ELSEVIER

European Journal of Operational Research 93 (1996) 1–33

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

Invited Review

# The job shop scheduling problem: Conventional and new solution techniques

Jacek Błażewicz [a], Wolfgang Domschke [b], Erwin Pesch [c,*]

[a] *Politechnika Poznańska, Instytut Informatyki, ul. Piotrowo 3a, P-60965 Poznań, Poland*
[b] *Institut für Betriebswirtschaftslehre, Operations Research, Technische Hochschule, Hochschulstraße 1, D-64289 Darmstadt, Germany*
[c] *Institut Gesellschafts- und Wirtschaftswissenschaften, BWL 3, Universität Bonn, Adenauerallee 24–42, D-53113 Bonn, Germany*

Received October 1995

## 1. Introduction

### 1.1. The problem

A job shop consists of a set of different machines (like lathes, milling machines, drills etc.) that perform operations on jobs. Each job has a specified processing order through the machines, i.e. a job is composed of an ordered list of operations each of which is determined by the machine required and the processing time on it. There are several constraints on jobs and machines: (i) There are no precedence constraints among operations of different jobs; (ii) operations cannot be interrupted (non-preemption) and each machine can handle only one job at a time; (iii) each job can be performed only on one machine at a time. While the machine sequence of the jobs is fixed, the problem is to find the job sequences on the machines which minimize the makespan, i.e. the maximum of the completion times of all operations. It is well known that the problem is NP-hard (Lenstra and Rinnooy Kan, 1979) and belongs to the most intractable problems considered (Lawler et al., 1993).

A huge amount of literature on machine scheduling, including scheduling of job shops, has been published within the last 30 years, among others the books by Conway et al. (1967), who wrote the first book on scheduling theory, Ashour (1972), Baker (1974), Ecker (1977), Brucker (1981), French (1982), the dissertations by Rinnooy Kan (1976), Lenstra (1977), Van de Velde (1991), Nuijten (1994), Vaessens (1995), the recent books by Tanaev et al. (1994) or Chrétienne et al. (1995), and the edited books by Muth and Thompson (1963) and Coffman (1976). A broader view on production and operation scheduling is provided in Pinedo (1995), Domschke et al. (1993), Błażewicz et al. (1996), Hax and Candea (1984), Johnson and Montgomery (1974).

A variety of scheduling rules for certain types of job shops have been considered in an enormous number of publications, see the excellent surveys, also covering complexity issues and mathematical programming formulations, by Błażewicz et al. (1991), Lawler et al. (1982, 1993), Rodammer and White (1988), Błażewicz (1987), Lageweg et al. (1982), Johnson (1983), Lawler (1982), Graham et al. (1978, 1979), Salvador (1978), Day and Hottenstein (1970), Bakshi and Arora (1969), Elmaghraby (1968), Moore and Wilson (1967), and Mellor (1966).

This survey is devoted to an overview of the solution techniques available for solving the job shop problem. After a brief recollection of all the trends and techniques we will concentrate on branching

* Corresponding author. E-mail: E. Pesch@uni-bonn.de


*SSDI* 0377-2217(95)00362-2

strategies and approximation algorithms belonging to the class of opportunistic and local search scheduling. The organization of the paper is as follows: In Section 2 some exact methods will be described in detail. Section 3 is concerned with approximation algorithms divided into two categories: opportunistic problem solvers and local search heuristics.

### 1.2. Modelling

There are different problem formulations, those by Bowman (1959), Wagner (1959), and the mixed integer formulation of Manne (1960) are the first ones published; see also Fisher (1973), Fisher et al. (1983), Błażewicz et al. (1991), Brah et al. (1991) as well as extensions in Stafford (1988). We have adopted the one presented by Adams et al. (1988). Let $V = \{0, 1, \ldots, n\}$ denote the set of operations, where 0 and $n$ are considered as dummy operations "start" (the first operation of all jobs) and "end" (the last operation of all jobs), respectively. Let $M$ denote the set of $m$ machines and $A$ be the set of ordered pairs of operations constrained by the precedence relations for each job. For each machine $k$, the set $E_k$ describes the set of all pairs of operations to be performed on machine $k$, i.e. operations which cannot overlap (cf. (ii)). For each operation $i$, its processing time $p_i$ is fixed, and the earliest possible process start of $i$ is $t_i$, a variable that has to be determined during the optimization. Hence, the job shop scheduling problem can be modelled as

$$\min\ t_n$$

s.t.

$$t_j - t_i \geqslant p_i \quad \forall (i, j) \in A, \tag{1}$$

$$t_j - t_i \geqslant p_i \text{ or } t_i - t_j \geqslant p_j \quad \forall \{i, j\} \in E_k, \quad \forall k \in M, \tag{2}$$

$$t_i \geqslant 0 \quad \forall i \in V. \tag{3}$$

Restrictions (1) ensure that the processing sequence of operations in each job corresponds to the predetermined order. Constraints (2) demand that there is only one job on each machine at a time, and (3) assures completion of all jobs. Any feasible solution to the constraints (1), (2), and (3) is called a schedule.

An illuminating problem representation is the disjunctive graph model due to Roy and Sussmann (1964). It has mostly replaced the solution representation by Gantt charts as described in Gantt (1919), Clark (1922), and Porter (1968). The latter, however, is useful in user interfaces to graphically represent a solution to a problem.

In the edge-weighted graph there is a vertex for each operation, additionally there exist two dummy operations, vertices 0 and $n$, representing the start and end of a schedule, respectively. For every two consecutive operations of the same job there is a directed arc; the start vertex 0 is considered to be the first operation of every job and the end vertex $n$ is considered to be the last operation of every job. For each pair of operations $\{i, j\} \in E_k$ that require the same machine there are two arcs $(i, j)$ and $(j, i)$ with opposite directions. The operations $i$ and $j$ are said to define a disjunctive arc pair or a disjunctive edge. Thus, single arcs between operations represent the precedence constraints on the operations and opposite directed arcs between two operations represent the fact that each machine can handle at most one operation at the same time. Each arc $(i, j)$ is labeled by a weight $p_i$ corresponding to the processing time of operation $i$. All arcs from 0 have label 0.

Fig. 1 illustrates the disjunctive graph for a problem instance with three machines M1, M2, M3 and three jobs J1, J2, J3 and eight operations. The machine sequences of job J1, J2, and J3 (see the rows of Fig. 1a) are M1 → M2 → M3, M3 → M2, and

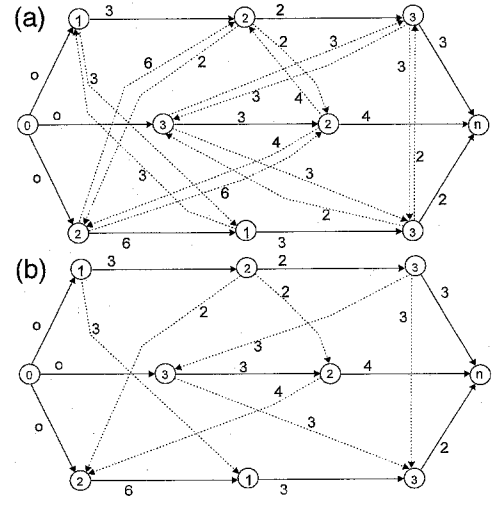


Fig. 1. (a) The disjunctive graph for the problem instance of Table 1. (b) A feasible schedule for the same problem.

M2 → M1 → M3, respectively. The processing times are presented in Table 1.

The job shop scheduling problem requires to find an order of the operations on each machine, i.e. to select one arc among all opposite directed arc pairs such that the resulting graph $G$ is acyclic (i.e. there are no precedence conflicts between operations) and the length of the maximum weight path between the start and end vertex is minimal. Obviously, the length of a maximum weight or longest path in $G$ connecting vertices 0 and $i$ equals the earliest possible starting time $t_i$ of operation $i$; the makespan of the schedule is equal to the length of the critical path, i.e. the weight of a longest path from start vertex 0 to end vertex $n$. Any arc $(i, j)$ on a critical path is said to be critical; if $i$ and $j$ are operations from different jobs then $(i, j)$ is called a disjunctive critical arc, otherwise it is a conjunctive critical arc.

In order to improve a current schedule, we have to modify the machine order of jobs (i.e. the sequence of operations) on longest paths. Therefore a neighbourhood structure can be defined by (i) reversing a disjunctive critical arc or (ii) reversing a disjunctive critical arc such that this arc is incident to an arc of the arc set $A$. For details we refer to Matsuo et al. (1988), Aarts et al. (1994), Van Laarhoven et al. (1992), and Vaessens et al. (1995).

For the problem instance of Table 1 and Fig. 1a let us consider the schedule defined by the jobs processing sequence J1 → J3 on machine M1, and J1 → J2 → J3 on machine M2 and M3. Hence all operations are lying on a longest path of length 26 (see Fig. 1b). Reversing the processing order of jobs J2 and J3 on machine M2 yields a reduced makespan of 16 for the new schedule. Extensions of the disjunctive graph representation including additional job shop constraints are discussed in White and Rogers (1990).

### 1.3. Complexity

The minimum makespan problem of job shop scheduling is a classical combinatorial optimization problem that has received considerable attention in the literature. It belongs to the most intractable problems considered. Only a few particular cases are efficiently solvable:

(i) Scheduling two jobs by the graphical method as described in Brucker (1988) and first introduced by Akers (1956). In general this idea can be used to compute good lower bounds sometimes superior to the one-machine bounds of Carlier (1982); see Brucker and Jurisch (1993).

(ii) The two-machine flow shop case, i.e. the machine sequences of all jobs are the same (Johnson, 1954); see Gonzalez/Sahni (1978).

(iii) The two-machine job shop problem where each job consists of at most two operations (Jackson, 1956).

(iv) The two-machine job shop case with unit processing times (Hefetz and Adiri, 1982; Kubiak et al., 1994).

(v) The two-machine job shop case with a fixed number of jobs (and, of course, repetitious processing of jobs on the machines (Brucker, 1994).

Slight modifications turn out to be difficult. The two-machine job shop problem where each job consists of at most three operations, the three-machine job shop problem where each job consists of at most two operations, the three-machine job shop problem with three jobs are NP-hard (Lenstra et al., 1977; Gonzalez and Sahni, 1978; Sotskov and Shakhlevich, 1993). The job shop problems with two and three machines and operation processing times equal to 1 or 2, and equal to 1, respectively, are NP-hard even in the case of preemption (Lenstra and Rinnooy Kan, 1979).

Table 1
Processing times of a 3-job 3-machine instance

| | J1 | J2 | J3 |
|---|---|---|---|
| M1 | 3 | – | 3 |
| M2 | 2 | 4 | 6 |
| M3 | 3 | 3 | 2 |

### 1.4. Flow shops

As mentioned, the two-machine flow shop case, i.e. the machine sequences of all jobs are the same, is polynomially solvable (Johnson, 1954). The idea of Johnson can be extended to a special case of three-machine flow shop scheduling (Burns and Rooker, 1978; Jackson, 1956). However, in general the three machine flow shop case is NP-hard (Röck, 1984; Sotskov, 1991) even in the case of preemption, i.e. if operations need not be continuously processed

on a machine (Gonzalez and Sahni, 1978). Flow shop scheduling on two machines with operation release dates is NP-hard (Garey, Johnson and Sethi, 1976) also in the preemptive case (Cho and Sahni, 1981).

Permutation flow shop problems, i.e. a schedule with identical job processing order on all machines, are in some cases polynomially solvable (Rinnooy Kan, 1976; Monma and Rinnooy Kan, 1983) or admit to develop good lower bounds (Lageweg et al., 1978; Ignall and Schrage, 1965) or dominance rules (McMahon, 1969; Szwarc, 1971, 1973). The latter have been applied in early enumeration approaches (Potts, 1980a; Gupta and Reddi, 1978; Szwarc, 1978). Potts et al. (1991) showed that there are instances for which the objective value of the optimal permutation schedule is much worse (by a factor of the number of machines) than that of the regular optimal flow shop schedule. The current champions for solving permutation flow shops are the fast insertion method of Nawaz et al. (1983), cf. Turner and Booth (1987), the extension of Palmer's heuristic (1965) by Hundal and Rajgopal (1988), the effective simulated annealing algorithm of Osman and Potts (1989), and the heuristical minimization of gaps between successive operations by Ho and Chang (1991). The latter consider a neighbour of a permutation schedule as a schedule that can be obtained by moving a single job to another position.

A survey on earlier approaches in order to schedule flow shops exactly can be found in Elmaghraby and Elshafei (1976), Baker (1975), and Kawaguchi and Kyan (1988). Dudek et al. (1992) review flow shop sequencing research since 1954.

Noteworthy flow shop heuristics, parallelizing the work on optimal procedures for the makespan criterion, are those of Campbell et al. (1970) and Dannenbring (1977). Both used Johnson's algorithm, the former to solve a series of two-machine approximations to obtain a complete schedule, which the latter locally improved by switching adjacent jobs in the sequence. For a survey on problem solutions on which Johnson's algorithm and underlying ideas have had an influence see Proust (1992).

Widmer and Hertz (1989) and Taillard (1990) solved the flow shop sequencing problem using tabu search. Neighbours are defined mainly as in the traveling salesman case: exchanging the order of processing for two operations. Computational results are compared to a number of previously published procedures. They made little use of features such as different strategies of intensification and diversification of the search with respect to the dynamics of the tabu list in order to reinforce attributes of attractive solutions to drive the search into new regions, as demonstrated by Hübscher and Glover (1992) for parallel machine scheduling. The currently most effective heuristic for scheduling flow shops with even 500 jobs and 20 machines is the tabu search approach by Nowicki and Smutnicki (1994). Their work parallels their earlier tabu search approach on job shop scheduling.

### *1.5. The history*

The history of the job shop scheduling problem, starting more than 30 years ago, is also the history of a well known benchmark problem consisting of ten jobs and ten machines and introduced by Fisher and Thompson in 1963. This particular instance of a 10-job 10-machine problem opposed its solution for 25 years leading to a competition among researchers for the most powerful solution procedure. Since then branch and bound procedures have received substantial attention from numerous researchers. Early work was performed by Brooks and White (1965), followed by Greenberg (1968), whose method was based on Manne's integer programming formulation. Further papers included Balas (1969), Charlton and Death (1970), Florian et al. (1971), Ashour et al. (1974), Ashour and Hiremath (1973), and Fisher (1973), who obtained lower bounds by the use of Lagrange multipliers.

A long time the algorithm of McMahon and Florian (1975) was the best exact solution method. Instead of using the worse bounds of Charlton and Death (1970) they combined the bounds for the one-machine scheduling problem with operation release dates and the objective function to minimize maximum lateness with the enumeration of active schedules (see Giffler and Thompson, 1960), among which are also optimal ones. An alternative approach whereby at each stage one disjunctive arc of some crucial pair is selected leads to a computationally inferior method (Lageweg et al., 1977).

Considerable effort has been invested in the empirical testing of various priority rules, see Gere (1966) and the survey papers of Day and Hottenstein (1970), Panwalkar and Iskander (1977) and Haupt (1989). Giffler et al. (1963) applied a probabilistic priority rule or random sampling for operation selection in order to build a feasible schedule.

During the last ten years substantial algorithmic improvements were achieved and accurately reflected by the stepwise optimum approach for the notorious 10-job 10-machine problem. Fisher et al. (1983) applied computationally costly surrogate duality relaxations, weighting and aggregating into a single constraint, either machine capacity constraints or job operation precedence constraints. An augmented Lagrangian relaxation approach is considered in Hoitomt et al. (1993) for more practical-like job shops of a size about 143 jobs and 43 machines. The idea of an augmented Lagrangian relaxation basically is a Lagrangian relaxation with constraint-related penalty terms added to the objective function. Violation of constraints is associated with the addition of a term of high cost to the objective function. A penalty coefficient serves as a measure for the penalty severity. If this coefficient is increased, then the solution of the augmented Lagrangian relaxation problem approaches the solution of the underlying job shop problem. A first attempt to obtain bounds by polyhedral techniques was made by Balas (1985). The neighbourhood structure used in some recent local search algorithms is also mainly employed as branching structure in the exact method of Barker and McMahon (1985). They rearrange operations on a longest path if the operations use the same machine. Lawler et al. (1993) report that, with respect to the famous $10 \times 10$ problem "Lageweg (1984) found a schedule of 930, without proving optimality; he also computed a number of multi-machine lower bounds, ranging from a three-machine bound of 874 to a six-machine bound of 907". So he was the first who found an optimal solution. Optimality of a schedule of length 930 was first proven by Carlier and Pinson (1989). Their algorithm is based on bounds obtained for the one machine problems with precedence constraints, release dates and allowed preemptions. This problem is polynomially solvable. Additionally, they used several simple but effective inference rules on operation subsets.

Currently the job shop champions among the exact methods are, besides the branch and bound implementations of Applegate and Cook (1991), Martin and Shmoys (1995) and Perregaard and Clausen (1995), the branch and bound algorithms of Caseau and Laburthe (1995), Baptiste et al. (1995a,b), Carlier and Pinson (1990, 1994), Brucker, Jurisch and Sievers (1994, 1992), and Brucker, Jurisch and Krämer (1992) (see also Pinson, 1988, 1992). The power of their methods basically results from some inference rules which describe simple cuts, and a branching scheme such that operations which belong to a block (a sequence of operations on a machine) on the longest path are moved to the block ends, hence improving an idea of Grabowski et al. (1986).

Nowadays, tailored approximation methods viewed as an opportunistic (greedy-type) problem solving process can yield optimal or near-optimal solutions even for problem instances up to now considered as difficult (Adams, Balas and Zawack, 1988; Ow and Smith, 1988; Sadeh, 1991; Balas et al., 1995; Dauzere-Peres and Lasserre, 1993; Balas and Vazacopoulos, 1995). Hereby opportunistic problem solving or opportunistic reasoning characterizes a problem solving process where local decisions on which operations, jobs, or machines should be considered next, are concentrated on the most promising aspects of the problem, e.g. job contention on a particular machine. Hence subproblems often defining bottlenecks are extracted and separately solved and serve as a basis from which the search process can expand. Breaking down the whole problem into smaller pieces takes place until, eventually, sufficiently small subproblems are created for which effective exact or heuristic procedures are available. However the way in which a problem is decomposed affects the quality of the solution reached. Not only the type of decomposition such as machine/resource (Adams et al., 1988), job/order (Pesch, 1993), or event based (Sadeh, 1991) has a dramatic influence on the outcome but also the number of subproblems and the order of their consideration. In fact, an opportunistic view suggests that the initial decomposition be reviewed in the course of problem solving to see if changes are necessary. The shifting bottleneck heuristic from Adams et al. (1988) and its improving modifications from Balas et al. (1995) and Dauzere-Peres and Lasserre (1993) are typical repre-

sentatives of opportunistic reasoning. It is resource based as there are sequences of one-machine schedules successively solved and their solutions introduced into the overall schedule. In recent years local search based scheduling became very popular; for a survey see Barnes et al. (1992), Glass et al. (1992), Anderson et al. (1995), Brucker (1995), as well as Vaessens et al. (1995). These algorithms are all based on a certain neighbourhood structure and some rules defining how to obtain a new solution from existing ones. The first efforts to implement powerful general problem solvers such as simulated annealing (Van Laarhoven et al., 1992; Matsuo et al., 1988), parallel tabu search (Taillard, 1994), and genetic algorithms (Aarts et al., 1994; Nakano and Yamada, 1991; Yamada and Nakano, 1992; Storer et al., 1991, 1992, 1992a,b) finally culminated in the excellent tabu search implementation of Dell'Amico and Trubian (1993), Nowicki and Smutnicki (1993) and Balas and Vazacopoulos (1995). Among the genetic based methods only a few, e.g. Yamada and Nakano (1992), Dorndorf and Pesch (1993a,b), Pesch (1993) and Mattfeld (1995), could solve the notorious 10-job 10-machine problem optimally. Most of the current local search approaches rely on naive search neighbourhoods which fail to exploit problem specific knowledge. Applications of local and probabilistic search methods to sequencing problems are based on neighbourhoods defined in the solution space of the problem. The method of Storer et al. (1992a) is based on problem perturbation neighbourhoods, i.e. the original data is genetically perturbed and a neighbour is defined as a solution which is obtained when a base heuristic is applied to the perturbed problem. The obtained solution sequence for the perturbed problem is mapped to the original data, i.e. the non-perturbed operations are scheduled in the same way and the makespan of the solution to the original problem data defines the quality of the perturbed problem. Other exact and heuristic methods are described in the dissertations of Bräsel (1990) (and Bräsel and Werner, 1989) and Meyer (1992).

The local search heuristics like simulated annealing, tabu search, and genetic algorithms are modestly robust under different problem structures and require only a reasonable amount of implementation work with relatively little insight into the combinatorial structure of the problem. Problem specific characteristics are mainly introduced via some improvement procedures, the kind of representation of solutions as well as their modifications based on some neighbourhood structure.

Throughout this survey, experimental results are reported mainly for the $10 \times 10$ problem. Techniques that are giving good results on the $10 \times 10$ problem need not necessarily perform well on other instances of the job shop scheduling problem, even for instances of the same size ($10 \times 10$) like LA19, LA20, ORB2, ORB3, ORB4 (cf. Applegate and Cook, 1991). Applegate and Cook's algorithm is very efficient on the $10 \times 10$ problem, but much less on other instances, in particular LA19, ORB2 and ORB3. On the contrary, a lot of the successful approaches mentioned use important ideas from Applegate and Cook's paper. An interesting benchmark is LA21. Vaessens (1995) solved it with a modified version of Applegate and Cook's algorithm (about 40,000,000 nodes). Then it was solved by Baptiste, Le Pape and Nuijten (1995) in about 4,000,000 nodes and 48 hours, by Caseau and Laburthe in about 2,000,000 nodes and 24 hours and by Martin and Shmoys (1995) in about one hour.

In the next section we will go into detail and present ideas of some exact algorithms and heuristic approaches

## 2. Exact methods

In this section we will be concerned with branch and bound algorithms, exploring specific knowledge about the critical path of the job shop scheduling problem.

### *2.1. Branch and bound*

The principle of branch and bound is the enumeration of all feasible solutions of a combinatorial optimization problem, say a minimization problem, such that properties or attributes not shared by any optimal solution are detected as early as possible. An attribute (or branch of the enumeration tree) defines a subset of the set of all feasible solutions of the original problem where each element of the subset satisfies this attribute. In general, attributes are chosen such that the union of all attribute-defined sub-

sets equals the set of all feasible solutions of the problem and any two of these subsets do not intersect. For each subset the objective value of its best solution is estimated by a lower bound (bounding). An optimal solution of a relaxation of the original problem such that this optimal solution also satisfies the subset defining attribute, serves as a lower bound. In case the lower bound exceeds the value of the best (smallest) known upper bound (a heuristic solution of the original problem) the attribute-defined subset can be dropped from further consideration. Otherwise, search is continued departing from the most promising subset which is devided into smaller subsets through the definition of additional attributes. Hence, at any search stage a subset of solutions is defined by a set of attributes all of which are satisfied by these solutions.

We shall see that the attributes of a branch and bound process exactly correspond to attributes forbidding moves in tabu search. Branching from one solution subset to a new smaller one can be associated with a tabu search move.

### 2.2. Lower bounds

One of the main drawbacks of all branch and bound methods is the lack of strong lower bounds in order to cut branches of the enumeration tree as early as possible. Several types of lower bounds are applied in the literature, for instance, bounds based on Lagrangian relaxation (Van de Velde, 1991), bounds based on the optimal solution of a subproblem consisting of only two or three jobs and all machines (Akers, 1956; Brucker, 1988; Brucker and Jurisch, 1993). However the most prominent bounding procedure has been described by Carlier (1982) and Potts (1980b). Consider any operation $i$ in the job shop or in the disjunctive graph that may include already a partial selection of arcs from disjunctive arc pairs. Then there is a longest path from the artifical vertex 0 to $i$ of length $r_i$ as well as a longest path of length $q_i$ connecting the end of operation $i$ to the last one, the dummy operation $n$. Operation $i$ cannot start to be processed earlier than its release date $r_i$ (also called head) and its processing has to be finished no later than its due date $t_n - q_i$ in order to cause no schedule delay. The time $q_i$ is said to be the tail of operation $i$. There exist $m$ one-machine lower bounds for the optimal makespan of the job shop scheduling problem where each bound is obtained from the exact solution of a one-machine scheduling problem with release dates and due dates. Although this problem is NP-complete, Carlier's algorithm quickly solves the one machine problems optimally for all problem sizes in the job shop under consideration. Balas, Lenstra, and Vazacopoulos (1995) describe an even better branch and bound procedure that can yield improved lower bounds. Their method additionally takes minimum delays between pairs of operations into account. That means, if there is a directed path connecting operations $i$ and $j$ in the disjunctive graph, then

$$t_j - t_i \geqslant L(i, j), \tag{4}$$

where $L(i, j)$ is the length of the path connecting $i$ and $j$.

While the one-machine scheduling problem with heads $r_i$ and tails $q_i$, for all operations $i$, can be solved in $O(n \cdot \log n)$ time if $r_i = r_j$, for all $i, j$ (use the longest tail rule, i.e. schedule the released jobs in order of decreasing tails) or if $q_i = q_j$, for all $i, j$ (use the shortest head rule, i.e. schedule the jobs in order of increasing heads), this is not true any longer if time lags $L(i, j)$ are imposed (Balas et al., 1995).

The branch and bound algorithms of Carlier (1982) and Balas et al. (1995) extensively make use of the fact that the shortest makespan of a one machine schedule cannot fall below

$$\mathrm{LB1}(C) := \min\{r_i \mid i \in C\} + \sum_{i \in C} p_i + \min\{q_i \mid i \in C\} \tag{5}$$

for any subset $C$ of all operations which have to be scheduled on a particular machine, where $p_i$ is the processing time of operation $i$. This lower bound can be calculated in $O(n \cdot \log n)$ time by solving the preemptive one machine problem without time lags. It is well known that the strongest bound $\mathrm{LB1}(C)$ equals the minimum makespan of the preemptive version of the problem. Let us consider the idea of branching. Consider a schedule produced by the longest tail rule. Let $C := (0, i_1, i_2, \ldots, i_p, n)$ be a sequence of operations constituting a critical path. Further, let $c$ be the last operation encountered in $C$ such that $q_c < q_{i_p}$, i.e. all operations in $C$ between $c$ and $n$ have tails at least $q_{i_p}$. Let $J$ denote the set of

these operations excluding $c$ and $n$. Then branching basically is based on the following observation: If $r_i \geqslant \max\{t_{i_1}, t_c\}$, for all $i \in J$, and if the part $C(c, i_p)$ of the critical path $C$ connecting $c$ to $i_p$ contains no precedence relation, then the longest tail schedule is optimal in case $c = 0$. Otherwise, if $c > 0$, in any schedule better than the current one, job $c$ either precedes or succeeds all jobs in $J$. For details see Balas et al. (1995).

Carlier (1982) and Balas et al. (1995) (see also Carlier and Pinson, 1989, 1990, 1994; Brucker et al., 1994, 1992; Caseau and Laburthe, 1995; Applegate and Cook, 1991) applied a lot of additional inference rules – several are summarized in the sequel – in order to cut the enumeration tree during a preprocessing or the search phase.

### 2.3. Branching

Consider once more the one-machine scheduling problem consisting of the set $N$ of operations, release dates $r_i$ and tails $q_i$ for all $i \in N$. Let $\mathscr{C}_{\max}$ be the maximum completion time of a feasible schedule, i.e. $\mathscr{C}_{\max}$ is an upper bound for the makespan of an optimal one machine schedule. Let $E_C$, $S_C$ and $C$ be subsets of $N$ such that $E_C$, $S_C \subseteq C$ and any operation $j \in C$ also belongs to $E_C$ ($S_C$) if there is an optimal one-machine schedule such that $j$ is first (last) among all operations in $C$. Then the following conditions hold for any operation k of N:

$$\text{If } r_k + \sum_{i \in C} p_i + \min\{q_i \mid i \in S_C,\ i \neq k\} > \mathscr{C}_{\max}$$
$$\text{then } k \notin E_C. \tag{6}$$

$$\text{If } \min\{r_i \mid i \in E_C,\ i \neq k\} + \sum_{i \in C} p_i + q_k > \mathscr{C}_{\max}$$
$$\text{then } k \notin S_C. \tag{7}$$

$$\text{If } k \notin E_C \text{ and } \mathrm{LB1}(C - \{k\}) + p_k > \mathscr{C}_{\max}$$
$$\text{then } k \in S_C. \tag{8}$$

$$\text{If } k \notin S_C \text{ and } \mathrm{LB1}(C - \{k\}) + p_k > \mathscr{C}_{\max}$$
$$\text{then } k \in E_C. \tag{9}$$

The preceding results tell us that, if $C$ contains only two operations $i$ and $k$ such that $r_k + p_k + p_i + q_i > \mathscr{C}_{\max}$, then operation $i$ is processed before $k$, i.e. from the disjunctive arc pair connecting $i$ and $k$ arc $(i, k)$ is selected. Moreover (8) and (9) can be used in order to adapt heads and tails within the branch and bound process (Carlier, 1982; Grabowski et al., 1986; Carlier and Pinson, 1989). If (8) or (9) holds, then one can fix all arcs $(i, k)$ or $(k, i)$, respectively, for all operations $i \in C$, $i \neq k$. Application of (6) to (9) guarantees an immediate selection of certain arcs from disjunctive arc pairs before branching into a subtree. There are problem instances such that conditions (6) to (9) cut the enumeration tree substantially.

The branching structure of Carlier and Pinson (1989, 1990, 1994) is based on the disjunctive arc pairs which define exactly two subtrees. Let $i$ and $j$ be such a pair of operations which have to be scheduled on a critical machine, i.e. a machine with longest initial lower bound (= the preemptive one-machine solution). Then, roughly speaking, according to Bertier and Roy (1965), both sequences of the two operations are checked with respect to their regrets if they increase the best lower bound LB. Let

$$d_{ij} := \max\{0,\ r_i + p_i + p_j + q_j - \mathrm{LB}\},$$
$$d_{ji} := \max\{0,\ r_j + p_j + p_i + q_i - \mathrm{LB}\}, \tag{10}$$
$$a_{ij} := \min\{d_{ij},\ d_{ji}\}, \quad b_{ij} := |\, d_{ij} - d_{ji}\,|,$$

Among all possible candidates of disjunctive arc pairs with respect to the critical machine that one is chosen that maximizes $b_{ij}$ and, in case of a tie, the pair is chosen with the maximum $a_{ij}$. Carlier and Pinson were the first to prove that an optimal solution of the $10 \times 10$ benchmark has a makespan of 930. In order reach this goal a lot of work had to be done. In 1971 Florian, Trepant and McMahon proposed a branch and bound algorithm where at a certain time the set of available operations is considered, i.e. all operations without any unscheduled predecessor are possible candidates for branching. At each node of the enumeration tree the number of branches generated correponds to the number of available operations competing for a particular machine. Branching continues from that node with smallest lower bound regarding the node corresponding partial schedule. As lower bounds Florian et al. (1971) used a one machine lower bound without considering tails, i.e. the optimal sequencing of the operations on this particular machine is in increasing order of the earliest possible start times. They could find a solution of 1041 for the $10 \times 10$ benchmark, thus, a slight improvement compared to Balas' best

solution of 1177 obtained two years earlier. His work was based on the disjunctive graph concept where he considered two successor nodes in the enumeration tree instead of as many nodes as there are conflicting operations. McMahon and Florian (1975) laid the foundation for Carlier's one-machine paper. Contrary to earlier approaches where the nodes of the enumeration tree correspond to incomplete (partial) schedules, they associated a complete solution with each search tree node. An initial solution and upper bound is obtained by Schrage's algorithms, i.e. among all available (released) jobs choose always that one with longest tail (earliest due date). Their problem is to minimize maximum lateness on one machine where each job is described by its release time, the processing time, and its due date. At any search node a critical job $j$ (with respect to the node associated schedule) is defined to be a job that realizes the value of the maximum lateness in the given schedule. Hence, an improvement is only possible if $j$ is scheduled earlier. The idea of branching is to consider those jobs having greater due dates than the critical job (i.e. having smaller tails than the critical job) and to schedule these jobs after the critical one. They continuously apply Schrage's algorithm in order to obtain a feasible schedule while the heads are adapted appropriately. McMahon and Florian also used their branching structure in order to solve the minimum makespan job shop scheduling problem and reached a value of 972 for the $10 \times 10$ problem. Moreover, they were the first to solve the Fisher and Thompson $5 \times 20$ benchmark to optimality.

Lageweg et al. (1977) were first to use the one-machine lower bound, hence extending the previously used lower bounds. They generated all active schedules branching over the conflict set in Giffler and Thompson's algorithm (see Section 3) or branching over the disjunctive arcs. A priority rule at each node of the search tree delivers an upper bound. There is no report on the notorious 10-jobs 10-machines problem.

Barker and McMahon (1985) associated with each node in their enumeration tree a subproblem whose solutions are a subset to the solution set of the original problem; a complete schedule; a critical block in the schedule which is used to determine the descendant subproblems; and a lower bound on the value of the solutions of the subproblem. The lower bound is a single-machine lower bound as computed in McMahon and Florian (1975). Each node of the search tree is associated with a different subproblem, i.e. at each node in the search tree there is a complete schedule containing a critical operation (a critical operation is the earliest scheduled operation $i$ where $t_i + q_i$ is at least the value of the best solution) and an associated critical block (a continuous sequence of operations on a single machine ending with a critical operation). The critical operation must be scheduled earlier if this subproblem is to yield an improved solution. Thus, a set of subproblems is explored, in each of which a different member of the critical block is made to precede all other members or to be the last of the operations in the block to be scheduled. Earliest start times and tails are adapted accordingly. While Barker and McMahon (1985) reached a value of 960 for the $10 \times 10$ problem they were not able to solve the $5 \times 20$ problem to optimality. Only a value of 1303 is obtained.

Branching in the algorithm of Brucker, Jurisch and Sievers (1992, 1994) is also restricted to moves of operations which belong to a critical path of a solution obtained by a heuristic based on dispatching rules. For a block $B$, i.e. successively processed operations on the same machine, that belongs to a critical path, new subtrees are generated if an operation is moved to the very beginning or the very end of this block. In any case the critical path is modified and additional disjunctive arcs are selected according to formulas (5)–(9) proposed by Carlier and Pinson (1989). Brucker et al. (1994) calculated different lower bounds: one machine relaxations and two jobs relaxation (Brucker and Jurisch, 1993). Moreover, if an operation $i$ is moved before the block $B$, all disjunctive arcs $\{(i, j) \mid j \in B \text{ and } j \neq i\}$ are fixed. Hence,

$$r_i + p_i + \max\left\{ \max_{\substack{j \in B \\ j \neq i}} (p_j + q_j);\ \sum_{\substack{j \in B \\ j \neq i}} p_j + \min_{\substack{j \in B \\ j \neq i}} q_j \right\}$$

is a simple lower bound for the search tree node. Similarly, the value

$$\max\left\{ \max_{\substack{j \in B \\ j \neq i}} (r_j + p_j);\ \sum_{\substack{j \in B \\ j \neq i}} p_j + \min_{\substack{j \in B \\ j \neq i}} r_j \right\} + p_i + q_i$$

is a lower bound for the search tree node if operation $i$ is moved to the very end position of block $B$. In order to keep the generated subproblems nonintersecting it is necessary to fix some additional arcs. Promissing subproblems are heuristically detected. Brucker et al. (1992, 1994) were able to improve and accelerate the branch and bound algorithm of Carlier and Pinson (1989) substantially and easily reached an optimal schedule for the $10 \times 10$ problem. However, they were unable to find an optimal solution for the $5 \times 20$ problem within a reasonable amount of time.

The heads and tails update rules (6) to (9) can also be considered in terms of equations. This results in the assignment of time windows of possible start times of operations to operation sets supposed to be scheduled on the same machine. The branching idea of Martin and Shmoys (1995) uses the tightness of these windows as a branching criterion. For tight or almost tight windows, where the window size equals (almost) the sum of the processing times of the operation set $C$, branching depends on which operation in $C$ is processed first. When an operation is chosen to be first the size of its window is reduced. The size of the windows of the other operations in $C$ are updated in order to reflect the fact that they cannot start until the chosen operation is completed. Comparable propagation ideas (after branching on disjunctive arcs) based on time window assignments to operations are considered by Caseau and Laburthe (1995). They found an optimal schedule to the $10 \times 10$ problem within less than 3 minutes, Martin and Shmoys (1995) needed about 9 minutes. In both papers, the updating of windows on operations of one machine causes further updates on all other machines. This iterated one-machine windows reduction algorithm generated lower bounds superior to the one-machine lower bound.

Perregaard and Clausen (1995) obtained excellent results through a parallel branch and bound algorithm on a 16-processor system based on Intel i860 processors each with 16 MB internal memory. There is a peak performance of about 500 MIPS. As a lower bound Jackson's preemtive schedule is used. One branching strategy is the one by Carlier and Pinson (1989) where one new disjunctive arc pair describes the branches originating from a node; this is done in analogy to the rules (6)–(10). Another branching strategy considered is the one described in Brucker et al. (1994), i.e. moving operations to block ends. Perregaard and Clausen (1995) easily found an optimal solution to the $10 \times 10$ or $5 \times 20$ problem, both in much less than one minute (of course including the optimality proof). For some other even more difficult problems they could prove optimality or obtained results unknown up to now.

### 2.4. Valid inequalities

Among the most efficient algorithms for solving the job shop scheduling problem is the branch and bound approach by Applegate and Cook (1991). In order to obtain good lower bounds they developed cutting plane procedures for both the disjunctive and the mixed integer problem formulation (Manne, 1960). In the latter case the disjunctive constraints can be modelled introducing a binary variable $y_{ij}^m$ for any operation pair $i$, $j$ supposed to be processed on the same machine $m$. The interpretation is that $y_{ij}^m$ equals 1 if $i$ is scheduled before $j$ on machine $m$, and 0 if $j$ is scheduled before $i$. Let $\Omega$ be some large constant. Then the following inequalities hold for all operations $i$ and $j$ on machine $m$:

$$t_i \geqslant t_j + p_j - \Omega\, y_{ij}^m, \tag{11}$$

$$t_j \geqslant t_i + p_i - \Omega\big(1 - y_{ij}^m\big). \tag{12}$$

Starting from the LP-relaxation valid inequalities involving the variables $y_{ij}$ as well inequalities developed for the disjunctive programming formulation are generated. Consider a feasible solution and a set $C$ of operations processed on the same machine $m$. Let $i \in C$ be an operation processed on $m$ and assume all members of the subset $C_i$ of $C$ are scheduled before $i$ on $m$. Then the start time $t_i$ of operation $i \notin C_i$ satisfies

$$\begin{aligned} t_i &\geqslant \min\{r_j \mid j \in C_i\} + \sum_{j \in C_i} p_j \\ &\geqslant \min\{r_j \mid j \in C\} + \sum_{j \in C_i} p_j. \end{aligned} \tag{13}$$

In order to become independent of the considered schedule we multiply this inequality by $p_i$ and sum

over all members of $C$. Hence, we get the basic cuts (Applegate and Cook, 1991):

$$\sum_{i\in C} t_i p_i \geqslant \left(\sum_{i\in C} p_i\right) \min\{r_j \mid j\in C\} + \frac{1}{2} \sum_{\substack{i\in C \\ i\neq j}} \sum_{\substack{j\in C \\ j\neq i}} p_j p_i. \tag{14}$$

With the addition of the variables $y_{ij}^m$ we can easily reformulate (13) and obtain the half cuts

$$t_i \geqslant \min\{r_j \mid j\in C\} + \sum_{\substack{j\in C \\ j\neq i}} y_{ji}^m p_j \tag{15}$$

because $y_{ji}^m = 1$ if and only if $j\in C_i$.

Let $i$ and $j$ be two operations supposed to be scheduled on the same machine. Let $\alpha$ and $\beta$ be any two nonnegative parameters. Assume $i$ is supposed to be processed before $j$, then $\alpha t_i + \beta t_j \geqslant \alpha r_i + \beta(r_i + p_i)$ because $t_i \geqslant r_i$ and operation $j$ cannot start earlier until $i$ is finished. Similarly, under the assumption that $j$ is scheduled before $i$, we get $\alpha t_i + \beta t_j \geqslant \alpha(r_j + p_j) + \beta r_j$. Thus, both inequalities hold, for instance, if the right hand sides of these inequalities are equal. Both inequalities are satisfied if $\alpha = p_i + r_i - r_j$ and $\beta = p_j + r_j - r_i$. Hence, the two-job cuts (Balas, 1985)

$$(p_i + r_i - r_j)t_i + (p_j + r_j - r_i)t_j \geqslant p_i p_j + r_i p_j + r_j p_i \tag{16}$$

sharpen (14) if $r_i + p_i > r_j$ and $r_j + p_j > r_i$.

The lower bounds in the branch and bound algorithm of Applegate and Cook (1991) are based on these inequalities, the corresponding reverse ones, i.e. considering the jobs in reverse order, and a couple of additional cuts. The cutting plane based lower bounds are superior to the one machine lower bounds, however, on the cost of additional computation time. For instance, for the $10\times 10$ problem Applegate and Cook were able to improve the one-machine bound of 808 obtained in less than one second up to 824 (in 300 seconds) or 827 in 7500 seconds. The branch and bound tree is established continuing the search from a tree node where the preemptive one-machine lower bound is minimum. The branching scheme results from the disjunctive model, i.e. each disjunctive edge defines two subproblems according to the corresponding disjunctive arcs. However, among all possible disjunctive edges that one is chosen, connecting operations $i$ and $j$, which maximizes the minimum $\{\mathrm{LB}(i\to j), \mathrm{LB}(j\to i)\}$ where $\mathrm{LB}(i\to j)$ and $\mathrm{LB}(j\to i)$ are the two preemptive one-machine lower bounds for the generated subproblems where the disjunctive arcs $(i, j)$ or $(j, i)$, respectively, are selected. Furthermore, in a more sophisticated branch and bound method branching is realized on the basis of the values $a_{ij}$ and $b_{ij}$ of (10). Then, based on the work of Carlier and Pinson (1989), inequalities (6) and (7) are applied for all machines $m$ and all operation subsets on this particular machine in order to eliminate disjunctive edges, simplifying the problem. In order to get a high-quality feasible solution Applegate and Cook modified the shifting bottleneck procedure of Adams et al. (1988). After scheduling all machines except $s$ ones, for the remaining $s$ machines the bottleneck criterion (see below) is replaced by complete enumeration. Not necessarily the machine with largest makespan is included into the partial schedule but each of the remaining $s$ machines is considered to be the one introduced next into the partial schedule. The value $s$ has to be small in order to keep the computation time low.

In order to obtain better feasible solutions they proceed as follows. Given a complete schedule the processing orders of the jobs on a small number $s$ of machines is kept fixed. The processing orders on the remaining machines are skipped. The resulting par-

$t := 0$; $Q(t) := \{1,\ldots,n-1\}$;
**repeat**
Among all unscheduled operations in $Q(t)$ let $j^*$ be the one with the smallest completion time, i.e. $c_{j^*} = \min\{c_j \mid j\in Q(t)\}$. Let $m^*$ denote the machine $j^*$ has to be processed on.
Randomly choose an operation $i$ from the conflict set $\{j\in Q(t) \mid j$ has to be processed on machine $m^*$ and $r_j < c_{j^*}\}$.
$Q(t) := Q(t)\setminus\{i\}$; Modify $c_j$ for all operations $j\in Q(t)$. Set $t$ to the next possible operation to machine assignment.
**until** $Q(t)$ is empty.

Fig. 2. The algorithm of Giffler and Thompson (1960).

tial schedule can be quickly completed to a new schedule using the forementioned branch and bound procedure. If the new schedule is shorter than the original, then this process is repeated with the restriction that the set of machines whose schedules are kept fixed is modified. The number $s$ of machines to fix is dictated by the need to have enough structure to rapidly fill in the remainder of the schedule, and, leaving sufficient amount of freedom for improving the processing orders. See Applegate and Cook (1991), for additional information on how to choose $s$, running times and the quality of the feasible solutions. Applegate and Cook easily found an optimal solution to the $10 \times 10$ job shop in less than 2 minutes (including the proof of optimality).

## 3. Approximation algorithms

### 3.1. Priority rules

Priority rules are probably the most frequently applied heuristics for solving (job shop) scheduling problems in practice because of their ease of implementation and their low time complexity. The algorithm of Giffler and Thompson (1960) can be considered as a common basis of all priority rule based heuristics. Let $Q(t)$ be the set of all unscheduled operations at time $t$. Let $r_i$ and $c_i$ denote the earliest possible start and the earliest possible completion time, respectively, of operation $i$. The algorithm of Giffler and Thompson assigns available operations to machines, i.e. operations which can start being processed. Conflicts, i.e. operations competing for the same machine, are solved randomly. A brief outline of the algorithm is given in Fig. 2.

The Giffler and Thompson algorithm can generate all active schedules among which are also optimal schedules. As the conflict set consists only of operations, i.e. jobs, competing for the same machine, the random choice of an operation or job from the conflict set may be considered as the simplest version of a priority rule where the priority assigned to each operation or job in the conflict set corresponds to a certain probability. Many other priority rules can be considered, for instance the total processing time of all subsequent job operations of operation $i$ may be a criterion. Some rules are collected in Table 2; for an extended summary and discussion see Panwalkar and Iskander (1977) as well as Blackstone et al. (1982) and Haupt (1989). The first column of Table 2 contains an abbreviation and name of the rule while the last column describes which operation or job in the conflict set gets highest priority.

### 3.2. The shifting bottleneck heuristic

The shifting bottleneck heuristic from Adams, Balas and Zawack (1988) and recently from Balas, Lenstra and Vazacopoulos (1995) is one of the powerful procedures among heuristics for the job shop scheduling problem. The idea is to solve for each machine a one-machine scheduling problem to optimality under the assumption that a lot of arc direc-

Table 2
Priority rules

| | Rule | Description |
|---|---|---|
| 1. | SOT (shortest operation time) | An operation with shortest processing time on the considered machine |
| 2. | LOT (longest operation time) | An operation with longest processing time on the machine considered. |
| 3. | LRPT (longest remaining processing time) | An operation with longest remaining job processing time. |
| 4. | SRPT (shortest remaining processing time) | An operation with shortest remaining job processing time. |
| 5. | LORPT (longest operation remaining processing time) | An operation with highest sum of tail and operation processing time. |
| 6. | Random | The operation for the considered machine is randomly chosen. |
| 7. | FCFS (first come first served) | The first operation in the queue of jobs waiting for the same machine. |
| 8. | SPT (shortest processing time) | A job with smallest total processing time. |
| 9. | LPT (longest processing time) | A job with longest total processing time. |
| 10. | LOS (longest operation successor) | An operation with longest subsequent operation processing time. |
| 11. | SNRO (smallest number of remaining operations) | An operation with smallest number of subsequent job operations. |
| 12. | LNRO (largest number of remaining operations) | An operation with largest number of subsequent job operations. |

tions in the optimal one machine schedules coincide with an optimal job shop schedule. Consider all operations of a job shop scheduling instance that have to be scheduled on machine $m$. In the (disjunctive) graph including a partial selection of opposite directed arcs (corresponding to a partial schedule) there exists a longest path of length $r_i$ from dummy operation 0 to each operation $i$ scheduled on machine $m$. Processing of operation $i$ cannot start before $r_i$. There is also a longest path of length $q_i$ from $i$ to the dummy operation $n$. Obviously, when $i$ is finished, it will take at least $q_i$ time units to finish the whole schedule. Although the one-machine scheduling problem with heads and tails is NP-complete, there is the powerful branch and bound method (see Section 2.2) proposed by Potts (1980b) and Carlier (1982, 1987) which dynamically changes heads and tails in order to improve the operations' sequence, see also Larson et al. (1985) and Nowicki and Zdrzalka (1986).

The shifting bottleneck heuristic consists of two subroutines. The first one (SB1) repeatedly solves one-machine scheduling problems while the second one (SB2) builds a partial enumeration tree where each path from the root to a leaf is similar to an application of SB1. As the very name suggests, the shifting bottleneck heuristic always schedules bottleneck machines first. As a measure of the bottleneck quality of machine $m$, the value of an optimal solution of a certain one-machine scheduling problem on machine $m$ is used. The one machine scheduling problems in consideration are those which arise from the disjunctive graph model when certain machines are already sequenced. The operation orders on sequenced machines are fully determined. Hence sequencing an additional machine probably results in a change of heads and tails of those operations whose machine order is still open. For all machines not sequenced, the maximum makespan of the corresponding optimal one-machine schedules, where the arc directions of the already sequenced machines are fixed, determines the bottleneck machine. In order to minimize the makespan of the job shop scheduling problem the bottleneck machine should be sequenced first. A brief statement of the shifting bottleneck procedure is given in Fig. 3.

The one-machine scheduling problems, although they are NP-hard (contrary to the preemptive case, cf. Baker et al. (1983)), can quickly be solved using the algorithm of Carlier (1982). Unfortunately, adjusting heads and tails does not take into account a possible already fixed processing order of operations connecting two operations $i$ and $j$ on the same machine, whereby this particular machine is still unscheduled. So, we get one-machine scheduling problems with heads, tails, and time lags (minimum delay between two operations), problems which cannot be handled with Carlier's algorithm. In order to overcome these difficulties an improved SB1 version is suggested by Dauzere-Peres and Lasserre (1993) using approximate one-machine solutions. Balas et al. (1995) solved the one-machine problems exactly,

Let $M$ be the set of all machines and let $M' := \{\}$ be the set of all sequenced machines;
**repeat**
  **for** $m \in M \setminus M'$ **do**
  **begin**
    Compute head and tail for each operation $i$ that has to be scheduled on machine $m$.
    Solve the one machine scheduling problem to optimality for machine $m$; let $v(m)$ be the resulting makespan for this machine.
  **end;**
  Let $m^*$ be the bottleneck machine, i.e. $v(m^*) \geq v(m)$ for all $m \in M \setminus M'$.
  $M' := M' \cup \{m^*\}$;

  /* local reoptimization */
  **for** $m \in M'$ in the order of its inclusion **do**
  **begin**
    Delete all arcs between operations on $m$ while all arc directions between operations on machines from $M' \setminus \{m\}$ are fixed.
    Compute heads and tails of all operations on machine $m$ and solve the one-machine scheduling problem.
  **end;**
**until** $M = M'$.

Fig. 3. Outline of the SB1 heuristic.

see also Dell'Amico (1993). So, there is a SB1 heuristic superior to the SB1 heuristic proposed by Adams et al. (1988). However, on average, the results are slightly worse than those obtained by Balas et al.'s SB2 heuristic.

During the local reoptimization part of the SB1 heuristic, for each machine, the operation sequence is redetermined keeping the sequences of all other already scheduled machines untouched. As the one-machine problems use only partial knowledge of the whole problem, it is not surprising, that optimal solutions will not be found easily. This is even more the case because Carlier's algorithm considers the one-machine problem as consisting of independent jobs while some dependence between jobs of a machine might exist in the job shop scheduling problem. Moreover, a monotonic decrease of the makespan is not quaranteed in the reoptimization step of Adams et al. Dauzere-Peres and Lasserre (1993) were the first to improve the robustness of SB1 and to ensure a monotonic decrease of the makespan in the reoptimization phase and therefore eliminate sensitivity to the number of local reoptimization cycles. Contrary to Carlier's algorithm, they update the operation release dates each time they select a new operation by Schrage's procedure. They obtained a solution of 950 with the modified version of the SB1 heuristic.

The quality of the schedules obtained by the SB1 heuristic heavily depends on the sequence in which the one machine problems are solved and thus on the order these machines are included in the set $M'$. Sequence changes may yield substantial improvements. This is the idea behind the second version of the shifting bottleneck procedure, i.e. the SB2 heuristic, as well as behind the second genetic algorithm approach by Dorndorf and Pesch (1995). The SB2 heuristic applies a slightly modified SB1 heuristic to the nodes of a partial enumeration tree. A node corresponds to a set $M'$ of machines that have been sequenced in a particular way. The root of the search tree corresponds to $M' = \{\}$. A branch corresponds to the inclusion of a machine $m$ into $M'$, thus the branch leads to a node representing an extended set $M' \cup (m)$. At each node of the search tree a single step of the SB1 heuristic is applied, i.e. machine $m$ is included followed by a local reoptimization. Each node in the search tree corresponds to a particular sequence of inclusion of the machines into the set $M'$. Thus, the bottleneck criterion no longer determines the inclusion into $M'$. Obviously a complete enumeration of the search tree is not acceptable. Therefore a breadth-first search up to depth $l$ is followed by a depth-first search. In the former case, for a search node corresponding to set $M'$ all possible branches are considered which result from inclusion of machine $m \notin M'$. Hence the successor nodes of node $M'$ correspond to machine sets $M' \cup \{m\}$ for all $m \in M \setminus M'$. Beyond the depth $l$ an extended bottleneck criterion is applied, i.e. instead of $|M \setminus M'|$ successor nodes there are several successor nodes generated corresponding to the inclusion of the bottleneck machine as well as several other machines $m$ to $M'$.

### 3.3. Opportunistic scheduling

A long time priority rules were the only possible way to tackle job shops of at least 100 operations (Crowston et al., 1963). Recently, generally applicable approximation procedures such as tabu search, simulated annealing or genetic algorithm learning strategies became very attractive and successful solution strategies. There general idea is to modify current solutions in a certain sense, where the modifications are defined by a neighbourhood operator, such that new feasible solutions are generated, the so called neighbours, which hopefully have an improved or at most limited deterioration of their objective function value. In order to reach this goal problem specific knowledge, incorporated by problem specific heuristics, has to be introduced into the local search process of the general problem solvers (Crama et al., 1995).

Knowledge based scheduling systems have been built by various people using various techniques (Kusiak and Chen, 1988; and Sauer, 1995). Some of them are rule based systems, others are based on frame representations. Some of them only use heuristic rules to construct a schedule, others conduct a constraint directed state space search. ISIS (Fox, 1987; Fox and Smith, 1984) is a constraint directed reasoning system for the scheduling of factory job shops. The main feature is that it formalizes various scheduling influences as constraints on the system's knowledge base and uses these constraints to guide

the search in order to generate heuristically the schedule. In each scheduling cycle it first selects an order of jobs to be scheduled according to the priority rules and then proceeds through a level of analysis of existing schedules, a level of constraint directed search and a level of detailed assignment of resources and time intervals for each operation in order. A large amount of work done by ISIS actually involves the extraction and organization of constraints that are created specifically for the problem under consideration. Scheduling relies only on job based problem decomposition. The system OPIS (Ow and Smith, 1988; Smith et al., 1986), which is a direct descendant of ISIS, attempts to make some progress by concentrating more on bottlenecks and scheduling under the perspective of resource based decomposition (Adams et al., 1988; Chu et al., 1992; Balas et al., 1995; Dauzere-Peres and Lasserre, 1993). The term "opportunistic reasoning" has been used to characterize a problem-solving process whereby activity is consistently directed toward those actions that appear most promising in terms of the current problem-solving state. The strategy is to identify the most "solvable" aspects of the problem (e.g. those aspects with the least number of choices or where powerful heuristics are known) and develop candidate solutions to these subproblems. However the way in which a problem is decomposed affects the quality of the solution reached. No subproblem contains all the information of the original problem. Subproblems should be as independent as possible in terms of effects of decisions on other subproblems. OPIS is an opportunistic scheduling system using a genetic opportunistic scheduling procedure. For instance, constantly redirect the scheduling effort towards those resources that are likely to be the most difficult to schedule (so-called bottleneck resources). Decomposing the job shop into single-machine scheduling problems bottleneck machines might get a higher priority for being scheduled first. Hence, dynamically revised decision making based on heuristic rules focuses on the most critical decision points and the most promising decisions at these points (Sadeh, 1991). The average complexity of the procedures is kept on a very low level by interleaving the search with application of consistency enforcing techniques and a set of look-ahead techniques that help decide which operation to schedule next (i.e. so-called variable-ordering and value-ordering techniques). Clearly, start times of operations competing for highly contended machines are more likely to become unavailable than those of other operations. A critical variable is one that is expected to cause backtracking, namely one whose remaining possible values are expected to conflict with the remaining possible values of other variables. A good value is one that is expected to participate in many solutions. Contention between unscheduled operations for a machine over some time interval is determined by the number of unscheduled operations competing for that machine/time interval and the reliance of each one of these operations on the availability of this machine/time interval. Typically, operations with few possible starting times left will heavily rely on the availability of any one of these remaining starting times in competition, whereas operations with many remaining starting times will rely much less on any one of these times. Each starting time is assigned a subjective probability to be assigned to a particular operation. The operation with the highest contribution to the demand for the most contended machine/time interval is considered the most likely to violate a capacity constraint, see Pesch and Tetzlaff (1995) as well as Caseau and Laburthe (1995) and Sadeh (1991).

### 3.4. Local search

An important issue is the extent to which problem specific knowledge must be used in the construction of learning algorithms (in other words the power and quality of inferencing rules) capable to provide significant performance improvements. Very general methods having a wide range of applicability in general are weak with respect to their performance. Problem specific methods achieve a highly efficient learning but with little use in other problem domains. Local search strategies (Pirlot, 1992) are falling somewhat in between these two extremes, where genetic algorithms or neural networks tend to belong to the former category while tabu search or simulated annealing etc. are counted as instances of the second category. Anyway, these methods can be viewed as tools for searching a space of legal alternatives in order to find a best solution within reasonable time limitations. What is required are techniques for rapid location of high-quality solutions in large-

size and complex search spaces and without any guarantee of optimality. When sufficient knowledge about such search spaces is available a priori, one can often exploit that knowledge (inference) in order to introduce problem specific search strategies capable of supporting to find rapidly solutions of higher quality. Whithout such a priori knowledge, or in cases where close to optimum solutions are indispensible, information about the problem has to be accumulated dynamically during the search process. Likewise obtained long-term as well as short-term memorized knowledge constitutes one of the basic parts in order to control the search process and in order to avoid getting stuck in a locally optimal solution. Previous approaches to deal with combinatorially explosive search spaces about which little knowledge is known a priori are unable to learn how to escape a local optimum. For instance, consider a random search. This can be effective if the search space is reasonably dense with acceptable solutions, such that the probability to find one is high. However, in most cases finding an acceptable solution within a reasonable amount of time is impossible because any kind of random search is not using any knowledge generated during the search process in order to improve its performance. Consider hill-climbing in which better solutions are found by exploring solutions "close" to a current and best one found so far. Hill-climbing techniques work well within a search space with relatively "few" hills. Iterated hill-climbing from randomly selected solutions can frequently improve the performance, however, any global information assessed during the search will not be exploited. Statistical sampling techniques are typical alternative approaches which emphasize the accumulation and exploitation of more global information. Generally speaking they operate by iteratively dividing the search space into regions to be sampled. Regions unlikely to produce acceptable solutions are discarded while the remaining ones will be subdivided for further sampling, and so on. If the number of useful subregions is small, this search process can be effective. However, in case that the amount of a priori search space knowledge is pretty small, as is the case for many applications in business and engineering, this strategy frequently is not satisfactory.

Combining hill-climbing as well as random sampling in a creative way and introducing concepts of learning and memory can overcome the above mentioned deficiencies. The obtained strategies dubbed "local search based learning" are known, for instance, under the names tabu search and genetic algorithms. They provide general problem solving strategies incorporating and exploiting problem-specific knowledge capable even to explore search spaces containing an exponentially growing number of local optima with respect to the problem defining parameters.

To be more specific consider the minimization problem $\min\{f(x) \mid x \in S\}$, where $f$ is the objective function, i.e. the desired goal, and $S$ is the search space, i.e. the set of feasible solutions of the problem. One of the most intuitive solution approaches to this optimization problem is to start with a known feasible solution and slightly perturb it while decreasing the value of the objective function. In order to operationalize the concept of slight perturbation let us associate with every $x \in S$ a subset $N(x)$ of $S$, called neighbourhood of $x$. The solutions in $N(x)$, or neighbours of $x$, are viewed as perturbations of $x$. They are considered to be "close" to $x$. Now the idea of a local search algorithm is to start with some initial solution and move from neighbour to neighbour as long as possible while decreasing the objective value.

The attractiveness of local search procedures stems from their wide applicability and (usually) low empirical complexity (see Johnson et al. (1988) and Yannakakis (1990) for more information on the theoretical complexity of local search). All that is needed here is a reasonable definition of neighbourhoods, and an efficient way of searching them. There is usually no guarantee that the value of the objective function at an arbitrary local optimum comes close to the optimal value.

In simulated annealing procedures (Metropolis et al., 1953; Aarts and Korst, 1989; Van Laarhoven and Aarts, 1987), the sequence of solutions does not roll monotonically down towards a local optimum. First, a neighbour of $x$, say $y \in N(x)$, is selected (usually, but not necessarily, at random). Then, based on the amplitude of $\Delta = f(x) - f(y)$, a transition from $x$ to $y$ (i.e., an update of $x$ by $y$) is either accepted or rejected. This decision is made nondeterministically. The probability of acceptance should be so that

escaping local optima is relatively easy during the first iterations, and the procedure explores the set $S$ freely. But, as the iteration count increases, only improving transitions tend to be accepted, and the solution path is likely to terminate in a local optimum.

One of the central ideas of tabu search (Glover, 1989, 1990a,b) is to guide deterministically the local search process out of local optima (in contrast with the non-deterministic approach of simulated annealing). This can be done using different criteria, which ensure that the loss incurred in the value of the objective function in such an "escaping" step (a move) is not too important, or is somehow compensated for.

A straightforward criterion for leaving local optima is to replace the improvement step in the local search procedure by a "least deteriorating" step. The resulting procedure replaces the current solution $x$ by a solution $y \in N(x)$ which maximizes $\Delta = f(x) - f(y)$. If during a number (a termination parameter) of iterations no improvements are found, the procedure stops. Notice that $\Delta$ may be negative, thus resulting in a deterioration of the objective function. In order to avoid to reverse the transition, and go back to the local optimum $x$ (since $x$ improves on $y$) a list of forbidden transitions (tabu list) is maintained throughout the search. To be more specific, the transition to the neighbour solution, i.e. a move, may be described by one or more attributes. These attributes (when properly chosen) can become the foundation for creating a so-called attribute based memory, cf. the attribute definition in branch and bound.

*3.4.1. Tabu search and simulated annealing based job shop scheduling*

In recent years local search based scheduling of job shops became very popular; for a survey see Barnes et al. (1992), Vaessens et al. (1995), Vaessens (1995), as well as Anderson et al. (1995). These algorithms are all based on a certain neighbourhood structure how to obtain a new solution from existing ones. Van Laarhoven et al. (1992) used the very simple neighbourhood structure (N1) in their simulated annealing procedure:

N1: A transition from a current solution to a new one is generated by replacing in the disjunctive graph representation of the current solution a disjunctive arc $(i, j)$ on a critical path by its opposite arc $(j, i)$.

In other words, N1 means reversing the order in which two operations $i$ and $j$ (or jobs) are processed on a machine where these two operations belong to a longest path. This parallels the early branching structures of exact methods. It is possible to construct a finite sequence of transitions leading from a locally optimal solution to the global optimum. This is a necessary and sufficient condition for asymptotic convergence of simulated annealing. On average (on a VAX 785 over five runs on each instance) it took about 16 hours to solve the $10 \times 10$ benchmark to optimality. A value of 937 was reached within almost 100 minutes. The $5 \times 20$ benchmark problem was solved to optimality within almost 18 hours.

Lourenço (1993, 1995) introduces a combination of small step moves based on the neighbourhood N1 and large step moves in order to reach new search areas. The small steps are responsible for search intensification in a relatively narrow area. Therefore a simple hill-climbing as well as simulated annealing are used, both with respect to neighbourhood N1. The large step moves modify the current schedule and drive the search to a new region. Simultaneously a modest optimization is performed to obtain a schedule reasonably close to a local optimum. This large step optimization then is followed by local search such as hill-climbing or simulated annealing. The large steps considered are the following: Randomly select two machines and remove all disjunctive arcs connecting operations on these two machines in the current schedule. Then solve the two one-machine problems – using Carlier's algorithm or allowing preemption and considering time lags – and return the obtained arcs according to their one-machine solutions into the whole schedule.

Starting solutions are generated through some randomized dispatching rules, one for instance, in the same way as Bierwirth (1995) (see below) represents and generates feasible schedules. Lourenço (1993, 1995) never found an optimal solution to the $10 \times 10$ problem; the best result she obtained was a schedule with makespan 937. The running times are comparable to those by Van Laarhoven et al. (1992).

More powerful neighbourhood definitions are necessary. Matsuo et al. (1988) defined a neighbourhood (N2) which has also been applied in the local search improvement steps of Aarts et al.'s (1994) genetic algorithms:

N2: Consider a feasible solution and a critical arc $(i, j)$ defining the processing order of operations $i$ and $j$ on the same machine, say machine $m$. Define $p(i)$ and $s(i)$ to be the predecessor and successor of $i$, respectively, on machine $m$. Restrict the choice of arc $(i, j)$ to those vertices such that at least one of the arcs $(p(i), i)$ or $(j, s(j))$ is not on a longest path, i.e. $i$ or $j$ are block end vertices (cf. the branching structure of Brucker et al. (1994)). Reverse $(i, j)$ and, additionally also reverse $(p(h), h)$ and $(k, s(k))$ – provided they exist – where $h$ directly precedes $i$ in its job, and $k$ is the immediate successor of $j$ in its job. The latter arcs are reversed only if a reduction of the makespan can be achieved.

Thus, a neighbour of a solution with respect to N2 may be found by reversing more than one arc. Within a time bound of 99 seconds the results of two simulated annealing algorithms based on the two different neighbourhood structures N1 and N2 are 969 and 977, respectively, for the $10 \times 10$ problem as well as 1216 and 1245, respectively, for the $5 \times 20$ problem (Aarts et al., 1994).

Dell'Amico and Trubian (1993) considered the problem as being symetric and scheduled operations bidirectional, i.e. from the beginning and from the end, in order to obtain a priority rules based feasible solution. The resulting two parts finally are put together in order to constitute a complete solution. The neighbourhood structure (N3) employed in their tabu search extends the connected neighbourhood structure N1:

N3: Let $(i, j)$ be a disjunctive critical arc. Consider all permutations of the three operations $\{p(i), i, j\}$ and $\{i, j, s(j)\}$ in which $(i, j)$ is inverted.

Again, it is possible to construct a finite sequence of moves with respect to N3 which leads from any feasible solution to an optimal one. In a restricted version N3′ of N3, arc $(i, j)$ is chosen such that either $i$ or $j$ is the end vertex of a block. In other words, arc $(i, j)$ is not considered as candidate when both $(p(i), i)$ and $(j, s(j))$ are on a longest path in the current solution. N3′ is not any longer a connected neighbourhood. Another branching scheme is considered to define a neighbourhood structure N4:

N4: For all operations $i$ in a block move $i$ to the very beginning or to the very end of this block.

Once more, N4 is connected, i.e. for each feasible solution it is possible to construct a finite sequence of moves, with respect to N4, leading to a globally optimal solution. For a while the tabu search by Dell'Amico and Trubian (1993) was the most powerful method to solve jobs shop. They were able to find an optimal solution to the $5 \times 20$ problem within 2.5 minutes and a solution of 935 to the $10 \times 10$ problem in about the same amount of time.

N1 and N4 are also the two neighbourhood structures Sun et al. (1995) used in their tabu search. In 40 benchmark problems they always obtained better solutions or reduced running times compared to the shifting bottleneck procedure. For instance, they generated an optimal solution to the $10 \times 10$ problem within 157 seconds.

In the parallel tabu search Taillard (1994) used the N1 neighbourhood. Every 15 iterations the length of the tabu list is randomly changed between 8 and 14. He obtained high quality solutions even for very large problem instances up to 100 jobs and 20 machines.

Barnes and Chambers (1994) also used N1 in their tabu search algorithm. They fixed the tabu list length and whenever no feasible move is available the list entries are deleted. Start solutions are obtained through dispatching rules.

Nowadays, the most efficient tabu search implementations are described in Nowicki and Smutnicki (1993) and Balas and Vazacopoulos (1995). The size of the neighbourhood N1 depends on the number of critical paths in a schedule and the number of operations on each critical path. It can be pretty large. Nowicki and Smutnicki (1993) consider a smaller neighbourhood (N5) restricting N1 (or N4) to reversals on the border of a block. Moreover, they restrict to a single critical path arbitrarily selected.

N5: A move is defined by the interchange of two successive operations $i$ and $j$, where $i$ or $j$ is the first or last operation in a block that belongs to a critical path. In the first block only the last two operations and symmetrically in the last block of the critical path only the first two operations are swapped.

The set of moves is not empty only if the number of blocks is more than one and if at least one block consists of more than one operation. In other words, if the set of moves is empty then the schedule is optimal. If we consider neighbourhoods N1 and N5 in more detail, then we can deduce: A schedule obtained from reversing any disjunctive arc which is not critical cannot improve the makespan; a move that belongs to N1 but not to N5 cannot reduce the makespan. Let us go into more detail of Nowicki and Smutnicki (1993).

The neighbourhood searching strategy includes an aspiration criterion as shown in Fig. 4.

The design of a classical tabu search algorithm is straightforward. A stopping criterion is when the optimal schedule is detected or the number of iterations without any improvement exceeds a certain limit. The initial solution can be generated using an insertion technique as described by Nawaz et al. (1983). Nowicki and Smutnicki (1993) note that the essential disadvantage of this approach consists of loosing information about previous runs. Therefore they suggest to build up a list of the $l$ best solutions and their associated tabu lists during the search. Whenever the classical tabu search has finished, go to the most recent entry, i.e. the best schedule $x$ from this list of at most $l$ solutions, and restart the classical tabu search. Whenever a new best solution is encountered the list of best solutions is updated. This extended tabu search "with backtracking" continues until the list of best solutions is empty. The results Nowicki and Smutnicki (1993) obtained are excellent; for instance, they could solve the notorious $10 \times 10$ problem within 30 seconds to optimality, even on a small personal computer. They solved the $5 \times 20$ problem within 3 seconds to optimality.

The idea of Balas and Vazacopoulos' (1995) guided local search procedure is based on reversing more than one disjunctive arc at a time. This leads to a considerably larger neighbourhood than in the previous cases. Moreover, neighbours are defined by interchanging a set of arcs of varying size, hence the search is of variable depth and supports search diversification in the solution space. The employed neighbourhood structure (N6) is an extension of all previously encountered neighbourhood structures. Consider any feasible schedule $x$ and any two operations $i$ and $j$ to be performed on the same machine, such that $i$ and $j$ are on the same critical path, say $P(0, n)$, but not necessarily adjacent. Assume $i$ is processed before $j$. Besides $p(i)$, $p(j)$ and $s(i)$, $s(j)$, the immediate machine predecessors and machine successors of $i$ and $j$ in $x$, let $a(i)$, $a(j)$ and $b(i)$ and $b(j)$ denote the job predecessors and job successors of operations $i$ and $j$, respectively. Moreover, let $r(i) := r_i + p_i$ and $q(i) := p_i + q_i$ be the length of a longest path (including the processing time of $i$, $p_i$) connecting 0 and $i$, or $i$ and $n$. An interchange on $i$ and $j$ either is a move of $i$ right after $j$ (forward interchange) or a move of $j$ right before $i$ (backward interchange). We have seen that the schedule $x$ cannot be improved by an interchange on $i$ and $j$ if $i$ and $j$ are adjacent and none of them is the first or last operation of a block in

Let $x$ be a current schedule (feasible solution) and $C(x)$ is its makespan; $N(x)$ denotes the set of all neighbours of $x$; $\mathscr{C}_{max}$ is the makespan of the currently best solution and $T$ contains the set of tabu moves (tabu list).
Let $A$ be the set $\{x' \in N(x) \mid \text{Move}(x \to x') \in T \text{ and } C(x') < \mathscr{C}_{max}\}$; i.e. all schedules in $A$ satisfy the aspiration criterion to improve the currently best makespan.
**if** $\{N(x) \mid \text{Move}(x \to x') \text{ is not tabu}\} \cup A$ is not empty **then**
  select $y = \text{argmin}\{C(x') \mid x' \in N(x)$ or if $\text{Move}(x \to x')$ is tabu then $x' \in A\}$
**else**
  **repeat**
    Drop the "oldest" entry in $T$ and append a copy of the last element in $T$
  **until** there is a non-tabu move $\text{Move}(x \to y)$
Assume $\text{Move}(x \to y)$ is defined by arc $(i, j)$ in $x$ then append arc $(j, i)$ to $T$.

Fig. 4. Neighbourhood searching strategy of Nowicki and Smutnicki (1993).

$P(0, n)$. In other words, in order to achieve an improvement either $a(i)$ or $b(j)$ must be contained in $P(0, n)$. This statement can easily be generalized to the case where $i$ is not an immediately preceding operation of $j$. Thus for an interchange on $i$ and $j$ to reduce the makespan, it is necessary that the critical path $P(0, n)$ containing $i$ and $j$ also contains at least one of the operations $a(i)$ or $b(j)$. Hence, the number of "attractive" interchanges reduces drastically and the question remains, under which conditions an interchange on $i$ and $j$ is guaranteed not to create a cycle. It is easy to derive that a forward interchange on $i$ and $j$ yields a new schedule $x'$ (obtained from $x$) if there is no directed path from $b(i)$ to $j$ in $x$. Similarly, a backward interchange on $i$ and $j$ will not create a cycle if there is no directed path from $i$ to $a(j)$ in $x$.

Now, the neighbourhood structure N6 can be introduced.

N6: A neighbour $x'$ of a schedule $x$ is obtained by an interchange of two operations $i$ and $j$ in one block of a critical path. Either operation $j$ is the last one in the block and there is no directed path in $x$ connecting the job successor of $i$ to $j$, or, operation $i$ is the first one in the block and there is no directed path in $x$ connecting $i$ to the job predecessor of $j$.

Whereas the neighbourhood N1 involves the reversal of a single arc $(i, j)$ on a critical path the more general move defined by N6 involves the reversal of potentially a large number of arcs.

Assume that an interchange on an operations pair $i, j$ results in a makespan increase of the new schedule $x'$ compared to the old one $x$. Then it is obvious that every critical path in $x'$ contains the arc $(j, i)$. The authors make use of this fact in order to further reduce the neighbourhood size. Consider a forward interchange resulting in a makespan increase: Since $(j, i)$ is a member of any critical path in $x'$ the arc $(i, b(i))$ is as well (because $i$ became the last operation in its block). We have to distinguish two cases. Either the length of a longest path from $b(i)$ to $n$ in $x$, say $q(b(i))$, exceeds the length of a longest path from $b(j)$ to $n$ in $x$, say $q(b(j))$ or $q(b(i)) \leqslant q(b(j))$. In the former case $q(b(i))$ is responsible for the makespan increase. In the latter case $s(i)$ is the first operation in its block in $x'$. Hence, the length $r(j)$ of a longest path in $x$ connecting 0 to $j$ is smaller than the length $r'(i)$ of a longest path in $x'$ connecting 0 to $i$. Thus, the number of interchange candidates can be reduced. In a forward interchange a right guidepost $h$ is reached if

$$q(b(h)) < q(b(i));$$

a left guidepost $h$ is reached if

$$r(j) < r'(h)$$

holds. Equivalently, in a backward interchange that worsens the makespan of schedule $x$ a left guidepost $h$ is reached if

$$r(p(h)) < r(p(j));$$

a right guidepost $h$ is reached if

$$q(i) < q'(h)$$

holds.

After an interchange that increased the makespan, if a left guidepost is reached the list of candidates for an interchange is restricted to those operation pairs on a critical path in $x'$ between 0 and $j$. If a right guidepost is reached candidates for an interchange are chosen from the segment on a critical path in $x'$ between $j$ and $n$. If both guideposts are reached, the set of candidates is not restricted. Thus, in summary, if the makespan increases after an interchange, available guideposts restrict the neighbourhood.

The guided local search procedure by Balas and Vazacopoulos (1995) uses the neighbourhood structure N6 inclusive the restrictions aforementioned. The procedure builds up an incomplete enumeration (called neighbourhood) tree. Each node of the tree corresponds to a schedule, an edge of the tree joins two schedules $x$ and $x'$ where descendant $x'$ is obtained through an interchange on two operations $i$ and $j$ lying on a critical path in $x$. The arc $(j, i)$ is fixed in all schedules corresponding to the nodes of the subtree rooted at $x'$. The number of direct descendants of $x'$, i.e. the number of possible moves, is the entire neighbourhood if $x'$ is a shorter schedule than $x$. It is the resticted (with respect to the guideposts) neighbourhood if the makespan of $x'$ is worse than the one of $x$. The children of a node corresponding to a schedule $x$ are ranked by their evaluations. The number of children is limited by a de-

creasing function of the depth in the neighbourhood tree. After an interchange on $i$, $j$ leading from $x$ to schedule $x'$ the arc $(j, i)$ remains fixed in all schedules of the subtree rooted in $x'$. Additionally, the arc $(j, i)$ is also fixed in all schedules corresponding to brothers of $x'$ (i.e. children of $x$) having a makespan worse than $x'$ (in the sequence of the ranked list). Finally, besides arc fixing and limits on the number of children a third factor is applied to keep the size of the tree small. The depth of the tree is limited by a logarithmic function of the number of operations on the tree's level. Altogether, the size of the neighbourhood tree is bounded by a linear function of the number of operations.

The number of neighourbood trees generated is governed by some rules. The root of a new neighbourhood tree corresponds to the best schedule available if it is generated in the current tree. Otherwise, if the current tree is not a step into a better local optimum the root of the new tree is randomly chosen among the nodes of the current tree. The root of the first tree is generated using the most work remaining dispatching rule.

In order to combine local search procedures operating on different neighbourhoods (which makes it more likely to escape local optima and explore regions not available by any single neighbourhood structure) Balas and Vazacopoulos (1995) combined their guided local search with the shifting bottleneck procedure. Remember, every time a new machine has been sequenced the shifting bottleneck procedure reoptimizes the sequence of each previously processed machine, by again solving a one machine problem with the sequence on the other machines held fixed. The idea of Balas and Vazacopoulos is to replace the reoptimization cycle of the shifting bottleneck procedure with the neighbourhood trees of the guided local search procedure. Whenever there are $m_0$ fixed machine sequences defining a partial schedule the shifting bottleneck guided local search (SB–GLS) generates $2 \cdot m_0 \cdot \#\text{jobs}$ neighbourhood trees instead of starting a reoptimization cycle. The root of the first tree is defined by the partial schedule of the $m_0$ already sequenced machines. The roots of the other trees are obtained as described above. The best schedule obtained from this incorporated guided local search then is used as a starting point for continuation of the shifting bottleneck procedure. A couple of modifications of SB–GLS ideas are applied which basically differ from SB–GLS in the number of sequenced machines (hence the root of the first neighbourhood tree) in the shifting bottleneck part (Balas and Vazacopoulos, 1995). SB–GLS and its modifications currently is the most powerful heuristic to solve job shop scheduling problems. It outperforms many others in solution quality and computation time. Needless to say that all versions of GLS and SB–GLS easily could solve the $10 \times 10$ problem to optimality in a time between 12 seconds up to a couple of minutes. We recommend the reader to study the extensive computational work provided in Balas and Vazacopoulos (1995).

### *3.4.2. Genetic based job shop scheduling*

Genetic algorithms are motivated by the theory of evolution and date back to the early work of Holland (1975). They have been designed as general search strategies and optimization methods.

Roughly speaking, a genetic algorithm aims at producing near-optimal solutions by letting a population of random solutions undergo a sequence of unary and binary transformations governed by a selection scheme biased towards high-quality solutions.

These transformations constitute the recombination step of a genetic algorithm and are performed on the population by three simple operators. The effect of the operators is that implicitly good properties are identified and combined into a new population which hopefully has the property that the best solution and the average value of the solutions are better than in previous populations. The process is then repeated until some stopping criteria are met (Goldberg, 1989; Michalewicz, 1992).

Compared to standard heuristics genetic algorithms are not well suited for fine-tuning structures which are very close to optimal solutions. Therefore it is essential if a competitive genetic algorithm is desired, to incorporate (local search) improvement operators (Davis, 1985; Husbands et al., 1991; Starkweather et al., 1992; Hilliard and Liepins, 1988).

Putting things into a more general framework, a solution of a combinatorial optimization problem may be considered as a sequence of local decisions. A local decision for the job shop scheduling problem

might be the choice of an operation to be scheduled next. In an enumeration tree of all possible decision sequences a solution of the problem is represented as a path corresponding to the different decisions from the root of the tree to some leaf. Genetics can guide a search process in order to learn to find the most promising decisions. The scheme of a so called genetic enumeration algorithm is described in Fig. 5, it requires further refinements in order to design a successful algorithm.

During the recombination step for reproduction the fitness value of a solution has to be defined. Usually the fitness value is the value of the objective function or some scaled version of it when all local decision rules and the improvement step have been applied. Next a new population is constructed. Therefore a new temporary population is generated where each member is a replica of a member of the old population. A copy of an old string is produced with probability proportional to its fitness value, i.e. better strings probably get more copies. The generation of new individuals is handled by the crossover operator.

In order to apply the crossover operator the population is randomly partitioned into pairs. Next, for each pair, the crossover operator is applied with a certain probability by choosing a position randomly in the string and exchanging the tails (defined as the substring starting at the chosen position) of the two strings.

The mutation operator which makes random changes to single elements of the string only plays a secondary role in genetic algorithms. Mutation serves to maintain diversity in the population.

The traditional genetic algorithm based on a binary string repesentation of a solution is often unsuitable for many optimization problems involving integral variables because it is very difficult to represent a solution such that substrings have a meaningful interpretation. Choosing a more natural representation of solutions, however, involves more intricated recombination operators, in particular crossover operators in order to get feasible offspring. To overcome these difficulties and apply the simple crossover operator one can use an interpretation of an individual solution as a sequence of decision rules as described first in Dorndorf and Pesch (1995). An individual of a population is considered to be a subset of feasible schedules from the set of all feasible schedules.

Each individual of the priority rule based genetic algorithm (P-GA) is a string of $n-1$ entries $(p_1, p_2, \ldots, p_{n-1})$ where $n-1$ is the number of operations in the underlying problem instance. An entry $p_i$ represents one rule of the set of priority rules described in Table 2. The entry in the $i$-th position says that a conflict in the $i$-th iteration of the Giffler–Thompson algorithm should be resolved using priority rule $p_i$. More precisely, an operation from the conflict set has to be selected by rule $p_i$; ties are broken by a random choice. Within a genetic framework a best sequence of priority rules has to be determined. An analogous encoding scheme has been used by Della Croce et al. (1995). The difference

**Initialization.** Construct an initial population of individuals each of which is a string of local decision rules.

**Assessment / Improvement.** Assess each individual in the current population introducing problem specific knowledge by special purpose heuristics (such as local search) which are guided by the sequence of local decisions.

**if** special purpose heuristics lead to a new string of local decision rules **then**
replace each individual by the new one, for instance a locally optimal one.

**repeat**
**Recombination.** Extend the current population by adding individuals obtained by unary and binary transformations (crossover, mutation) on one or two individuals in the current population.
**Assessment / Improvement.** Assess each individual in the current population introducing problem specific knowledge by special purpose heuristics (such as local search) which are guided by the sequence of local decisions.
**if** special purpose heuristics lead to a new string of local decision rules **then**
replace each individual by the new one, for instance a locally optimal one.
**Selection.** Reduce the extended population to its original size according to the selection rules.
**until** some stopping criterion is met.

Fig. 5. Genetic enumeration.

being that an individual is divided into substrings of preference lists such that a substring concentrates operation's selection for a particular machine.

The crossover operator is straightforward. Obviously, the simple crossover, where the substrings of two cut strings are exchanged, applies and always yields feasible offspring. Heuristic information already occurs in the encoding scheme and a particular improvement step – contrary to genetic local search approaches (Aarts et al., 1994; Ulder et al., 1991) – is dropped. The mutation operator applied with a very small probability simply switches a string position to another one, i.e. the priority rule of a randomly chosen string entry is replaced by a new rule randomly chosen among the remaining ones. The approach of Dorndorf and Pesch (1995) to search a best sequence of decision rules for selecting operations is just in line with the ideas of Fisher and Thompson (1963) on probabilistic learning of sequences consisting of two priority rules, and Crowston et al. (1963) or O'Grady and Harrison (1985) on learning how to find promising linear combinations of basic priorities. Fisher and Thompson (1963) were amongst the first to suggest an adaptive approach by using a combination of rules in a sequencing system. They proposed using two separate sequencing criteria and, when a decision was taken, a random choice of a rule was made. Initially, there was an equal probability of selecting each rule but as the system progressed these probabilities were adjusted according to a predefined learning procedure. Crowston et al. (1963) considered the following rules: SOT, LOT, LRPT, FCFS, least remaining job slack per operation, least remaining machine slack. Their idea was to create a rule (as a linear combination of the above mentioned priority rules) capable of decisions which cannot be specified by any of the rules in isolation. Furthermore, the projection of the combined rule should yield each individual rule (see also O'Grady and Harrison, 1985). In their experiments they restricted consideration to SOT and LRT.

Besides using the genetic algorithm as a metastrategy to optimally control the use of priority rules, another genetic algorithm described in Dorndorf and Pesch (1995) controls the selection of nodes in the enumeration tree of the shifting bottleneck heuristic (shifting bottleneck based genetic algorithm, SB-GA). Remember that the SB2 heuristic is only a repeated application of a part of the SB1 heuristic where the sequence in which the one machine problems are solved is predetermined. Up to some depth $l$, a complete enumeration tree is generated and a partial tree for the remaining search levels. The SB2 heuristic tries to determine the best single machine sequence for the SB1 heuristic within a reasonable amount of time. This can also be achieved by a genetic strategy, even in a more effective way. Scheduling of one machine alone does not in general provide a highly constrained solution space.

The length of a string representation of an individual in the population equals the number of machines in the problem which is equal to the depth of the enumeration tree in the SB2 heuristic. Hence, an individual is encoded over the alphabet from 1 to the number of machines and a partial string from the first to the $k$-th entry just describes the sequence in which the single machines are considered in the SB1 heuristic. As a crossover operator one can use any traveling salesman crossover; Dorndorf and Pesch (1995) chose the cycle crossover as described in Goldberg (1989). The best solutions Dorndorf and Pesch found for the $10 \times 10$ and $5 \times 20$ problem, were 960 (P-GA)/938 (SB-GA) and 1249 (P-GA)/1178 (SB-GA), respectively. The running times are about 15 (P-GA)/2 (SB-GA) and 25 (P-GA)/1.5 (SB-GA) minutes.

Della Groce et al. (1995) reached a value of 946 as well as 1178 for the two benchmarks, both in about 10 minutes.

Another genetic local search approach based on an arc solution representation is described in Aarts et al. (1994) or in Nakano and Yamada (1991). Their ideas are stimulated by the encouraging results obtained for the travelling salesman problem (Ulder et al., 1991). Aarts et al. (1994) devise a multi-start local search embedded into a genetic framework; hence the name genetic local search. Each solution in the population is replaced by a locally optimal one with respect to moves based on the neighbourhoods N1 and N2. The crossover idea is to implant a subset of arcs from one solution to another. The parent solutions are randomly chosen. The algorithm terminates when either all solutions in the population have equal cost, or the best makespan in the population has not changed for 10 generations. Within a time bound of 99 or 88 seconds for the $10 \times 10$ or $5 \times 20$

problem the results of the genetic local search algorithms are worse than those from simulated annealing. However the results are better than a multi-start local search on randomly generated initial solutions. The basic contribution of Nakano and Yamada (1991) is the individuals representation in the population. An individual representing a schedule is described by a 0, 1 matrix consisting of a column for each machine and a row for each pair of different jobs. Hence, the number of rows is limited to $|J|(|J|-1)$ ordered job pairs from set $J$ of jobs. A 1-entry in row $(i, j)$ and column $m$ indicates that job $i$ is supposed to be processed before job $j$ on machine $m$. Otherwise, the entry is 0. The simple crossover (a random individual cut and tail exchange) and the simple mutation operators (flip an 0, 1 entry) are applied. A harmonization algorithm turns a possibly inconsistent result through cycle elimination into a feasible schedule. Even for a population size of 1000 and 150 generations the $10 \times 10$ problem and the $5 \times 20$ problem could not be solved better than 965 and 1215, respectively.

A different approach has been followed by Storer et al. (1992). They map the original data of the underlying problem instance to slightly disturbed and genetically controlled data representing new problem instances. The latter are solved heuristically and the solutions, i.e. the operations' processing orders, are considered to be solutions of the original problem. Thus, the proposed neighbourhood definition is based on the fact that a heuristic is a mapping of a problem to a solution; hence a heuristic–problem pair is an encoding of a solution. A subset of solutions may be generated by the application of a single heuristic to perturbed versions of the original problem. That is, neighbouring solutions are generated by applying the base heuristic to the perturbed problem, which is obtained through adding uniformly distributed random numbers to the job shop data. Then the solution is evaluated using the original problem data. The simple crossover applies. Their results are 976 for the $10 \times 10$ and 1186 for the $5 \times 20$ problem.

Yamada and Nakano (1992) were the first to use the Giffler–Thompson algorithm as crossover operator. The random selection of a next operation is replaced by a choice of the operation with respect to one of the parent schedules. That is, in order to resolve a conflict (i.e. choice of a next operation from a set of operations competing for the same machine) randomly choose one parent schedule. Select that operation from the set of operations in conflict which is also the first one processed from the conflict set of the parent schedule. A huge population size of 2000 individuals led them find an optimal schedule for the $10 \times 10$ problem. Their result on the $5 \times 20$ problem was not better than 1184.

Bierwirth's (1995) representation is motivated by the idea to employ the traveling salesman crossover operators also in a job shop framework. He represented an individual as a string of length of the number of operations in the job shop. An entry in this string is a job identification. The number of operations of a job is the number of not necessarily consecutive string entries with the same job identification. For instance, if there are three jobs a, b, c having 3, 4, 3 operations, respectively, then a randomly generated string (b, a, b, b, c, a, c, c, b, a) says, that string entries 1, 3, 4, and 9 correspond to the 1st, 2nd, 3rd, and 4th operation of job b. Further, if the first operation of job c and the last operation of job b happen to need the same machine then c will come first. Now a TSP crossover (Kolen and Pesch, 1994) can be used to implant a substring of one parent schedule to another one. Within run times of about 9 to 10 minutes he reached a makespan of 936 and 1181 for the $10 \times 10$ and $5 \times 20$ problem. The population's size is 100.

### *3.4.3. Constraint propagation*

The job shop scheduling problem is a typical representative of a binary constraint satisfaction problem (CSP), i.e., generally speaking, there is a set of variables each of which has its own domain of values. Find an assignment of values to variables such that a set of constraints on variable pairs is satisfied (Dechter and Pearl, 1988; Meseguer, 1989; Minton et al., 1992). Assume that there is an upper bound on the makespan of an optimal schedule of the underlying job shop scheduling problem. Then computing heads and tails assigns to each operation an interval of possible start times. Considering variable domains as possible operation start times where the variables define the operations in a schedule then the disjunctive graph illustrates the job shop schedul-

ing constraint satisfaction problem, hence it corresponds to the constraint graph (Montanari, 1974).

A set of $k$ variables is said to be $k$-consistent if it is $k-1$-consistent and for each subset of $k-1$ variables holds: if a set of $k-1$ values each of which belongs to another of the $k-1$ variable domains violates none of the constraints on the considered $k-1$ variables, then there is a value in the domain of the remaining variable such that the set of all $k$ values satisfies the set of constraints on the $k$ variables. Let us assume that 0-consistency is always satisfied by definition. A set of variables is $k$-consistent if each subset of $k$ variables is $k$-consistent. A 2-consistent set of variables is also said to be arc-consistent in order to emphasize the relation with the edges in the constraint graph. (Van Hentenryck et al., 1992). Consider a pair $i$, $j$ of operations. If for any two start times $t_i$ and $t_j$ of operations $i$ and $j$, respectively, and any third operation $k$ there exists a start time $t_k$ of operation $k$ such that $t_i$, $t_j$, $t_k$ satisfy constraints (1) to (3) then operations $i$ and $j$ are said to be path consistent. Hence, consistency checks, or roughly speaking propagation of constraints will make implicitly defined constraints more visible and will prune the search tree in a branch and bound algorithm. The job shop scheduling problem is said to be path consistent if all operation pairs are path consistent (Mackworth, 1977; Mohr and Henderson, 1986; Han and Lee, 1988). Obviously, $n$-consistency, where $n$ is the number of operations, immediately implies that a feasible schedule can be generated easily, however, to achieve $n$-consistency is in general not practicable. Moreover, worse upper bounds on the makespan of an optimal schedule will hardly reduce variable domains, i.e. only a few arc directions are fixed during the constraint propagation process. The better the bounds, the more arc directions can be fixed, see also the comments in Section 2.3 on Caseau and Laburthe (1995) and Martin and Shmoys (1995).

Pesch (1993, 1994) introduced other genetic approaches. In the first one, the one-machine constraint propagation based genetic algorithm (1MCP-GA), each entry of an individual of the SB-GA is replaced by an upper bound on the makespan of the corresponding one-machine problem. In the second approach, the two job constraint propagation based genetic algorithm (2JCP-GA), each entry of an individual is an upper bound on the makespan of a subproblem consisting of a job pair. Whenever a new population is generated, a local decision rule in the sense of constraint propagation in order to achieve arc- and path-consistency is applied simultaneously to each subproblem (corresponding to an entry of an individual) with respect to its upper bound which is $\alpha\%$ above the optimal makespan of the subproblem. The number of newly fixed arc directions divided by the number of arcs which were included into a cycle during the constraint propagation process on the subproblems, defines the fitness of an individual. An individual of the population corresponds to a partial schedule. However each population is transformed to a population of feasible solutions, where each individual of a population is assessed in order to judge its contribution to a schedule. Therefore Giffler and Thompson's algorithm is applied with respect to the partial schedule representing individual. Ties are broken with respect to the complete schedules that are attached to the parents (partial schedules) of the considered offspring (partial schedule). Hence, the next operation is chosen as in one of the parents corresponding complete schedules with the same probability (cf. GT-crossover in Yamada and Nakano (1992)). In the first population of complete schedules ties were broken randomly. For both problem, the $10 \times 10$ and $5 \times 20$, the optimal solution was reached within about 2 minutes using the 1MCP-GA. It was impossible to reach an optimum using 2JCP-GA. Only values of 937 and 1175 could be found.

The main interest of propagation of constraints (constraint programming) is the enormous flexibility that results from the fact that each constraint propagates independently from the existence or non-existence of other constraints. It appears that, within each constraint, considered separately, any type of technique (in particular OR algorithms) can be used. It appears that the propagation process can be organized to guarantee that propagation steps will occur in an order consistent with Ford's flow algorithm (hence with the same time complexity, Caseau et al., 1996). Aggoun and Beldiceanu (1993) present a new construct called the "cumulative" constraint, incorporated in the CHIP constraint programming language. Using the cumulative constraint, Aggoun and Beldiceanu find the optimal solution of the $10 \times 10$ problem in about 30 minutes (but cannot prove its

optimality). Nuijten (1994) presents a variant of the algorithm by Carlier and Pinson (1990) to update time-bounds of activities. It seems that this variant can easily be incorporated in a constraint satisfaction framework, cf. Nuijten and Aarts (1996). Baptiste and Le Pape (1995) explore various techniques based on "edge-finding" and "energetic reasoning" with the aim of integrating such techniques in Ilog Schedule, an industrial software tool for constraint-based scheduling. In all of these cases, the flexibility inherent to consatraint programming is maintained, but more efficient techniques can be archived using the wealth of the OR algorithmic work.

*3.4.4. Ejection chains*

Variable-depth procedures, whose terminology was popularized by Papadimitriou and Steiglitz (1982), have had an important role in heuristic procedures for optimization problems. A class of these procedures, called ejection chain methods, has proved highly effective in a variety of applications (Glover, 1991, 1992; Dorndorf and Pesch, 1994b; Pesch, 1994; Glover and Pesch, 1995).

Ejection chain methods extend ideas exemplified by certain types of shortest path and alternating path constructions. The basic moves for transitioning from one solution to another are compound moves composed of a sequence of paired steps. The first component of each paired step in an ejection chain approach introduces a change that creates a dislocation (i.e., an inducement for further change), while the second component creates a change designed to restore the system. The dislocation of the first component may involve a form of infeasibility, or may be heuristically defined to create conditions that can be usefully exploited by the second component. Typically, the restoration of the second component may not be complete, and hence in general it is necessary to link the paired steps into a chain that ultimately achieves a desired outcome.

The ejection terminology comes from the typical graph theory setting where each of the paired steps begins by introducing an element (such as a node, edge or path) that disrupts the graph's preferred structure, and then is followed by ejecting a corresponding element, in a way that recovers a critical portion of the structure. A chain of such steps is controlled to assure the preferred structure eventually will be fully recovered (and preferably, fully recovered at various intermediate stages by means of trial solutions). The candidate element to be ejected in such instances may not be unique, but normally comes from a limited set of alternatives.

As our construction proceeds, we therefore note the trial solutions that would result by applying these feasibility-recovering transformations after each step, keeping track of the best. At the conclusion of the construction we simply select this best trial solution to replace the current solution, provided it yields an improvement. In this process, the moves at each

**begin**
  Start with an initial solution $x^*$ and the corresponding acyclic graph $G(x^*)$;
  $x := x^*$;
  **repeat**
    $T := \emptyset$; {$T$ is the tabu list}
    $d := 0$ {$d$ is the current search depth}
    **while** there are non-tabu critical arcs in $G(x(d))$ **do**
      $d := d + 1$
      Find the best move, i.e. the disjunctive critical arc $(i^*, j^*)$ for which
      $g(i^*, j^*) = \max\{g(i, j) | (i, j)$ is a disjunctive critical arc which is not in $T\}$;
        {note that $g(i^*, j^*)$ can be negative}
      Make this move, i.e. replace arc $(i^*, j^*)$, thus obtaining the solution $x(d)$ and its acyclic graph $G(x(d))$ at search depth $d$;
      $T := T \cup \{(j^*, i^*)\}$;
    Let $d^*$ denote the search depth at which the best solution $x(d^*)$ with
      $f(x(d^*)) = \min\{f(x(d)) | 0 < d \leq \frac{1}{2}m|J|(|J| - 1)\}$ has been found;
      **if** $d^* > 0$ **then begin** $x^* := x(d^*)$; $x := x^*$ **end**
  **until** $d^* = 0$
**end**

Fig. 6. An ejection chain procedure.

level cannot be obtained by a collection of independent and non-intersecting moves of previous levels. The list of forbidden (tabu) moves grows dynamically during a variable depth search iteration and is reset at the beginning of the next iteration. Details can be found in Glover and Pesch (1995).

An application with respect to neighbourhood structure N1 describes a move to a neighbouring solution in which the processing order of operations $i$ and $j$ is changed. This neighbourhood is connected, i.e. for any two solutions (including the optimal one) $x$ and $y$ there is a sequence of moves, with respect to N1, connecting $x$ to $y$. The gain $g(i, j)$ affected by such a move from $x$ to $y$ can be estimated based on considerations about the minimal length of the critical path of the resulting disjunctive graph $G(y)$. Finding the exact gain of a move would generally involve a longest path calculation. The gain of a move can be negative, thus leading to a deterioration of the objective function. Dorndorf and Pesch (1994a) present a local search procedure that is based on a compound neighbourhood structure, each component consists of the neighbourhood defined above. It is a variable depth search or ejection chain consisting of a simple neighbourhood structure at each depth which is composed to complex and powerful moves. The basic idea is similar to the one used in tabu search, the main difference being that the list of forbidden (tabu) moves grows dynamically during a variable depth search iteration and is reset at the beginning of the next iteration. The algorithm is outlined in Fig. 6; in our case $f(x)$ is the objective function value (makespan).

Starting with an initially best solution $x^*(0)$, the procedure looks ahead for a certain number of moves and then sets the new currently best solution $x^*(d)$ for the next iteration to the best solution found in the look-ahead phase at depth $d^*$. These steps are repeated as long as an improvement is possible. The maximal look-ahead depth is reached if all critical disjunctive arcs in the current solution are set tabu. The step leading from a solution $x$ in iteration $k$ to a new solution in the next iteration consists of a varying number $d^*$ of moves in the neighbourhood, hence the name variable depth search where a complex compound move results from a sequence of compressed simpler moves. The algorithm can escape local optima because moves with negative gain are possible. A continuously increasing growing tabu list avoids cycling of the search procedure. As an extension of the algorithm, the whole 'repeat... until' part could easily be embedded in yet another control loop (not shown here) leading to a multi-level (parallel) search algorithm (Glover, 1992).

A genetic algorithm with variable depth search has been implemented by Dorndorf and Pesch (1994), i.e. each individual of a population is made locally optimal with respect to the ejection chain based embedded neighbourhood described in Fig. 6. The algorithm was run five times on each problem instance, and all instances have been solved to optimality within a CPU time of ten minutes for a single run. While the $6 \times 6$ problem and the $5 \times 20$ problem are relatively easy, it is quite remarkable that the algorithm has always solved the notoriously difficult $10 \times 10$ instance.

## 4. Conclusions

Although the $10 \times 10$ problem is not any longer a challenge, it provides a way to briefly get an impression of how powerful a certain method can be. For detailed comparisons of solution procedure–if this is possible at all under different machine environments–this is obviously not enough and there are many other benchmark problems some of them with unkown optimal solution, see Taillard (1993). It is apparent from the discussion that local search methods are the most powerful tool to schedule job shops. However, a stand alone local search cannot be competitive to those methods incorporating problem specific knowledge either by problem decomposition, special purpose heuristics, constraints and propagation of variables' domain modification, neighbourhood structures (e.g. neighbourhoods where each neighbour of a feasible schedule is locally optimal, cf. Brucker et al. (1994a,b)), etc. or any composition of these tools.

The analogy of branching structures in exact methods and neighbourhood structures reveals parallelism that is largely unexpored.

## References

Aarts, E.H.L., and Korst, J.H.M. (1989), *Simulated Annealing and Boltzmann Machines*, Wiley, Chichester.

Aarts, E.H.L., van Laarhoven, P.J.M., Lenstra, J.K., and Ulder, N.L.J. (1994), "A computational study of local search shop scheduling," *ORSA Journal on Computing* 6, 118–125.

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

Aggoun, A., and Beldiceanu, N. (1993), "Extending CHIP in order to solve complex scheduling and placement problems", *Mathematical and Computer Modelling* 17, 57–73.

Akers, S.B. (1956), "A graphical approach to production scheduling problems", *Operations Research* 4, 244–245.

Anderson, E.J., Glass, C.A., and Potts, C.N. (1995), "Local search in combinatorial optimization: Applications in machine scheduling", Research Report No. OR56, University of Southampton.

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

Ashour, S. (1972), *Sequencing Theory*, Springer, Berlin.

Ashour, S., and Hiremath, S.R. (1973), "A branch-and-bound approach to the job-shop scheduling problem", *International Journal of Production Research* 11, 47–58.

Ashour, S., Moore, T.E., and Chiu, K.-Y. (1974), "An implicit enumeration algorithm for the nonpreemptive shop scheduling problem", *AIIE Transactions* 6, 62–72.

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

Baker, K.R. (1975), "A comparative study of flow shop algorithms", *Operations Research* 23, 62–73.

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

Bakshi, M.S., and Arora, S.R. (1969), "The sequencing problem", *Management Science* 16, B247–B263.

Balas, E. (1969), "Machine sequencing via disjunctive graphs: An implicit enumeration algorithm", *Operations Research* 17, 941–957.

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

Balas, E., Lenstra, J.K., and Vazacopoulos, A. (1995), "One machine scheduling with delayed precedence constraints," *Management Science* 41, 94–109.

Balas, E., and Vazacopoulos, A. (1995), "Guided local search with shifting bottleneck fo job shop scheduling", Management Science Research Report #MSRR-609.

Baptiste, P., and Le Pape, C. (1995), "A theoretical and experimental comparison of constraint propagation techniques for disjunctive scheduling", *Proc. 14th Int. Joint Conference on Artificial Intelligence (IJCAI)*, Montreal, Canada.

Baptise, P., Le Pape, C., and Nuijten, W. (1995a), "Constrained-based optimization and approximation for job-shop scheduling", *Proc. AAAI-SIGMAN Workshop on Intelligent Manufacturing Systems, 14th Int. Joint Conference on Artificial Intelligence (IJCAI)*, Montreal, Canada.

Baptiste, P., Le Pape, C., and Nuijten, W. (1995b), "Incorporating efficient operations research algorithms in constraint-based scheduling", *Proc. 1st Joint Workshop on Artificial Intelligence and Operations Research*, Timberline Lodge, OR (to appear).

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

Barnes, J.W., and Chambers, J.B. (1994), "Solving the job shop scheduling problem using tabu search," *IIE Transactions* 27/2.

Barnes, J.W., Laguna, M., and Glover, F. (1992), "An overview of tabu search approaches to production scheduling problems", *Proc. Symposium Intelligent Scheduling Systems*, San Francisco, 30–50.

Bertier, P., and Roy, B. (1965), "Trois exemples numeriques d'application de la procedure SEP", Note de travail No. 32 de la Direction Scientifique de la SEMA.

Bierwirth, C. (1995), "A generalized permutation approach to job shop scheduling with genetic algorithms," *OR Spektrum* 17, 87–92.

Blackstone, J.H., Phillips, D.T., and Hogg, G.L. (1982), "A state of the art survey of dispatching rules for manufacturing job shop operations", *International Journal Production Research* 20, 27–45.

Błazewicz, J. (1987), "Selected topics in scheduling theory", *Annals of Discrete Mathematics* 31, 1–60.

Błazewicz, J., Dror, M., and Weglarz, J. (1991), "Mathematical programming formulations for machine scheduling: A survey," *European Journal of Operational Research* 51, 283–300.

Błazewicz, J., Ecker, K., Schmidt, G., and Weglarz, J. (1994), *Scheduling in Computer and Manufacturing Systems*, 2nd edition, Springer, Heidelberg.

Bowman, E.H. (1959), "The scheduling sequencing problem", *Operations Research* 7, 621.

Brah, S., Hunsucker, J., and Shah, J. (1991), "Mathematical modeling of scheduling problems", *Journal of Information & Optimization Sciences* 12, 113–137.

Brăsel, H. (1990), "Lateinische Rechtecke und Maschinenbelegung", Dissertation B, Technical University of Magdeburg.

Brăsel, H., and Werner, F. (1989), "The job-shop problem–modelling by latin rectangles exact and heuristic solution," Working paper Math 8/89, Technical University of Magdeburg.

Brooks, G.H., and White, C.R. (1965), "An algorithm for finding optimal or near-optimal solutions to the production scheduling problem", *Journal of Industrial Engineering* 16, 34–40.

Brucker, P. (1981), *Scheduling*, Akademische Verlagsgesellschaft, Wiesbaden.

Brucker, P. (1988), "An efficient algorithm for the job-shop problem with two jobs", *Computing* 40, 353–359.

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

Brucker, P. (1995), *Scheduling Algorithms*, Springer, Berlin.

Brucker, P., Hurink, J., and Werner, F. (1994a), "Improving local search heuristics for some scheduling problems", Working paper, University of Osnabrück.

Brucker, P., Hurink, J., and Werner, F. (1994b), "Improving local search heuristics for some scheduling problems, Part II", Working paper, University of Osnabrück.

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

Brucker, P., Jurisch, B., and Krämer, A. (1992), "The job-shop problem and immediate selection", Working paper, University of Osnabrück.

Brucker, P., Jurisch, B., and Sievers, B. (1992), "Job-shop (C codes)", *European Journal of Operational Research* 57, 132–133.

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

Burns, F., and Rooker, J. (1978), "Three-stage flow-shops with recessive second stage", *Operations Research* 26, 207–208.

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

Carlier, J. (1982), "The one machine sequencing problem," *European Journal of Operational Research* 11, 42–47.

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

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

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

Carlier, J., and Pinson, E. (1994), "Adjustments of heads and tails for the job-shop problem", *European Journal of Operational Research* 78, 146–161.

Caseau, Y., and Laburthe, F. (1995), "Disjunctive scheduling with task intervals", Working paper, Ecole Normale Supérieure, Paris

Caseau, Y., Le Pape, C., and Nuijten, W. (1996), Private communication.

Charlton, J.M., and Death, C.C. (1970), "A generalized machine scheduling algorithm", *Operations Research Quarterly* 21, 127–134.

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

Chrétienne, P., Coffman, Jr., E.G., Lenstra, J.K., and Liu, Z. (1995), *Scheduling Theory and its Applications*, Wiley, Chichester.

Chu, C., Portmann, M.C., and Proth, J.M. (1992), "A splitting-up approach to simplify job-shop scheduling problems", *International Journal of Production Research* 30, 859–870.

Clark, W. (1922), *The Gantt Chart: A Working Tool of Management*, The Ronald Press, 3rd ed., Pittman, New York.

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

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

Crama, Y., Kolen, A., and Pesch, E. (1995), "Local search in combinatorial optimization", *Lecture Notes in Computer Science* 931, 157–174.

Crowston, W.B., Glover, F., Thompson, G.L., and Trawick, J.D. (1963), "Probabilistic and parametric learning combinations of local job shop scheduling rules", ONR Research Memorandum No. 117, GSIA, Carnegie-Mellon University, Pittsburg, PA.

Dannenbring, D.G. (1977), "An evaluation of flow shop scheduling heuristics", *Management Science* 23, 1174–1182.

Dauzere-Peres, S., and Lasserre, J.-B. (1993), "A modified shifting bottleneck procedure for job-shop scheduling", *International Journal of Production Research* 31, 923–932.

Davis, L. (1985), "Job shop scheduling with genetic algorithms", *Proc. International Conf. on Genetic Algorithms and Their Applications* (J.J. Grefenstette, ed.), Lawrence Erlbaum Ass., 136–140.

Day, J.E., and Hottenstein, M.P. (1970), "Review of sequencing research", *Naval Research Logistics Quarterly* 17, 11–39.

Dechter, R., and Pearl, J. (1988), "Network-based heuristics for constraint satisfaction problems," *Artificial Intelligence* 34, 1–38.

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

Dell'Amico, M. (1993), "Shop problems with two machines and time lags," Working paper, Politecnico di Milano.

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

Domschke, W., Scholl, A., and Voß, S.(1993), *Produktionsplanung*, Springer, Berlin.

Dorndorf, U., and Pesch, E. (1993a), "Combining genetic and local search for solving the job shop scheduling problem," *Symposium on Applied Mathematical Programming and Modeling APMOD93* (I. Maros, ed.), Akaprint, Budapest, 142–149.

Dorndorf, U., and Pesch, E. (1993b), "Genetic algorithms for job shop scheduling", in: K.-W. Hansmann et al., (eds.), *Operations Research Proc. 1992*, Springer, Berlin, 243–250.

Dorndorf, U., and Pesch, E. (1994a), "Variable depth search and embedded schedule neighbourhoods for job shop scheduling", *4th International Workshop on Project Management and Scheduling*, 232–235.

Dorndorf, U., and Pesch, E. (1994b), "Fast clustering algorithms", *ORSA Journal on Computing* 6, 141–153.

Dorndorf, U., and Pesch, E. (1995), "Evolution based learning in a job shop scheduling environment", *Computers & Operations Research* 22, 25–40.

Dudek, R.A., Panwalkar, S.S., and Smith, M.L. (1992), "The lessons of flowshop scheduling research", *Operations Research* 40, 7–13.

Ecker, K. (1977), *Organisation von parallelen Prozessen*, BI-Wissenschaftsverlag, Mannheim.

Elmaghraby, S.E. (1968), "The machine sequencing problem–Review and extensions", *Naval Research Logistics Quarterly* 15, 205–232.

Elmaghraby, S.E., and Elshafei, A.N. (1976), "Branch-and-bound revised: A survey of basic concepts and their applications in scheduling", in: W.H. Marlow (ed.), *Modern Trends in Logistics Research*, MIT Press, Cambridge, MA.

Fisher, M.L. (1973), "Optimal solution of scheduling problems

using Lagrange multipliers: Part I", *Operations Research* 21, 1114–1127.

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

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

Florian, M., Trépant, P., and McMahon, G. (1971), "An implicit enumeration algorithm for the machine sequencing problem", *Management Science* 17, B782–B792.

Fox, M.S. (1987), *Constraint-directed search: A case study of job shop scheduling*, Pitman, London.

Fox, M.S., and Smith, S.F. (1984), "ISIS–A knowledge based system for factory scheduling," *Expert Systems* 1, 25–49.

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

Gantt, H.L. (1919), "Efficiency and democracy", *Transactions of the American Society of Mechanical Engineers*. 40, 799–808.

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

Gere, W.S. (1966), "Heuristics in job-shop scheduling", *Management Science* 13, 167–190.

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

Giffler, B., Thompson, G.L., and van Ness, V. (1963), "Numerical experience with the linear and Monte Carlo algorithms for solving production scheduling problems", in: J.F. Muth and G.L. Thompson (eds.), *Industrial Scheduling*, Prentice-Hall, Englewood Cliffs, NJ.

Glass, C.A., Potts, C.N., and Shade, P. (1992), "Genetic algorithms and neighbourhood search for scheduling unrelated parallel machines", Working paper No. OR47, University of Southampton.

Glover, F. (1989), "Tabu search – Part I", *ORSA Journal on Computing* 1, 190–206.

Glover, F. (1990a), "Tabu search – Part II", *ORSA Journal on Computing* 2, 4–32.

Glover, F. (1990b), "Tabu search: A tutorial", *Interfaces* 20, 74–94.

Glover, F. (1991), "Multilevel tabu search and embedded search neighborhoods for the traveling salesman problem", Working paper, University of Colorado, Boulder.

Glover, F. (1992), "Ejection chains, reference structures and alternating path methods for the traveling salesman problem," Working paper, University of Colorado, Boulder.

Glover, F., and Pesch, E. (1995), "TSP ejection chains", *Discrete Applied Mathematics*, to appear.

Goldberg, D.E. (1989), *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, MA.

Gonzalez, T., and Sahni, S. (1978), "Flowshop and jobshop schedules: Complexity and approximation", *Operations Research* 20, 36–52.

Grabowski, J., Nowicki, E., and Zdrzalka, S.S. (1986), "A block approach for single machine scheduling with release dates and due dates", *European Journal of Operational Research* 26, 278–285.

Graham, R.L., Lawler, E.L., Lenstra, J.K., and Rinnooy Kan, A.H.G. (1978), "Optimization and approximation in deterministic sequencing and scheduling: A survey", in: J.K. Lenstra, A.H.G. Rinnooy Kan, and P. van Emde Boas (eds.), *Interfaces between Computer Science and Operations Research*, Mathematical Centre Tracts 99, Amsterdam, 169–231.

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

Green, G.I., and Appel, L.B. (1981), "An empirical analysis of job shop dispatch rule selection," *Journal of Operations Management* 1, 197–203.

Greenberg, H.H. (1968), "A branch and bound solution to the general scheduling problem", *Operations Research* 16, 353–361.

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

Han, C.C., and Lee, C.H. (1988), "Comments on Mohr and Henderson path consistency algorithm," *Artificial Intelligence* 36, 125–130.

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

Hax, A.C., and Candea, D. (1984), *Production and Inventory Management*, Prentice-Hall, Englewood Cliffs, NJ.

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

van Hentenryck, P., Deville, Y., and Teng, C.-M. (1992), "A generic arc-consistency algorithm and its specializations", *Artificial Intelligence* 57, 291–321.

Hilliard, M.R., and Liepins, G.E. (1988), "Machine learning applications to job shop scheduling", *Proceedings Workshop on Production Planning and Scheduling*, AAAI-SIGMAN, St. Paul, MN.

Ho, J.C., and Chang, Y-L. (1991), "A new heuristic for the $n$-job, $M$-machine flow-shop problem", *European Journal of Operational Research* 52, 194–202.

Hoitomt, D.J., Luh, P.B., and Pattipati, K.R. (1993), "A practical approach to job-shop scheduling problems", *IEEE Transactions on Robotics and Automation* 9, 1–13.

Holland, J.H. (1975), *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, MI.

Hübscher, R., and Glover, F. (1992), "Applying tabu search with influential diversification to multiprocessor scheduling", Working paper, University of Colorado, Boulder.

Hundal, T.S., and Rajgopal, J. (1988), "An extension of Palmer's heuristic for the flow-shop scheduling problem", *International Journal of Production Research* 26, 1119–1124.

Husbands, P., Mill, F., and Warrington, S. (1991), "Genetic algorithms, production plan optimisation and scheduling", *Lecture Notes in Computer Science* 496, 80–84.

Ignall, E., and Schrage, L. (1965), "Application of the branch-and

bound technique to some flow-shop scheduling problem", *Operations Research* 13, 400–412.

Jackson, J.R. (1956), "An extension of Johnson's results on job lot scheduling," *Naval Research Logistics Quarterly* 3, 201–203.

Johnson, D.S. (1983), "The NP-completeness column: An ongoing guide", *Journal of Algorithms* 4, 189–203.

Johnson, D.S., Papadimitriou, C.H., and Yannakakis, M. (1988), "How easy is local search?", *Journal of Computer System Science* 37, 79–100

Johnson, L.A., and Montgomery, D.C. (1974) *Operations Research in Production Planning, Scheduling and Inventory Control*, Wiley, New York.

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

Kawaguchi, T., and Kyan, S. (1988), "Deterministic scheduling in computer systems: A survey," *Journal of the Operational Research Society Japan* 31, 190–217.

Kolen, A., and Pesch, E. (1994), "Genetic local search in combinatorial optimization," *Discrete Applied Mathematics* 48, 273–284.

Kubiak, W., Sethi, S., and Srishkandarajah, C. (1994), "An efficient algorithm for a job shop problem", *Mathematics of Industrial Systems*, to appear.

Kusiak, A., and Chen, M. (1988), "Expert systems for planning and scheduling manufacturing systems", *European Journal of Operational Research* 34, 113–130.

van Laarhoven, P.J.M., and Aarts, E.H.L. (1987), *Simulated Annealing: Theory and Applications*, Reidel, Dordrecht, Netherlands.

van Laarhoven, P.J.M., Aarts, E.H.L., and Lenstra, J.K. (1992), "Job shop scheduling by simulated annealing", *Operations Research* 40, 113–125.

Lageweg, B., Lawler, E.L., Lenstra, J.K., and Rinnooy Kan, A.H.G. (1982), "Computer aided complexity classification of combinatorial problems", *Communications of the ACM* 25, 817–822.

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

Lageweg, B., Lenstra, J.K., and Rinnooy Kan, A.H.G. (1978), "A general bounding scheme for the permutation flow-shop problem", *Operations Research* 26, 53–67.

Larson, R.E., Dessouky, M.I., and Devor, R.E. (1985), "A forward-backward procedure for the single machine problem to minimize maximum lateness", *IIE Transactions* 17, 252–260.

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

Lawler, E.L., Lenstra, J.K., and Rinnooy Kan, A.H.G. (1982), "Recent developments in deterministic sequencing and scheduling: A survey", in: M.A.H. Dempster, J.K. Lenstra and A.H.G. Rinnooy Kan (eds.), *Deterministic and Stochastic Scheduling*, Proc. of the NATO Advanced Study and Research Institute on Theoretical Approaches to Scheduling Problems, Reidel, Dordrecht, 35–73.

Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., and Shmoys, D.B. (1993), "Sequencing and scheduling: algorithms and complexity", in: S.C. Graves, A.H.G. Rinnooy Kan and P.H. Zipkin (eds.), *Handbooks in Operations Research and Management Science, Vol. 4: Logistics of Production and Inventory* Elsevier, Amsterdam.

Lenstra, J.K. (1977), *Sequencing by Enumerative Methods*, Mathematical Center Tract 69, Mathematisch Centrum, Amsterdam.

Lenstra, J.K., and Rinnooy Kan, A.H.G. (1979), "Computational complexity of discrete optimization problems", *Annals of Discrete Mathematics* 4, 121–140.

Lenstra, J.K., Rinnooy Kan, R.H.G., and Brucker, P. (1977), "Complexity of machine scheduling problems", *Annals of Discrete Mathematics* 4, 121–140.

Lourenço, H.R. (1993), "A computational study of the job-shop and flow shop scheduling problem", Dissertation, Cornell University, Ithaca.

Lourenço, H.R. (1995), "Job-shop scheduling: Computational study of local search and large-step optimization methods," *European Journal of Operational Research* 83, 347–364.

Mackworth, A.K. (1977), "Consistency in networks of relations," *Artificial Intelligence* 8, 99–118.

Manne, A.S. (1960), "On the job shop scheduling problem", *Operations Research* 8, 219–223.

Martin, P., and Shmoys, D. (1995), "A new approach to computing optimal schedules for the job shop scheduling problem", Extended Abstract, Cornell University, Ithaca.

Mattfeld, D.C. (1995), "Evolutionary search and the job shop", Dissertation, University of Bremen.

Matsuo, H., Suh, C.J., and Sullivan, R.S. (1988), "A controlled search simulated annealing method for the general jobshop scheduling problem", Working paper 03-04-88, University of Texas at Austin.

McMahon, G.B. (1969), "Optimal production schedules for flow shops", *Canadian Operations Research Society Journal* 7, 141–151.

McMahon, G.B., and Florian, M. (1975), "On scheduling with ready times and due dates to minimize maximum lateness", *Operations Research* 23, 475–482.

Mellor, P. (1966), "A review of job shop scheduling", *Operations Research Quarterly* 17, 161–171.

Meseguer, P. (1989), "Constraint satisfaction problems: An overview", *AICOM 2*, 3–17.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E. (1953), "Equation of state calculations by fast computing machines", *Journal Chemical Physics* 21, 1087–1092.

Meyer, W. (1992), "Geometrische Methoden zur Lösung von Job-Shop Problemen und deren Verallgemeinerungen", Dissertation, University of Osnabrück.

Michalewicz, Z. (1992), "*Genetic Algorithms + Data Structures = Evolution Programs*, Springer, Berlin.

Minton, S., Johnston, M.D., Philips, A.B., and Laird, P. (1992), "Minimizing conflicts: A heuristic repair method for con-

straint satisfaction and scheduling problems", *Artificial Intelligence* 58, 161–205.

Mohr, R., and Henderson, T.C. (1986), "Arc and path consistency revisited", *Artificial Intelligence* 28, 225–233.

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

Montanari, U. (1974), "Networks of constraints: Fundamental properties and applications to picture processing", *Informations Sciences* 7, 95–132.

Moore, J.M., and Wilson, R.C. (1967), "A review of simulation research in job shop scheduling", *Production and Inventory Management* 8, 1–10.

Muth, J.F., and Thompson, G.L. (eds.) (1963), *Industrial Scheduling*, Prentice-Hall, Englewood Cliffs, NJ.

Nakano, R., and Yamada, T. (1991), "Conventional genetic algorithm for job shop problems", in: R.K. Belew and L.B. Booker (eds.), *Proc. 4th. International Conf. on Genetic Algorithms*, Morgan Kaufmann, 474–479.

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

Nowicki, E., and Smutnicki, C. (1993), "A fast taboo search algorithm for the job shop problem", *Management Science*, to appear.

Nowicki, E., and Smutnicki, C. (1994), "A fast tabu search algorithm for the flow shop problem", Working paper, Technical University Wrocław.

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

Nuijten, W.P.M. (1994), *Time and Resource Constrained Scheduling*, Dissertation, Ponsen & Looijen, Wageningen, The Netherlands.

Nuijten, W.P.M., and Aarts, E.H.L. (1996), "A computational study of constraint satisfaction for multiple capacitated job shop scheduling", *European Journal of Operational Research* 90, 269–284.

O'Grady, P.J., and Harrison, C. (1985), "A general search sequencing rule for job shop sequencing", *International Journal of Production Research* 23, 951–973.

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

Ow, P.S., and Smith, S.F. (1988), "Viewing scheduling as an opportunistic problem-solving process", *Annals of Operations Research* 12, 85–108.

Palmer, D.S. (1965), "Sequencing jobs through a multi-stage process in the minimum total time–A quick method of obtaining a near optimum", *Operational Research Quarterly* 16, 101–107.

Panwalkar, S.S., and Iskander, W. (1977), "A survey of scheduling rules", *Operations Research* 25, 45–61.

Papadimitriou, C.H., and Steiglitz, K. (1982), *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, NJ.

Perregaard, M., and Clausen, J. (1995), "Parallel branch-and-bound methods for the job-shop scheduling problem", Working paper, University of Copenhagen.

Pesch, E. (1993), "Machine learning by schedule decomposition", Working paper, University of Limburg, Maastricht.

Pesch, E. (1994), *Learning in Automated Manufacturing*, Physica, Heidelberg.

Pesch, E., and Tetzlaff, U. (1995), "Constraint propagation based scheduling of job shops", *INFORMS Journal on Computing*, to appear.

Pinedo, M. (1995), *Scheduling Theory, Algorithms and Systems*, Prentice Hall, Englewood Cliffs, NJ.

Pinson, E. (1988), "Le problème de job-shop", Dissertation, University Paris VI, Paris.

Pinson, E. (1992), "The job shop scheduling problem: A concise survey and some recent developments", Working paper, University Catholique de l'Ouest, Angers, France.

Pirlot, M. (1992), "General local search heuristics in combinatorial optimization: A tutorial", *JORBEL* 32, 7–68.

Porter, D.B. (1968), "The Gantt chart as applied to production scheduling and control", *Naval Research Logistics Quarterly* 15, 311–317.

Potts, C.N. (1980a), "An adaptive branching rule for the permutation flow-shop problem", *European Journal of Operational Research* 5, 19–25.

Potts, C.N. (1980b), "Analysis of a heuristic for one machine sequencing with release dates and delivery times", *Operations Research* 28, 1436–1441.

Potts, C.N., Shmoys, D.B., and Williamson, D.P. (1991), "Permutation vs. non-permutation flow shop schedules", *Operations Research Letters* 10, 281–284.

Proust, C. (1992), "De l'influence des idées de S.M. Johnson dans la résolution des problèmes dÓdonnancement de type Flowshop", Working paper, University F. Rabelais, Tours, France.

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

Röck, H. (1984), "The three-machine no-wait flow shop problem is NP-complete", *Journal of the ACM* 51, 336–345.

Rodammer, F.A., and White, K.P. (1988), "A recent review of production scheduling", *IEEE Transactions on Systems, Man, and Cybernetics* 18, 841–852.

Roy, B., and Sussmann, B. (1964), "Les problèmes d'ordonnancement avec contraintes disjonctives", SEMA, Note D.S. No. 9., Paris.

Sadeh, N. (1991), "Look-ahead techniques for micro-opportunistic job shop scheduling", Dissertation, Carnegie Mellon University, Pittsburgh, PA.

Salvador, M.S. (1978), "Scheduling and sequencing", in: J.J. Moder and S.E. Elmaghraby, (eds.), *Handbook of Operations Research: Models and Applications*, Van Nostrand Reinhold, New York.

Sauer, J. (1995), "Scheduling and meta-scheduling", in: C. Beierle and L. Plümer (eds.), *Logic Programming: Formal Methods and Practical Applications*, Elsevier, Amsterdam.

Smith, S.F., Fox, M.S., and Ow, P.S. (1986), "Constructing and maintaining detailed production plans: Investigations into the

development of knowledge-based factory scheduling systems", *AI Magazine*, 46–61.

Sotskov, Y.N. (1991), "The complexity of scheduling problems with two and three jobs", *European Journal of Operational Research* 53, 326–336.

Sotskov, Y.N., and Shaklevich, N.V. (1995), "NP-hardness of shop scheduling problems with three jobs", *Discrete Applied Mathematics* 59, 237–266.

Stafford, E.F. (1988), "On the development of a mixed-integer linear programming model for the flowshop sequencing problem", *Journal of the Operational Research Society* 39, 1163–1174.

Starkweather, T., Whitley, D., Mathias, K., and McDaniel, S. (1992), "Sequence scheduling with genetic algorithms", in: G. Fandel, T. Gulledge, and J. Jones (eds.), *New Directions for Operations Research in Manufacturing*, Springer, Berlin, 129–148.

Storer, R.H., Wu, S.D., and Park, I. (1992), "Genetic algorithms in problem space for sequencing problems", in: G. Fandel, T. Gulledge, and A. Jones (eds.) *New Directions for Operations Research in Manufacturing*, Proc. of the 2nd Joint US/German Conf., Hagen.

Storer, R.H., Wu, S.D. and Vaccari, R. (1991), "Local search in problem and heuristic space for job shop scheduling genetic algorithms", in: G. Fandel, T. Gulledge, and A. Jones (eds.), *New Directions for Operations Research in Manufacturing*, Proc. of a Joint US/German Conf., Gaithersburg, Maryland.

Storer, R.H., Wu, S.D., and Vaccari, R. (1992a), "New search spaces for sequencing problems with application to job shop scheduling", *Management Science* 38, 1495–1509.

Storer, R.H., Wu, S.D., and Vaccari, R. (1992b), "Problem and heuristic space search strategies for job shop scheduling", Working paper, Lehigh University, Bethlehem, PA.

Sun, D., Batta, R., and Lin, L. (1995), "Effective job shop scheduling through active chain manipulation", *Computers & Operations Research* 22, 159–172.

Szwarc, W. (1971), "Elemination methods in the $m \times n$ sequencing problem", *Naval Research Logistics Quarterly* 18, 295–305.

Szwarc, W. (1973), "Optimal elemination methods in the $m \times n$ sequencing problem", *Operations Research* 21, 1250–1259.

Szwarc, W. (1978), "Dominance conditions for the three-machine flow shop problem", *Operations Research* 26, 203–206.

Taillard, E. (1994), "Parallel taboo search technique for the job shop scheduling problem", *ORSA Journal on Computing* 6, 108–117.

Taillard, E. (1990), "Some efficient heuristic methods for the flow shop sequencing problem", *European Journal of Operational Research* 47, 65–74.

Taillard, E. (1993), "Benchmarks for basic scheduling problems", *European Journal of Operational Research* 64, 278–285.

Tanaev, V.S., Gordon, V.S., and Shafransky, Y.M. (1994), *Scheduling Theory: Single-Stage Systems*, Kluwer Academic Publ., Dordrecht.

Tanaev, V.S., Sotskov, Y.N., and Strusevich, V.A. (1994), *Scheduling Theory: Multi-Stage Systems*, Kluwer Academic Publ., Dordrecht.

Thompson, G.L., and Zawack, D.J. (1985/6), "A problem expanding parametric programming method for solving the job shop scheduling problem", *Annals of Operations Research* 4, 327–342.

Turner, S., and Booth, D. (1987), "Comparison of heuristics for flow shop sequencing", *Omega* 15, 75–78.

Ulder, N.L.J., Aarts, E.H.L., Bandelt, H.-J., van Laarhoven, P.J.M., and Pesch, E. (1991), "Genetic local search algorithms for the traveling salesman problem", *Lecture Notes in Computer Science* 496, 109–116.

Vaessens, R.J.M. (1995), "Generalized job shop scheduling: Complexity and local search", Dissertation, University of Technology Eindhoven.

Vaessens, R.J.M., Aarts, E.H.L., and Lenstra, J.K. (1995), "Job shop scheduling by local search", Working paper, University of Technology, Eindhoven.

van de Velde, S. (1991), "Machine scheduling and Lagrangian relaxation", Dissertation, CWI Amsterdam.

Wagner, H.M. (1959), "An integer linear programming model for machine scheduling", *Naval Research Logistics Quarterly* 6, 131.

White, K.P., and Rogers, R.V. (1990), "Job-shop scheduling: Limits of the binary disjunctive formulation", *International Journal of Production Research* 28, 2187–2200.

Widmer, M., and Hertz, A. (1989), "A new heuristic method for the flow shop sequencing problem", *European Journal of Operational Research* 41, 186–193.

Yamada, T., and Nakano, R. (1992), "A genetic algorithm applicable to large-scale job-shop problems", in: R. Männer and B. Manderick (eds.), *Parallel Problem Solving from Nature 2*, Elsevier, Amsterdam, 281–290.

Yannakakis, M. (1990), "The analysis of local search problems and their heuristics", *Lecture Notes in Computer Science* 415, 298–311.