive a polynomial time algorithm for problem $JO|n=2, pmtn|f(C_1, C_2)$ with one open shop job and one job shop job, where f is an arbitrary regular objective function of the two job completion times.

The job shop with two counter routes

We now discuss a special case of the job shop (and a generalization of the flow shop) where the jobs can only take two routes through the m machines: The route $1-2-\cdots-m$ (as in the flow shop), and the reverse route $m-\cdots-2-1$. Hence, every job consists of a chain of exactly m operations. The job shop with two counter routes is denoted by $F^{\pm}$.

Since problem $F^{\pm}2||C_{\max}$ is equivalent to problem $J2|op \leq 2|C_{\max}$, it is solvable in polynomial time (see Section 7.3.1). On the other hand, problem $F^{\pm}3||C_{\max}$ is strongly $\mathcal{NP}$-hard, and all other $\mathcal{NP}$-hardness results also

carry over from the flow shop to the job shop with two counter routes. It is not known whether problem $F^{\pm}m||C_{\max}$ allows a PTAS (in this case, the techniques of Hall [232] cannot be generalized). Sevastianov [474] proves that the optimal makespan for problem $F^{\pm}m||C_{\max}$ is at most $\Pi_{\max}+2m^2p_{\max}$, a result which is just slightly weaker than that of Sevastianov [473] and Bárány & Fiala [40] for problem $Fm||C_{\max}$. For problem $F^{\pm}3||C_{\max}$, Neumytov & Sevastianov [397] improve the upper bound on the optimal makespan to $\Pi_{\max}+3p_{\max}$, and they show that the factor 3 in this bound is the best possible.

Other variants

Lev & Adiri [353] analyze the so-called *V-shop* problem, a special case of the job shop with machine repetition where each job takes the V-shaped route $1-2-\cdots-(m-1)-m-(m-1)-\cdots-2-1$ through the m machines. Matsuo [371] and Kamoun & Sriskandarajah [291] study cyclic flow shop problems in which each job has to go repeatedly through the machines.

8 Further Scheduling Models

8.1 Family Scheduling

In the family scheduling model, jobs are partitioned into F families according to their similarity, so that no setup is required for a job if it belongs to the same family of the previously processed job. For example, jobs may be assigned to the same family if they require the same machine tool, or if they are produced with the same material and the machine needs to be cleaned each time a different material is used. Reviews on models of this type are given by Potts & Van Wassenhove [436] and Webster & Baker [554].

A setup time is required at the start of the schedule and on each occasion when the machine switches from processing jobs in one family to jobs in another family. If the setup times are *sequence independent*, then the *family setup time* on machine i for family f is s_{if}. On the other hand, for *sequence dependent* setup times, then the setup time on machine i at the start of the schedule is s_{i0g} if a job of family g is processed first, and is s_{ifg} if a job of family g is processed immediately after a job of family f. Further, we make the reasonable assumption that the *triangle inequality* holds for each machine i, which means that $s_{ifh} \leq s_{ifg} + s_{igh}$, for all distinct families f, g and h, including the case $f = 0$. All setups are *anticipatory*, which means

that a setup on a machine does not require the presence of the job. In the case of a single machine, we omit the subscript i.

In this section, we use the following additional job characteristics in the second field:

- $\{\circ, s_f, s_{fg}\}$:
  - $\circ$: there are no set-up times;
  - s_f: there are sequence independent family set-up times;
  - s_{fg}: there are sequence dependent family set-up times.

A *batch* a maximal set of jobs that are scheduled contiguously on a machine and share a setup. Large batches have the advantage of high machine utilization because the number of setups is small. On the other hand, processing a large batch may delay the processing of an important job belonging to a different family.

For many problems with family setup times, $\mathcal{NP}$-hardness is deduced from a result without setups. Thus, in the discussion below, we mainly restrict our attention to problems that are polynomially solvable when there are no setup times.

8.1.1 Complexity

For several of the basic family scheduling models, results on the ordering of jobs within a family are available. For example, Monma & Potts [382] show that there exists an optimal schedule for problem $1|s_{fg}|L_{\max}$ and for the on-time jobs for problem $1|s_{fg}|\sum w_j U_j$ in which the jobs within each family are sequenced in EDD order. Also, for problem $1|s_{fg}|\sum w_j C_j$, jobs within each family are sequenced in SWPT order. Thus, to solve these problems, the ordered job within the different families are merged to produce a schedule (and for problem $1|s_{fg}|\sum w_j U_j$ the jobs are to be on time are selected). These merging problems can be solved by dynamic programming.

The dynamic programming algorithms differ according to whether a forward or backward approach is used. In a forward dynamic programming algorithm, initial partial schedules are build by appending a job to the the previous partial schedule. On the other hand, a backward dynamic programming algorithm inserts a job at the start of a previous final partial schedule, thereby causing a delay to each job in the partial schedule.

Monma & Potts [382] suggest forward dynamic programming algorithms with state variables to indicate the number of sequenced jobs in each family,

the number of setups of each type in the partial schedule (this enables the completion time of the partial schedule to be evaluated), and the family to which the last job in the partial schedule belongs. This yields algorithms for problems $1|s_{fg}|L_{\max}$ and $1|s_{fg}|\sum w_jC_j$, each with a time complexity of $O(F^2n^{F^2+2F})$. However, Ghosh & Gupta [199], and Ghosh [198] develop backward dynamic programming algorithms that avoid having numbers of setups as state variables. This reduces the time complexities for solving each of the problems $1|s_{fg}|L_{\max}$ and $1|s_{fg}|\sum w_jC_j$ to $O(F^2n^F)$. For problem $1|s_{fg}|\sum w_jU_j$, Monma & Potts develop a forward dynamic programming algorithm in which the weighted number of late jobs is a state variable, and the completion time of the partial of on-time jobs is a function value thereby avoiding state variables that define numbers of setups. This algorithm requires $O(F^2n^FW)$ time, where $W=\sum_{j=1}^{n} w_j$.

Problems $1|s_{fg}|L_{\max}$ and $1|s_{fg}|\sum U_j$ are is polynomially solvable by dynamic programming for fixed F, as indicated above. For arbitrary F (the number of families is part of the input), Bruno & Downey [73] prove that problems $1|s_f|L_{\max}$ and $1|s_f|\sum U_j$ are $\mathcal{NP}$-hard, although they are open with respect to pseudo-polynomial solvability. Similarly, problem $1|s_{fg}|\sum w_jC_j$ is polynomially solvable by dynamic programming for fixed F, and Ghosh [198] shows that problem $1|s_{fg}|\sum C_j$ is strongly $\mathcal{NP}$-hard for arbitrary F. However, the complexity status of problems $1|s_f|\sum C_j$ and $1|s_f|\sum w_jC_j$ is open for arbitrary F.

For parallel machines, Monma & Potts [382] show that problem $P2|\ s_f,\ pmtn|\ C_{\max}$ is $\mathcal{NP}$-hard, and Webster [553] shows that problem $P|\ s_f|\ \sum C_j$ is strongly $\mathcal{NP}$-hard.

For multi-stage problems, Kleinau [300] shows that problem $O2|s_f|C_{\max}$ is $\mathcal{NP}$-hard, even for the case of three families. He also shows that problem $F2|s_f|C_{\max}$ is $\mathcal{NP}$-hard for an arbitrary number of families, and that there exists an optimal solution for this problem that is a permutation schedule. Further, Monma & Potts [382] show that jobs within each family can be sequenced using the algorithm of Johnson [286]. Thus, as observed by Potts & Van Wassenhove [436], for fixed F, there is a polynomial time dynamic programming algorithm of the type described above for problem $F2|s_f|C_{\max}$.

8.1.2 Enumerative Algorithms

Hariri & Potts [245] propose a branch and bound algorithm for problem $1|s_f|L_{\max}$. They obtain an initial lower bound by ignoring setups, except for those associated with the first job in each family, and solve the resulting problem with the EDD rule. This lower bound is improved by a procedure that considers whether or not certain families are split into two or more batches. A binary branching rule fixes adjacent jobs (with respect to the EDD ordering) in the same family to be in the same or in different batches. Computational results show that the algorithm is successful in solving instances with up to about 60 jobs. Schutten, Van de Velde & Zijm [469] develop a branch and bound algorithm for problem $1|s_f, r_j|L_{\max}$. In the presence of release dates, no results are known about the order of jobs within a family. A key component of their algorithm is the use of dummy jobs to represent setups. Quickly computed lower bounds are obtained by relaxing setups and solving the corresponding preemptive problem, and a forward branching rule is used. Computational results show that the algorithm is effective in solving instances with up to about 40 jobs.

Mason & Anderson [370] and Crauwels, Hariri, Potts & Van Wassenhove [115] propose branch and bound algorithms for problem $1|s_f|\sum w_jC_j$. Mason & Anderson use a forward branching rule, and make extensive use of dominance rules to restrict the size of the branch and bound search tree. Their lower bound is derived using objective splitting: the total weighted completion time can be partitioned into contributions from the processing times and from the setup times, which are optimized separately. Crauwels, Hariri, Potts & Van Wassenhove compare three algorithms. The most efficient uses a forward sequencing branching rule. Lower bound are obtained by performing a Lagrangean relaxation of the machine capacity constraints in a time-index formulation of the problem, and a constructive method is used to compute values of the multipliers. Their algorithm is successful in solving instances with up to 70 jobs, and is superior to Mason & Anderson's algorithm.

8.1.3 Local Search

There are two studies that develop local search heuristics for problem $1|s_f|\sum w_jC_j$. Mason [369] designs a genetic algorithm from the observation that knowledge of the first job in each batch enables a solution to be constructed by ordering the batches using a generalization of the SWPT

rule. Thus, he uses a binary representation of solutions to which standard genetic operators are applied. Crauwels, Potts & Van Wassenhove [117] develop several neighborhood search heuristics (descent, simulated annealing, threshold accepting and tabu search). They use a neighborhood that selects a sub-batch of jobs at the beginning (end) of a batch and moves this sub-batch to an earlier (later) position in the sequence. The temperature in simulated annealing follows a periodic pattern, and a descent algorithm is applied before each temperature change. Threshold accepting is applied in an analogous way to simulated annealing. In their tabu search method, sub-batches are restricted to contain a single job, and a limited reordering of batches according to an SWPT rule applied to batches is used. Computational tests for instances with up to 100 jobs and up to 20 families show that all local search methods generate solutions which are close in value to the optimum. The best results are obtained with a multi-start version of tabu search when the number of families is small, and with Mason's genetic algorithm when the number of families is large.

Crauwels, Potts & Van Wassenhove [116] propose multi-start versions of descent, simulated annealing and tabu search, and a genetic algorithm, for problem $1|s_f|\sum U_j$. The neighborhood search algorithms use either a job or batch neighborhood. In the job neighborhood, a job is removed from the sequence of on-time jobs, and an attempt is made to insert one or more late jobs into the resulting sequence so that all of these jobs are on-time. The batch neighborhood is similar except that a complete batch is removed and then insertions of late jobs are attempted. The genetic algorithm uses a representation in which two binary elements are associated with each job, one indicating whether the job is on time or late, and the other indicating whether the job ends a batch. To obtain the corresponding schedule of on-time jobs, a due date is associated with each batch of on-time jobs, and the batches are sequenced in EDD order. If the resulting solution is infeasible, the algorithm of Moore [385] is applied to remove the smallest number of batches. In computational tests for instances with up to 50 jobs and up to 10 families, all heuristics perform well. The best quality solutions are obtained with a version of the genetic algorithm, which includes a procedure that attempts to improve each solution of the final population.

Sotskov, Tautenhahn & Werner [498] develop constructive and neighborhood search heuristics (simulated annealing, threshold accepting and tabu search) for problems $F|s_f|C_{\max}$ and $F|s_f|\sum C_j$ in which the search is restricted to permutation schedules. They use the same type of neighborhood as Crauwels, Potts & Van Wassenhove [117]. In computational tests for

instances with up to 80 jobs, and with 5 and 10 machines, the best results are obtained with simulated annealing and tabu search, with the former providing slightly better quality solutions.

8.1.4 Approximation

Monma & Potts [383] propose a heuristic for problem $Pm|s_f, pmtn|C_{\max}$ that resembles McNaughton's algorithm [378] for the classical scheduling problem $P|pmtn|C_{\max}$. It has a ratio guarantee of $2 - 1/(\lfloor m/2 \rfloor + 1)$. For a special class of instances in which the setup plus total processing time for each single family does not exceed the optimal makespan, Monma & Potts [383] and Chen [91] show that this performance can be improved through the use of a heuristic that first uses list scheduling for complete families, and then splits families between selected pairs of machines. In particular, Chen's heuristic has a worst-case ratio of $\max\{3m/(2m+1), (3m-4)/(2m-2)\}$.

Chen, Potts & Strusevich [92] propose two heuristics for problem $F2|s_f|C_{\max}$, each of which requires $O(n \log n)$ time. The first heuristic, which assigns all jobs of a family to a single batch and then schedules the batches, has a worst-case ratio of 3/2. The second heuristic uses properties of the schedule created by the first heuristic to generate another schedule by splitting each family into at most two batches. The heuristic selects the better of the two schedules, and has a worst-case ratio of 4/3.

8.2 Scheduling Multiprocessor Jobs

A *multiprocessor job* requires more than one machine at a time. This notion is in contrast to the classical scheduling assumption that a job can only be executed on at most one machine at a time. However, it seems to be the right way of modeling modern parallel computer systems with shared memory.

In this subsection, we only deal with two basic variants of scheduling multiprocessor jobs. In the first variant, processing of job j simultaneously needs precisely $size_j$ machines and it does not matter *which* machines are used for processing the job. In second variant, processing of job j simultaneously needs a prespecified subset fix_j of the *dedicated machines*. In these variants, the symbol Δ is used to denote $\max_j size_j$ or $\max_j |fix_j|$.

We note that many other variants of scheduling multiprocessor jobs appear in the literature. For example, sometimes every job j has an associated

upper bound δ_j that indicates the maximum number of machines that can simultaneously process j. The actual processing time of a job then depends on the number of machines that are used for running the job. This dependence may be linear (in the *linear speedup* model) or more involved. For more information on these variants, we refer the reader to the survey article by Drozdowski [146]. Also, a discussion of various speedup functions in an on-line setting, is given by Edmonds, Chinn, Brecht & Deng [156].

8.2.1 Parallel Machines

The first papers on multiprocessor job scheduling mainly investigate variants with unit execution times. Lloyd [362] proves that problem $P2|\ size_j,\ prec,\ p_j = 1|\ C_{\max}$ is solvable in linear time, whereas problem $P3|size_j \in \{1,2\}, prec, p_j = 1|C_{\max}$ and problem $P|size_j, p_j = 1|C_{\max}$ are both strongly $\mathcal{NP}$-hard. He also presents an approximation algorithm for problem $P|size_j, p_j = 1|C_{\max}$ that is based on list scheduling and has ratio guarantee of $(2m - \Delta)/(m - \Delta + 1)$. Błażewicz, Drabowski & Węglarz [53] show that problem $P|size_j, p_j = 1|C_{\max}$ is polynomially solvable if Δ is fixed and not part of the input.

Du & Leung [147] show that problems $P2|size_j|C_{\max}$ and $P3|size_j|C_{\max}$ are $\mathcal{NP}$-hard, but are pseudo-polynomially solvable. On the other hand, problem $P5|size_j|C_{\max}$ is strongly $\mathcal{NP}$-hard. It is still open whether problem $P4|size_j|C_{\max}$ is strongly $\mathcal{NP}$-hard. For problem $P2|size_j, chain|C_{\max}$, Du & Leung prove strong $\mathcal{NP}$-hardness. Steinberg [501] presents a 2-approximation algorithm for problem $P|size_j|C_{\max}$.

Drozdowski [145] shows that problem $P|\ size_j,\ pmtn|\ C_{\max}$ is $\mathcal{NP}$-hard. Błażewicz, Drabowski & Węglarz [53] show that problem $P|\ size_j \in \{1,\ \Delta\}$, $pmtn|\ C_{\max}$ is solvable in linear time. Moreover, they show that if Δ is fixed, then problem $Pm|size_j, pmtn|C_{\max}$ can be formulated as an integer program of fixed dimension. Consequently, problem $Pm|size_j, pmtn|C_{\max}$ is polynomially solvable. Błażewicz, Drozdowski, Schmidt & de Werra [54] present an $O(n \log n + nm)$ algorithm for problem $Q|size_j \in \{1,2\}, pmtn|C_{\max}$.

Finally, we mention some on-line scheduling results. Feldmann, Sgall & Teng [167] study problem $P|on\text{-}line\text{-}list, size_j|C_{\max}$. They show that list scheduling, which always schedule the current job as early as possible, is $(2 - 1/m)$-competitive. Feldmann, Kao, Sgall & Teng [166] show that for problem $P|on\text{-}line\text{-}list, size_j, chain|C_{\max}$, the best possible competitive ratio is m. For the variant of $P|on\text{-}line\text{-}list, size_j, prec|C_{\max}$ with linear speedup, they present an algorithm with a competitive ratio of $(\sqrt{5}+3)/2 \approx 2.6180$.

This result is the best possible for on-line scheduling, and it also constitutes the best currently known off-line approximation algorithm for this problem.

8.2.2 Dedicated Machines

In one of the first papers on the dedicated machine model, Krawczyk & Kubale [308] show that problem $P|\mathit{fix}_j|C_{\max}$ is $\mathcal{NP}$-hard even if $|\mathit{fix}_j| = 2$ for all jobs. Hoogeveen, Van de Velde & Veltman [269] and Drozdowski [145] show that problems $P3|\mathit{fix}_j|C_{\max}$ and $P2|\mathit{fix}_j|L_{\max}$ are strongly $\mathcal{NP}$-hard. Hoogeveen, Van de Velde & Veltman also establish $\mathcal{NP}$-hardness for the following problems with unit processing times: $P|\mathit{fix}_j, p_j = 1|C_{\max} \leq 3$, $P2|\mathit{fix}_j, p_j = 1, \mathit{chain}|C_{\max}$, and $P2|\mathit{fix}_j, p_j = 1, r_j|C_{\max}$. Problem $Pm|\mathit{fix}_j, p_j = 1|C_{\max}$ may be formulated as an integer program of fixed dimension, and hence is polynomially solvable.

Hoogeveen, Van de Velde & Veltman [269] argue that, unless $\mathcal{P}$=$\mathcal{NP}$, problem $P|\mathit{fix}_j|C_{\max}$ does not possess a polynomial time approximation algorithm with worst-case ratio better than 4/3, which is a consequence of the $\mathcal{NP}$-completeness of determining whether there is a schedule with $C_{\max} \leq 3$ for the case of unit processing times. Amoura, Bampis, Kenyon & Manoussakis [22] present a PTAS for problem $Pm|\mathit{fix}_j|C_{\max}$. This PTAS is based on a linear programming formulation for the corresponding preemptive problem, and it dominates a flood of previous results on the approximability of this problem.

Kubale [313] shows that the preemptive problem $P|\mathit{fix}_j, \mathit{pmtn}|C_{\max}$ is strongly $\mathcal{NP}$-hard, even if $|\mathit{fix}_j| = 2$ for all jobs. On the other hand, he shows that problem $Pm|\mathit{fix}_j, \mathit{pmtn}|C_{\max}$ with $|\mathit{fix}_j| = 2$ may be formulated as a linear program, and therefore can be solved in polynomial time. Krämer [306] shows that problem $Pm|\mathit{fix}_j, r_j, \mathit{pmtn}|L_{\max}$ can be solved by solving $O(\log n)$ linear programs, each of which has $O(n^{m+1})$ variables and $O(n)$ constraints. Amoura, Bampis, Kenyon & Manoussakis [22] propose an $O(n)$ solution for problem $Pm|\mathit{fix}_j, \mathit{pmtn}|C_{\max}$.

Cai, Lee & Li [77] show that problem $P2|\mathit{fix}_j|\sum C_j$ is strongly $\mathcal{NP}$-hard. Also, Hoogeveen, Van de Velde & Veltman [269] show that the problems $P3|\mathit{fix}_j|\sum C_j$, $P2|\mathit{fix}_j|\sum w_j C_j$, $P2|\mathit{fix}_j, p_j = 1, \mathit{chain}|\sum w_j C_j$, and $P|\mathit{fix}_j, p_j = 1|\sum C_j$ are all strongly $\mathcal{NP}$-hard.

8.3 Scheduling with Communication Delays

Scheduling with communication delays models computer systems with data transmissions between the jobs where the data transmission times are significant. There are m parallel machines and a set of precedence constrained jobs. With every arc (j,k) between two jobs in the precedence constraints, there is an associated *communication delay* c_{jk}. If jobs j and k are processed on different machines, then the processing of k cannot start before c_{jk} time units have elapsed after the completion of j. However, if jobs j and k are processed on the same machine, then k cannot start before j has been completed, but no further delay is imposed. The communication does not interfere with the availability of the machines, i.e., during a communication delay all machines may process other jobs. The occurrence of communication delays is indicated by one of the following entries in the job characteristics field.

- $\{\circ, c_{jk}, c_{j*}, c_{*k}, c, c=1, dup\}$:
  - $\circ$: there are no communication delays;
  - c_{jk}: there are general communication delays;
  - c_{j*}: the communication delays depend on the sending job only;
  - c_{*k}: the communication delays depend on the receiving job only;
  - c: all communication delays are equal (*uniform* communication delays);
  - $c=1$: all communication delays take one time unit (*unit* communication delays);
  - dup: job duplication is allowed.

If *job duplication* is allowed, then the scheduler may create several copies of a job. Note that creation of job copies may be favorable in circumventing high communication times. In the machine environment field, an entry "$P\infty$" means that the number of machines is unrestricted and may be chosen by the scheduler. For example, makespan minimization with job duplication, uniform communication delays, and an unrestricted number of machines is denoted by $P\infty|prec, c, dup|C_{\max}$. For more information on scheduling with communication delays (than that presented in this short section), we refer the reader to the survey articles by Chrétienne & Picouleau [107] and by Veltman, Lageweg & Lenstra [544], and to the Ph.D. theses by Picouleau [418] and by Veltman [543].

8.3.1 Complexity

In a seminal paper, Papadimitriou & Yannakakis [415] show by a transformation from the clique problem in graphs that problems $P\infty|prec, p_j = 1, c|C_{\max}$ and $P\infty|prec, p_j = 1, c, dup|C_{\max}$ are both $\mathcal{NP}$-hard. Jung, Kirousis & Spirakis [287] design an $O(n^{c+2})$ algorithm that is based on dynamic programming for the problem with job duplication. Hence, if the uniform communication delay c is not part of the input, then problem $P\infty|prec, p_j = 1, c, dup|C_{\max}$ is solvable in polynomial time. This demonstrates that problem $P\infty|prec, p_j = 1, c = 1, dup|C_{\max}$ is polynomially solvable. In contrast, for the corresponding problem $P\infty|prec, p_j = 1, c = 1|C_{\max}$ without job duplication, deciding whether there exists a schedule with makespan that does not exceed six is shown by Hoogeveen, Lenstra & Veltman [260], based on results of Picouleau [419], to be $\mathcal{NP}$-complete. Notably, it is shown by Hoogeveen, Lenstra & Veltman that deciding whether there exists a schedule with makespan that does not exceed five is solvable in polynomial time.

Colin & Chrétienne [112] show that problem $P\infty|prec, c_{jk}, dup|C_{\max}$ is polynomially solvable if the communication delays are *small*, i.e., if for all jobs k, $\min\{p_j|j \to k\} \geq \max\{c_{jk}|j \to k\}$ holds, where $j \to k$ means that job j is a direct predecessor of job k. Jakoby & Reischuk [285] investigate problems with tree-like precedence constraints. They prove that problem $P\infty|p_j = 1, intree, c, dup|C_{\max}$ is $\mathcal{NP}$-hard if the precedence constraints are given by a binary intree, but is polynomially solvable if the precedence constraints form a *complete* binary intree, in which all leaves are in the same distance from the root and every non-leaf vertex has precisely two predecessors. Problem $P\infty|p_j = 1, intree, c_{jk}, dup|C_{\max}$ is $\mathcal{NP}$-hard, even for complete binary intrees.

Most of the cases where the number of machines is given as part of the input are $\mathcal{NP}$-hard, since even the two simplest variants of deciding whether there exists a schedule with makespan that does not exceed four for problems $P|prec, p_j = 1, c = 1|C_{\max}$ and $P|prec, p_j = 1, c = 1, dup|C_{\max}$ are shown by Hoogeveen, Lenstra & Veltman [260], based on results of Rayward-Smith [444], to be $\mathcal{NP}$-complete. It is also shown by Hoogeveen, Lenstra & Veltman that problem $P|prec, p_j = 1, c = 1|C_{\max} \leq 3$ is solvable in polynomial time. The complexity of problem $Pm|prec, p_j = 1, c = 1|C_{\max}$ with a fixed number of machines is open, even for $m = 2$.

On the other hand, although problem $P|intree, p_j = 1, c = 1|C_{\max}$ is strongly $\mathcal{NP}$-hard (Lenstra, Veldhorst & Veltman [351]), Varvarigou, Roy-

chowdhury, Kailath & Lawler [542] show that the corresponding problem $Pm|intree, p_j = 1, c = 1|C_{\max}$ with a fixed number of machines is solvable in $O(n^{2m})$. Lenstra, Veldhorst & Veltman show that problem $P2|p_j = 1, intree, c = 1|C_{\max}$ is solvable even in linear time. Rayward-Smith [443] studies problem $P|chain, c|C_{\max}$, whose special case $P|chain, c = 1|C_{\max}$ is equivalent to problem $P|pmtn|C_{\max}$ and hence is solvable in polynomial time by McNaughton's wrap-around rule [378]. Surprisingly, for any fixed $c \geq 2$, problem $P|chain, c|C_{\max}$ is $\mathcal{NP}$-hard. Ali & El-Rewini [17] give a polynomial time algorithm for the special case of problem $P|prec, p_j = 1, c = 1|C_{\max}$ where the precedence constraints arise from an interval order.

8.3.2 Approximation

We first discuss approximation results for the variant with an unrestricted number of machines. Papadimitriou & Yannakakis [415] construct a polynomial time approximation algorithm with a ratio guarantee of 2 for problem $P\infty|prec, c_{j*}, dup|C_{\max}$. Two outstanding open problems are to decide whether their exists a polynomial time approximation algorithm for problem $P\infty|prec, p_j = 1, c, dup|C_{\max}$ with ratio guarantee better than 2, and to find an approximation algorithm for problem $P\infty|prec, p_j = 1, c|C_{\max}$ with bounded ratio guarantee.

Munier & König [391] present a (4/3)-approximation algorithm for problem $P\infty|prec, p_j = 1, c = 1|C_{\max}$, which is based on the solution of the linear programming relaxation for their integer linear programming formulation. Hoogeveen, Lenstra & Veltman [260] show that, unless $\mathcal{P}$=$\mathcal{NP}$, problem $P\infty|prec, p_j = 1, c = 1|C_{\max}$ does not possess a polynomial time approximation algorithm with ratio guarantee better than 7/6. Möhring, Schäffter & Schulz [380] generalize the results of Munier & König [391] to problem $P\infty|prec, p_j = 1, c = 1|\sum w_j C_j$, and thus derive a (4/3)-approximation algorithm for this problem. Based on the results of Hoogeveen, Lenstra & Veltman [260], Hoogeveen, Schuurman & Woeginger [263] observe that, unless $\mathcal{P}$=$\mathcal{NP}$, problem $P\infty|prec, p_j = 1, c = 1|\sum C_j$ does not possess a polynomial time approximation algorithm with ratio guarantee better than 9/8.

When the number of machines is given as part of the input, Rayward-Smith [444] investigates *greedy* schedules for problem $P|prec, p_j = 1, c = 1|C_{\max}$, where a greedy schedule is one in which no machine is idle unless there is no job available for processing. He proves that any greedy scheduling algorithm has a ratio guarantee of $3 - 2/m$. Munier & Hanen [390]

provide an improvement by presenting a $(7/3 - 4/(3m))$-approximation algorithm for this problem. Hoogeveen, Lenstra & Veltman [260] show that, unless $\mathcal{P}$=$\mathcal{NP}$, problem $P|prec, p_j = 1, c = 1|C_{\max}$ does not possess a polynomial time approximation algorithm with ratio guarantee better than 5/4. Möhring, Schäffter & Schulz [380] extend the results of Munier & Hanen [390] to problem $P|prec, p_j = 1, c = 1|\sum w_j C_j$ by derive a $(10/3 - 4/(3m))$-approximation algorithm. Based on the results of Hoogeveen, Lenstra & Veltman [260], Hoogeveen, Schuurman & Woeginger [263] observe that, unless $\mathcal{P}$=$\mathcal{NP}$, problem $P|prec, p_j = 1, c = 1|\sum C_j$ does not possess a polynomial time approximation algorithm with ratio guarantee better than 11/10. Munier & Hanen [389] present a $(2 - 1/m)$-approximation algorithm for problem $P|prec, p_j = 1, c = 1, dup|C_{\max}$.

8.4 Resource Constrained Scheduling

In this section, we consider a natural extension of the classical deterministic machine scheduling problems, namely problems that involve the presence of additional resources, where each resource is limited in quantity and each job requires the use of a given quantity of each resource during its execution. This topic itself is so broad that we are mainly concerned here with a very limited subclass of off-line scheduling problems with discrete renewable resources, i.e., only their total usage over each time period is constrained. Nevertheless this covers a tremendous variety of problem types, through which the domain of deterministic scheduling theory is considerably extended. For more general models of resource constrained scheduling and more comprehensive treatment of the topic, we refer the reader to a monograph by Błażewicz, Cellary, Słowiński & Węglarz [52].

Let us start with a problem classification scheme of Błażewicz, Lenstra & Rinnooy Kan [56] to accommodate the additional resource constraints into the existing classification scheme. Suppose that there are l *resources* $R_1, \ldots, R_l$. For each resource R_h, there is a positive integer *size* s_h, which is the total amount of R_h that is available at any given time. In single-stage models, for each resource R_h and job j, there is a non-negative integer *requirement* r_{hj}, which is the amount of R_h required by j throughout its execution. A schedule is *feasible* with respect to the resource constraints if and only if the total requirement for resource R_h by all jobs being executed at any time does not exceed s_h, for $h = 1, \ldots, l$. In multi-stage models, there is for each resource R_h and operation O_{ij} a non-negative integer requirement r_{hij}, with a similar condition for the feasibility of a schedule.

The presence of resource constraints is indicated in the second field of our existing scheme by an entry $res\lambda\sigma\rho$, where λ, σ, and ρ specify the number of resources, their sizes, and the required amounts, respectively. More precisely, they are characterized as follows.

- $\lambda \in \{\cdot, \bar{\lambda}\}$:
  - $\lambda = \cdot$: the number of resources is arbitrary;
  - $\lambda = \bar{\lambda}$: there is a fixed upper bound $\bar{\lambda}$ on the number of resources.
- $\sigma \in \{\cdot, \bar{\sigma}\}$:
  - $\sigma = \cdot$: the resource sizes are arbitrary;
  - $\sigma = \bar{\sigma}$: all resource sizes s_h are constants and equal to $\bar{\sigma}$.
- $\rho \in \{\cdot, \bar{\rho}\}$:
  - $\rho = \cdot$: the resource requirements r_{hj} (r_{hij}) are arbitrary;
  - $\rho = \bar{\rho}$: all resource requirements r_{hj} (r_{hij}) have a constant upper bound equal to $\bar{\rho}$.

8.4.1 No Precedence Constraints

Błażewicz, Lenstra & Rinnooy Kan [56] investigate problem $Q|res\lambda\sigma\rho, p_j = 1|C_{\max}$ for $\lambda, \sigma, \rho \in \{1, \cdot\}$. They provide an exhaustive complexity classification by identifying all maximal polynomially solvable problems and minimal $\mathcal{NP}$-hard problems.

There are three basic problems with resource constraints that are solvable in polynomial time. First, problem $P2|res\cdot\cdot, p_j = 1|C_{\max}$ can be solved in $O(n^2 l + n^{2.5})$ time by computing a maximum matching (Garey & Johnson [185]); recall that l denotes the number of resources. Second, problem $Q2|res1\cdot, p_j = 1|C_{\max}$ can be solved in $O(n \log n)$ time by the following simple algorithm of Błażewicz, Lenstra & Rinnooy Kan [56]. Start by scheduling all jobs on the faster machine in order of non-increasing resource requirements. Next, successively remove the last job from this machine and schedule it as early as possible on the slower machine, as long as this reduces the makespan. Third, Błażewicz, Lenstra & Rinnooy Kan [56] show that problem $Q|res1 \cdot 1, p_j = 1|C_{\max}$ can be solved in $O(n^3)$ time by solving a bottleneck transportation problem. In contrast, problems $P3|res \cdot 11, p_j = 1|C_{\max}$, $Q2|res \cdot 11, p_j = 1|C_{\max}$ and $P3|res1\cdot, p_j = 1|C_{\max}$ all are strongly $\mathcal{NP}$-hard (Błażewicz, Lenstra & Rinnooy Kan [56] and Garey & Johnson [185]).

Most of the above results can be easily extended to other optimality criteria. Błażewicz, Lenstra & Rinnooy Kan [56] also observe that, if preemption is allowed, then the very general problem $Rm|pmtn, res\cdots|C_{\max}$ can be solved by linear programming.

Now consider multi-stage models. Research has revealed that virtually all except the simplest problems are $\mathcal{NP}$-hard. Similar to solving $P2|res\cdots, problemp_j = 1|C_{\max}$, the $O2|res\cdots, p_{ij} = 1|C_{\max}$ reduces to finding a maximum matching, while Błażewicz, Cellary, Słowiński & Węglarz [52] show that problems $O3|res \cdot 11, p_{ij} = 1|C_{\max}$ and $O3|res1\cdot\cdot, p_{ij} = 1|C_{\max}$ are strongly $\mathcal{NP}$-hard. Kubiak [310] and Lushchakova & Strusevich [364] solve problem $O2|res111|C_{\max}$ in linear time. Recently, Jurisch & Kubiak [288] formulate problem $O2|res1\cdot\cdot|C_{\max}$ as a maximum flow problem in a network, which can be solved in $O(n^3)$ time, and then convert the optimal preemptive schedule into a non-preemptive one in $O(n^2)$ time without increasing the makespan. Problems $O2|res211|C_{\max}$ and $O2|res \cdot 11|C_{\max}$ are shown by Lushchakova & Strusevich [363] and Jurisch & Kubiak, respectively, to be strongly $\mathcal{NP}$-hard. These results show, in terms of computational complexity, that it is much easier to minimize makespan in the resource-constrained open shops with a single resource than it is with many specialized resources.

Flow shop and job shop problems are more difficult, and results for these models are limited. Błażewicz, Lenstra & Rinnooy Kan [56] and Błażewicz, Cellary, Słowiński & Węglarz [52] observe that, while problem $F2|res111, p_{ij} = 1|C_{\max}$ is solvable in linear time by appropriately grouping jobs together according to their overall resource requirements, each of the problems $F2|res \cdot 11, p_{ij} = 1|C_{\max}$, $F2|res111|C_{\max}$ and $J2|res111, p_{ij}|C_{\max}$ is strongly $\mathcal{NP}$-hard.

8.4.2 Precedence Constraints

In the presence of precedence constraints, essentially *all* scheduling problems with resource constraints are strongly $\mathcal{NP}$-hard, since Błażewicz, Lenstra & Rinnooy Kan [56] establish strong $\mathcal{NP}$-hardness for the simplest such problem $P2|res111, chain, p_j = 1|C_{\max}$. Research is mainly focused on enumerative algorithms.

A detailed literature review on research in this area can be found in Davis [123, 124], Herroelen [249], Herroelen & Demeulemeester [250]. A basic conceptual formulation for the classical resource-constrained project scheduling problem is to minimize C_n subject to precedence constraints $C_j + p_j \leq C_k$ whenever $j \to k$, and resource constraints $\sum_{j \in I_t} r_{hj} \leq s_h$

for all R_h and all t, where job n succeeds all others, and I_t is the index sets of jobs executed at time period t. Various integer programming formulations and algorithmic procedures are developed. Typically, these include the bounded enumeration procedure of Davis & Heidorn [125], the branch and bound algorithm of Stinson, Davis & Khumawala [504] and the implicit enumeration procedure of Talbot & Patterson [523]. Later, Demeulemeester & Herroelen [135] develop a branch and bound algorithm, which we call the DH-procedure, that is based on a depth-first solution strategy. In their search tree, the nodes represent partial feasible schedules. Branches emanating from a parent node correspond to exhaustive and minimal combinations of jobs whose delay resolves resource conflicts at each parent node. Computational evidence indicates that the DH-procedure is more efficient than other algorithms.

Following earlier work by Balas [35], Bartusch, Möhring & Radermacher [46] introduce into their solution procedure the notion of *forbidden sets*, which are subsets of jobs that cannot be executed simultaneously due to their collective resource requirements, and *temporal* constraints between pairs of jobs. This enables their algorithm to allow for precedence constraints of minimal and maximal time lags between jobs, for resources whose availability may change in discrete jumps over time, for time-dependent (in discrete jumps) resource consumption per job, and for job release dates and due dates. Unfortunately, when the problem has a large degree of parallelism, i.e., if there are many forbidden sets or if there are few temporal constraints, then their algorithm is computationally too time consuming.

Recently, Demeulemeester & Herroelen [136] extend their earlier branch and bound algorithm (the DH-procedure) to deal with *generalized* resource-constrained project scheduling problems. In such a problem, not only the availability of each resource varies over time and jobs have release dates and due dates, but also *precedence diagramming* is introduced to accommodate the specification of all four possible minimum time lags between the start and finish times of a predecessor and those of a successor. Results from computational experience are promising, and the extended DH-procedure is able to solve to optimality typical test instances in the literature reasonably effectively.

8.5 Scheduling with Controllable Processing Times

In most classical scheduling models, we assume *fixed* job processing times. However, in real-life applications, the processing of a job often requires ad-

ditional resources such as facilities, manpower, funds, etc., and hence the processing time can change with the allocation of these additional resources. These changes can be *continuous* (see Nowicki & Zdrzałka [405]) and *discrete* (see Chen, Lu & Tang [100]). These situations are usually modeled as follows. In the case of continuously controllable processing times, the processing requirement of job j is specified by three positive parameters a_j, u_j and c_j with $u_j \leq a_j$. By assigning additional resources to the processing of job j, the *actual* processing time of job j may be compressed down to $a_j - x_j$, where $0 \leq x_j \leq u_j$. The cost of performing this compression is equal to $c_j x_j$. In the case of discretely controllable processing times, the processing requirement of job j is specified by positive values a_{ij} and c_{ij}, for $i = 1, \ldots, k_j$. The compression parameter x_j for job j may take any integer value between 1 and k_j. The *actual* processing time of job j is equal to $a_{x_j,j}$, and the cost for this compression is equal to $c_{x_j,j}$.

Solving a scheduling problem with controllable processing times amounts to specifying a schedule σ for the jobs together with a compression vector $\vec{x}$ that encodes the compressions x_j of the jobs. For an optimality criterion f such as $C_{\max}$ and $L_{\max}$, we denote by $F_1(f, \sigma, \vec{x})$ the cost of schedule σ under criterion f with processing times compressed by $\vec{x}$. Moreover, we denote by $F_2(\vec{x})$ the total compression cost of $\vec{x}$, i.e., $F_2(\vec{x}) \doteq \sum_j c_j x_j$ in the continuous case and $F_2(\vec{x}) \doteq \sum_j c_{x_j,j}$ in the discrete case. Scheduling with controllable job processing times essentially is a bicriteria problem, and the following four basic optimization problems arise.

P1. Minimization of $F_1(f, \sigma, \vec{x}) + F_2(\vec{x})$.

P2. Minimization of $F_1(f, \sigma, \vec{x})$ subject to $F_2(\vec{x}) \leq \kappa$.

P3. Minimization of $F_2(\vec{x})$ subject to $F_1(f, \sigma, \vec{x}) \leq \tau$.

P4. Identify the set of Pareto-optimal points $(\vec{x}, \sigma)$ for (F_1, F_2).

A pair $(\vec{x}, \sigma)$ is called *Pareto-optimal* if there does not exist another pair $(\vec{x'}, \sigma')$ that improves on $(\vec{x}, \sigma)$ with respect to one of F_1 and F_2, and stays equal or improves with respect to the other one. Note that a solution to problem P4 also solves problems P1–P3 as a by-product.

8.5.1 Continuously Controllable Processing Times

For problems $1||L_{\max}$, $1||T_{\max}$, and $1|r_j|C_{\max}$, an optimal schedule can be computed without knowledge of the job processing times by the EDD rule or variants of this rule (see Section 4.1.1). As a consequence, the three

problems P1, P2 and P3 become linear programs, and hence are solvable in polynomial time. Vickson [548] observes that for problem $1||T_{\max}$, the linear program for P3 reduces to a production-inventory problem, which yields an $O(n^2)$ algorithm. Van Wassenhove & Baker [540] describe a simple greedy algorithm that solves P4 for these three scheduling problems in $O(n^2)$ time. The boundary of the set of Pareto-optimal points is a piecewise linear curve with up to $n+1$ breakpoints. The greedy algorithm identifies the breakpoints one-by-one. As a consequence, this also yields algorithms of the same time complexity for problems P1–P3.

The delivery time version of problem $1|r_j|L_{\max}$ is strongly $\mathcal{NP}$-hard even for fixed processing times (see Section 4.1.1). For the corresponding problem P1, Zdrzałka [572] gives a polynomial time approximation algorithm with a worst-case ratio of $3/2+\varepsilon$, where $\varepsilon > 0$ can be made arbitrarily small. This result is based on the PTAS of Hall & Shmoys [235] for the delivery time version of problem $1|r_j|L_{\max}$.

We now consider problem $1||f_{\max}$ (cf. Section 4.1.1). Van Wassenhove & Baker [540] investigate the special case where the functions f_j are totally ordered, i.e., $f_j(t) \leq f_{j+1}(t)$ holds for all t and all $1 \leq j \leq n-1$. For the case of linear functions f_j, they give an $O(n^3)$ algorithm for problems P1–P4. Tuzikov [529] presents a polynomial time algorithm for problem P3. Hoogeveen & Woeginger [271] prove that P1–P4 are polynomially solvable for regular functions f_j that are piecewise linear.

Vickson [548, 549] studies P1 for problems $1||\sum C_j$ and $1||\sum w_jC_j$. For $1||\sum C_j$, problem P1 can be formulated as an assignment problem and hence can be solved in $O(n^{2.5})$ time. For problem $1||\sum C_j$, Ruiz Diaz & French [454] develop an enumerative algorithm for P4. They also note that the set of Pareto-optimal points in general is not convex. Hoogeveen & Woeginger [271] prove that P1 is $\mathcal{NP}$-hard for problem $1||\sum w_jC_j$. Vickson [549] describes an enumerative algorithm for P4 for $1||\sum w_jC_j$. Inspired by Vickson's work, Panwalkar & Rajagopalan [410] consider P1 for problem $1|d_j = d|\sum(wE_j + w'T_j)$, where d is an unrestrictively large due date, and formulate it as an assignment problem. Alidaee & Ahmadian [18] observe that the results of Vickson [548] and Panwalkar & Rajagopalan [410] can be extended to parallel machines, which yields polynomial time algorithms for P1 for problems $P||\sum C_j$ and $P|d_j = D|\sum \alpha E_j + \beta T_j$.

Nowicki & Zdrzałka [406] give an $O(n^2)$ greedy algorithm to solve P4 for problem $P|pmtn|C_{\max}$. The boundary of the set of Pareto-optimal points is a piecewise linear curve with up to $2n+1$ breakpoints. Their approach also works for P4 for problem $Q2|pmtn|C_{\max}$. The computational complexity of

P4 for problem $Q3|pmtn|C_{\max}$ is unknown.

For $O2||C_{\max}$, problem P1 can be formulated as a linear program, and thus is solvable in polynomial time (see Section 7.1.1). This observation also yields polynomial time algorithms for problems P2–P4. Nowicki & Zdrzałka [404] show that P1 for problem $F2||C_{\max}$ is $\mathcal{NP}$-hard. They also develop a polynomial time approximation algorithm with a ratio guarantee of 3/2 for this problem. Grabowski & Janiak [222] propose a branch and bound algorithm for P3 for problem $J||C_{\max}$.

8.5.2 Discretely Controllable Processing Times

Chen, Lu & Tang [100] formulate P1 for problems $1||\sum C_j$ and $1|d_j = D|\sum(wE_j + w'T_j)$ as assignment problems, thus giving polynomial time algorithms for these problems. These are the only known polynomially solvable problems for scheduling with discretely controllable processing times.

Vickson [548] shows that P1 for problem $1||T_{\max}$ is $\mathcal{NP}$-hard. Chen, Lu & Tang [100] prove that P1 for each of the problems $1|r_j|C_{\max}$, $1|d_j = d|T_{\max}$ and $1|d_j = d|\sum w_jU_j$ is $\mathcal{NP}$-hard. They also give pseudo-polynomial algorithms for these problems. The complexity of P1 for problem $1||\sum w_jC_j$ is unknown.

9 Concluding Remarks

In this review, we have displayed results for all of the classical machine scheduling problems, including single machine, parallel machine, open shop, flow shop and job shop models. Some of the well-known variants of these classical models have also been covered.

Our review discusses the complexity of the various models, stating whether they are polynomially solvable, $\mathcal{NP}$-hard, strongly $\mathcal{NP}$-hard, or open. Much progress has been made in the area of complexity classification over the last twenty years, and the number of open problems is fairly small. Examples of problems that have resisted attempts to classify them are $R|pmtn|\sum C_j$ (for which the non-preemptive counterpart is polynomially solvable), and $O3||C_{\max}$ which is $\mathcal{NP}$-hard but is open with respect to pseudo-polynomial solvability.

For the $\mathcal{NP}$-hard problems, we have described a variety of enumerative methods, most of which use branch and bound. Except for the most structured of these problems, the currently available branch and bound algorithms cannot solve problems of practical size. The main difficulty is

deriving a lower bounding scheme that is powerful enough to restrict the search. Even the polyhedral-based approaches that use linear programming to compute lower bounds are not very effective, although they are very successful in other areas of combinatorial optimization.

Research on local search methods has only gained momentum relatively recently. Nevertheless, there is evidence from flow shop and job shop scheduling, where comparative studies of multi-start descent, simulated annealing, threshold accepting, tabu search and genetic algorithms are reported in the literature, that these methods are capable of generating near-optimal solutions at reasonable computational expense. Used as a 'black box' technique, local search methods only perform satisfactorily. However, by incorporating some problem-specific features, and allowing variations in the basic local search technique, the performance often improves dramatically. Our current understanding of why these methods work well in some situations, but not in others, is still very superficial. Further research is needed to produce meaningful guidelines as to what type of local search method should be used for a problem with particular characteristics, and what special features should be incorporated into the method to improve its performance.

Until the last few years, research on approximation algorithms mainly focused on deriving ratio guarantees for problems of minimizing the makespan or maximum lateness. Recent techniques use the solution of a linear programs to guide the construction of the schedule. This approach allows ratio guarantees to be derived for a variety of models involving the total weighted completion time criterion. There are some non-approximability results, which state that a particular ratio guarantee cannot be achieved in polynomial time unless $\mathcal{P}=\mathcal{NP}$. However, there is often a large difference between the ratio indicated in the non-approximability result, and the best available ratio guarantee. An important research topic is the further development of techniques to help close the gap between ratios for non-approximability and approximability.

Acknowledgements

We gratefully acknowledge the comments and suggestions of Edwin Cheng, Han Hoogeveen, Chung-Lun Li, Petra Schuurman, Jiří Sgall, Vitaly Strusevich, Guochun Tang, Steef van de Velde and Wenci Yu. Partial

support for the research by the first author was provided by the Management Research Fellowship of the ESRC (Economic & Social Research Council) of Britain and the Research Initiatives Fund of the Warwick Business School, by the second author was provided by INTAS (Project INTAS-93-257 and INTAS-93-257-Ext), and by the third author was provided by the START program Y43-MAT of the Austrian Ministry of Science.

References

[1] E.H.L. Aarts and J.K. Lenstra (eds.), *Local Search in Combinatorial Optimization*, Wiley, Chichester, 1997.

[2] E.H.L. Aarts, P.J.M. van Laarhoven, J.K. Lenstra and N.L.J. Ulder, A computational study of local search algorithms for job shop scheduling, *ORSA Journal on Computing* 6 (1994), 118-125.

[3] T.S. Abdul-Razaq and C.N. Potts, Dynamic programming state-space relaxation for single-machine scheduling, *Journal of the Operational Research Society* 39 (1988), 141-152.

[4] T.S. Abdul-Razaq, C.N. Potts and L.N. Van Wassenhove, A survey of algorithms for the single machine total weighted tardiness scheduling problem, *Discrete Applied Mathematics* 26 (1990), 235-253.

[5] J.O. Achugbue and F.Y. Chin, Bounds on schedules for independent tasks with similar execution times. *Journal of the Association for Computing Machinery* 28 (1981), 81-99.

[6] J.O. Achugbue and F.Y. Chin, Scheduling the open shop to minimize mean flow time, *SIAM Journal on Computing* 11 (1982), 709-720.

[7] J. Adams, E. Balas and D. Zawack, The shifting bottleneck procedure for job shop scheduling, *Management Science* 34 (1988), 391-401.

[8] B. Adenso-Días, Restricted neighborhood in the tabu search for the flowshop problem, *European Journal of Operational Research* 62 (1992), 27-37.

[9] I. Adiri and N. Aizikowitz, Open shop scheduling problems with dominated machines, *Operations Research, Statistics and Economics Mimeograph Series 383*, Technion, Haifa, Israel, 1986.

[10] D. Adolphson and T.C. Hu, Optimal linear ordering, *SIAM Journal on Applied Mathematics* 25 (1973), 403-423.

[11] R.H. Ahmadi and U. Bagchi, Lower bounds for single-machine scheduling problems, *Naval Research Logistics Quarterly* 37 (1990), 967-979.

[12] R.H. Ahmadi and U. Bagchi, Improved lower bounds for minimizing the sum of completion times on n jobs over m machines in a flow shop, *European Journal of Operational Research* 44 (1990), 331-336.

[13] S.B. Akers, A graphical approach to production scheduling problems, *Operations Research* 4 (1956), 244-245.

[14] S.B. Akers and J. Friedman, A non-numerical approach to production scheduling problems, *Operations Research* 3 (1955), 429-442.

[15] V.A. Aksjonov, A polynomial-time algorithm for an approximate solution of a scheduling problem (in Russian), *Upravlyaemye Sistemy* 28 (1988), 8-11.

[16] S. Albers, Better bounds for online scheduling, *Proceedings of the 29th Annual ACM Symposium on Theory of Computing* (1997), 130-139.

[17] H.H. Ali and H. El-Rewini, An optimal algorithm for scheduling interval ordered tasks with communication on N processors, *Journal of Computing and System Sciences* 51 (1995), 301-306.

[18] B. Alidaee and A. Ahmadian, Two parallel machine sequencing problems involving controllable job processing times, *European Journal of Operational Research* 70 (1993), 335-341.

[19] B. Alidaee and S. Gopalan, A note on the equivalence of two heuristics to minimize total tardiness, *European Journal of Operational Research* 96 (1997), 514-517.

[20] N. Alon, Y. Azar, G.J. Woeginger and T. Yadid, Approximation schemes for scheduling, *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms* (1997), 493-500.

[21] N. Alon, Y. Azar, G.J. Woeginger and T. Yadid, Approximation schemes for scheduling on parallel machines, Technical Report Woe-18, Department of Mathematics, TU Graz, Graz, Austria, 1997. To appear in *Journal of Scheduling*.

[22] A.K. Amoura, E. Bampis, C. Kenyon and Y. Manoussakis, Scheduling independent multiprocessor tasks, *Proceedings of the 5th Annual European Symposium on Algorithms* (1997), 1-12.

[23] E.J. Anderson, C.A. Glass and C.N. Potts, Machine scheduling, in E.H.L. Aarts and J.K. Lenstra (eds.) *Local Search in Combinatorial Optimization*, Wiley, Chichester, 1997, 361-414.

[24] D. Applegate and W. Cook, A computational study of the job-shop scheduling problem, *ORSA Journal on Computing* 3 (1991), 149-156.

[25] J. Aspnes, Y. Azar, A. Fiat, S. Plotkin and O. Waarts, On-line load balancing with applications to machine scheduling and virtual circuit routing, *Journal of the Association for Computing Machinery* 44 (1997), 486-504.

[26] A. Avidor, Y. Azar and J. Sgall, Ancient and new algorithms for load balancing in the L_p norm, *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms* (1998), 426-435.

[27] B. Awerbuch, Y. Azar, E. Grove, M. Kao, P. Krishnan and J. Vitter, Load balancing in the L_p norm, *Proceedings of the 36th IEEE Symposium on Foundations of Computer Science* (1995), 383-391.

[28] Y. Azar, J. Naor and R. Rom, The competitiveness of on-line assignments, *Proceedings of the 3rd Annual ACM-SIAM Symposium on Discrete Algorithms* (1992), 203-210.

[29] U. Bagchi, R.S. Sullivan and Y.-L. Chang, Minimizing mean squared deviations of completion times about a common due date, *Management Science* 33 (1987), 894-906.

[30] K.R. Baker, *Introduction to Sequencing and Scheduling*, Wiley, New York, 1974.

[31] K.R. Baker and J.W. Bertrand, A dynamic priority rule for scheduling against due dates, *Journal of Operations Management* 3 (1982), 37-42.

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

[33] K.R. Baker and G.D. Scudder, Sequencing with earliness and tardiness penalties: a review, *Operations Research* 38 (1990), 22-36.

[34] K.R. Baker and Z.-S. Su, Sequencing with due-dates and early start times to minimize maximum tardiness, *Naval Research Logistics Quarterly* 21 (1974), 171-176.

[35] E. Balas, Project scheduling with resource constraints, in E.M.L. Beale (ed.) *Applications of Mathematical Programming Techniques*, English University Press, London, 1970, 187-200.

[36] E. Balas, On the facial structure of scheduling polyhedra, *Mathematical Programming Study* 24 (1985), 179-218.

[37] E. Balas, J.K. Lenstra and A. Vazacopoulos, The one-machine problem with delayed precedence constraints and its use on job shop scheduling, *Management Science* 41 (1995), 94-109.

[38] E. Balas and A. Vazacopoulos, Guided local search with shifting bottleneck for job shop scheduling, Management Science Research Report MSSR-609, Carnegie Mellon University, Pittsburgh, USA, 1994.

[39] I. Bárány, A vector-sum theorem and its application to improving flow shop guarantees, *Mathematics of Operations Research* 6 (1981), 445-452.

[40] I. Bárány and T. Fiala, Nearly optimum solution of multimachine scheduling problems (in Hungarian), *Szigma Mathematika Közgazdasági Folyóirat* 15 (1982), 177-191.

[41] J.R. Barker and G.B. McMahon, Scheduling the general job-shop, *Management Science* 31 (1985), 594-598.

[42] J.W. Barnes and J.J. Brennan, An improved algorithm for scheduling jobs on identical machines, *AIIE Transactions* 9 (1977), 25-31.

[43] J.W. Barnes and J.B. Chambers, Solving the job shop scheduling problem using tabu search, *IIE Transactions* 27 (1995), 257-263.

[44] Y. Bartal, A. Fiat, H. Karloff and R. Vohra, New algorithms for an ancient scheduling problem, *Journal of Computing and System Sciences* 51 (1995), 359-366.

[45] Y. Bartal, S. Leonardi, A. Marchetti-Spaccamela, J. Sgall and L. Stougie, Multiprocessor scheduling with rejection, *Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms* (1996), 95-103. To appear in *SIAM Journal on Discrete Mathematics*.

[46] M. Bartusch, R.H. Möhring and F.J. Radermacher, Scheduling project networks with resource constraints and time windows, *Annals of Operations Research* 16 (1988), 201-240.

[47] H. Belouadah, M.E. Posner and C.N. Potts, Scheduling with release dates on a single machine to minimize total weighted completion time, *Discrete Applied Mathematics* 36 (1992), 213-231.

[48] H. Belouadah and C.N. Potts, Scheduling identical parallel machines to minimize total weighted completion time, *Discrete Applied Mathematics* 48 (1994), 201-218.

[49] I.S. Belov and J.I. Stolin, An algorithm for the flow shop problem (in Russian), in *Mathematical Economics and Functional Analysis*, Nauka, Moscow, 1974, 248-257.

[50] P. Berman, M. Charikar and M. Karpinski, On-line load balancing for related machines, *Proceedings of the 5th Workshop on Algorithms and Data Structures* (1997), 116-125.

[51] L. Bianco and S. Ricciardelli, Scheduling a single machine to minimize total weighted completion time subject to release dates, *Naval Research Logistics Quarterly* 29 (1982), 151-167.

[52] J. Błażewicz, W. Cellary, R. Słowiński and J. Węglarz, *Scheduling under Resource Constraints–Deterministic Models*, (Annals of Operations Research, Volume 7), J.C. Baltzer AG, Basel, Switzerland, 1986.

[53] J. Błażewicz, M. Drabowski and J. Węglarz, Scheduling multiprocessor tasks to minimize schedule length, *IEEE Transactions on Computing* 35 (1986), 389-393.

[54] J. Błażewicz, M. Drozdowski, G. Schmidt and D. de Werra, Scheduling independent two processor tasks on a uniform k-processor system, *Discrete Applied Mathematics* 28 (1990), 11-20.

[55] J. Błażewicz, K.H. Ecker, G. Schmidt and J. Węglarz, *Scheduling in Computer and Manufacturing System*, Springer-Verlag, Berlin, 1994.

[56] J. Błażewicz, J.K. Lenstra and A.H.G. Rinnooy Kan, Scheduling subject to resource constraints: Classification and complexity, *Discrete Applied Mathematics* 5 (1983), 11-24.

[57] H. Bräsel, T. Tautenhahn and F. Werner, Constructive heuristic algorithms for the open shop problem, *Computing* 51 (1993), 95-110.

[58] P. Bratley, M. Florian and P. Robillard, On sequencing with earliest starts and due dates with application to computing bounds for the $(n/m/G/F_{\max})$ problem, *Naval Research Logistics Quarterly* 20 (1973), 57-67.

[59] P. Bratley, M. Florian and P. Robillard, Scheduling with earliest start and due date constraints on multiple machines, *Naval Research Logistics Quarterly* 22 (1975), 165-173.

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

[61] P. Brucker, A polynomial time algorithm for the two machine job-shop scheduling problem with a fixed number of jobs, *OR Spektrum* 16 (1994), 5-7.

[62] P. Brucker, J. Hurink, B. Jurisch and B. Wöstmann, A branch & bound algorithm for the open-shop problem, *Discrete Applied Mathematics* 76 (1997), 43-59.

[63] P. Brucker, J. Hurink and F. Werner, Improved local search heuristics for some scheduling problems. Part I, *Discrete Applied Mathematics* 65 (1996), 97-122.

[64] P. Brucker, J. Hurink and F. Werner, Improved local search heuristics for some scheduling problems. Part II, *Discrete Applied Mathematics* 72 (1997), 47-69.

[65] P. Brucker and B. Jurisch, A new lower bound for the job-shop scheduling problem, *European Journal of Operational Research* 64 (1993), 156-167.

[66] P. Brucker, B. Jurisch and M. Jurisch, Open shop problems with unit time operations, *ZOR – Mathematical Methods of Operations Research* 37 (1993), 59-73.

[67] P. Brucker, B. Jurisch and A. Krämer, The job-shop problem and immediate selection, *Annals of Operations Research* 50 (1994), 73-114.

[68] P. Brucker, B. Jurisch and B. Sievers, A branch & bound algorithm for the job-shop scheduling problem, *Discrete Applied Mathematics* 49 (1994), 107-127.

[69] P. Brucker, S.A. Kravchenko and Y.N. Sotskov, On the complexity of two machine job-shop scheduling with regular objective functions, *OR Spektrum* 19 (1997), 5-10.

[70] P. Brucker, S.A. Kravchenko and Y.N. Sotskov, Preemptive job-shop scheduling problems with a fixed number of jobs, Osnabrücker Schriften zur Mathematik, Heft 184, Universität Osnabrück, Germany, 1997.

[71] J.L. Bruno, E.G. Coffman, Jr., and R. Sethi, Scheduling independent tasks to reduce mean finishing time, *Communications of the ACM* 17 (1974), 382-387.

[72] J.L. Bruno, E.G. Coffman, Jr., and R. Sethi, Algorithms for minimizing mean flow time, *Proceedings of the IFIP Congress*, North-Holland, Amsterdam, 1974, 504-510.

[73] J.L. Bruno and P.J. Downey, Complexity of task sequencing with deadlines, set-up times and changeover costs, *SIAM Journal on Computing* 7 (1978), 393-404.

[74] J.L. Bruno and T. Gonzalez, Scheduling independent tasks with release dates and due dates on parallel machines, Technical Report 213, Computer Science Department, Pennsylvania State University, USA, 1976.

[75] H. Buer & R.H. Möhring, A fast algorithm for the decomposition of graphs and posets, *Mathematics of Operations Research* 8 (1983), 170-184.

[76] X. Cai, Minimization of agreeably weighted variance in single-machine systems, *European Journal of Operational Research* 85 (1995), 576-592.

[77] X. Cai, C.-Y. Lee and C.-L. Li, Minimizing total completion time in two-processor task systems with prespecified processor allocations, *Naval Research Logistics* 45 (1998), 231-242.

[78] H.G. Campbell, R.A. Dudek and M.L. Smith A heuristic algorithm for the n job, m machine sequencing problem, *Management Science* 16B (1970), 630-637.

[79] J. Carlier, The one-machine sequencing problem, *European Journal of Operational Research* 11 (1982), 42-47.

[80] J. Carlier, Scheduling jobs with release dates and tails on identical machines to minimize the makespan, *European Journal of Operational Research* 29 (1987), 298-306.

[81] J. Carlier and E. Pinson, An algorithm for solving the job-shop problem, *Management Science* 35 (1989), 164-176.

[82] J. Carlier and E. Pinson, A practical use of Jackson's preemptive schedule for solving the job shop problem, *Annals of Operations Research* 26 (1990), 269-287.

[83] J. Carlier and E. Pinson, Adjustment of heads and tails for the job-shop problem, *European Journal of Operational Research* 78 (1994), 146-161.

[84] S. Chakrabarti, C.A. Phillips, A.S. Schulz, D.B. Shmoys, C. Stein, and J. Wein, Improved scheduling algorithms for minsum criteria, *Proceedings of the 23rd International Colloquium on Automata, Languages and Programming* (1996), 646-657.

[85] L.M.A. Chan, P. Kaminsky, A. Muriel and D. Simchi-Levi, Machine scheduling, linear programming and list scheduling heuristics, Manuscript, Department of Industrial Engineering, Northwestern University, Evanston, USA, 1995.

[86] S. Chang, Q. Lu, G. Tang and W. Yu, On decomposition of the total tardiness problem, *Operations Research Letters* 17 (1995), 221-229.

[87] S. Chang, H. Matsuo and G. Tang, Worst-case analysis of local search heuristics for the one-machine total tardiness problem, *Naval Research Logistics Quarterly* 37 (1990), 111-121.

[88] J.M. Charlton and C.C. Death A generalized machine scheduling algorithm, *Operational Research Quarterly* 21 (1970), 127-134.

[89] C. Chekuri, R. Motwani, B. Natarajan and C. Stein, Approximation techniques for average completion time scheduling, *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms* (1997), 609-618.

[90] B. Chen, Tighter bounds for MULTIFIT scheduling on uniform processors, *Discrete Applied Mathematics* 31 (1991), 227-260.

[91] B. Chen, A better heuristic for preemptive parallel machine scheduling with batch setup times, *SIAM Journal on Computing* 22 (1993), 1303-1318.

[92] B. Chen, C.N. Potts and V.A. Strusevich, Approximation algorithms for two-machine flow shop scheduling with batch setup times, *Mathematical Programming* (1998), to appear.

[93] B. Chen and V.A. Strusevich, Approximation algorithms for three-machine open shop scheduling, *ORSA Journal on Computing* 5 (1993), 321-326.

[94] B. Chen, A. van Vliet and G.J. Woeginger, New lower and upper bounds for on-line scheduling, *Operations Research Letters* 16 (1994), 221-230.

[95] B. Chen, A. van Vliet and G.J. Woeginger, An optimal algorithm for preemptive on-line scheduling, *Operations Research Letters* 18 (1995), 127-131.

[96] B. Chen and A. Vestjens, Scheduling on identical machines: How good is LPT in an on-line setting? *Operations Research Letters* 21 (1998), 165-169.

[97] B. Chen, A.P.A. Vestjens and G.J. Woeginger, On-line scheduling of two-machine open shops where jobs arrive over time, *Journal of Combinatorial Optimization* 1 (1997), 355-365.

[98] B. Chen and G.J. Woeginger, A study of on-line scheduling two-stage shops, in D.-Z. Du and P.M. Pardalos (eds.) *Minimax and Applications*, Kluwer Academic Publishers, 1995, 97-107.

[99] C.-L. Chen and R.L. Bulfin, Complexity of single machine, multi-criteria scheduling problems, *European Journal of Operational Research* 70 (1993), 115-125.

[100] Z.L. Chen, Q. Lu and G. Tang, Single machine scheduling with discretely controllable processing times, *Operations Research Letters* 21 (1997), 69-76.

[101] Z.L. Chen and W.B. Powell, Solving parallel machine total weighted completion time problems by column generation, Manuscript, Department of Civil Engineering and Operations Research, Princeton University, Princeton, USA, 1995.

[102] T.C.E. Cheng, Optimal common due date with limited completion time deviation, *Computers and Operations Research* 15 (1988), 91-96.

[103] T.C.E. Cheng, A note on the equivalence of the Wilkerson-Irwin and Modified Due-Date rules for the mean tardiness sequencing problem, *Computers and Industrial Engineering* 22 (1992), 63-66.

[104] T.C.E. Cheng and C.C.S. Sin, A state-of-the-art review of parallel-machine scheduling research, *European Journal of Operational Research* 47 (1990), 271-292.

[105] Y. Cho and S. Sahni, Bounds for list schedules on uniform processors, *SIAM Journal on Computing* 9 (1980), 91-103.

[106] Y. Cho and S. Sahni, Preemptive scheduling of independent jobs with release and due times on open, flow and job shops, *Operations Research* 29 (1981), 511-522.

[107] P. Chrétienne and C. Picouleau, Scheduling with communication delays: a survey, in P. Chrétienne, E.G. Coffman, J.K. Lenstra and Z.Liu (eds.) *Scheduling theory and its applications*, Chichester, John Wiley & Sons, 1995, 65-90.

[108] C. Chu, A branch-and-bound algorithm to minimize total tardiness with different release dates, *Naval Research Logistics* 39 (1992), 265-283.

[109] C. Chu, A branch-and-bound algorithm to minimize total flow time with unequal release dates, *Naval Research Logistics* 39 (1992), 859-875.

[110] F. Chudak and D.B. Shmoys, Approximation algorithms for precedence-constrained scheduling problems on parallel machines that run at different speeds, *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms* (1997), 581-590.

[111] E.G. Coffman, Jr., M.R. Garey and D.S. Johnson, An application of bin-packing to multiprocessor scheduling, *SIAM Journal on Computing* 7 (1978), 1-17.

[112] J.-Y. Colin and P. Chrétienne, CPM scheduling with small communication delays. *Operations Research* 39 (1991), 680-684.

[113] R.K. Congram, C.N. Potts and S.L. van de Velde, Dynasearch—iterative local improvement by dynamic programming: The total weighted tardiness problem, Preprint, Faculty of Mathematical Studies, University of Southampton, Southampton, UK, 1998.

[114] R.W. Conway, W.L. Maxwell and L.W. Miller, *Theory of Scheduling*, Addison-Wesley, Reading, 1967.

[115] H.A.J. Crauwels, A.MA. Hariri, C.N. Potts and L.N. Van Wassenhove, Branch and bound algorithms for single machine scheduling with batch set-up times to minimize total weighted completion time, *Annals of Operations Research*, to appear.

[116] H.A.J. Crauwels, C.N. Potts and L.N. Van Wassenhove, Local search heuristics for single-machine scheduling with batching to minimize the number of late jobs, *European Journal of Operational Research* 90 (1996), 200-213.

[117] H.A.J. Crauwels, C.N. Potts and L.N. Van Wassenhove, Local search heuristics for single machine scheduling with batch set-up times to minimize total weighted completion time, *Annals of Operations Research* 70 (1997), 261-279

[118] H.A.J. Crauwels, C.N. Potts and L.N. Van Wassenhove, Local search heuristics for the single machine total weighted tardiness scheduling problem, *INFORMS Journal on Computing*, to appear.

[119] H.A.J. Crauwels, C.N. Potts and L.N. Van Wassenhove, Local search heuristics for single machine scheduling with batching to minimize total weighted completion time, Unpublished manuscript, 1995.

[120] D.G. Dannenbring, An evaluation of flow-shop sequencing heuristics, *Management Science* 23 (1977), 1174-1182.

[121] S. Dauzère-Pérès and J.-B. Lasserre, A modified shifting bottleneck procedure for job-shop scheduling, *International Journal of Production Research* 31 (1993), 923-932.

[122] S. Dauzère-Pérès, Minimizing late jobs in the general one machine scheduling problem, *European Journal of Operational Research* 81 (1995), 134-142.

[123] E.W. Davis, Resource allocation in project network models–A survey, *Journal of Industrial Engineering* 17 (1966), 177-188.

[124] E.W. Davis, Project scheduling under resource constraints–Historical review and categorization of procedures, *AIIE Transactions* 5 (1973), 297- 313.

[125] E.W. Davis and G.E. Heidorn, An algorithm for optimal project scheduling under multiple resource constraints. *Management Science* 17 (1971), 803-816.

[126] E. Davis and J.M. Jaffe, Algorithms for scheduling tasks on unrelated processors, *Journal of the Association for Computing Machinery* 28 (1981), 721-736.

[127] J.S. Davis and J.J. Kanet, Single-machine scheduling with early and tardy completion costs, *Naval Research Logistics* 40 (1993), 85-101.

[128] P. De, J.B. Ghosh and C.E. Wells, On the minimization of completion time variance with a bicriteria extension, *Operations Research* 40 (1992), 1148-1155.

[129] F. Della Croce, V. Narayan and R. Tadei, The two-machine total completion time flow shop problem, *European Journal of Operational Research*, to appear.

[130] F. Della Croce, R. Tadei and G. Volta, A genetic algorithm for the job shop problem, *Computers and Operations Research* 22 (1995), 15-24.

[131] F. Della Croce, W. Szwarc, R. Tadei, P. Baracco and R. Di Tullio, Minimizing the weighted sum of quadratic completion times on a single machine, *Naval Research Logistics* 42 (1995), 1263-1270.

[132] F. Della Croce, R. Tadei, P. Baracco and A. Grosso, A new decomposition approach for the single machine total tardiness problem, *Journal of the Operational Research Society*, to appear.

[133] M. Dell'Amico and S. Martello, Optimal scheduling of tasks on identical parallel processors, *ORSA Journal on Computing* 7 (1995), 191-200

[134] M. Dell'Amico and M. Trubian, Applying tabu-search to the job-shop scheduling problem, *Annals of Operations Research* 41 (1993), 231-252.

[135] E. Demeulemeester and W. Herroelen, A branch-and-bound procedure for the multiple resource-constrained project scheduling problem, *Management Science* 38 (1992), 1803-1818.

[136] E. Demeulemeester and W. Herroelen, A branch-and-bound procedure for the generalized resource-constrained project scheduling problem, *Operations Research* 45 (1997), 201-212.

[137] J.S. Deogun, On scheduling with ready times to minimize mean flow time, *The Computer Journal* 26 (1983), 320-328.

[138] M.I. Dessouky and J.S. Deogun, Sequencing jobs with unequal ready times to minimize mean flow time, *SIAM Journal on Computing* 10 (1981), 192-202.

[139] M.I. Dessouky, B.J. Lageweg, J.K. Lenstra and S.L. van de Velde, Scheduling identical jobs on uniform parallel machines, *Statistica Neerlandica* 44 (1990), 115-123.

[140] P. Dileepan and T. Sen, Bicriterion static scheduling research for a single machine, *OMEGA* 16 (1988), 53-59.

[141] G. Dobson, Scheduling independent tasks on uniform processors. *SIAM Journal on Computing* 13 (1984), 705-716.

[142] U. Dorndorf and E. Pesch, Evolution based learning in a job shop scheduling environment, *Computers and Operations Research* 22 (1993), 25-40.

[143] I.G. Drobouchevitch and V.A. Strusevich, Heuristics for the two-stage job shop scheduling problem with a bottleneck machine, CASSM R&D Paper 11, Centre for Applied Statistics and Systems Modelling, University of Greenwich, London, UK, 1997.

[144] I.G. Drobouchevitch and V.A. Strusevich, A polynomial algorithm for the three machine open shop with a bottleneck machine, CASSM R&D Paper 13, Centre for Applied Statistics and Systems Modelling, University of Greenwich, London, UK, 1997.

[145] M. Drozdowski, On the complexity of multiprocessor task scheduling, *Bulletin of the Polish Academy of Sciences. Technical Sciences* 43 (1995), 381-392.

[146] M. Drozdowski, Scheduling multiprocessor tasks - An overview, *European Journal of Operational Research* 94 (1996), 215-230.

[147] J. Du and J.Y.-T. Leung, Complexity of scheduling parallel task systems, *SIAM Journal on Discrete Mathematics* 2 (1989), 473-487.

[148] J. Du and J.Y.-T. Leung, Minimizing total tardiness on one processor is NP-hard, *Mathematics of Operations Research* 15 (1990), 483-495.

[149] J. Du and J.Y.-T. Leung, Minimizing mean flow time with release time and deadline constraints, *Journal of Algorithms* 14 (1993), 45-68.

[150] J. Du and J.Y.-T. Leung, Minimizing mean flow time in two-machine open shops and flow shops, *Journal of Algorithms* 14 (1993), 341-364.

[151] J. Du, J.Y.-T. Leung and C.S. Wong, Minimizing the number of late jobs with release time constraints, Technical Report, Computer Science Program, University of Texas, Dallas, USA, 1989.

[152] J. Du, J.Y.-T. Leung and G.H. Young, Minimizing mean flow time with release time constraints, Technical Report, Computer Science Program, University of Texas, Dallas, USA, 1988.

[153] J. Du, J.Y.-T. Leung and G.H. Young, Scheduling chain-structured tasks to minimize makespan and mean flow time, *Information and Computation* 92 (1991), 219-236.

[154] M.E. Dyer and L.A. Wolsey, Formulating the single machine sequencing problem with release dates as a mixed integer program, *Discrete Applied Mathematics* 26 (1990), 255-270.

[155] W.L. Eastman, S. Even and I.M. Isaacs, Bounds for the optimal scheduling of n jobs on m processors, *Management Science* 11 (1964), 268-279.

[156] J. Edmonds, D.D. Chinn, T. Brecht and X. Deng, Non-clairvoyant multiprocessor scheduling of jobs with changing execution characteristics, *Proceedings of the 29th Annual ACM Symposium on Theory of Computing* (1997), 120-129.

[157] S. Eilon and I.G. Chowdhury, Minimizing waiting time variance in the single machine problem, *Management Science* 23 (1977), 567-575.

[158] S.E. Elmaghraby, The one-machine sequencing problem with delay costs, *Journal of Industrial Engineering* 19 (1968), 105-108.

[159] S.E. Elmaghraby and S.H. Park, Scheduling jobs on a number of identical machines, *AIIE Transactions* 6 (1974), 1-12.

[160] H. Emmons, One-machine sequencing to minimize certain functions of job tardiness, *Operations Research* 17 (1969), 701-715.

[161] L. Epstein, J. Noga, S.S. Seiden, J. Sgall and G.J. Woeginger, On-line scheduling for two related machines, unpublished manuscript, 1997.

[162] P. Erdős and L. Lovász, Problems and results on 3-chromatic hypergraphs and some related questions, In: *Infinite and Finite Sets* (A. Hajnal, R. Rado and V.T. Sós, eds.), North-Hollnd, Amsterdam, 1975. 609-628.

[163] U. Faigle, W. Kern and G. Turan, On the performance of on-line algorithms for partition problems, *Acta Cybernetica* 9 (1989), 107-119.

[164] A. Federgruen and H. Groenevelt, Preemptive scheduling of uniform machines by ordinary network flow techniques, *Management Science* 32 (1986), 341-349.

[165] U. Feige and C. Scheideler, Improved bounds for acyclic job shop scheduling, manuscript, Weizmann Institute, Rehovot, Israel, 1997.

[166] A. Feldmann, M.-Y. Kao, J. Sgall and S.-H. Teng, Optimal online scheduling of parallel jobs with dependencies, *Proceedings of the 25th Annual ACM Symposium on Theory of Computing* (1993), 642-651.

[167] A. Feldmann, J. Sgall and S.-H. Teng, Dynamic scheduling on parallel machines, *Theoretical Computer Science* 130 (1994), 49-72.

[168] T. Fiala, An algorithm for the open-shop problem, *Mathematics of Operations Research* 8 (1983), 100-109.

[169] A. Fiat and G.J. Woeginger, On-line scheduling on a single machine: Minimizing the total completion time, Technical Report Woe-04, Department of Mathematics, TU Graz, Graz, Austria, 1997.

[170] G. Finn and E. Horowitz, A linear time approximation algorithm for multiprocessor scheduling, *BIT* 19 (1979), 312-320.

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

[172] M.L. Fisher, A dual algorithm for the one-machine scheduling problem, *Mathematical Programming* 11 (1976), 229-251.

[173] M.L. Fisher, B.J. Lageweg, J.K. Lenstra and A.H.G. Rinnooy Kan, Surrogate duality relaxation for job shop scheduling, *Discrete Applied Mathematics* 5 (1983), 65-75.

[174] P.M. França, M. Gendreau, G. Laporte and F.M. Müller, A composite heuristic for the identical parallel machine scheduling problem with minimum makespan objective, *Computers and Operations Research* 21 (1994), 205-210.

[175] G.M. Frederickson, Scheduling unit-time tasks with integer release times and deadlines, *Information Processing Letters* 16 (1983), 171-173.

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

[177] D.K. Friesen, Tighter bounds for the multifit processor scheduling algorithm, *SIAM Journal on Computing* 13 (1984), 170-181.

[178] D.K. Friesen, Tighter bounds for LPT scheduling on uniform processors, *SIAM Journal on Computing* 16 (1987), 554-660.

[179] D.K. Friesen and M.A. Langston, Bounds for multifit scheduling on uniform processors, *SIAM Journal on Computing* 12 (1983), 60-70.

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

[181] T.D. Fry, R.D. Armstrong and H. Lewis, A framework for single machine multiple objective sequencing research, *OMEGA* 17 (1989), 595-607.

[182] T.D. Fry, L. Vicens, K. MacLeod and S. Fernandez, A heuristic solution procedure to minimize $\overline{T}$ on a single-machine, *Journal of the Operational Research Society* 40 (1989), 293-297.

[183] M. Fujii, T. Kasami and K. Ninomiya, Optimal sequencing of two equivalent processors, *SIAM Journal on Applied Mathematics* 17 (1969), 784-789. Erratum: *SIAM Journal on Applied Mathematics* 20 (1971), 141.

[184] G. Galambos and G.J. Woeginger, An on-line scheduling heuristic with better worst case ratio than Graham's List Scheduling, *SIAM Journal on Computing* 22 (1993), 349-355. Erratum: R. Chandrasekaran, B. Chen, G. Galambos, P.R. Narayanan, A. van Vliet and G.J. Woeginger, A note on "An on-line scheduling heuristic with better worst case ratio than Graham's List Scheduling", *SIAM Journal on Computing* 26 (1997), 870-872.

[185] M.R. Garey and D.S. Johnson, Complexity results for multiprocessor scheduling under resource constraints, *SIAM Journal on Computing* 4 (1975), 397-411.

[186] M.R. Garey and D.S. Johnson, Scheduling tasks with nonuniform deadlines on two processors, *Journal of the Association for Computing Machinery* 23 (1976), 461-467.

[187] M.R. Garey and D.S. Johnson, Two-processor scheduling with start-times and deadlines, *SIAM Journal on Computing* 6 (1977), 416-426.

[188] M.R. Garey and D.S. Johnson, Strong NP-completeness results: motivation, examples and implications, *Journal of the Association for Computing Machinery* 25 (1978), 499-508.

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

[190] M.R. Garey, D.S. Johnson and R. Sethi, The complexity of flowshop and jobshop scheduling, *Mathematics of Operations Research* 1 (1976), 117-129.

[191] M.R. Garey, D.S. Johnson, B.B. Simons and R.E. Tarjan, Scheduling unit-time tasks with arbitrary release dates and deadlines, *SIAM Journal on Computing* 10 (1981), 256-269.

[192] M.R. Garey, D.S. Johnson, R.E. Tarjan and M. Yannakakis, Scheduling opposing forests, *SIAM Journal on Algebraic Discrete Mathematics* 4 (1983), 72-93.

[193] M.R. Garey, R.E. Tarjan and G.T. Wilfong, One-processor scheduling with symmetric earliness and tardiness penalties, *Mathematics of Operations Research* 13 (1988), 330-348.

[194] L. Gelders and P.R. Kleindorfer, Coordinating aggregate and detailed scheduling decisions in the one-machine job shop: Part I. Theory, *Operations Research* 22 (1974), 46-60.

[195] L. Gelders and P.R. Kleindorfer, Coordinating aggregate and detailed scheduling decisions in the one-machine job shop: II–Computation and structure, *Operations Research* 23 (1975), 312-324.

[196] G.V. Gens and E.V. Levner, Approximation algorithms for certain universal problems in scheduling theory, *Engineering Cybernetics* 16 (1978), 31-36.

[197] G.V. Gens and E.V. Levner, Fast approximation algorithm for job sequencing with deadlines. *Discrete Applied Mathematics* 3 (1981), 313-318.

[198] J.B. Ghosh, Batch scheduling to minimize total completion time, *Operations Research Letters* 16 (1994), 271-275.

[199] J.B. Ghosh and J.N.D. Gupta, Batch scheduling to minimize maximum lateness, *Operations Research Letters* 21 (1997),, 77-80.

[200] B. Giffler and G.L. Thompson, Algorithms for solving production-scheduling problems, *Operations Research* 8 (1960), 487-503.

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

[202] C.A. Glass, J.N.D. Gupta and C.N. Potts, Two-machine no-wait flow shop scheduling with missing operations, Preprint OR75, Faculty of Mathematical Studies, University of Southampton, Southampton, UK, 1995.

[203] C.A. Glass and C.N. Potts, A comparison of local search methods for permutation flow shop scheduling, *Annals of Operations Research* 63 (1996), 489-509.

[204] C.A. Glass, C.N. Potts and P. Shade, Unrelated parallel machine scheduling using local search, *Mathematical and Computer Modelling* 20 (1994), 41-52.

[205] J.N.D. Gupta and S.S. Reddi, Improved dominance conditions for the three-machine flowshop scheduling problem, *Operations Research* 26 (1978), 200-203.

[206] M.X. Goemans, Improved approximation algorithms for scheduling with release dates, *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms* (1997), 591-598.

[207] M.X. Goemans and J. Kleinberg, An improved approximation ratio for the minimum latency problem, *Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms* (1996), 152-157.

[208] L.A. Goldberg, M. Paterson, A. Srinivasan and E. Sweedyk, Better approximation guarantees for job shop scheduling, *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms* (1997), 599-608.

[209] T. Gonzalez, Optimal mean finish time preemptive schedules, Technical Report 220, Computer Science Department, Pennsylvania State University, USA, 1977.

[210] T. Gonzalez, A note on open shop preemptive schedules, Unit execution time shop problems, *IEEE Transactions on Computing* 28 (1979), 782-786.

[211] T. Gonzalez, Unit execution time shop problems, *Mathematics of Operations Research* 7 (1982), 57-66.

[212] T. Gonzalez, O.H. Ibarra and S. Sahni, Bounds for LPT schedules on uniform processors, *SIAM Journal on Computing* 6 (1977), 155-166.

[213] T. Gonzalez and D.B. Johnson, A new algorithm for preemptive scheduling of trees, *Journal of the Association for Computing Machinery* 27 (1980), 287-312.

[214] T. Gonzalez, E.L. Lawler and S. Sahni, Optimal preemptive scheduling of two unrelated processors, *ORSA Journal on Computing* 2 (1990), 219-224.

[215] T. Gonzalez and S. Sahni, Open shop scheduling to minimize finish time, *Journal of the Association for Computing Machinery* 23 (1976), 665-679.

[216] T. Gonzalez and S. Sahni, Flow shop and job shop schedules: complexity and approximation, *Operations Research* 26 (1978), 36-52.

[217] T. Gonzalez and S. Sahni, Preemptive scheduling of uniform processor systems, *Journal of the Association for Computing Machinery* 25 (1978), 92-101.

[218] V.S. Gordon and V.S. Tanaev On minimax problems of scheduling theory for a single machine (in Russian), *Vetsi Akademii Navuk BSSR. Ser. fizika-matematychnykh navuk* (1983) 3-9.

[219] S.K. Goyal and C. Sriskandarajah, No-wait scheduling: computational complexity and approximation algorithms, *Opsearch* 25 (1988), 220-244.

[220] J. Grabowski, On two-machine scheduling with release and due dates to minimize maximum lateness, *Opsearch* 17 (1980), 133-154.

[221] J. Grabowski, A new algorithm of solving the flow-shop problem, in G. Feichtinger and P. Kall (eds.), *Operations Research in Progress*, Reidel, Dordrecht, 1982, 57-75.

[222] J. Grabowski and A. Janiak, Job-shop scheduling with resource-time models of operations, *European Journal of Operational Research* 28 (1986), 58-73.

[223] J. Grabowski, E. Nowicki and S. Zdrzałka, A block approach for single-machine scheduling with release dates and due dates, *European Journal of Operational Research* 26 (1986), 278-285.

[224] J. Grabowski, E. Skubalska and C. Smutnicki, On flow shop scheduling with release and due dates to minimize maximum lateness, *Journal of the Operational Research Society* 34 (1983), 615-620.

[225] R.L. Graham, Bounds for certain multiprocessing anomalies, *Bell System Technical Journal* 45 (1966), 1563-1581.

[226] R.L. Graham, Bounds on multiprocessing timing anomalies, *SIAM Journal on Applied Mathematics* 17 (1969), 416-429.

[227] R.L. Graham, unpublished manuscript, see [335].

[228] R.L. Graham, E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey, *Annals of Operations Research* 5 (1979), 287-326.

[229] M.C. Gupta, Y.P. Gupta and A. Kumar, Minimizing flow time variance in a single machine system using genetic algorithms, *European Journal of Operational Research* 70 (1993), 289-303.

[230] D. Gusfield, Bounds for naive multiple machine scheduling with release times and deadlines, *Journal of Algorithms* 5 (1984), 1-6.

[231] L.A. Hall, A polynomial time approximation scheme for a constrained flow-shop scheduling problem, *Mathematics of Operations Research* 19 (1994), 68-85.

[232] L.A. Hall, Approximability of flow shop scheduling, *Proceedings of the 36th IEEE Symposium on Foundations of Computer Science* (1995), 82-91.

[233] L.A. Hall, A.S. Schulz, D.B. Shmoys and J. Wein, Scheduling to minimize average completion time: Off-line and on-line approximation algorithms, *Mathematics of Operations Research* 22 (1997), 513-544.

[234] L.A. Hall and D.B. Shmoys, Approximation algorithms for constrained scheduling problems, *Proceedings of the 30th IEEE Symposium on Foundations of Computer Science* (1989), 134-139.

[235] L.A. Hall and D.B. Shmoys, Jackson's rule for one-machine scheduling: making a good heuristic better, *Mathematics of Operations Research* 17 (1992), 22-35.

[236] L.A. Hall, D.B. Shmoys and J. Wein, Scheduling to minimize average completion time: Off-line and on-line algorithms, *Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms* (1996), 142-151.

[237] N.G. Hall, W. Kubiak and S.P. Sethi, Earliness-tardiness scheduling problems, II: deviation of completion times about a restrictive common due date, *Operations Research* 39 (1991), 847-856.

[238] N.G. Hall and M.E. Posner, Earliness-tardiness scheduling problems, I: weighted deviation of completion times about a common due date, *Operations Research* 39 (1991), 836-846.

[239] N.G. Hall and C. Sriskandarajah, A survey of machine scheduling with blocking and no-wait in process, *Operations Research* 44 (1996), 510-525.

[240] A.M.A. Hariri and C.N. Potts, An algorithm for single machine sequencing with release dates to minimize total weighted completion time, *Discrete Applied Mathematics* 5 (1983), 99-109.

[241] A.M.A. Hariri and C.N. Potts, Algorithms for two-machine flow-shop sequencing with precedence constraints, *European Journal of Operational Research* 17 (1984), 238-248.

[242] A.M.A. Hariri and C.N. Potts, A branch and bound algorithm to minimize the number of late jobs in a permutation flow-shop, *European Journal of Operational Research* 38 (1989), 228-237.

[243] A.M.A. Hariri and C.N. Potts, Heuristics for scheduling unrelated parallel machines. *Computers and Operations Research* 18 (1991), 323-331.

[244] A.M.A. Hariri and C.N. Potts, Single machine scheduling with deadlines to minimize the weighted number of tardy jobs. *Management Science* 40 (1994), 1712-1719.

[245] A.M.A. Hariri and C.N. Potts, Single machine scheduling with batch set-up times to minimize maximum lateness, *Annals of Operations Research* 70 (1997), 75-92.

[246] A.M.A. Hariri, C.N. Potts and L.N. Van Wassenhove, Single machine scheduling to minimize total weighted late work, *ORSA Journal on Computing* 7 (1995), 232-242.

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

[248] N. Hefetz and I. Adiri, An efficient optimal algorithm for the two-machines, unit-time, job shop, schedule-length problem, *Mathematics of Operations Research* 7 (1982), 354-360.

[249] W.S. Herroelen, Resource-constrained project scheduling–the state of the art, *Operational Research Quarterly* 23 (1972), 261-275.

[250] W.S. Herroelen and E.L. Demeulemeester, Recent advances in branch-and-bound procedures for resource-constrained project scheduling problems, in P. Chrétienne et al. (eds.) *Proceedings of the Summer School on Scheduling Theory and Its Applications*, John Wiley & Sons, Chichester, 1995, 259-276.

[251] D.S. Hochbaum and D.B. Shmoys, Using dual approximation algorithms for scheduling problems: Theoretical and practical results, *Journal of the Association for Computing Machinery* 34 (1987), 144-162.

[252] D.S. Hochbaum and D.B. Shmoys, A polynomial approximation scheme for machine scheduling on uniform processors: Using the dual approximating approach, *SIAM Journal on Computing* 17 (1988), 539-551.

[253] J.E. Holsenback and R.M. Russell, A heuristic algorithm for sequencing on one machine to minimize total tardiness, *Journal of the Operational Research Society* 43 (1992), 53-62.

[254] K.S. Hong and J.Y.-T. Leung, Preemptive scheduling with release times and deadlines, *Journal of Real-Time Systems* 1 (1989), 265-281.

[255] K.S. Hong and J.Y.-T. Leung, On-line scheduling of real-time tasks, *IEEE Transactions on Computing* 41 (1992), 1326-1331.

[256] J.A. Hoogeveen, Single-machine bicriteria scheduling, *Ph.D. thesis*, Center for Mathematics and Computer Science, Amsterdam, The Netherlands, 1992.

[257] J.A. Hoogeveen, Minimizing maximum promptness and maximum lateness on a single machine, *Mathematics of Operations Research* 21 (1996), 100-114.

[258] J.A. Hoogeveen, Single-machine scheduling to minimize a function or two or three maximum cost criteria, *Journal of Algorithms* 21 (1996), 415-433.

[259] J.A. Hoogeveen and T. Kawaguchi, Minimizing total completion time in a two-machine flowshop: Analysis of special cases, *Proceedings of the 5th IPCO Conference* (1996), 374-388.

[260] J.A. Hoogeveen, J.K. Lenstra and B. Veltman, Three, four, five, six, or the complexity of scheduling with communication delays, *Operations Research Letters* 16 (1994), 129-137.

[261] J.A. Hoogeveen, J.K. Lenstra and B. Veltman, Preemptive scheduling in a two-stage multiprocessor flow shop is NP-hard, *European Journal of Operational Research* 89 (1996), 172-175.

[262] J.A. Hoogeveen, H. Oosterhout and S.L. van de Velde, New lower and upper bounds for scheduling around a small common due date, *Operations Research* 42 (1994), 102-110.

[263] J.A. Hoogeveen, P. Schuurman and G.J. Woeginger, Non-approximability results for scheduling problems with minsum criteria, Technical Report Woe-15, Department of Mathematics, TU Graz, Graz, Austria, 1997.

[264] J.A. Hoogeveen and S.L. van de Velde, Scheduling around a small common due date, *European Journal of Operational Research* 55 (1991), 237–242.

[265] J.A. Hoogeveen and S.L. van de Velde, Minimizing total completion time and maximum cost simultaneously is solvable in polynomial time. *Operations Research Letters* 17 (1995), 205-208.

[266] J.A. Hoogeveen and S.L. van de Velde, Stronger Lagrangian bounds by use of slack variables: applications to machine scheduling problems. *Mathematical Programming* 70 (1995), 173-190.

[267] J.A. Hoogeveen and S.L. van de Velde, A branch-and-bound algorithm for single-machine earliness-tardiness scheduling with idle time, *INFORMS Journal on Computing* 8 (1996), 402-412.

[268] J.A. Hoogeveen and S.L. van de Velde, Earliness-tardiness scheduling around almost equal due date, *INFORMS Journal on Computing* 9 (1997), 92-99.

[269] J.A. Hoogeveen, S.L. van de Velde and B. Veltman, Complexity of scheduling multiprocessor tasks with prespecified processor allocations, *Discrete Applied Mathematics* 55 (1994), 259-272.

[270] J.A. Hoogeveen and A.P.A. Vestjens, Optimal on-line algorithms for single-machine scheduling, *Proceedings of the 5th IPCO Conference* (1996), 404-414.

[271] J.A. Hoogeveen and G.J. Woeginger, Scheduling with controllable processing times, Manuscript, Department of Mathematics, TU Graz, Graz, Austria, 1998.

[272] W.A. Horn, Single-machine job sequencing with treelike precedence ordering and linear delay penalties, *SIAM Journal on Applied Mathematics* 23 (1972), 189-202.

[273] W.A. Horn, Minimizing average flow time with parallel machines, *Operations Research* 21 (1973), 846-847.

[274] W.A. Horn, Some simple scheduling algorithms, *Naval Research Logistics Quarterly* 21 (1974), 177-185.

[275] E. Horowitz and S. Sahni, Exact and approximate algorithms for scheduling nonidentical processors, *Journal of the Association for Computing Machinery* 23 (1976), 317-327.

[276] E.C. Horvath, S. Lam and R. Sethi, A level algorithm for preemptive scheduling, *Journal of the Association for Computing Machinery* 24 (1977), 32-43.

[277] T.C. Hu, Parallel sequencing and assembly line problems, *Operations Research* 9 (1961), 841-848.

[278] O.H. Ibarra and C.E. Kim, Heuristic algorithms for scheduling independent tasks on nonidentical processors, *Journal of the Association for Computing Machinery* 24 (1977), 280-289.

[279] O.H. Ibarra and C.E. Kim, Approximation algorithms for certain scheduling problems, *Mathematics of Operations Research* 3 (1978), 197-204.

[280] I. Ignall and L. Schrage, Applications of the branch-and-bound technique to some flow-shop scheduling problems, *Operations Research* 13 (1965), 400-412.

[281] J.R. Jackson, Scheduling a production line to minimize maximum tardiness, Research Report 43, Management Science Research Project, University of California, Los Angeles, USA, 1955.

[282] J.R. Jackson, An extension of Johnson's result on job lot scheduling, *Naval Research Logistics Quarterly* 3 (1956), 201-203.

[283] J.M. Jaffe, Efficient scheduling of tasks without full use of processor resources, *Theoretical Computer Science* 12 (1980), 1-17.

[284] J.M. Jaffe, An analysis of preemptive multiprocessor job scheduling, *Mathematics of Operations Research* 5 (1980), 415-521.

[285] A. Jakoby and R. Reischuk, The complexity of scheduling problems with communication delays for trees, *Proceedings of the 3rd Scandinavian Workshop on Algorithm Theory* (1992), 165-177.

[286] S.M. Johnson, Optimal two- and three-stage production schedules with setup times included, *Naval Research Logistics Quarterly* 1 (1954), 61-68.

[287] H. Jung, L.M. Kirousis and P. Spirakis, Lower bounds and efficient algorithms for multiprocessor scheduling of directed acyclic graphs with communication delays, *Information and Computation* 105 (1993), 94-104.

[288] B. Jurisch and W. Kubiak, Two-machine open shops with renewable resources, *Operations Research* 45 (1997), 544-552.

[289] B. Jurisch and W. Kubiak, Algorithms for minclique scheduling problems, *Discrete Applied Mathematics* 72 (1997), 115-139.

[290] H.G. Kahlbacher, SWEAT–A program for a scheduling problem with earliness and tardiness penalties, *European Journal of Operational Research* 43 (1989), 111-112.

[291] H. Kamoun and C. Sriskandarajah, The complexity of scheduling jobs in repetitive manufacturing systems, *European Journal of Operational Research* 70 (1993), 350-364.

[292] J.J. Kanet, Minimizing the average deviation of jobs completion times about a common due date, *Naval Research Logistics Quarterly* 28 (1981), 643-651.

[293] D.R. Karger, S.J. Phillips and E. Torng, A better algorithm for an ancient scheduling problem, *Journal of Algorithms* 20 (1996), 400-430.

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

[295] M.T. Kaufman, An almost-optimal algorithm for the assembly line scheduling problem, *IEEE Transactions on Computing* C-23 (1974), 1169-1174.

[296] T. Kawaguchi and S. Kyan, Worst case bound of an LRF schedule for the mean weighted flow-time problem, *SIAM Journal on Computing* 15 (1986), 1119-1129.

[297] H. Kellerer, T. Tautenhahn and G.J. Woeginger, Approximability and nonapproximability results for minimizing total flow time on a single machine, *Proceedings of the 28th Annual ACM Symposium on Theory of Computing* (1996), 418-426. To appear in *SIAM Journal on Computing*.

[298] Y.-D. Kim, Heuristics for flowshop scheduling problems minimizing mean tardiness, *Journal of the Operational Research Society* 44 (1993), 19-28.

[299] H. Kise, T. Ibaraki and H. Mine, Performance analysis of six approximation algorithms for the one-machine maximum lateness scheduling problem with ready times, *Journal of the Operational Research Society of Japan* 22 (1979), 205-224.

[300] U. Kleinau, Two-machine shop scheduling problems with batch processing, *Mathematical and Computer Modelling* 17 (1993), 55-66.

[301] W.H. Kohler and K. Steiglitz, Exact, approximate and guaranteed accuracy algorithms for the flow-shop problem $n/2/F/\overline{F}$, *Journal of the Association for Computing Machinery* 22 (1975), 106–114.

[302] C. Koulamas, The total tardiness problem: Review and extensions, *Operations Research* 42 (1994), 1025-1041.

[303] M.Y. Kovalyov, On one machine scheduling to minimize the number of late items and the total tardiness, Preprint N4, Institute of Engineering Cybernetics, Academy of Sciences of Byelorussian SSR, Minsk, Byelorussia, 1991.

[304] M.Y. Kovalyov and W. Kubiak, A fully polynomial time approximation scheme for the weighted earliness-tardiness problem, *Operations Research*, to appear.

[305] M.Y. Kovalyov, C.N. Potts and L.N. Van Wassenhove, A fully polynomial approximation scheme for scheduling a single machine to minimize total weighted late work, *Mathematics of Operations Research* 19 (1994), 86-93.

[306] A. Krämer, Scheduling multiprocessor tasks on dedicated processors, *Ph.D. thesis*, Fachbereich Mathematik/Informatik, Universität Osnabrück, Osnabrück, Germany, 1995.

[307] S.A. Kravchenko and Y.N. Sotskov, Optimal makespan schedule for three jobs on two machines, *ZOR – Mathematical Methods of Operations Research* 43 (1996), 233-238.

[308] H. Krawczyk and M. Kubale, An approximation algorithm for diagnostic test scheduling in multicomputer systems, *IEEE Transactions on Computing* 34 (1985), 869-872.

[309] M.J. Krone and K. Steiglitz, Heuristic-programming solution of a flowshop-scheduling problem, *Operations Research* 22 (1974), 629–638.

[310] W. Kubiak, Złożoność Obliczeniowa Algorytmów i Problemów Szeregowania Zadań przy Ograniczeniach Zasobowych (in Polish), *Ph.D. thesis*, ICS-Polish Academy of Sciences, Warsaw, Poland, 1987.

[311] W. Kubiak, Completion time variance minimization on a single machine is difficult, *Operations Research Letters* 14 (1993), 49-59.

[312] W. Kubiak, C. Sriskandarajah and K. Zaras, A note on the complexity of open shop scheduling problems, *INFOR* 29 (1991), 284-294.

[313] M. Kubale, Preemptive scheduling of two-processor tasks on dedicated processors (in Polish), *Zeszyty Naukowe Politechniki Śląskiej. Seria Automatyka* 100/1082 (1990), 145-153.

[314] M. Kunde, Nonpreemptive LP-scheduling on homogeneous multiprocessor systems, *SIAM Journal on Computing* 10 (1981), 151-173.

[315] J. Labetoulle, E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan, Preemptive scheduling of uniform machines subject to release dates, in W.R. Pulleyblank (ed.) *Progress in Combinatorial Optimization*, Academic Press, New York, 1984, 245-261.

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

[317] B.J. Lageweg, J.K. Lenstra and A.H.G. Rinnooy Kan, Job-shop scheduling by implicit enumeration, *Management Science* 24 (1977), 441-450.

[318] B.J. Lageweg, J.K. Lenstra and A.H.G. Rinnooy Kan, A general bounding scheme for the permutation flow-shop problem, *Operations Research* 26 (1978), 53-67.

[319] S. Lam and R. Sethi, Worst case analysis of two scheduling algorithms, *SIAM Journal on Computing* 6 (1977), 518-536.

[320] R.E. Larson, M.I. Dessouky and R.E. Devor, A forward-backward procedure for the single machine problem to minimize maximum lateness, *IIE Transactions* 17 (1985), 252-260.

[321] E.L. Lawler, Optimal sequencing of a single machine subject to precedence constraints, *Management Science* 19 (1973), 544-546.

[322] E.L. Lawler, Sequencing to minimize the weighted number of tardy jobs, *RAIRO Recherche opérationnelle* S10(5) (1976), 27-33.

[323] E.L. Lawler, A 'pseudopolynomial' algorithm for sequencing jobs to minimize total tardiness, *Annals of Operations Research* 1 (1977), 331-342.

[324] E.L. Lawler, Sequencing jobs to minimize total weighted completion time subject to precedence constraints, *Annals of Operations Research* 2 (1978), 75-90.

[325] E.L. Lawler, Preemptive scheduling of uniform parallel machines to minimize the weighted number of late jobs, Report BW105, Centre for Mathematics and Computer Science, Amsterdam, The Netherlands, 1979.

[326] E.L. Lawler, Efficient implementation of dynamic programming algorithms for sequencing problems, Report BW106, Centre for Mathematics and Computer Science, Amsterdam, The Netherlands, 1979.

[327] E.L. Lawler, A fully polynomial approximation scheme for the total tardiness problem, *Operations Research Letters* 1 (1982), 207-208.

[328] E.L. Lawler, Preemptive scheduling of precedence-constrained jobs on parallel machines, in M.A.H. Dempster, J.K. Lenstra and A.H.G. Rinnooy Kan (eds.) *Deterministic and Stochastic Scheduling*, Reidel, Dordrecht, 1982, 101-123.

[329] E.L. Lawler, 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, 1983, 202-234.

[330] E.L. Lawler, Scheduling a single machine to minimize the number of late jobs, Report No.UCB/CSD 83/139, Computer Science Division, University of California, Berkeley, USA, 1983.

[331] E.L. Lawler, A dynamic programming algorithm for preemptive scheduling of a single machine to minimize the number of late jobs, *Annals of Operations Research* 26 (1990), 125-133.

[332] E.L. Lawler, Knapsack-like scheduling problems, the Moore-Hodgson algorithm and the 'tower of sets' property, *Mathematical and Computer Modelling* 20 (1994), 91-106.

[333] E.L. Lawler and J. Labetoulle, On preemptive scheduling of unrelated parallel processors by linear programming, *Journal of the Association for Computing Machinery* 25 (1978), 612-619.

[334] E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan, Minimizing maximum lateness in a two-machine open shop, *Mathematics of Operations Research* 6 (1981), 153-158. Erratum: *Mathematics of Operations Research* 7 (1982), 635.

[335] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys, Sequencing and scheduling: algorithms and complexity, in S. Graves, A.H.G. Rinnooy Kan and P. Zipkin (eds.) *Handbooks in Operations Research and Management Science, Volume 4, Logistics of Production and Inventory*, North Holland, Amsterdam, 1993, 445-522.

[336] E.L. Lawler, M.G. Luby and V.V. Vazirani, Scheduling open shops with parallel machines, *Operations Research Letters* 1 (1982), 161-164.

[337] E.L. Lawler and C.U. Martel, Preemptive scheduling of two uniform machines to minimize the number of late jobs, *Operations Research* 37 (1989), 314-318.

[338] E.L. Lawler and J.M. Moore, A functional equation and its application to resource allocation and sequencing problems, *Management Science* 16 (1969), 77-84.

[339] S. Lawrence, Resource constrained scheduling: an experimental investigation of heuristic scheduling techniques, Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, USA, 1984.

[340] C.-Y. Lee, T.C.E. Cheng and B.M.T. Lin, Minimizing the makespan in the 3-machine assembly-type flowshop scheduling problem, *Management Science* 39 (1993), 616-625.

[341] C.-Y. Lee and S.J. Kim, Parallel genetic algorithms for the earliness tardiness job scheduling problem with general penalty weights, *Computers and Industrial Engineering* 28 (1995), 231-243.

[342] C.-Y. Lee and G.L. Vairaktarakis, Complexity of single machine hierarchical scheduling: A survey, in P.M. Pardalos (ed.) *Complexity in Numerical Optimization*, World Scientific Publishing Company, 1993, 269–298.

[343] T. Leighton, B. Maggs and S. Rao, Packet routing and job shop scheduling in O(Congestion+Dilation) steps, *Combinatorica* 14 (1994), 167-186.

[344] T. Leighton, B. Maggs and A. Richa, Fast algorithms for finding O(Congestion+Dilation) packet routing schedules, Technical Report CMU-CS-96-152, School of Computer Science, Carnegie Mellon University, Pittsburgh, USA, 1996.

[345] J.K. Lenstra, unpublished manuscript.

[346] J.K. Lenstra and A.H.G. Rinnooy Kan, Complexity of scheduling under precedence constraints, *Operations Research* 26 (1978), 22-35.

[347] J.K. Lenstra and A.H.G. Rinnooy Kan, Computational complexity of discrete optimization problems, *Annals of Operations Research* 4 (1979), 121-140.

[348] J.K. Lenstra and A.H.G. Rinnooy Kan, Complexity results for scheduling chains on a single machine, *European Journal of Operational Research* 4 (1980), 270-275.

[349] J.K. Lenstra, A.H.G. Rinnooy Kan and P. Brucker, Complexity of machine scheduling problems, *Annals of Operations Research* 1 (1977), 343-362.

[350] J.K. Lenstra, D.B. Shmoys and É. Tardos, Approximation algorithms for scheduling unrelated parallel machines, *Mathematical Programming* 46 (1990), 259-271.

[351] J.K. Lenstra, M. Veldhorst and B. Veltman, The complexity of scheduling trees with communication delays, *Journal of Algorithms* 20 (1996), 157-173.

[352] S. Leonardi and D. Raz, Approximating total flow time on parallel machines, *Proceedings of the 29th Annual ACM Symposium on Theory of Computing* (1997), 110-119.

[353] V. Lev and I. Adiri, V-shop scheduling, *European Journal of Operational Research* 18 (1984), 51-56.

[354] C.-L. Li, Z.L. Chen and T.C.E. Cheng, A note on one-processor scheduling with asymmetric earliness and tardiness penalties, *Operations Research Letters* 13 (1993), 45-48.

[355] R. Li and L. Shi, An on-line algorithm for some uniform processor scheduling, *SIAM Journal on Computing* (1998), to appear.

[356] C.Y. Liu and R.L. Bulfin, On the complexity of preemptive open-shop scheduling problems, *Operations Research Letters* 4 (1985), 71-74.

[357] C.Y. Liu and R.L. Bulfin, Scheduling open shops with unit execution times to minimize functions of due dates, *Operations Research* 36 (1988), 553-559.

[358] J.W.S. Liu and C.L. Liu, Bounds on scheduling algorithms for heterogeneous computing systems, in J.L. Rosenfeld (ed.) *Information Processing*, North-Holland, Amsterdam, 1974, 349-353.

[359] J.W.S. Liu and C.L. Liu, Bounds on scheduling algorithms for heterogeneous computing systems, Technical Report UIUCDCS-R-74-632, Department of Computer Science, University of Illinois at Urbana-Champaign, USA, 1974.

[360] J.W.S. Liu and C.L. Liu, Performance analysis of heterogeneous multiprocessor computing systems, in E. Gelenbe and R. Mahl (eds.) *Computer Architectures and Networks*, North-Holland, Amsterdam, 1974, 331-343.

[361] J.W.S. Liu and A. Yang, Optimal scheduling of independent tasks on heterogeneous computing systems, *Proceedings of the ACM Annual Conference* (1974), 38-45.

[362] E.L. Lloyd, Concurrent task systems, *Operations Research* 29 (1981), 85-92.

[363] I.N. Lushchakova and V.A. Strusevich, The complexity of open shop scheduling under resource constraints (in Russian), in *Solution Methods for Extremal Problems*, Minsk, 1989, 57-65.

[364] I.N. Lushchakova and V.A. Strusevich, Two-stage open shop systems with resource constraints (in Russian), *Zhurnal Vychistlitel'noj Matematiki i Matematicheskoj Fiziki* 29 (1989), 1393-1407.

[365] W. Mao, R.K. Kincaid and A. Rifkin, On-line algorithms for a single machine scheduling problem, in S.G. Nash and A. Sofer (eds.), *The Impact of Emerging Technologies on Computer Science and Operations Research*, Kluwer Academic Publishers, Boston, 1995, 157-173.

[366] C.U. Martel, Preemptive scheduling with release times, deadlines and due times, *Journal of the Association for Computing Machinery* 29 (1982), 812-829.

[367] S. Martello, F. Soumis and P. Toth, Exact and approximation algorithms for makespan minimization on unrelated parallel machines, *Discrete Applied Mathematics* 75 (1997), 169-188.

[368] P. Martin, D.B. Shmoys, A new approach to computing schedules for the job shop scheduling problem, *Proceedings of the 5th IPCO Conference* (1996), 389-403.

[369] A.J. Mason, Genetic algorithms and scheduling problems, *Ph.D. thesis*, Department of Engineering, University of Cambridge, U.K., 1992.

[370] A.J. Mason and E.J. Anderson, Minimizing flow time on a single machine with job classes and setup times, *Naval Research Logistics* 38 (1991), 333-350.

[371] H. Matsuo, Cyclic sequencing in the two-machine permutation flow shop: complexity, worst-case and average-case analysis, *Naval Research Logistics Quarterly* 37 (1990), 679-694.

[372] H. Matsuo, C.J. Suh and R.S. Sullivan, A controlled search simulated annealing method for the single machine weighted tardiness problem, Working paper 87-12-2, Department of Management, The University of Texas at Austin, TX, USA, 1987.

[373] H. Matsuo, C.J. Suh and R.S. Sullivan, A controlled search simulated annealing method for the general jobshop scheduling problem, Working paper 03-44-88, Department of Management, The University of Texas at Austin, TX, USA, 1988.

[374] S.T. McCormick and M.L. Pinedo, Scheduling n independent jobs on m uniform machines with both flow time and makespan objectives: A parametric analysis, *ORSA Journal on Computing* 7 (1995), 63-77.

[375] G.B. McMahon, Optimal production schedules for flow shops, *Canadian Operational Research Society Journal* 7 (1969), 141-151.

[376] G.B. McMahon and P.G. Burton, Flow-shop scheduling with the branch-and-bound method, *Operations Research* 15 (1967), 473-481.

[377] G.B. McMahon and M. Florian, On scheduling with ready times and due dates to minimize maximum lateness, *Operations Research* 23 (1975), 475-482.

[378] R. McNaughton, Scheduling with deadlines and loss functions, *Management Science* 6 (1959), 1-12.

[379] J. Mittenthal, M. Raghavachari and A.I. Rana, A hybrid simulated annealing approach for single machine scheduling problems with non-regular penalty functions, *Computers and Operations Research* 20 (1993), 103-111.

[380] R.H. Möhring, M.W. Schäffter and A.S. Schulz, Scheduling jobs with communication delays: using infeasible solutions for approximation, *Proceedings of the 4th Annual European Symposium on Algorithms* (1996), 76-90.

[381] C.L. Monma, Linear-time algorithms for scheduling on parallel processors, *Operations Research* 30 (1982), 116-124.

[382] C.L. Monma and C.N. Potts, On the complexity of scheduling with batch setup times, *Operations Research* 37 (1989), 798-804.

[383] C.L. Monma and C.N. Potts, Analysis of heuristics for preemptive parallel machine scheduling with batch setup times, *Operations Research*4193 981-993.

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

[385] J.M. Moore, An n job, one machine sequencing algorithm for minimizing the number of late jobs, *Management Science* 15 (1968), 102-109.

[386] J.F. Morrison, A note on LPT scheduling, *Operations Research Letters* 7 (1988), 77-79.

[387] R. Motwani, S. Phillips and E. Torng, Non-clairvoyant scheduling, *Theoretical Computer Science* 130 (1994), 17-47.

[388] J.H. Muller and J. Spinrad, Incremental modular decomposition. *Journal of the Association for Computing Machinery* 36 (1989), 1-19.

[389] A. Munier and C. Hanen, Using duplication for scheduling unitary tasks on m processors with unit communication delays, *Theoretical Computer Science* 178 (1997), 119-127.

[390] A. Munier and C. Hanen, An approximation algorithm for scheduling unitary tasks on m processors with communication delays, Internal Report LITP 12, Université P. et M. Curie, Paris, France, 1995.

[391] A. Munier and J.-C. König, A heuristic for a scheduling problem with communication delays, *Operations Research* 45 (1997), 145-147.

[392] R.R. Muntz and E.G. Coffman, Jr., Optimal scheduling on two-processor systems, *IEEE Transactions on Computing* C-18 (1969), 1014-1020.

[393] R.R. Muntz and E.G. Coffman, Jr., Preemptive scheduling of real tasks on multiprocessor systems, *Journal of the Association for Computing Machinery* 17 (1970), 324-338.

[394] R. Nakano and T. Yamada, Conventional genetic algorithm for job shop problems, in R.K. Belew and L.B. Booker (eds.) *Proceedings of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann, San Mateo, 1991, 474-479.

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

[396] E. Németi, Das reihenfolgeproblem in der fertinungsprogrammierung und linearplanung mit logischen bedingungen, *Mathematica (Cluj)* 6 (1964), 87-99.

[397] Ju.D. Neumytov and S.V. Sevastianov, An approximation algorithm with best possible bound for the counter routes problem with three machines (in Russian), *Upravlyaemye Sistemy* 31 (1993), 53-65.

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

[399] E. Nowicki and C. Smutnicki, Worst-case analysis of an approximation algorithm for flow-shop scheduling, *Operations Research Letters* 8 (1989), 171-177.

[400] E. Nowicki and C. Smutnicki, An approximation algorithm for single-machine scheduling problem with release times and delivery times, *Discrete Applied Mathematics* 48 (1993), 69-79.

[401] E. Nowicki and C. Smutnicki, A fast tabu search algorithm for the permutation flow shop problem, *European Journal of Operational Research* 91 (1996), 160-175.

[402] E. Nowicki and C. Smutnicki, A fast taboo search algorithm for the job shop problem, *Management Science* 42 (1996), 797-813.

[403] E. Nowicki and S. Zdrzałka, A note on minimizing maximum lateness in a one-machine sequencing problem with release dates, *European Journal of Operational Research* 23 (1986), 266-267.

[404] E. Nowicki and S. Zdrzałka, A two-machine flow shop scheduling problem with controllable job processing times, *European Journal of Operational Research* 34 (1988), 208-220.

[405] E. Nowicki and S. Zdrzałka, A survey of results for sequencing problems with controllable processing times, *Discrete Applied Mathematics* 26 (1990), 271-287.

[406] E. Nowicki and S. Zdrzałka, A bicriterion approach to preemptive scheduling of parallel machines with controllable job processing times, *Discrete Applied Mathematics* 63 (1995), 237-256.

[407] F.A. Ogbu and D.K. Smith, The application of the simulated annealing algorithm to the solution of the $n|m|C_{\max}$ flowshop problem, *Computers and Operations Research* 17 (1990), 243-253.

[408] F.A. Ogbu and D.K. Smith, Simulated annealing for the permutation flowshop problem, *OMEGA* 19 (1991), 64-67.

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

[410] S.S. Panwalkar and R. Rajagopalan, Single-machine sequencing with controllable processing times, *European Journal of Operational Research* 59 (1992), 298-302.

[411] S.S. Panwalkar, M.L. Smith and C.P. Koulamas, A heuristic for the single machine tardiness problem, *European Journal of Operational Research* 70 (1993), 304-310.

[412] S.S. Panwalkar, M.L. Smith and A. Seidmann, Common due date assignment to minimize total penalty for the one machine sequencing problem, *Operations Research* 30 (1982), 391-399.

[413] C.H. Papadimitriou, *Computational Complexity*, Addison-Wesley, 1994.

[414] C.H. Papadimitriou and P.C. Kannelakis, Flow shop scheduling with limited temporary storage, *Journal of the Association for Computing Machinery* 27 (1980), 533-549.

[415] C.H. Papadimitriou and M. Yannakakis, Towards an architecture-independent analysis of parallel algorithms, *SIAM Journal on Computing* 19 (1990), 322-328.

[416] E. Pesch, Machine learning by schedule decomposition, Working Paper, Faculty of Economics and Business Administration, University of Limburg, Maastricht, The Netherlands, 1993.

[417] C. Phillips, C. Stein and J. Wein, Scheduling jobs that arrive over time, *Proceedings of the 4th Workshop on Algorithms and Data Structures* (1995), 86-97. To appear in *Mathematical Programming*.

[418] C. Picouleau, Etude de problèmes les systèmes distribués, *Ph.D. thesis*, Univ. Pierre et Marie Curie, Paris, France, 1992.

[419] C. Picouleau, New complexity results on scheduling with small communication delays, *Discrete Applied Mathematics* 60 (1995), 331-342.

[420] M.E. Posner, Minimizing weighted completion times with deadlines, *Operations Research* 33 (1985), 562-574.

[421] C.N. Potts, The job-machine scheduling problem, *Ph.D. thesis*, University of Birmingham, U.K., 1974.

[422] C.N. Potts, An algorithm for the single machine sequencing problem with precedence constraints, *Mathematical Programming Study* 13 (1980), 78-87.

[423] C.N. Potts, An adaptive branching rule for the permutation flow-shop problem, *European Journal of Operational Research* 5 (1980), 19-25.

[424] C.N. Potts, Analysis of a heuristic for one machine sequencing with release dates and delivery times, *Operations Research* 28 (1980), 1436-1441.

[425] C.N. Potts, A Lagrangean based branch and bound algorithm for single machine sequencing with precedence constraints to minimize total weighted completion time, *Management Science* 31 (1985), 1300-1311.

[426] C.N. Potts, Analysis of a linear programming heuristic for scheduling unrelated parallel machines, *Discrete Applied Mathematics* 10 (1985), 155-164.

[427] C.N. Potts, Analysis of heuristics for two-machine flow-shop sequencing subject to release dates, *Mathematics of Operations Research* 10 (1985), 576-584.

[428] C.N. Potts, S.V. Sevastianov, V.A. Strusevich, L.N. Van Wassenhove and C.M. Zwaneveld, The two-stage assembly scheduling problem: complexity and approximation, *Operations Research* 43 (1995), 346-355.

[429] C.N. Potts, D.B. Shmoys and D.P. Williamson, Permutation vs. non-permutation flow shop schedules, *Operations Research Letters* 10 (1991), 281-284.

[430] C.N. Potts and L.N. Van Wassenhove, A decomposition algorithm for the single machine total tardiness problem, *Operations Research Letters* 1 (1982), 177-181.

[431] C.N. Potts and L.N. Van Wassenhove, An algorithm for single machine sequencing with deadlines to minimize total weighted completion time, *European Journal of Operational Research* 12 (1983), 379-387.

[432] C.N. Potts and L.N. Van Wassenhove, A branch and bound algorithm for the total weighted tardiness problem, *Operations Research* 33 (1985), 363-377.

[433] C.N. Potts and L.N. Van Wassenhove, Dynamic programming and decomposition approaches for the single machine total tardiness problem, *European Journal of Operational Research* 32 (1987), 404-414.

[434] C.N. Potts and L.N. Van Wassenhove, Algorithms for scheduling a single machine to minimize the weighted number of late jobs, *Management Science* 34 (1988), 843-858.

[435] C.N. Potts and L.N. Van Wassenhove, Single machine tardiness sequencing heuristics, *IEE Transactions* 23 (1991), 346-354.

[436] C.N. Potts and L.N. Van Wassenhove, Integrating scheduling with batching and lot-sizing: a review of algorithms and complexity, *Journal of the Operational Research Society* 43 (1992), 395-406.

[437] C.N. Potts and L.N. Van Wassenhove, Single machine scheduling to minimize total late work, *Operations Research* 40 (1992), 586-595.

[438] C.N. Potts and L.N. Van Wassenhove, Approximation algorithms for scheduling a single machine to minimize total late work, *Operations Research Letters* 11 (1992), 261-266.

[439] M. Queyranne, Personal communication cited in [233].

[440] M. Queyranne and A.S. Schulz, Polyhedral approaches to machine scheduling, Preprint No. 408/1994, Department of Mathematics, Technical University of Berlin, Berlin, Germany, 1994. To appear in *Mathematical Programming*.

[441] M. Queyranne and Y. Wang, Single-machine scheduling polyhedra with precedence constraints, *Mathematics of Operations Research* 16 (1991), 1-20. Erratum: *Mathematics of Operations Research* 20 (1995), 768.

[442] M. Queyranne and Y. Wang, A cutting-plane procedure for precedence-constrained single-machine scheduling. Working paper, University of British Columbia, Vancouver, Canada, 1991.

[443] V.J. Rayward-Smith, The complexity of preemptive scheduling given interprocessor communication delays, *Information Processing Letters* 25 (1987), 123-125.

[444] V.J. Rayward-Smith, UET Scheduling with unit interprocessor communication delays, *Discrete Applied Mathematics* 18 (1987), 55-71.

[445] C.R. Reeves, Improving the efficiency of tabu search for machine sequencing problems, *Journal of the Operational Research Society* 44 (1993), 375-382.

[446] C.R. Reeves, A genetic algorithm for flowshop sequencing, *Computers and Operations Research* 22 (1995), 5-13.

[447] A.H.G. Rinnooy Kan, B.J. Lageweg and J.K. Lenstra, Minimizing total costs in one-machine scheduling, *Operations Research* 23 (1975), 908-927.

[448] H. Röck, The three-machine no-wait flow shop is NP-complete, *Journal of the Association for Computing Machinery* 31 (1981), 336-345.

[449] H. Röck, Some new results in flow shop scheduling, *ZOR – Mathematical Methods of Operations Research* 28 (1984), 1-16.

[450] H. Röck and G. Schmidt, Machine aggregation heuristics in shop-scheduling, *Methods of Operations Research* 45 (1983), 303-314.

[451] G. Rote and G.J. Woeginger, Time complexity and linear-time approximation of the ancient two machine flow shop, Technical Report Woe-14, Department of Mathematics, TU Graz, Graz, Austria, 1997.

[452] M.H. Rothkopf, Scheduling independent tasks on parallel processors, *Management Science* 12 (1966), 437-447.

[453] M.H. Rothkopf and S.A. Smith, There are no undiscovered priority index rules for minimizing total delay costs, *Operations Research* 32 (1984), 451-456.

[454] F.M. Ruiz Diaz and S. French, A note on SPT scheduling of a single machine with controllable processing times, Note 154, Department of Decision Theory, University of Manchester, Manchester, UK, 1984.

[455] R.M. Russell and J.E. Holsenback, Evaluation of leading heuristics for the single machine tardiness problem, *European Journal of Operational Research* 96 (1997), 538-545.

[456] R.M. Russell and J.E. Holsenback, Evaluation of greedy, myopic and less-greedy heuristics for the single machine total tardiness problem, *Journal of the Operational Research Society* 48 (1997), 640-646.

[457] S. Sahni, Algorithms for scheduling independent tasks, *Journal of the Association for Computing Machinery* 23 (1976), 116-127.

[458] S. Sahni, Preemptive scheduling with due dates, *Operations Research* 27 (1979), 925-934.

[459] S. Sahni and Y. Cho, Nearly on line scheduling of a uniform processor system with release times, *SIAM Journal on Computing* 8 (1979), 275-285.

[460] S. Sahni and Y. Cho, Complexity of scheduling shops with no wait in process, *Mathematics of Operations Research* 4 (1979), 448-457.

[461] S. Sahni and Y. Cho, Scheduling independent tasks with due times on a uniform processor system, *Journal of the Association for Computing Machinery* 27 (1980), 550-563.

[462] S.C. Sarin, S. Ahn and A.B. Bishop, An improved branching scheme for the branch and bound procedure of scheduling n jobs on m parallel machines to minimize total weighted flowtime, *International Journal of Production Research* 26 (1988), 1183-1191.

[463] J.P. Schmidt, A. Siegel and A. Srinivasan, Chernoff-Hoeffding bounds for applications with limited independence, *SIAM Journal on Discrete Mathematics* 8 (1995), 223-250.

[464] L. Schrage, A proof of the shortest remaining processing time processing discipline, *Operations Research* 16 (1968), 687-690.

[465] L. Schrage, Solving resource-constrained network problems by implicit enumeration—nonpreemptive case, *Operations Research* 18 (1970), 263-278.

[466] L. Schrage, Obtaining optimal solutions to resource constrained network scheduling problems, Unpublished manuscript, 1971.

[467] L. Schrage and K.R. Baker, Dynamic programming solution of sequencing problems with precedence constraints, *Operations Research* 26 (1978), 444-449.

[468] A.S. Schulz, Scheduling to minimize total weighted completion time: Performance guarantees of LP-based heuristics and lower bounds, *Proceedings of the 5th IPCO Conference* (1996), 301-315.

[469] J.M.J. Schutten, S.L. van de Velde and W.H.M. Zijm, Single-machine scheduling with release dates, due dates and family setup times, *Management Science* 42 (1996), 1165-1174.

[470] P. Schuurman and G.J. Woeginger, A polynomial time approximation scheme for the two-stage multiprocessor flow shop problem, Technical Report Woe-01, Department of Mathematics, TU Graz, Graz, Austria, 1997. To appear in *Theoretical Computer Science*.

[471] P. Schuurman and G.J. Woeginger, Approximation algorithms for the multiprocessor open shop problem, Technical Report Woe-13, Department of Mathematics, TU Graz, Graz, Austria, 1997.

[472] R. Sethi, On the complexity of mean flow time scheduling, *Mathematics of Operations Research* 2 (1977), 320-330.

[473] S.V. Sevastianov, Approximation algorithms for Johnson's and vector summation problems (in Russian), *Upravlyaemye Sistemy* 20 (1980), 64-73.

[474] S.V. Sevastianov, Some generalizations of the Johnson problem (in Russian), *Upravlyaemye Sistemy* 21 (1981), 45-61.

[475] S.V. Sevastianov, Algorithms with estimates for the Johnson's and Akers-Friedman problems in the case of three machines (in Russian), *Upravlyaemye Sistemy* 22 (1982), 51-57.

[476] S.V. Sevastianov, On some geometric methods in scheduling theory: a survey, *Discrete Applied Mathematics* 55 (1994), 59-82.

[477] S.V. Sevastianov, Vector summation in Banach space and polynomial time algorithms for flow shops and open shops, *Mathematics of Operations Research* 20 (1995), 90-103.

[478] S.V. Sevastianov, Nonstrict vector summation in multi-operation scheduling, Technical Report COSOR 95-37, Department of Mathematics and Computing, TU Eindhoven, Eindhoven, The Netherlands, 1995.

[479] S.V. Sevastianov and G.J. Woeginger, Makespan minimization in preemptive two machine job shops, *Computing* 60 (1998), 73-79.

[480] S.V. Sevastianov and G.J. Woeginger, Makespan minimization in open shops: a polynomial time approximation scheme, *Mathematical Programming* (1998), to appear.

[481] J. Sgall, On-line scheduling, in A. Fiat and G.J. Woeginger (eds.) *On-line Algorithms: The State of the Art*, Springer, 1998.

[482] N.V. Shaklevich, J.A. Hoogeveen and M. Pinedo, Minimizing total weighted completion time in a proportionate flow shop, Technical Report COSOR 96-03, Department of Mathematics and Computing, TU Eindhoven, Eindhoven, The Netherlands, 1996.

[483] N.V. Shaklevich and Y.N. Sotskov, Scheduling two jobs with fixed and non-fixed routes, *Computing* 52 (1994), 17-30.

[484] N.V. Shaklevich and V.A. Strusevich, Two machine open shop scheduling problem to minimize an arbitrary machine usage regular penalty function, *European Journal of Operational Research* 70 (1993), 391-404.

[485] A.H. Sharary and N. Zaguia, Minimizing the number of tardy jobs in single machine sequencing, *Discrete Applied Mathematics* 117 (1993), 215-223.

[486] D.B. Shmoys, C. Stein and J. Wein, Improved approximation algorithms for shop scheduling problems, *SIAM Journal on Computing* 23 (1994), 617-632.

[487] D.B. Shmoys and É. Tardos, An approximation algorithm for the generalized assignment problem, *Mathematical Programming* 62 (1993), 461-474.

[488] D.B. Shmoys, J. Wein and D.P. Williamson, Scheduling parallel machines on-line, *SIAM Journal on Computing* 24 (1995), 1313-1331.

[489] J. Shwimer, On the N-jobs, one machine, sequence-independent scheduling problem with penalties: A branch-and-bound solution, *Management Science* 18B (1972), 301-313.

[490] J.B. Sidney, 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, Volume 86, Springer, Berlin, 1973, 393-398.

[491] J.B. Sidney, Decomposition algorithms for single-machine sequencing with precedence relations and deferral costs, *Operations Research* 23 (1975), 283-298.

[492] J.B. Sidney, Optimal single-machine scheduling with earliness and tardiness penalties, *Operations Research* 25 (1977), 62-69.

[493] B.B. Simons, A fast algorithm for single processor scheduling, *Proceedings of the 19th IEEE Symposium on Foundations of Computer Science* (1978), 246-252.

[494] S.P. Smith, An experiment on using genetic algorithms to learn scheduling heuristics, *Proceedings of the SPIE-Conference on Applications of Artificial Intelligence X: Knowledge-Based Systems*, SPIE—The International Society for Optical Engineering, Bellingham, WA, Volume 1707, 1992, 378-386

[495] W.E. Smith, Various optimizers for single-stage production, *Naval Research Logistics Quarterly* 3 (1956), 59-66.

[496] Y.N. Sotskov, The complexity of shop-scheduling problems with two or three jobs, *European Journal of Operational Research* 53 (1991), 326-336.

[497] Y.N. Sotskov and N.V. Shaklevich, NP-hardness of shop-scheduling problems with three jobs, *Discrete Applied Mathematics* 59 (1995), 237-266.

[498] Y.N. Sotskov, T. Tautenhahn and F. Werner, Heuristics for permutation flow shop scheduling with batch setup times, *OR Spektrum* 18 (1996), 67-80.

[499] J.P. Sousa and L.A. Wolsey, A time indexed formulation of non-preemptive singe machine scheduling problems, *Mathematical Programming* 54 (1992), 353-367.

[500] C. Stein and J. Wein, On the existence of schedules that are near-optimal for both makespan and total weighted completion time, *Operations Research Letters* 21 (1997), 115-122.

[501] A. Steinberg, A strip-packing algorithm with absolute performance bound two, *SIAM Journal on Computing* 26 (1997), 401-409.

[502] G. Steiner, Minimizing the number of tardy jobs with precedence constraints and agreeable due dates. *Discrete Applied Mathematics* 72 (1997), 167-177.

[503] E. Steinitz, Bedingt konvergente Reihen und konvexe Systeme, *Journal für Reine und Angewandte Mathematik* 143 (1913), 128-175.

[504] J.P. Stinson, E.W. Davis and B.M. Khumawala, Multiple resource-constrained scheduling using branch-and-bound, *AIIE Transactions* 10 (1978), 252-259.

[505] R.H. Storer, S.D. Wu and R. Vaccari, New search spaces for sequencing problems with application to job shop scheduling, *Management Science* 38 (1992), 1495-1509.

[506] L. Stougie, Personal communication (1995), cited in [547].

[507] V.A. Strusevich, The two-machine super-shop scheduling problem, *Journal of the Operational Research Society* 42 (1991), 479-492.

[508] V.A. Strusevich, Shop scheduling problems under precedence constraints, *Annals of Operations Research* 69 (1997), 351-377.

[509] W. Szwarc, Elimination methods in the $m \times n$ sequencing problem, *Naval Research Logistics Quarterly* 18 (1971), 295-305.

[510] W. Szwarc, Optimal elimination methods in the $m \times n$ sequencing problem, *Operations Research* 21 (1973), 1250-1259.

[511] W. Szwarc, Dominance conditions for the three-machine flow-shop problem *Operations Research* 26 (1978), 203-206.

[512] W. Szwarc, Single-machine scheduling to minimize absolute deviation of completion times from a common due date, *Naval Research Logistics* 36 (1989), 663-673.

[513] W. Szwarc, Parametric precedence relations in single machine scheduling, *Operations Research Letters* 9 (1990), 133-140.

[514] W. Szwarc, Single machine total tardiness problem revisited, in Y. Ijiri (ed.) *Creative and Innovative Approaches to the Science of Management*, Quorum Books, 1993, 407-417.

[515] W. Szwarc, Adjacent orderings in single-machine scheduling with earliness and tardiness penalties, *Naval Research Logistics* 40 (1993), 229-243.

[516] W. Szwarc and S.K. Mukhopadhyay, Minimizing a quadratic cost function of waiting times in single-machine scheduling, *Journal of the Operational Research Society* 46 (1995), 753-761.

[517] W. Szwarc and S.K. Mukhopadhyay, Decomposition of the single-machine total tardiness problem, *Operations Research Letters* 19 (1996), 243-250.

[518] W. Szwarc, M.E. Posner and J.J. Liu, The single machine problem with a quadratic cost function of completion times, *Management Science*3488 1480-1488.

[519] R. Tadei, J.N.D. Gupta, F. Della Croce and M Cortesi Minimizing makespan in the two-machine flow-shop with release times, *Journal of the Operational Research Society* 49 (1998), 77-85.

[520] E. Taillard, Some efficient heuristic methods for the flow shop sequencing problem, *European Journal of Operational Research* 47 (1990), 65-74.

[521] E. Taillard, Benchmarks for basic scheduling problems. *European Journal of Operational Research* 64 (1993), 278-285.

[522] E. Taillard, Parallel taboo search techniques for the job shop scheduling problem, *ORSA Journal on Computing* 6 (1994), 108-117.

[523] F.B. Talbot and J.H. Patterson, An efficient integer programming algorithm with network cuts for solving resource-constrained scheduling problems. *Management Science* 24 (1978), 1163-1174.

[524] V.S. Tanaev, V.S. Gordon and Y.M. Shafransky, *Scheduling Theory: Single-Stage System*, Kluwer Academic Publishers, Dordrecht, 1994.

[525] V.S. Tanaev, Y.N. Sotskov and V.A. Strusevich, *Scheduling Theory: Multi-Stage Systems*, Kluwer Academic Publishers, Dordrecht, 1994.

[526] B.C. Tansel and I. Sabuncuoglu, New insights on the single machine total tardiness problem, *Journal of the Operational Research Society* 48 (1997), 82-89.

[527] W. Townsend, Minimizing the maximum penalty in the two-machine flow shop, *Management Science* 24 (1977), 230-234.

[528] W. Townsend, The single machine problem with quadratic penalty function of completion times: A branch-and-bound solution, *Management Science* 24 (1978), 530-534.

[529] A.V. Tuzikov, A bi-criteria scheduling problems subject to variation of processing times, *Zhurnal Vychistlitel'noj Matematiki i Matematicheskoj Fiziki* 24 (1984), 1585-1590.

[530] J.D. Ullman, NP-Complete scheduling problems, *Journal of Computing and System Sciences* 10 (1975), 384-393.

[531] J.D. Ullman, Complexity of sequencing problems, in E.G. Coffman, Jr., (ed.) *Computer and Job-Shop Scheduling Theory*, Wiley, New York, 1976, 139-164.

[532] R.J.M. Vaessens, E.H.L. Aarts and J.K. Lenstra, Job shop scheduling by local search, *INFORMS Journal on Computing* 8 (1996), 302-317.

[533] J.M. van den Akker, LP-Based Solution Methods for Single-Machine Scheduling Problems, *Ph.D. thesis*, Department of Mathematics and Computing Science, Eindhoven University of Technology, Eindhoven, The Netherlands, 1994.

[534] J.M. van den Akker, J.A. Hoogeveen and S.L. van de Velde, Parallel machine scheduling by column generation, Report COSOR 95-35, Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands, 1995.

[535] J.M. van den Akker, J.A. Hoogeveen and S.L. van de Velde, A column generation algorithm for common due date scheduling, Report, Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands, 1998.

[536] S.L. van de Velde, Minimizing the sum of job completion times in the two-machine flow shop by Lagrangian relaxation, *Annals of Operations Research* 26 (1990), 257-268.

[537] S.L. van de Velde, Duality-based algorithms for scheduling unrelated parallel machines, *ORSA Journal on Computing* 5 (1993), 192-205.

[538] S.L. van de Velde, Duality decomposition of a single-machine scheduling problem. *Mathematical Programming* 69 (1995), 413-428.

[539] P.J.M. van Laarhoven, E.H.L. Aarts and J.K. Lenstra, Job shop scheduling by simulated annealing, *Operations Research* 40 (1992), 113-125.

[540] L.N. Van Wassenhove and K.R. Baker, A bicriterion approach to time/cost tradeoffs in sequencing, *European Journal of Operational Research* 11 (1982), 48-54.

[541] L.N. Van Wassenhove and L.F. Gelders, Solving a bicriterion scheduling problem, *European Journal of Operational Research* 4 (1980), 42-48.

[542] T.A. Varvarigou, V.P. Roychowdhury, T. Kailath and E.L. Lawler, Scheduling in and out forests in the presence of communication delays, *IEEE Transactions on Parallel and Distributed Systems* 7 (1996), 1065-1074.

[543] B. Veltman, Multiprocessor scheduling with communication delays, *Ph.D. thesis*, CWI, Amsterdam, The Netherlands, 1993.

[544] B. Veltman, B.J. Lageweg and J.K. Lenstra, Multiprocessor scheduling with communication delays, *Parallel Computing* 16 (1990), 173-182.

[545] J.A. Ventura and M.X. Weng, Minimizing single-machine completion time variance *Management Science* 41 (1995), 1448-1455.

[546] S. Verma and M. Dessouky Single-machine scheduling of unit-time jobs with earliness and tardiness penalties, *Mathematics of Operations Research*, to appear.

[547] A.P.A. Vestjens, On-Line Machine Scheduling, *Ph.D. thesis*, Department of Mathematics and Computing Science, Eindhoven University of Technology, Eindhoven, The Netherlands, 1997.

[548] R.G. Vickson, Two single machine sequencing problems involving controllable job processing times, *AIIE Transactions* 12 (1980), 258-262.

[549] R.G. Vickson, Choosing the job sequence and processing times to minimize total processing plus flow cost on a single machine, *Operations Research* 28 (1980), 1155-1167.

[550] F.J. Villarreal and R.L. Bulfin, Scheduling a single machine to minimize the weighted number of tardy jobs, *IIE Transactions* 15 (1983), 337-343.

[551] S.T. Webster, New bounds for the identical parallel processor weighted flow time problem, *Management Science* 38 (1992), 124-136.

[552] S.T. Webster, Weighted flow time bounds for scheduling identical processors, *European Journal of Operational Research* 80 (1995), 103-111.

[553] S.T. Webster, The complexity of scheduling job families about a common due date, *Operations Research Letters* 20 (1997), 65-74.

[554] S.T. Webster and K.R. Baker, Scheduling groups of jobs on a single machine, *Operations Research* 43 (1995), 692-703.

[555] M.X. Weng and J.A. Ventura, Scheduling about a large common due date with tolerance to minimize absolute deviation in completion times, *Naval Research Logistics* 41 (1994), 843-851.

[556] F. Werner, On the heuristic solution of the permutation flow shop problem by path algorithms, *Computers and Operations Research* 20 (1993), 707-722.

[557] M. Widmer and A. Hertz, A new heuristic method for the flow shop sequencing problem, *European Journal of Operational Research* 41 (1989), 186-193.

[558] L.J. Wilkerson and J.D. Irwin, An improved method for scheduling independent tasks, *AIIE Transactions* 3 (1971), 239-245.

[559] D.P. Williamson, L.A. Hall, J.A. Hoogeveen, C.A.J. Hurkens, J.K. Lenstra, S.V. Sevastianov and D.B. Shmoys, Short shop schedules, *Operations Research* 45 (1997), 288-294.

[560] D.A. Wismer, Solution of the flow shop scheduling problem with no intermediate queues, *Operations Research* 20 (1972), 689-697.

[561] G.J. Woeginger, An approximation scheme for minimizing agreeably weighted variance on a single machine, Technical Report Woe-21, Department of Mathematics, TU Graz, Graz, Austria, 1998.

[562] G.J. Woeginger, When does a dynamic programming formulation guarantee the existence of an FPTAS?, Technical Report Woe-27, Department of Mathematics, TU Graz, Graz, Austria, 1998.

[563] D.L. Woodruff and M.L. Spearman, Sequencing and batching for two classes of jobs with deadlines and setup times, *Production and Operations Management* 1 (1992), 87-102.

[564] T. Yamada and R. Nakano, A genetic algorithm applicable to large-scale job-shop problems, in R. Männer and B. Manderick (eds.) *Parallel Problem Solving from Nature, Vol.2*, Elsevier, Amsterdam, 1992, 281-290.

[565] T. Yamada, B.E. Rosen and R. Nakano, A simulated annealing approach to job shop scheduling using critical block transition operators, Presented at IEEE World Congress of Computational Intelligence, Orlando, FL, USA, 1994.

[566] C.A. Yano and Y.-D. Kim, Algorithms for a class of single-machine weighted tardiness and earliness problems, *European Journal of Operational Research* 52 (1991), 167-178.

[567] W. Yu, An approximation algorithm for the total tardiness problem (in Chinese), *Journal of East China University of Science & Technology* 18 (1992), 671-677.

[568] W. Yu, Augmentations of consistent partial orders for the one-machine total tardiness problem, *Discrete Applied Mathematics* 68 (1996), 189-202.

[569] W. Yu and Z. Liu, The performance ratio of the time forward algorithm for the total tardiness problem (in Chinese), *Chinese Transaction of Operations Research* 1 (1997), 89-96.

[570] W. Yu and M. Yu, Key position method for total tardiness problem (in Chinese), *Chinese Journal of Operations Research* 14 (1995), 11-17.

[571] M. Yue, On the exact upper bound for the multifit processor scheduling algorithms, *Annals of Operations Research* 24 (1990), 233-259.

[572] S. Zdrzałka, Scheduling jobs on a single machine with release dates, delivery times, and controllable processing times: worst-case analysis, *Operations Research Letters* 10 (1991), 519-532.

[573] S. Zdrzałka, Analysis of approximation algorithms for single-machine scheduling with delivery times and sequence independent batch setup times, *European Journal of Operational Research* 80 (1995), 371-380.

[574] S. Zdrzałka and J. Grabowski, An algorithm for single machine sequencing with release dates to minimize maximum cost, *Discrete Applied Mathematics* 23 (1989), 73-89.