RAIRO. RECHERCHE OPÉRATIONNELLE

YURI N. SOTSKOV
THOMAS TAUTENHAHN
FRANK WERNER

On the application of insertion techniques for job shop problems with setup times

RAIRO. Recherche opérationnelle, tome 33, n° 2 (1999), p. 209-245

<http://www.numdam.org/item?id=RO_1999__33_2_209_0>

RAIRO Rech. Opér.
(vol. 33, n° 2, 1999, p. 209-245)

ON THE APPLICATION OF INSERTION TECHNIQUES FOR JOB SHOP PROBLEMS WITH SETUP TIMES (*)

by Yuri N. SOTSKOV ([1]), Thomas TAUTENHAHN ([2]) and Frank WERNER ([3])

Communicated by Jacques CARLIER

Abstract. – *Constructive heuristics for shop scheduling problems are often based on priority (or dispatching) rules. However, recent work has demonstrated that insertion algorithms that step by step insert operations or jobs into partial schedules usually clearly outperform priority rules. In this paper, we consider various job shop scheduling problems with setup times. For each job a specific technological route and a release date are given. Moreover, the jobs are partitioned into groups. A sequence independent setup time s_{rj} is required on machine j when a job of the r-th group is processed after a job of another group. We consider different types of job availability, namely item and batch availability. As objective function we use both regular and nonregular criteria. For such problems we apply insertion techniques combined with beam search. Especially we consider different insertion orders of the operations or jobs. A refined variant of the insertion algorithm is presented, where several operations are inserted in parallel. The proposed variants have been tested on a large collection of test problems and compared with other constructive algorithms based on priority rules.*

Keywords: Job shop scheduling, setup times, constructive heuristics.

Résumé. – *Des heuristiques constructives sont basées souvent sur des règles de priorité. Néanmoins, les publications les plus récentes ont montré que des algorithmes d'insertion qui insèrent pas à pas des opérations dans les plans partiels déjà existants sont plus efficaces.*

Dans notre publication nous considérons différents problèmes de « job shop scheduling » avec des temps de préparation. Pour chaque ordre sont données une route technologique et la date de mise à disposition. Les ordres sont partagés en groupes. Un temps de préparation s_{rj} indépendant de la suite est nécessaire pour la machine j au cas où un ordre pour le r-ième groupe est exécuté après un ordre d'un autre groupe. Nous considérons des types différents de la mise à disposition d'un ordre, plus précisément Item- et Batch-disponibilité.

Comme fonction de but seront utilisés aussi bien des critères réguliers que non-réguliers. Pour des problèmes de ce genre nous utilisons des techniques d'insertion combinées avec la recherche « Beam ». En particulier nous considérons des ordres différents d'insertion des demandes (resp. des opérations).

Une variante avec des insertions parallèles pour plusieurs opérations est considérée également. Les algorithmes déduits sont testés pour un grand nombre de problèmes et comparés avec d'autres algorithmes de priorité.

Mots clés : « *Job Shop Scheduling* », *temps de préparation, heuristiques constructives.*

(*) Received June 1996.

Supported by INTAS (Project 93-257) and by Deutsche Forschungsgemeinschaft (Project ScheMA).

([1]) Institute of Engineering Cybernetics of the Byelorussian Academy of Sciences, Surganov St. 6, 220012 Minsk, Belarus

([2]) University of Southampton, Highfield, Southampton, SO17 1BJ, United Kingdom

([3]) Otto-von-Guericke-Universität Magdeburg, PSF 4120, 39016 Magdeburg, Germany

Recherche opérationnelle/Operations Research, 0399-0559/99/02/$ 7.00

1. INTRODUCTION

The job shop problem belongs to the scheduling problems that are the most difficult ones in classical scheduling theory: n jobs have to be processed on m machines and each job i consists of a set of operations, which have to be performed in a given order. Each operation represents a processing of a job on a certain machine and it may not be interrupted. Often the makespan $C_{\max} = \max_{1 \leq i \leq n} \{C_i\}$ has to be minimized, where C_i is the completion time of job i. Using the 3-parameter classification suggested by Graham *et al.* [15], this problem is denoted as $J//C_{\max}$. Note that a well-known problem with 10 jobs and 10 machines formulated in 1963 [24] has been solved optimally only in 1989 [10]. Only some special cases with a small number of jobs or machines can be solved in polynomial time.

For the job shop problem without setup times several branch and bound algorithms and heuristics have been proposed. Among the algorithms of the former type, we mention the algorithms by Carlier and Pinson [10], Brucker *et al.* [8] and Applegate and Cook [4]. Concerning heuristics, various constructive and iterative algorithms have been proposed. In constructive algorithms often priority (or dispatching) rules have been applied (*see* for instance [6, 14, 16, 30]). They assign available operations with the largest priority to the machines, *i.e.* operations which can start with its processing. It has been stated in [6] that no single priority rule has been shown to consistently produce better results than other rules under a variety of shop configurations and operating conditions. In [17], a mixed dispatching rule has been proposed for short-term scheduling which can assign different priority rules to the machines. Most of the iterative algorithms are based on neighbourhood search. Usually some metaheuristic such as simulated annealing, tabu search or threshold accepting is included (*see* for instance [20] or [27]). Also some genetic algorithms for the heuristic solution of the job shop problem have been developed (*see* for instance [29]). A combination of construction of a solution and iterative improvement is the shifting bottleneck procedure [1] which is based on the repeated solution of single machine problems.

In modern production one is often concerned with problems, where setup times or change-over costs between the processing of certain jobs on the machines have to be considered. Jobs are classified into different groups or families to take advantage of similar processing requirements. A setup time is necessary if a job is processed on a machine after a job of another group. The setup times are often assumed to be sequence independent. Such

problems with family setup times are also referred to as *group technology* or *part family scheduling.*

A related type of problems arises when jobs have to be put together in a "batch" such that the processing of the batch can begin only when all jobs of a batch are available on the corresponding machine, *i.e.* the preceding operations of all jobs have been completed. In this case several consecutive batches of jobs of the same group are possible and a batch setup time is necessary before the processing of each batch. This situation occurs for instance when jobs have to be placed on a palette in a Flexible Manufacturing System and afterwards this palette is inserted in the machine. Combining this case with group technology, we may form a batch only by jobs of the same group.

Up to now, mostly single-stage scheduling problems with family or batch setup times have been considered in the literature (*cf.* [2, 5, 11, 13, 23, 25, 31]). An overview of results on both mentioned types of problems in the single machine case can be found in [36]. In connection with multi-stage problems, usually the first mentioned type of problems in connection with the flow shop case (identical technological routes for all jobs) has been considered. Contrary to the classical 2-machine flow shop problem, where the makespan can be minimized in polynomial time, the corresponding problem with family setup times (and hence the job shop problem, too) is already NP-hard [18]. Therefore, mostly heuristics have been proposed for such problems. Allison [3] gave some 2-stage constructive heuristics for the permutation flow shop problem (when each machine processes the jobs in the same order) with family setup times and an arbitrary number of machines. A similar constructive heuristic as well as an iterative simulated annealing algorithm have been proposed by Vakharia and Chang [34]. In the mentioned papers the minimization of the makespan has been considered and groups of jobs are not split, *i.e.* one always sequences complete groups of jobs. Several types of constructive and iterative algorithms for the more general case that groups of jobs can be split have been discussed in [32]. In the latter paper along with the minimization of the makespan other regular criteria have been considered.

In [9] a branch and bound algorithm for the general shop problem with sequence dependent setup times has been proposed. In that paper, also job shop problems with family setup times have been tested. The results from [9] confirm that the problems with setup times are much harder than the corresponding problems without setup times.

Related problems with some kinds of batching decisions are so-called lot-sizing problems which will not be considered in this paper. An overview of such problems can be found in [19] and [35]. Some recent trends in scheduling research which attempt to make it more relevant and applicable in operations management and manufacturing systems have been given in [22].

In this paper, we deal with job shop problems with setup times, where the groups of jobs to be processed are known in advance. Moreover, each job is allowed to be available for processing only after a certain release date. We consider both mentioned types of problems with several objective functions.

Whereas in the literature usually regular criteria are considered, we also treat objective functions that are not nondecreasing in the completion times of the jobs. The latter type of objective function is typical for "just-in-time" production. More precisely, we shall consider job shop scheduling to minimize the weighted sum of job earliness and tardiness penalties (the weighted squared deviation from the given due dates). This problem with arbitrary positive weights assigned to the jobs is NP-hard in the strong sense even for the single machine case without setups, since it is a generalization of the weighted tardiness problem which is NP-hard in the strong sense [21]. Moreover, it is NP-hard in the ordinary sense even when all job weights are equal [12].

Here we deal only with heuristic constructive algorithms. In particular, we investigate and test several insertion strategies and some refinements in dependence on the objective function of the considered type of problems and we perform a computational comparison with some standard priority rules.

The paper is organized as follows. The considered problems are described in detail in Section 2. Then we state in Section 3 the basic variant of the insertion algorithm and we propose different refinements. We demonstrate how these insertion techniques can be applied to the case of a nonregular criterion. Finally, the results of our computational experiments are presented and discussed in Section 4.

2. THE PROBLEM

The job shop problem with setup times can be formulated as follows: n jobs are partitioned into g groups G_1, G_2, ..., G_g. For each job i, a technological route $q^i = (j_1, j_2, ..., j_{k_i})$ is given, where k_i denotes the number of operations of job i and j_h with $1 \leq j_h \leq m$ is the h-th machine on which job i has to be processed. For simplicity of the description we

assume that each job has to be processed at most once on each machine but all presented algorithms can be applied to the more general case when a job has to be processed more than once on a machine. Times t_{ij} for processing an operation (i, j) of job i on machine j are given. Moreover, we have setup times $s_{hj} \geq 0$ assigned to group G_h on machine j. For the first type of considered problems, a setup occurs on a machine before the first job starts its processing on this machine and when a job has to be processed after a job of another group. This situation is denoted as item availability (ia). The processing of a job on a certain machine can start when

- the processing of the preceding job on this machine has been finished and, if necessary, a setup has been done and
- the processing of this job on the preceding machine has been finished.

The completion time of this operation is the time when the processing of this operation has been finished. The setup times are assumed to be sequence independent, *i.e.* they do not depend on the preceding job.

For the second type of problems, the jobs are processed within batches. A batch consists of one or more jobs of the same group, which are processed together on a machine (batch availability: ba). The processing of a batch on a certain machine can start when

- the processing of the preceding batch on this machine has been finished and a setup has been done and
- the processing of all jobs contained in the batch on their preceding machines has been finished.

The processing time of a batch is the sum of the processing times of the operations in the batch. The completion time of each operation in the batch is equal to the completion time of the whole batch. A setup times s_{rj} occurs before each batch of jobs of group G_r on machine j. Notice that it can be advantageous in the case of batch availability to form several batches of consecutive jobs of the same group. Although this leads to additional setups, the value of the objective function can possibly be reduced.

However, the case is also possible that we have batch availability only on a subset of the machines. Let the notation $ba\,(m_1, ..., m_h)$ indicate that we have batch availability on the machines $m_1, ..., m_h$ whereas on the remaining machines item availability is assumed.

In principle, two types of setup times are possible:

Anticipatory setup times. The setup can start before the processing of the job of the preceding machine is finished, *i.e.* the setup can be done "in anticipation" of the job;

Nonanticipatory setup times. The setup cannot start until the processing of the job on the preceding machine is finished.

In the following we always assume anticipatory setup times.

In this paper, we consider several objective functions that are based on the completion times C_i of the jobs. Among the regular criteria (objective functions which are nondecreasing in the completion times $C_1, C_2, ..., C_n$), we consider the minimization of the makespan $C_{\max}$ and the minimization of the weighted sum of completion times $\sum w_i C_i$, where w_i is a weight assigned to job i. In the case of a regular criterion it is sufficient for the description of a schedule to give the sequence of operations on each machine since each operation starts with the processing as early as possible. However, this is not the case if so-called nonregular criteria are considered which are typical for "just-in-time" production. Often such criteria are based on given due dates, where both tardiness and earliness penalties are included. Among the criteria of this type, we consider the minimization of $\sum w_i (C_i - d_i)^2$, where d_i is the due date of job i.

Moreover, in our experiments we also consider the case when a release date $r_i \geq 0$ is assigned to each job, *i.e.* the processing of the first operation of each job i cannot start before time r_i. The release dates are assumed to be known in advance.

The considered problems are denoted as follows:

- $J/s_{hj} \geq 0,\ ia,\ r_i \geq 0/C_{\max}$;
  $J/s_{hj} \geq 0,\ ba(M),\ r_i \geq 0/C_{\max}$
- $J/s_{hj} \geq 0,\ ia,\ r_i \geq 0/\sum w_i C_i$;
  $J/s_{hj} \geq 0,\ ba(M),\ r_i \geq 0/\sum w_i C_i$

and

- $J/s_{hj} \geq 0,\ ia,\ r_i \geq 0/\sum w_i (C_i - d_i)^2$;
  $J/s_{hj} \geq 0,\ ba(M),\ r_i \geq 0/\sum w_i (C_i - d_i)^2$.

In our tests we consider certain sets $M \subseteq \{1, 2, ..., m\}$ which we shall precise later.

For describing partial solutions in the case of a regular criterion, we use a similar way as it has been done in [37] for the job shop problem without considering setup times. It is known that a feasible combination of

the given technological routes and the chosen job orders on the machines can be represented by a digraph $D = (V, E)$, where the vertex set V represents the set of operations (i, j) and the arc set E describes both the technological routes and the job orders, *i.e.* the digraph D completely describes the sequences in which the operations are performed.

Example 1: Assume that we have $n = 6$ jobs and $m = 3$ machines. Let the given technological routes be as follows:

$$q^1 = (1, 3, 2), \quad q^2 = (3, 2, 1), \quad q^3 = (1, 3, 2),$$
$$q^4 = (1, 2, 3), \quad q^5 = (1, 3, 2), \quad q^6 = (2, 1, 3).$$

A combination of the technological routes and the job orders is described by the digraph $D = (V, E)$ given in Figure 1. The "horizontal arcs" describe the given technological routes and the "vertical arcs" describe the chosen job orders on the machines. From Figure 1 we obtain the job orders:

$$p^1 = (1, 3, 2, 5, 4, 6), \quad p^2 = (6, 2, 1, 3, 4, 5), \quad p^3 = (2, 1, 3, 5, 6, 4),$$

where p^j describes the job order on machine j. Moreover, the schedule presented in Figure 1 is feasible for the job shop problem since all given technological routes are satisfied and the corresponding digraph does not

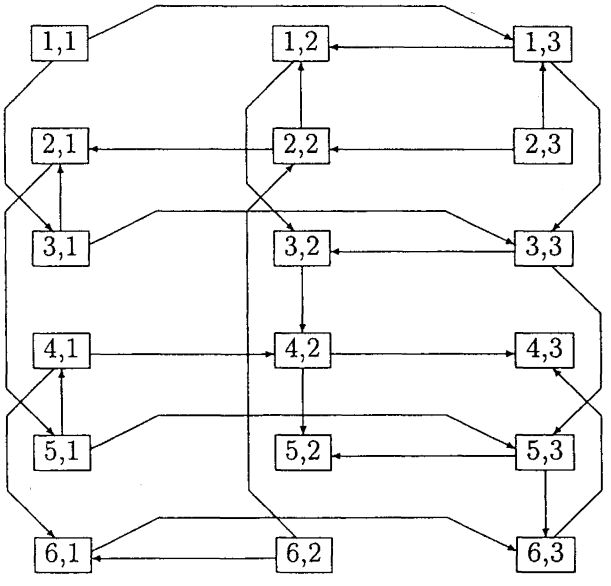


Figure 1. – Digraph $D = (V, E)$.

contain any cycle. A convenient representation of such a schedule may be given by the rank matrix A of the above digraph $D = (V, E)$, *i.e.* we assign to each vertex (operation) (i, j) the maximal number of vertices on a path from a source to vertex (i, j). We note that this representation can also be used for partial schedules, where only a subset of operations occurs in the matrix.

In the case of item availability the occurrence of setup times is automatically determined from the matrix A by the fact that a setup occurs on a machine if and only if the first job on this machine starts or a job of a group other than the group of the preceding job starts with processing on this machine.

However, in the case of batch availability, it may be useful to divide a set of successive jobs of the same group into two or more batches. Moreover, because all jobs of a batch enter and leave this machine simultaneously, the sequence of operations within this batch does not influence the objective function value. It is convenient to consider each batch as a unit. Thus, in the digraph D described above, all vertices representing operations of such a batch are combined into a single vertex. Consequently, to all operations of such a batch the same rank value is assigned in the matrix A.

To illustrate, we return to the above example (see Fig. 1) and assume that three groups $G_1 = \{1, 2, 3\}$, $G_2 = \{4, 5\}$ and $G_3 = \{6\}$ are given. If we consider a problem with item availability, the matrix A of this example looks as follows:

$$A = \begin{bmatrix} 1 & 3 & 2 \\ 3 & 2 & 1 \\ 2 & 4 & 3 \\ 5 & 6 & 8 \\ 4 & 7 & 5 \\ 6 & 1 & 7 \end{bmatrix}.$$

Additionally we assume that the processing times and the setup times are given by

$$S = \begin{bmatrix} 6 & 4 & 3 \\ 7 & 3 & 2 \\ 5 & 2 & 5 \end{bmatrix} \quad \text{and} \quad T = \begin{bmatrix} 6 & 4 & 6 \\ 4 & 6 & 7 \\ 3 & 2 & 7 \\ 2 & 18 & 5 \\ 3 & 13 & 12 \\ 4 & 7 & 10 \end{bmatrix}.$$

Figure 2 represents the schedule given by the matrix A using a Gantt chart. In particular, we have $C_1 = C_2 = 23$, $C_3 = 28$, $C_4 = 67$, $C_5 = 66$ and $C_6 = 60$. Thus, we get the objective function values $C_{\max} = 67$ and $\sum C_i = 267$. However, it is possible that a schedule that is feasible in the case of item availability is not feasible in the case of batch availability. For the above example with batch availability on machines 1 and 3, combining all consecutive jobs of the same group on a machine into the same batch would result in an infeasible schedule. The batch on machine 1 consisting of the jobs of group G_1 must be processed before the batch of these jobs on machine 3 and vice versa, *i.e.* we have a contradiction.

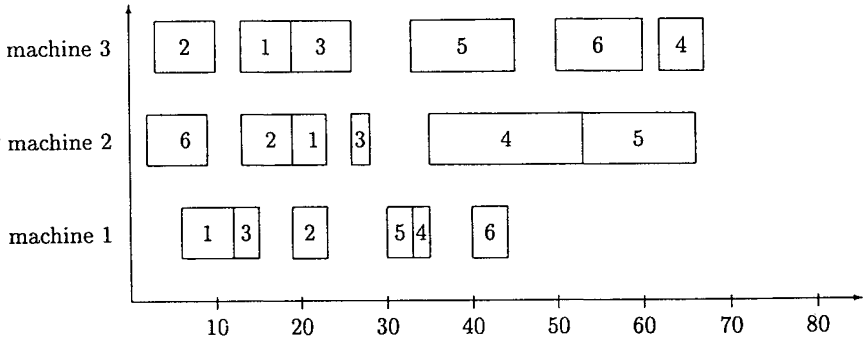


Figure 2. – Gantt chart of the schedule for item availability.

If we keep the sequence of all operations but split the jobs of group G_1 into one batch consisting of the jobs 1 and 3, and one batch consisting only of job 2 on each machine, we get the schedule presented in the Gantt chart in Figure 3. Thus, we have 4 batches on the machines 1 and 5 batches on machine 3. In this case we get $C_1 = 34$, $C_2 = 25$, $C_3 = 34$, $C_4 = 71$, $C_5 = 68$ and $C_6 = 64$. Consequently, we get the objective function values $C_{\max} = 71$ and $\sum C_i = 296$. This schedule is represented by the rank matrix

$$A' = \begin{bmatrix} 1 & 3 & 2 \\ 3 & 2 & 1 \\ 1 & 4 & 2 \\ 4 & 5 & 7 \\ 4 & 6 & 5 \\ 5 & 1 & 6 \end{bmatrix}.$$

We note, that it is not necessary to form the same batches on each machine.

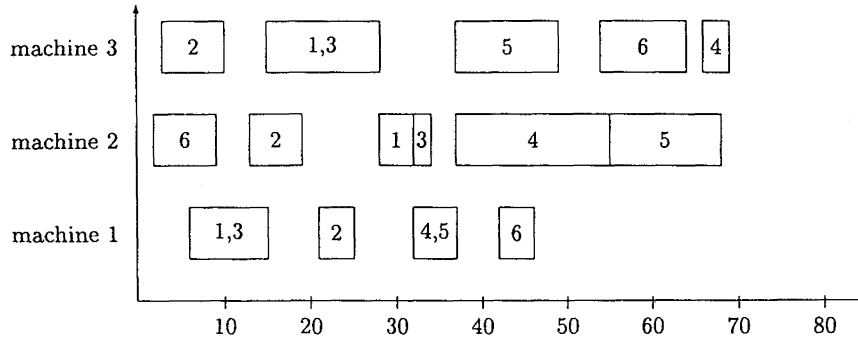


Figure 3. – Gantt chart of the schedule for batch availability on the machines 1 and 3.

3. CONSTRUCTIVE ALGORITHMS

For job shop problems without consideration of setup times often priority rules have been used. On the other hand, recently for classical shop scheduling problems often insertion techniques have been applied for constructing a rather good feasible schedule. Nawaz *et al.* [26] proposed such an algorithm for the permutation flow shop problem, where the jobs are successively inserted into a partial sequence. Similar algorithms have been given for job shop and open shop problems by Werner and Winkler [37] and Bräsel *et al.* [7]. However, for the latter problems operations are successively inserted into partial schedules. In both algorithms the basic principle of beam search [28] has also been used. Beam search is basically an adaptation of branch and bound such that at each level of the branching tree only the most promising nodes are selected as nodes to branch from. The remaining nodes at that level are disgarded permanently.

First we discuss the insertion of an operation in the case of the job shop problem without considering setup times. Assume that a rank matrix A for a partial schedule has been determined and that in accordance with the chosen insertion order operation (i, j) has to be inserted next. Then operation (i, j) is inserted on all possible positions in the job order of machine j, for which the resulting matrix after the insertion of operation (i, j) is again a rank matrix, all technological routes of the jobs are satisfied and all previously established precedence relations between certain operations remain valid.

Assume that on machine j a set of u operations have already been sequenced. Let $k_1 < k_2 < \ldots < k_u$ be the entries of column j of the matrix A and assume that $a_{i_h,j} = k_h$ for $1 \leq h \leq u$. Then the following insertions have to be considered:

1. Process job i as the first job on machine j: Determine

$$a_{ij} = \max_v \{a_{iv} | \text{operation } (i, v) \text{ precedes operation } (i, j)\} + 1,$$

modify the remaining entries of column j (already sequenced operations on machine j) and the entries of the matrix corresponding to succeeding operations of the already sequenced operations on machine j. If this leads to a modification of entry a_{ij}, then the considered insertion of operation (i, j) is not feasible since this operation must precede itself.

2. Process job i as the l-th job on machine j $(2 \leq l \leq u + 1)$: Determine

$$a_{ij} = \max \{\max_v \{a_{iv} | \text{operation } (i, v) \text{precedes operation} (i, j)\}; k_{l-1}\} + 1.$$

Then we test feasibility of the partial schedule in an analogous way as described above. Thus, all operations belonging to column j having an entry greater than k_l and their succeeding operations must also be a successor of the inserted operation. Analogously to the above case, if this leads to a cycle in the resulting digraph, the corresponding insertion is infeasible.

To illustrate, we consider the given technological routes of Example 1 introduced in the previous section and a partial schedule represented by the matrix

$$A = \begin{bmatrix} 1 & . & 4 \\ . & 2 & 1 \\ . & . & . \\ . & . & . \\ 3 & 6 & 5 \\ 2 & 1 & 3 \end{bmatrix}.$$

Let the operation (1, 2) be inserted next. We remind that the technological route of job 1 is $q^1 = (1, 3, 2)$. In the following A^l describes the matrix when job i is inserted on position l in the job order on machine j:

$$A^1 = \begin{bmatrix} 1 & 5 & \underline{[9]} \\ . & \underline{7} & 1 \\ . & . & . \\ . & . & . \\ \underline{8} & . & . \\ \underline{7} & \underline{6} & \underline{8} \end{bmatrix}, \qquad A^2 = \begin{bmatrix} 1 & 5 & 4 \\ . & \underline{6} & 1 \\ . & . & . \\ . & . & . \\ 3 & \underline{7} & 5 \\ 2 & 1 & 3 \end{bmatrix},$$

$$A^3 = \begin{bmatrix} 1 & 5 & 4 \\ . & 2 & 1 \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ 3 & 6 & 5 \\ 2 & 1 & 3 \end{bmatrix} \quad \text{and} \quad A^4 = \begin{bmatrix} 1 & 7 & 4 \\ . & 2 & 1 \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ 3 & 6 & 5 \\ 2 & 1 & 3 \end{bmatrix}.$$

If on machine 2 job 1 is processed first, we get $a_{12} = 5$ since operation (1, 2) is the direct successor of operation (1, 3). Now, all remaining operations on machine 2 and their successors must also be a successor of operation (1, 2). Therefore we modify the corresponding values of A beginning with the smallest ones. Thus, we get the matrix A^1 with $a^1_{62} = 6$, $a^1_{61} = 7$, $a^1_{51} = a^1_{63} = 8$ and we would obtain $a^1_{13} = 9$, which contradicts the technological route of job 1 (the underlined entries of A^1 denote the modified values). Hence, the processing of job 1 as the first job on machine 2 does not lead to a feasible partial schedule. The remaining insertions yield feasible partial schedules (*see* matrices $A^2 - A^4$).

To evaluate partial schedules, we first introduce a cost to vertex (*i.e.* to operation) (i, j) given by the processing time t_{ij}. Moreover, we introduce heads r_{ij} and tails z_{ij} of an operation (i, j) in the usual way, *i.e.* r_{ij} denotes the length of a longest path from one of the sources to (i, j) and z_{ij} represents the length of a longest path from vertex (i, j) to one of the sinks. In each case the vertex cost of (i, j) is not included. Note that the head r_{ij} takes also into account the operations of job i which precede (i, j) in the technological route but are not yet inserted. Analogously, z_{ij} includes also the job tail, *i.e.* the sum of processing times of all succeeding operations of job i, whether they are already inserted or not. Hence, the head r_{ij} gives the current earliest possible starting time of operation (i, j) by considering the already scheduled operations and the technological route of job i. Analogously, the tail z_{ij} gives the number of time units that are at least necessary after the completion of operation (i, j) to finish the processing of the remaining operations.

When considering a job shop problem with family setup times, we must additionally take a setup time into account if a job of another group starts. In the introduced digraph model this can be represented by introducing an arc cost given by the setup time on each arc between two vertices that represent two successive operations of jobs of different groups. To represent the setups for processing the first job on each machine, a fictitious source with a vertex cost equal to zero and an additional arc from this source to the operations

that are processed first on the machine are introduced (the fictitious source is considered to have rank zero). The arc costs are given by the corresponding setup times. Now the heads and tails of the operations are again represented by the corresponding longest paths in the modified digraph.

A partial schedule is then evaluated by its current objective function value, *i.e.* in the case of minimizing the makespan by $\max\{r_{ij} + t_{ij} + z_{ij} | (i, j) \in V\}$. When the objective function depends on the sum of the completion times, we estimate C_i by the lower bound $LBC_i = \max_j \{r_{ij} + t_{ij} + z_{ij} | (i, j) \in V\}$.

In addition to the approach proposed in [37], we consider only "active" insertions of operations in the case of regular criteria since there always exists an optimal active schedule (a schedule is called active if no operation can start earlier without delaying the processing of another operation). Hence an insertion of operation (i, j) at a certain position is not considered if it is possible that a succeeding operation (L, j) on the same machine would be completed before operation (i, j) can start, *i.e.* if

$$s^*_{lj} + t_{lj} \leq r_{ij}$$

holds, where s^*_{lj} is the earliest starting time of operation (l, j) after shifting it before operation (i, j).

We also include the beam search principle from [28]. The idea is to construct a limited number of partial schedules in parallel. A partial schedule has usually several "sons" which are the immediate successors in the branching tree, *i.e.* they result from all possible insertions of the current operation. If we use a beamwidth b, we select in each step the b best partial schedules with respect to the current objective function value (which form the beam), where the best b schedules are always selected from the whole set of generated sons, *i.e.* the selected sons must not necessarily have different fathers. We only note here that in [28] also filtered beam search as a special form of a "look-ahead" procedure has been proposed, which requires that in each step a certain number of partial schedules must be extended to a complete schedule by some priority rule and only then the decision is made which partial schedules are selected as the "best" sons.

3.1. Successive insertion of the operations for a regular criterion

First we consider the variant that in each step exactly one operation is inserted. This variant has been used in [7] and [37] for open shop and job

shop problems without considering setup times. In these papers the operations have been inserted according to nonincreasing processing times.

In connection with the different problems considered in this paper, we include the following insertion orders or the operations into our tests.

1. The operations are inserted

(a) according to nondecreasing processing times,

(b) according to nonincreasing processing times or

(c) in random order.

2. The operations are inserted job by job. The jobs are inserted

(a) according to nonincreasing weights,

(b) according to nondecreasing release dates,

(c) according to nondecreasing values $d_i - r_i - T_i$, where T_i denotes the time, which is at least necessary to process job i completely, or

(d) in random order.

3. The operations are inserted job by job, however, after having determining the first job to be inserted, all remaining jobs of this group are inserted by applying one of the orders (a)-(d) mentioned in 2) and only if all jobs of the current group have been inserted, the job (and hence the corresponding group) that has to be inserted next is determined by applying the same order.

Note that in dependence on the considered objective function only a subset of the orders (a)-(d) mentioned in 2) is of interest. For instance, if all weights are equal ($w_i = 1$ for each $i = 1, 2, ..., n$), the order 2a) is identical with a random order of the insertion of the jobs. Analogously, if release dates do not occur (*i.e.* $r_i = 0$ for each $i = 1, 2, ..., n$), the order 2b) is identical with a random order of the insertion of the jobs.

If we consider problem $J/s_{hj} \geq 0, ia/C_{\max}$ or problem $J/s_{hj} \geq 0, ba\,(M)/C_{\max}$, we can apply in addition to 2d) also order 2c), which reduces to an insertion of the jobs according to nonincreasing T_i values (in the case of these problems, we have $r_i = 0$ and we can define $d_i = 0$). As T_i we use the time that is necessary after r_i to complete job i if only this job is processed and setup times of the group of this job are taken into consideration. The criterion 2c) can be interpreted as a "bottleneck strategy", where the jobs with the strongest restrictions resulting from the release and due dates as well as from the minimal required time for processing a job completely are considered first.

It remains to discuss the question in which order the operations of the current job should be inserted when applying a jobwise insertion. In initial experiments we tested three insertion variants for the cases 2) and 3), namely the insertion of the operations of a job according to the technological route (*i.e.* starting with the first operation of a job, then the second operation and so on), in opposite order (*i.e.* starting with the last operation of a job, then the second last operation and so on) or in random order. Our initial tests have shown that the insertion according to the technological route has obtained the best results for all considered problems. Hence, in the following this insertion order has always been used in connection with regular criteria when jobwise insertion of the operations has been applied.

As already mentioned, in the case of batch availability it can be preferable to split jobs of the same group, which have to be processed consecutively on the same machine, into two or more batches. Although this leads to additional setup times, the objective function value can possibly be smaller since not all jobs of this group must wait for the next processing until all consecutive jobs of the same group have been completed. However, a formed batch is never split later but it can be combined with further jobs of the corresponding group. If a job is considered for insertion at a certain position and the preceding batch consists of jobs of the same group, we can include this job into the preceding batch or this job can form a new batch.

3.2. Successive insertion of the operations for a nonregular criterion

Next, we turn on the consideration of a nonregular criterion, where the objective function value depends on the completion times of the last operations of the jobs, too. Contrary to regular criteria, here the operations do not necessarily start as early as possible. Moreover, not only "active" insertions of operations should be considered. Since the objective function value depends on the completion time of the last operation of the jobs, it seems to be recommendable to insert the operations of each job "backwards".

Having scheduled the last operation of a job, we schedule all remaining (*i.e.* "nonlast") operations of this job in each case as late as possible with respect to the chosen sequence of the operations. The reason for that is to leave as much time as possible for the processing of the still unscheduled first operations of the job, *i.e.* we want to avoid that, due to a current head of an operation, the already scheduled operations must later be unnecessarily shifted to the right (*i.e.* to a later time) which could lead to a considerable increase in the objective function value. In the following we describe a

partial schedule by the matrix $C = [c_{ij}]$ of the completion times of the scheduled operations meeting the above convention, where c_{ij} denotes the completion time of job i on machine j.

To describe the modified insertion in the case of a nonregular criterion, we distinguish between the insertion of the last operation of a job and the insertion of a nonlast operation. First we describe the insertion of the last operation of a job i. Assume that this operation has to be performed on machine j and let $k_1, k_2, ..., k_u$ be the jobs that have already been scheduled on machine j. We consider the case of item availability.

When inserting job i on machine j at position $h+1$ with $h > 1$ (*i.e.* between operations (k_h, j) and (k_{h+1}, j)), we first check whether

$$c_{i_h,j} + f \cdot s_{gn[i],j} + t_{ij} \geq d_i \tag{1}$$

holds, where $f = 1$ if jobs i_h and i belong to different groups and $f = 0$ otherwise, and $gn[i]$ denotes the index of the group to which job i belongs. If inequality (1) holds, then the earliest possible completion time of job i sequenced at position $h+1$ on machine j is not smaller than its due date. In this case (*late insertion*) we schedule job i directly after its predecessor k_h. This could force a right shift of some of the operations $(k_{h+1}, j), ..., (k_u, j)$ and their successors.

If inequality (1) does not hold (*early insertion*), we try to schedule job i on machine j before job k_{h+1} "as close as possible" to the due date d_i on this machine without delaying the start of operation (k_{h+1}, j). However, this can cause possibly left shifts of some of the operations $(k_h, j), ..., (k_1, j)$ and their predecessors. These necessary left shifts may eventually not be feasible, *i.e.* the current head of some operation would be violated. In such a situation we get a "forced" right shift of operation (i, j) and possibly of some of the succeeding operations in the current partial schedule. The described procedure is analogously applied if job i is sequenced at the first position on machine j.

The computational expense can possibly be reduced by restricting the positions for the insertion of operation (i, j) as follows: If an operation (i, j) can be inserted at a certain position without shifting the processing of any preceding operation, then no insertion at smaller positions is considered.

Whenever a right shift of an operation has been carried out, caused either by a late insertion of a last operation or by a forced right shift, in the resulting schedule possibly some modifications must be made to ensure that our convention concerning the completion times of all nonlast operations

of the jobs is satisfied, *i.e.* each nonlast operation is scheduled as late as possible with respect to its successors. Furthermore, each early last operation of a job is shifted towards its due date as far as possible with respect to its successor, *i.e.* this operation will not be late.

Nonlast operations of a job are inserted according the conventions mentioned above, where we consider the possible positions in the sequence beginning from the right. To reduce the required expense, we immediately accept the first insertion beginning from the right, where this insertion does not increase the objective function value with respect to the previous step. This covers the case of an "active insertion from the right". Notice that an insertion on a smaller position could possibly even improve the objective function value but only due to a forced right shift. However, the occurrence of forced shifts when inserting an operation other than the first operation of a job can cause "bottlenecks" for the insertion of the remaining operations of the current job and, therefore, they can lead to a substantial increase of the objective function value.

Up to now we considered only "forced" shifts of operations caused by some head of an operation due to the inserted operation. One possibility to refine an obtained schedule is to allow also "unforced" shifts of a last operation of a job towards its due date. Hence, in the case that a last operation (i, j) is inserted late, we "push" this operation in steps of a certain number of time units to the left until one of the following conditions is satisfied:

a) the objective function value has not been increased with respect to the previous step or

b) the starting time of an operation is equal to its head or

c) the completion time c_{ij} is equal to the due date d_i.

Similarly, in the case of an early last operation we push this operation in steps of a certain number of time units to the right until condition a) or c) is satisfied (clearly, condition b) cannot occur since only right shifts are considered). Based on initial tests, we decided to use a steplength of $\lceil |C_i - d_i|/24 \rceil$.

This refinement can be applied to partial schedules as well as to complete schedules. When combining it with the insertion method described above, to reduce the expense we restrict this approach to the best insertion position found for any last operation of a job. In the case that we modify a complete schedule by this approach, we apply the unforced shifts to all last operations of jobs ordered according to nonincreasing contributions to the objective function value.

The case of batch availability on machine j can be handled in a similar way as it has been done in the case of a regular criterion but constructing the schedule from the right. To illustrate the insertion algorithm for a nonregular criterion, we consider the following example.

Example 2: Let $n = 5$, $m = 3$ and the jobs be partitioned into three groups $G_1 = \{1, 2, 3\}$, $G_2 = \{4\}$ and $G_3 = \{5\}$. Assume that the jobs 1 and 3 have the technological route $q^1 = q^3 = (1, 2, 3)$ and the jobs 2, 4 and 5 have the technological route $q^2 = q^4 = q^5 = (2, 1, 3)$. Moreover, item availability of the jobs is assumed and the objective is to minimize $F = \sum w_i (C_i - d_i)^2$. Let the setup and processing times be given by

$$S = \begin{bmatrix} 4 & 2 & 1 \\ 1 & 3 & 4 \\ 5 & 6 & 2 \end{bmatrix} \quad \text{and} \quad T = \begin{bmatrix} 4 & 6 & 2 \\ 2 & 5 & 3 \\ 1 & 3 & 5 \\ 4 & 2 & 6 \\ 5 & 7 & 1 \end{bmatrix}.$$

Moreover, let the release dates, job weights and due dates be as follows:

i	1	2	3	4	5
r_i	0	4	8	3	12
w_i	1	3	5	4	7
d_i	25	40	35	38	30

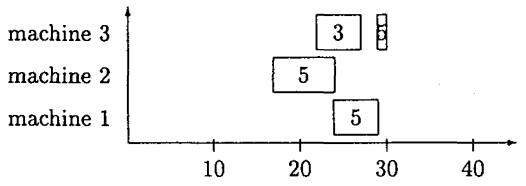
First, we get the following job heads r'_{ij} of the operations, *i.e.* the earliest possible starting time of operation (i, j) by considering only the data of job i and the setups of the corresponding group:

$$R' = \begin{bmatrix} 4 & 8 & 14 \\ 9 & 4 & 11 \\ 8 & 9 & 12 \\ 5 & 3 & 9 \\ 19 & 12 & 24 \end{bmatrix}.$$

To illustrate, we apply insertion order 2a) backwards. Hence, we start with job 5 and get successively the completion times $c_{53} = 30$, $c_{51} = 29$ and $c_{52} = 24$. Next we insert job 3. Operation (3, 3) can be sequenced before or after operation (5, 3). This yields the matrices $C^{(1)}$ and $C^{(2)}$, respectively

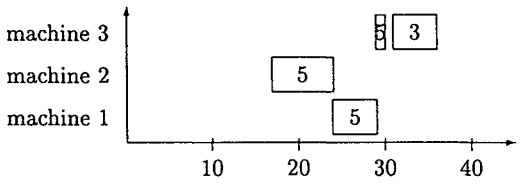
(for illustration, we present also the corresponding Gantt charts for the first three matrices $C^{(i)}$):

$$C^{(1)} = \begin{bmatrix} . & . & . \\ . & . & . \\ . & . & 27 \\ . & . & . \\ 29 & 24 & 30 \end{bmatrix},$$



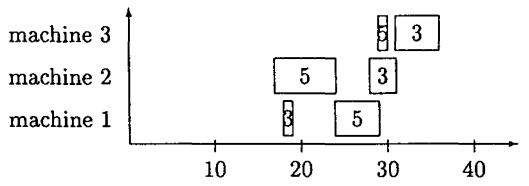
and

$$C^{(2)} = \begin{bmatrix} . & . & . \\ . & . & . \\ . & . & 36 \\ . & . & . \\ 29 & 24 & 30 \end{bmatrix},$$



where $F(C^{(1)}) = 320$ and $F(C^{(2)}) = 5$. Thus, we have to choose $C^{(2)}$. Inserting the remaining operations of job 3, we get the partial schedule

$$C^{(3)} = \begin{bmatrix} . & . & . \\ . & . & . \\ 19 & 31 & 36 \\ . & . & . \\ 29 & 24 & 30 \end{bmatrix},$$



with $F(C^{(3)}) = 5$. Inserting the last operation of job 4, we get

$$C^{(4)} = \begin{bmatrix} . & . & . \\ . & . & . \\ 19 & 31 & 36 \\ . & . & 46 \\ 29 & 24 & 30 \end{bmatrix}, \qquad C^{(5)} = \begin{bmatrix} . & . & . \\ . & . & . \\ 19 & 41 & 46 \\ . & . & 40 \\ 29 & 24 & 30 \end{bmatrix}$$

$$\text{and} \qquad C^{(6)} = \begin{bmatrix} . & . & . \\ . & . & . \\ 19 & 31 & 36 \\ . & . & 27 \\ 29 & 24 & 30 \end{bmatrix}$$

with $F(C^{(4)}) = 261$, $F(C^{(5)}) = 621$ and $F(C^{(6)}) = 489$, *i.e.* we have to choose $C^{(4)}$. The first two insertions represent late insertions and the last insertion is an early insertion. Inserting now the remaining operations of job 4, all operations of job 2 and the last operation of job 1 we get the partial schedule

$$C^{(7)} = \begin{bmatrix} . & . & 24 \\ 33 & 31 & 36 \\ 16 & 26 & 33 \\ 40 & 36 & 46 \\ 26 & 21 & 27 \end{bmatrix}$$

with $F(C^{(7)}) = 388$. Inserting operation (1, 2), we only need to consider the insertion at the positions 1 and 2 (since operation (3, 2) starts after the starting time $s^*_{13} = 22$ and no setup is required when inserting job 1 before job 3). When inserting this operation at position 1 (*i.e.* when trying to sequence operation (1, 2) in front of operation (5, 2)), then, due to $r'_{12} = 14$, operation (5, 2) must be shifted to the right for 6 time units. Thus we get in this case the partial schedule $C^{(8)}$ and additionally the partial schedule $C^{(9)}$ for the insertion at position 2:

$$C^{(8)} = \begin{bmatrix} . & 14 & 25 \\ 39 & 37 & 42 \\ 22 & 32 & 39 \\ 46 & 42 & 52 \\ 32 & 27 & 33 \end{bmatrix} \qquad \text{and} \qquad C^{(9)} = \begin{bmatrix} . & 27 & 29 \\ 38 & 36 & 41 \\ 21 & 31 & 38 \\ 45 & 41 & 51 \\ 31 & 19 & 32 \end{bmatrix},$$

where $F(C^{(8)}) = 939$ and $F(C^{(9)}) = 768$. Note that operation (1, 3) has been shifted to the right in $C^{(8)}$ only for one time unit since $d_1 = 25$. Finally, the insertion of operation (1, 1) does not change the objective function value of $C^{(9)}$.

When using the modified insertion of the last operations of the jobs with unforced shifts, we get the same partial schedules when inserting

jobs 5 and 3. When inserting operation (4, 3) and applying unforced shifts to schedule $C^{(4)}$ in steps of one time unit, we get the partial schedule

$$C^{(4)'} = \begin{bmatrix} . & . & . \\ . & . & . \\ 17 & 29 & 34 \\ . & . & 44 \\ 27 & 22 & 28 \end{bmatrix}$$

with $F(C^{(4)'}) = 177$. Example 2 illustrates that the consideration of unforced shifts can be preferable when in the further insertion process hardly forced right shifts or only very small forced right shifts occur.

3.3. Parallel insertion of the operations

Insertion algorithms can be improved by including some "look-ahead" procedure. The previously mentioned form of filtered beam search belongs to such algorithms. In this paper, we take a different form of such an approach into consideration. To get a better evaluation of the insertion of operations, it seems to be recommendable to insert several operations in parallel. In our implementation we restrict this procedure to the insertion of up to k operations of the same job. According to the chosen insertion order of the operations, we insert k succeeding operations (according to the technological route, *i.e.* starting with the first operation) in parallel. This means that in one step k operations are simultaneously inserted into the current partial schedule in all possible ways and only then the different new partial schedules are compared and the best variant is chosen. Thus, in one step the influence of k new inserted operations is evaluated. Notice that in the case of inserting the last operations of a job possibly less than k operations of this job can be inserted in one step.

3.4. Priority rules

For comparison purposes we included various well-known priority rules from job shop scheduling into our tests which have been adapted to the considered problems. We schedule an operation next on that machine, where the earliest completion time t^* among all operations available for processing

is possible. Then we select among all operations that can start before t^* on the chosen machine an operation according to a certain rule. Among the rules considered in our tests we present computational results for the following rules:

- *ECT*: Select the operation that leads to the earliest completion time among all available operations.
- *FIFO*: Select the operation which enters the queue at the machine first.
- *NINQ*: Select the operation of that job from the current queue whose next machine for processing according to the technological route has the shortest queue.

The above priority rules can be adapted to the considered problems as follows. In the case of item availability, the setup time is included if it is necessary due to the preceding job on this machine.

If in the case of batch availability on the currently considered machine, a job has been selected that belongs to the same group as the preceding job or batch, it has to be decided whether the new job is inserted into the previous batch or combined with the previous job into a batch, respectively, or not. This decision is made by comparing the completion times of the last operations on this machine in both variants and choosing the better one.

Finally we mention that we also tested variants, where jobs of the same group as the last sequenced job on the currently considered machine get a higher priority than jobs of other groups. However, in most cases such a modification has led to worse results.

In the case of considering a nonregular criterion involving due dates, we additionally include the following rules:

- *SLACK*: Select the job which has the minimum slack time, where the slack time is defined as due date minus starting time minus remaining processing time.
- *URG*: Select the job with the maximum urgency, where the urgency of a job is defined as remaining processing time minus due date.

When a feasible schedule has been generated, we consider unforced shifts applied to a complete schedule as described previously.

4. COMPUTATIONAL RESULTS

4.1. Generation of the test problems

First we describe the generation of the test problems, which has been done in a similar way as proposed by Taillard [33]. A problem is entirely defined by the initial value of a seed x_0 of a random generator. The (pseudo) random number generator is based on the recursive formula

$$x_{i+1} := (16\,807\, x_i) \bmod (2^{31} - 1).$$

This formula provides a uniformly distributed sequence of numbers in the closed interval $[0, 2^{31} - 2]$, which are transformed to integers z_i from the closed interval $[a, b]$ according to

$$z_i = a + \left\lfloor \frac{(b-a)\, x_i}{2^{31} - 1} \right\rfloor.$$

The considered job shop problems with setup times are characterized by the processing times t_{ij} of job i on machine j, by the setup times s_{rj} when starting with a batch of group r on machine j, by partitioning the jobs into groups, by assigning a technological route and a release date r_i to each job and by assigning a weight w_i to each job. By using the algorithm proposed by Taillard [33], first the matrix T of the processing times and then the matrix S of setup times are rowwise generated, where the processing times are uniformly distributed integers in $[a_1, b_1]$ and the setup times are uniformly distributed integers in $[a_2, b_2]$.

Then the release dates are assigned to the jobs. Job 1 has the release date $r_1 = 0$. Then the release dates of the jobs 2, 3, ..., n are generated, which are uniformly distributed integers in the closed interval $[0, r_{\max}]$, where $r_{\max} = \lfloor \frac{1}{2} b_1\, n \rfloor$. Next g groups of jobs are generated. This is done by assigning a value $gn[i]$ to each job i with $1 \leq i \leq n$, which means that job i belongs to group $G_{gn[i]}$. First, each job i with $1 \leq i \leq g$ is inserted into group G_i and then for $i = g+1, g+2, ..., n$ the value $gn[i]$ is determined as a uniformly distributed integer in $[1, g]$. In our experiments we always choose $g = \lfloor n/3 \rfloor$.

Then the technological routes of all jobs are generated. In all problems, each job has to be processed exactly once on each machine. For each job i, $1 \leq i \leq n$, the technological route is determined by inserting successively machine k, $k = 2, 3, ..., m$, into the partial sequence of the

first $k-1$ machines. For inserting machine k, the generated random number is transformed to an integer from the closed interval $[1, k]$, which gives the position of machine k among the machines $1, 2, ..., k$. Finally, in the case of minimizing the weighted sum of completion times, the weights w_i are generated, which are uniformly distributed integers in the closed interval $[1, 10]$.

In all tests we used $a_1 = 1$ and $a_2 = 0$. Moreover, we considered problems with dominant setup times (DST; $b_1 = 50$, $b_2 = 200$), with dominant processing times (DPT; $b_1 = 200$, $b_2 = 50$) and with similar processing and setup times (SIT; $b_1 = b_2 = 100$). Our experiments have been performed on a PC 486 (66 MHz) and we have coded all algorithms in $C++$. For generating the i-th problem of each type, we use the seed $x_0 = 2654321 + i \star 2123456$.

In the following we give the results in the case of a regular criterion for the algorithms I_b (insertion algorithm with beamwidth b), Ik (insertion algorithm with k parallel insertions) as well as for the priority rules *ECT*, *FIFO* and *NINQ*.

4.2. Insertion orders

First we tested the proposed insertion orders for the basic variant I_1 of the insertion algorithm. To this end, we have fixed $n = m = 10$ and considered problems with both regular criteria and with both kinds of job availability, while for the second type of problems batch availability is assumed on the machines 6-10. For each problem type we generated 20 problems with DST, 20 problems with DPT, and 20 problems with SIT and we compared the average objective function values for each problem type. We considered the insertion orders 1a)-1c), 2a)-2d) and 3a)-3d). In Table 1 we give the percentage of the average objective function value of the individual variants over the best average value of each class.

From Table 1 it can be seen that an operationwise insertion (1a)-1c)) works bad. Among the jobwise insertion orders, we see that a suitable insertion order depends on the given objective function. Whereas in the case of makespan minimization an insertion according to nondecreasing values $d_i - r_i - T_i$ turns out to be preferable, one can recommend an insertion according to nonincreasing weights when minimizing $\sum w_i C_i$. Additionally, in the case of DST problems a groupwise insertion of the jobs can be recommended, while for SIT and DPT problems a groupwise insertion of the jobs should

TABLE 1
A comparison of several insertion orders.

io	1a	1b	1c	2a	2b	2c	2d	3a	3b	3c	3d
$J/s_{hj} \geq 0,\ ia,\ r_i \geq 0/C_{\max}$											
DST	20.84	19.12	17.02	5.90	9.11	3.88	4.03	3.06	2.69	0.0	0.97
DPT	15.62	7.30	9.75	4.20	7.79	0.0	6.24	4.65	7.01	2.53	4.85
SIT	14.01	9.63	10.29	2.32	8.03	0.0	4.12	2.12	4.58	2.32	4.78
$J/s_{hj} \geq 0,\ ia,\ r_i \geq 0/\sum w_i C_i$											
DST	30.52	27.59	26.91	2.75	6.45	10.61	6.29	1.19	1.42	0.88	0.00
DPT	11.46	6.50	11.20	0.0	3.11	4.42	3.61	3.27	5.69	5.47	4.40
SIT	13.70	11.90	14.61	0.0	4.62	6.99	4.33	2.51	5.12	5.81	2.17
$J/s_{hj} \geq 0,\ ba\,(6, \ldots, 10),\ r_i \geq 0/C_{\max}$											
DST	14.80	18.03	17.17	6.08	3.90	2.91	2.84	0.0	0.79	0.99	3.17
DPT	15.42	7.18	10.00	5.08	8.76	0.0	5.81	5.25	7.91	1.25	6.09
SIT	10.56	5.84	9.19	3.04	5.59	0.0	1.61	2.86	3.73	1.55	2.17
$J/s_{hj} \geq 0,\ ba\,(6, \ldots, 10)/\sum w_i C_i$											
DST	26.82	21.41	23.96	0.0	3.52	4.92	3.46	1.47	4.51	2.33	4.38
DPT	11.65	7.40	9.00	0.0	3.63	4.66	2.86	2.00	3.24	4.24	4.60
SIT	14.75	11.78	14.87	0.0	2.47	4.19	3.46	1.79	2.89	4.55	4.00

be avoided. Note that the type of job availability does not seem to influence the choice of a suitable insertion order.

For our further tests we always use the insertion orders recommended above (*cf.* the underlined numbers in Tab. 1, which document the chosen insertion orders). This choice takes into account that we tried to avoid to have for each individual variant a different insertion order. As it can already be seen from Tab. 1, the structure of the problem and also the objective function has some influence on a good variant for inserting the operations! Even a random insertion, which usually does not lead to the best results can sometimes be suitable if no sufficient information about the choice of a suitable order is available.

4.3. Insertion versa priority rules

After having fixed the concrete insertion order for the different problem types, we have compared the basic variant I_1 of the insertion algorithm with the three priority rules *ECT*, *FIFO* and *NINQ*. We considered both objective functions $C_{\max}$ and $\sum w_i C_i$ combined in each case with item availability and batch availability on the machines $\lfloor m/2 \rfloor + 1, \ldots, m$. For each of the six

combinations of n and m, we again randomly generated 20 problems with DST, 20 problems with DPT and 20 problems with SIT.

In Table 2 we give for each problem type the average percentage improvement of algorithms I_1, *FIFO* and *NINQ* over the reference algorithm *ECT, i.e.* we compared the average objective function value of *ECT* with the average objective function value of 20 problems of the considered type and measured the percentage differences between them. In the case of minimizing makespan, the priority rule *FIFO* yields worse results than *ECT* for most of the DST problems. Also *NINQ* works worse than *ECT* and even worse than *FIFO* for DST problems with item availability when minimizing the makespan. However, for the makespan problems with DPT and SIT both algorithms *FIFO* and *NINQ* are superior to *ECT*. In the case of minimizing the weighted sum of completion times, *ECT* works clearly better than the other applied priority rules.

Comparing the basic insertion variant with the priority rules, we see that the variant I_1 already outperforms all other priority rules for all problem types, independently of the objective function and the type of job availability. In particular, for DST problems with batch availability and minimizing the weighted sum of completion times, we have obtained, on average, nearly 25% improvement over the best priority rule *ECT*. Contrary to that observation, the smallest percentage improvements of I_1 over *ECT* have also been obtained for certain DST problems (see item availability with minimizing makespan). A similar result has been obtained for the permutation flow shop problem (*see* minimization of the sum of completion times in [32]). Generally spoken, one can see that the average percentage differences between the different algorithms are much larger for the problems with batch availability than those with item availability.

4.4. Different degree of batch availability

Up to now we considered only one variant of batch availability (approximately on half of the machines). To test, whether the degree of batch availability has some influence on the results, we applied various algorithms to problems with $n = 20$, $m = 10$ and a different number of machines with batch availability. The results are given in Table 3. As it can be seen, the degree of batch availability does not substantially influence the results (even with 2 machines with batch availability the results are closer to those problems with 5 machines with batch availability than those with item availability). So further, we continue with considering problems with

TABLE 2

A comparison with priority rules for problems $J/s_{hj} \geq 0,\ x,\ r_i \geq 0/F$.

n, m	Type	I_1	*FIFO*	*NINQ*	I_1	*FIFO*	*NINQ*
		$x = ia,\ F = C_{\max}$			$x = ia,\ F = \sum w_i C_i$		
10,5	DST	22.91	0.87	−6.54	27.56	−8.00	−20.43
	DPT	9.42	0.0	3.27	10.52	−2.59	−7.95
	SIT	12.73	−2.07	0.61	14.46	−10.02	−16.14
10,10	DST	4.77	−3.91	−12.59	11.95	−13.73	−25.22
	DPT	11.18	4.42	3.45	8.56	−1.67	−2.40
	SIT	10.62	0.48	3.78	9.71	−5.41	−6.69
15,5	DST	6.66	−9.30	−19.45	17.22	−23.88	−40.70
	DPT	10.86	6.77	6.67	11.54	−2.70	−14.56
	SIT	7.54	−1.34	−1.56	11.75	−13.16	−24.41
15,10	DST	3.68	−7.47	−9.69	13.84	−23.28	−26.87
	DPT	8.25	−1.76	0.68	8.58	−4.73	−7.47
	SIT	7.25	0.88	0.53	13.51	−8.58	−15.02
20,5	DST	6.88	−11.87	−28.43	17.30	−29.30	−40.91
	DPT	14.00	7.15	5.72	12.75	−3.59	−19.15
	SIT	8.81	0.48	−1.21	12.69	−30.03	−31.87
20,10	DST	6.38	−9.03	−10.10	11.17	−30.03	−31.87
	DPT	13.80	7.49	7.60	12.78	−1.99	−6.32
	SIT	4.72	−0.82	1.32	11.42	−10.24	−14.82
		$x = ba\left(\lfloor \frac{m}{2} \rfloor + 1, \ldots, m\right),\ F = C_{\max}$			$x = ba\left(\lfloor \frac{m}{2} \rfloor + 1, \ldots, m\right),\ F = \sum w_i C_i$		
10,5	DST	20.38	0.01	0.01	14.27	0.01	−4.97
	DPT	11.49	3.72	4.60	25.33	−7.63	−12.39
	SIT	12.47	2.48	4.68	15.58	−5.32	−8.91
10,10	DST	14.77	0.06	2.45	20.27	−7.45	−9.61
	DPT	16.26	6.80	4.69	10.44	−0.66	−2.00
	SIT	11.97	5.80	4.76	11.16	−3.21	−4.43
15,5	DST	26.75	−1.66	−0.55	29.88	−16.85	−18.80
	DPT	9.44	4.87	4.91	12.46	−3.59	−14.80
	SIT	10.45	0.60	0.20	16.21	−10.82	−18.98
15,10	DST	12.27	−1.88	2.49	23.52	−14.91	−13.43
	DPT	11.80	2.15	3.13	10.54	−3.41	−6.87
	SIT	9.86	1.16	3.03	18.07	−7.43	−10.37
20,5	DST	26.55	−1.97	0.92	28.54	−19.32	−23.11
	DPT	11.51	7.03	5.09	12.22	−4.68	−19.77
	SIT	12.76	1.10	1.54	15.57	−11.12	−24.04
20,10	DST	15.01	2.98	0.46	23.67	−17.19	−12.10
	DPT	12.87	7.74	8.33	13.37	−1.64	−6.26
	SIT	9.43	1.29	5.16	15.59	−7.76	−10.65

TABLE 3
The influence of the degree of batch availability.

x	Type	I_1	*FIFO*	*NINQ*	I_1	*FIFO*	*NINQ*
		$J/s_{hj} \geq 0,\ x,\ r_i \geq 0/C_{\max}$			$J/s_{hj} \geq 0,\ x,\ r_i \geq 0/\sum w_i C_i$		
$x = ia$	DST	6.38	−9.03	−10.10	11.17	−30.03	−31.87
	DPT	13.80	7.49	7.60	12.78	−1.99	−6.32
	SIT	4.72	−0.82	1.32	11.42	−10.24	−14.82
$x = ba\,(1, \ldots, 10)$	DST	10.09	−5.30	−0.82	23.24	−16.84	−10.75
	DPT	13.16	5.68	6.62	13.70	−1.94	−6.53
	SIT	9.01	2.54	6.72	15.70	−7.45	−10.83
$x = ba\,(4, \ldots, 10)$	DST	14.11	2.26	−0.33	23.87	−16.88	−13.34
	DPT	14.21	8.70	9.36	13.47	−1.52	−6.39
	SIT	9.43	−1.29	5.16	15.60	−7.76	−10.65
$x = ba\,(6, \ldots, 10)$	DST	15.01	−2.98	−0.46	23.67	−17.19	−12.10
	DPT	12.87	7.71	8.33	13.37	−1.64	−6.26
	SIT	9.43	−1.29	5.16	15.60	−7.76	−10.65
$x = ba\,(8, \ldots, 10)$	DST	14.27	−3.50	−1.35	24.26	−17.43	−15.09
	DPT	14.52	−1.64	−6.26	12.72	−1.77	−5.66
	SIT	9.14	1.70	4.35	14.56	−7.51	−10.99

batch availability, where approximately half of the machines have this type of job availability.

4.5. Refinements of the insertion algorithm

Next, we tested the basic variant I_1 of the insertion algorithm against the proposed refinements beam search and parallel insertions. To this end, we concentrated on DST problems (which have mostly been considered in connection with such types of problems, *see* [9, 34]) and generated for each combination of n and m again 20 problems. The results are given in the Tables 4 and 5. In Table 4 we present the average percentage improvements obtained with beam search (beamwidths 2 and 3) and parallel insertions (2 and 3 insertions in parallel) with respect to the basic variant I_1 and in Table 5 we state how often each of the applied 5 variants has obtained the best value among the problems of each type. From Table 4 it can be seen that the largest percentage improvements over I_1 have been obtained for the problems with minimizing the weighted sum of completion times when $k = 3$ operations have been inserted in parallel (up to nearly 8%). Surprisingly, the largest average percentage improvements have been obtained for the problems with

TABLE 4
Results with beam search and parallel insertions for problems $J/s_{hj} \geq 0, x, r_i \geq 0/F$ with DST.

n, m	I_2	I_3	$I2$	$I3$	I_2	I_3	$I2$	$I3$
	$x = ia, F = C_{\max}$				$x = ia, F = \sum w_i C_i$			
10,5	2.55	2.22	-0.20	1.55	1.88	3.15	2.82	6.15
10,10	0.75	-0.30	0.90	1.57	2.35	3.92	2.30	3.94
15,5	1.23	2.05	2.30	2.94	1.69	3.62	3.03	4.54
15,10	0.34	0.58	-0.82	0.58	3.56	3.85	0.96	4.36
20,5	0.00	0.52	-0.57	1.70	2.34	2.92	5.87	7.92
20,10	0.27	0.20	-0.24	1.26	1.81	3.32	1.19	5.48
	$x = ba\left(\lfloor \frac{m}{2} \rfloor + 1, ..., m\right), F = C_{\max}$				$x = ba\left(\lfloor \frac{m}{2} \rfloor + 1, ..., m\right), F = \sum w_i C_i$			
10,5	2.06	2.32	3.35	5.24	1.25	0.51	3.20	0.82
10,10	0.53	1.33	-1.31	4.12	2.00	2.30	0.87	2.20
15,5	2.41	3.95	1.32	5.39	4.94	6.14	2.88	6.65
15,10	5.51	4.91	2.58	4.39	0.48	1.69	1.88	1.88
20,5	1.48	2.90	1.80	2.77	2.35	4.78	3.96	6.60
20,10	0.58	1.37	0.36	0.79	-0.16	1.71	2.29	2.93

TABLE 5
The number of best values obtained with different insertion variants for problems $J/s_{hj} \geq 0, x, r_i \geq 0/F$ with DST.

n, m	I_1	I_2	I_3	$I2$	$I3$	I_1	I_2	I_3	$I2$	$I3$
	$x = ia, F = C_{\max}$					$x = ia, F = \sum w_i C_i$				
10,5	8	11	10	5	5	1	2	4	3	11
10,10	6	7	4	6	7	2	2	6	5	7
15,5	0	1	4	8	8	0	2	5	4	9
15,10	4	4	5	3	9	1	6	7	1	8
20,5	2	3	5	3	8	0	3	3	3	11
20,10	2	5	6	5	5	1	3	5	3	10
	$x = ba\left(\lfloor \frac{m}{2} \rfloor + 1, ..., m\right), F = C_{\max}$					$x = ba\left(\lfloor \frac{m}{2} \rfloor + 1, ..., m\right), F = \sum w_i C_i$				
10,5	5	5	6	8	9	3	3	7	7	6
10,10	3	2	2	4	12	5	4	8	3	5
15,5	2	2	3	3	10	0	1	6	2	11
15,10	3	8	5	3	6	2	2	5	3	9
20,5	2	4	9	5	4	0	3	6	2	9
20,10	3	3	7	3	5	2	5	6	5	5

5 machines. As Table 6 shows, the variant $I3$ also yields most often the best value among these 5 constructive variants.

Although we primarily concentrated on the comparison of the quality of the obtained solutions, we add some remarks on the computational times. For

TABLE 6
Results for problems $J/s_{hj} \geq 0,\ x,\ r_i \geq 0/F$ with special technological routes.

n, m	Type	TR	*FIFO*	I_1	I_3	*I3*	*FIFO*	I_1	I_3	*I3*
			$x = ia,\ F = C_{\max}$				$x = ia,\ F = \sum w_i C_i$			
10,5	DST	A	-5.92	-7.36	-4.55	3.14	-8.20	6.29	7.96	13.13
		B	-2.23	-5.43	-0.64	3.62	-3.20	7.12	11.12	8.21
		C	-15.35	-4.21	3.61	0.15	-16.38	8.06	10.72	10.69
		D	-4.13	5.96	10.76	6.15	-9.24	4.96	13.42	14.99
10,10	DST	A	-8.50	-10.88	-6.46	-4.50	-12.62	-6.39	-2.04	5.10
		B	-8.88	-3.20	-0.90	-0.58	-12.18	-4.48	1.52	11.28
		C	-21.47	-6.79	1.34	4.10	-21.89	5.48	8.47	14.75
		D	-21.08	-5.46	0.30	-3.94	-28.02	2.70	7.89	10.74
20,10	DST	A	-14.10	-17.16	-11.40	-8.56	-17.76	-14.04	-9.84	-3.17
		B	-22.18	-11.49	-2.92	-3.42	-25.74	-0.21	0.97	11.15
		C	-36.42	-15.77	-6.85	-8.92	-43.46	-15.16	-6.72	6.71
		D	-24.91	-2.57	5.47	4.60	-32.77	1.52	7.00	13.79
10,10	DPT	A	0.65	5.87	7.88	6.62	-4.32	-1.97	0.55	1.86
		B	-1.32	3.67	5.00	4.37	-5.33	1.60	2.91	5.32
		C	-1.05	2.01	4.48	2.50	-4.62	0.80	2.85	3.53
		D	0.37	1.30	3.16	4.23	-5.33	3.88	6.80	5.73
10,10	SIT	A	-2.12	1.76	3.58	3.35	-8.20	-5.79	-2.15	0.91
		B	-7.33	-0.06	2.44	0.85	-11.80	-6.94	0.70	-2.79
		C	-6.35	-3.68	-0.58	-5.30	-8.14	-1.94	5.08	5.04
		D	-5.36	-2.62	0.90	-1.34	-10.08	2.78	9.49	4.62
			$x = ba\left(\lfloor \frac{m}{2} \rfloor + 1, \ldots, m\right),$ $F = C_{\max}$				$x = ba\left(\lfloor \frac{m}{2} \rfloor + 1, \ldots, m\right),$ $F = \sum w_i C_i$			
10,5	DST	A	-4.68	21.41	27.46	24.79	-5.72	21.51	22.62	27.53
		B	-7.32	19.18	25.99	23.85	-5.09	16.82	22.03	25.82
		C	-5.60	19.32	22.47	20.89	-8.46	11.91	21.78	20.32
		D	0.00	23.77	25.89	28.44	-6.16	17.06	21.43	21.56
10,10	DST	A	-3.43	10.01	11.53	13.70	-7.22	6.69	8.78	13.30
		B	-8.79	7.18	13.67	9.65	-8.70	8.97	10.01	18.02
		C	-10.99	7.59	14.96	10.63	-11.73	7.50	12.80	11.97
		D	-7.35	12.19	17.12	13.21	-9.75	11.41	15.80	19.61
20,10	DST	A	-4.94	10.77	13.58	14.80	-12.69	8.70	9.86	13.46
		B	-8.28	12.67	16.90	17.66	-17.54	4.88	9.54	13.25
		C	-17.45	4.00	12.01	10.86	-18.44	-0.63	5.59	15.15
		D	-10.38	12.93	18.99	16.20	-15.73	13.99	15.48	21.99
10,10	DPT	A	7.71	9.85	11.84	9.28	0.20	2.85	5.84	6.84
		B	0.03	3.59	4.55	5.25	-4.17	2.17	4.55	6.57
		C	1.18	2.03	4.39	2.46	-3.24	2.14	2.11	6.16
		D	2.17	1.95	3.57	2.79	-5.46	3.42	6.99	6.93
10,10	SIT	A	-2.62	2.03	4.43	2.88	-6.99	-6.14	0.54	0.61
		B	-2.49	3.13	6.07	3.63	-3.49	0.29	7.88	8.54
		C	-5.88	1.73	5.04	3.65	-7.91	1.05	7.86	6.66
		D	-2.38	2.48	5.05	3.92	-8.20	5.23	11.02	7.41

the large problems ($n = 20$, $m = 10$), the average running time per problem was 6 s for I_1 and 18 s for I_3 when minimizing $\sum w_i C_i$. When minimizing $C_{\max}$, where the objective function value after inserting an operation (i, j) can be obtained by considering only the old value and $r_{ij} + t_{ij} + z_{ij}$, the average running time per problem was 1.5 s for I_1 and approximately 4 s for I_3. For $I3$ the average running time was about 2 min for both objective functions. For the small problems ($n = 10$, $m = 5$), the running times were less than 2 s for I_3 and less than 1 s for I_1. For $I\,3$ the average running time was about 8 s. All applied priority rules run in less than 1 s.

4.6. Special technological routes

Next, we tested the two priority rules *ECT* and *FIFO* against three variants I_1, I_3 and $I\,3$ of the insertion algorithm for special technological routes:

TR A: Choose the same technological route for all jobs of a group;

TR B: Choose an individual technological route for each job but each job has to be processed first on the machines $1, ..., \lfloor m/2 \rfloor$ and then on the machines $\lfloor m/2 \rfloor + 1, ..., m$, where the sequence of machines within each of both sets of machines is randomly determined.

TR C: Choose randomly one "basic" technological route from which the concrete individual technological routes for each job are determined by performing exactly m inversions in the permutation of the machines in the basic technological route (*i.e.* m interchanges of adjacent machines are performed in the initial permutation).

TR D: Apply the same procedure as in TR C but for each group separately, *i.e.* choose one basic technological route for each group and determine then the individual technological routes of each job of the corresponding group from this basic route as in TR C.

The average percentage improvements of the individual variants over the reference algorithm *ECT* are given in Table 6. We included into our tests three combinations of n and m with DST and, to test whether different conclusions can be drawn for different types of processing and setup times, we selected one combination ($n = m = 10$) in connection with DPT and SIT problems. We give here all results in detail since various new tendencies can be observed in comparison with the previously obtained results.

First, in almost all cases *ECT* now outperforms the other priority rule *FIFO* (with a few exceptions for DPT problems with makespan minimization). Comparing *ECT* with the different insertion variants, we can observe that for

the problems with batch availability the insertion variants are much better than *ECT* independently of the objective function. Moreover, the refinements of the basic insertion variant often leads to substantially larger percentage improvements over I_1 than for the randomly generated problems. This is especially true for the parallel insertion variant $I3$ (*see* for instance TR B or TR C when minimizing the weighted sum of completion times). A different behaviour can be observed for the problems with item availability. Here in some cases the *ECT* rule works surprisingly good. Whereas in the cases of minimizing the weighted sum of completion times, the refinements of the basic insertion variant (in particular $I3$) clearly outperform *ECT*, for large DST problems even the refinements of the insertion algorithms do not beat *ECT*. However, in particular for minimizing $\sum w_i C_i$, the parallel insertion variant yields substantial improvements over the basic variant. One conclusion of the results in Table VI is that possibly for special technological routes another insertion order could be preferable than in the case of usual randomly generated problems but $I3$ is able to "compensate" this difficulty and it yields in most cases clearly the best results among the constructive algorithms.

4.7. Nonregular criterion

Finally, we considered job shop problems with family setup times and a nonregular criterion, namely, we consider the minimization of $\sum w_i (C_i - d_i)^2$ as a special case of earliness and tardiness penalties. In our tests we assumed correlated release and due dates. By means of the generated release dates we calculated due dates according to

$$d_i = r_i + y \cdot \sum_{j=1}^{m} t_{ij}, \qquad i = 1, 2, ..., n.$$

In the experiments we applied $y \in \{4, 6, 8, 10\}$. Since the objective function values vary in a rather large range for the considered problems, we present the average values in rounded thousands in Tables 7 and 8.

In Table 7 we give the results of Algorithm I_1 using different insertion orders for problems with 10 jobs and 10 machines. In initial tests we have found that it is often advantageous to reverse some of the insertion orders proposed above for regular criteria, since the schedules are constructed from the right. Therefore, we included the results for the orders a^* (nondecreasing job weights) and b^* (nonincreasing release dates) into the table. It can be seen that insertion order $3\,a^*$ is clearly the best one. However, the second

TABLE 7
A comparison of several insertion orders for a nonregular criterion.

io	2a	2b	2c	2d	2a*	2b*	3a	3b	3c	3d	3a*	3b*
						$y = 4$						
DST	12,029	12,709	7,453	10,719	8,503	4,317	4,972	7,698	6,643	5,379	7,721	5,564
DPT	553	1,099	2,470	3,392	1,076	2,171	1,026	437	2,755	624	973	1,524
SIT	1,159	310	813	913	337	2,137	411	316	1,453	323	855	504
						$y = 6$						
DST	717	1,254	1,781	225	755	674	416	434	908	451	406	1,115
DPT	3,701	2,336	3,099	6,508	1,514	3,647	1,302	3,764	4,668	2,117	927	2,637
SIT	206	958	1,464	1,833	571	1,123	411	316	1,453	323	380	1,000
						$y = 8$						
DST	747	523	264	1,028	379	946	1,352	180	392	460	176	408
DPT	5,544	2,641	2,706	8,624	3,550	5,283	2,558	1,766	1,348	1,332	819	1,317
SIT	1,397	678	1,406	2,191	969	1,622	1,336	819	2,390	962	374	409
						$y = 10$						
DST	723	1,094	384	1,534	339	291	1,430	221	273	173	204	322
DPT	9,861	4,476	5,855	1,843	1,012	1,930	4,963	3,472	3,181	381	814	881
SIT	2,653	1,016	1,532	2,959	486	1,193	1,297	412	1,020	550	275	714

best order is the random groupwise insertion (order 3d). This also indicates why sometimes even the opposite insertion of the jobs can be preferable in certain situations.

Then we tested some insertion variants against priority rules. The results are presented in Table 8. In all insertion algorithms we applied order $3a^*$. In addition to the basic variant I_1, we applied unforced shifts after each inserted job (*i.e.* n times during the algorithm; variant I_1/n) and unforced shifts one time to all jobs after a complete schedule has been obtained ($I_1/1$). Among the priority rules, we give the results for the three best rules with applied unforced shifts to all jobs in the complete schedule (*ECT*/1, *URG*/1 and *SLACK*/1). We present the results for problems with 10 jobs and 5 as well as 10 machines.

Although the objective function values for the priority rules vary in a large range, rule *URG* turns out to be the best one. However, only in 3 of 24 problem types considered in Table 8, the best average value has been obtained by a priority rule. In most cases the favourite is algorithm $I_1/1$, which yields partially substantially better objective function values than all other variants. It can be seen from Table 8 that the consideration of unforced

shifts for all jobs at the end is usually prefererable than trying unforced shifts after each inserted job.

TABLE 8
A comparison with priority rules for a nonregular criterion.

		ECT/1	*URG*/1	*SLACK*/1	I_1	$I_1/1$	I_1/n
				$n = 10,\ m = 5$			
$y = 4$	DST	6,586	14,335	12,305	11,287	11,287	11,617
	DPT	5,002	558	2,338	827	729	834
	SIT	2,706	811	1,927	747	746	911
$y = 6$	DST	4,848	4,749	4,850	4,131	4,123	3,835
	DPT	8,651	2,154	1,767	485	375	452
	SIT	3,858	388	1,198	367	295	293
$y = 8$	DST	5,801	1,632	2,682	943	926	992
	DPT	261,441	134,884	111,805	415	277	403
	SIT	5,510	356	883	350	233	356
$y = 10$	DST	7,167	1,077	2,654	644	596	1,055
	DPT	542,034	556,502	576,113	558	344	555
	SIT	8,534	1,220	800	252	155	191
				$n = 10,\ m = 10$			
$y = 4$	DST	3,598	7,932	7,373	8,503	8,503	7,500
	DPT	6,679	658	1,030	1,076	1,065	1,077
	SIT	2,619	711	1,220	337	329	377
$y = 6$	DST	4,791	23,734	29,216	755	749	870
	DPT	864,317	814,162	857,407	1,514	924	1,490
	SIT	4,362	518	669	571	299	564
$y = 8$	DST	6,361	1,810	2,643	379	306	353
	DPT	2,205,109	1,870,364	1,860,964	3,550	1,844	3,598
	SIT	175,202	962	1,429	969	540	957
$y = 10$	DST	8,392	1,374	2,317	338	255	315
	DPT	1,663,658	1,604,671	1,648,409	1,012	588	1,002
	SIT	610,297	509,206	535,353	485	299	508

5. CONCLUDING REMARKS

In this paper we performed a computational study on insertion algorithms for job shop problems with family and batch setup times, different types of job availabilities and different objective functions. In addition, a comparison with some typical priority rules has been performed. The following conclusions can be drawn from our experiments.

1. Generally spoken, insertion algorithms usually clearly outperform priority rules. Although the computational expense is higher, insertion algorithms should be applied to get a heuristic solution of various scheduling problems instead of priority rules, which are often more sensitive with respect to the problem structure.

2. A suitable insertion order of the operations depends on the type of the considered problem. The objective function strongly influences the choice of a suitable order but also the concrete structure of a special problem. We recommend to insert successively all operations of a job. In the case of minimizing the makespan it is preferable to insert the jobs according to increasing "slack" times (*i.e.* "due date" minus release date minus required time for processing). When minimizing the weighted sum of completion times, it is recommendable to insert the jobs according to nonincreasing weights. For problems with dominant setup times, it is preferable to insert all jobs of the same group consecutively.

3. In connection with the considered regular criteria, the proposed refinements improved the results in particular for the "hard" problems. However, for some special situations (*cf.* Tab. 6) the performance of the chosen insertion variant is still unsatisfactory. Here additional investigations seem to be necessary. One possibility consists in applying a filtered variant of beam search in such cases.

4. For problems with a nonregular criterion, it turns out to be very difficult to determine a good heuristic constructive solution. Here the investigations are only in an initial stage. For the considered problem, the objective function values vary for the individual algorithms in a large range. Whereas the modified priority rules perform poor in general, the insertion algorithm with the selected best insertion order (nondecreasing weights) works substantially better. However, even the objective function values for the various insertion orders vary in a large range. In comparison with regular criteria we observed that it is sometimes advantageous to apply the opposite insertion orders (*cf.* orders 2a*, 2b*, 3a* and 3b*). It turned out to be preferable to insert the jobs groupwise according to the selected insertion order.

5. Job shop problems with setup times are very hard to solve (only problems with very small dimensions can be solved exactly). Therefore, besides constructive algorithms the development of iterative algorithms based on local search is necessary. In this connection the main problem is to design suitable neighbourhoods that are sufficiently flexible to handle the different situations even within the same problem type.

REFERENCES

1. J. Adams, E. Balas and D. Zawack, The Shifting Bottleneck Procedure for Job Shop Scheduling, *Management Sci.*, 1988, *34*, p. 391-401.
2. S. Albers and P. Brucker, The Complexity of One-Machine Batching Problems, *Discrete Appl. Math.*, 1993, *47*, p. 87-107.
3. J. D. Allison, Combining Petrov's Heuristic and the CDS Heuristic in Group Scheduling problems, *Computers Ind. Engng.*, 1990, *19*, p. 457-461.
4. D. Applegate and W. Cook, A Computational Study of The Job-Shop Scheduling Problem, *ORSA J. Computing*, 1991, *3*, p. 149-156.
5. K. R. Baker, Scheduling The Production of Components at a Common Facility, *IEE Transactions*, 1988, *20*, p. 32-35.
6. J. N. Blackstone, D. T. Philips and C. L. Hogg, A State-of-the-Art Survey of Manufacturing Job Shop Operations, *International Journal of Production Research*, 1982, *20*, p. 27-45.
7. H. Bräsel, T. Tautenhahn and F. Werner, Constructive Heuristic Algorithms for the Open Shop Problem, *Computing*, 1993, *51*, p. 95-110.
8. P. Brucker B. Jurisch and B. Sievers, A Branch and Bound Algorithm for the Job-Shop Problem, *Discrete Appl. Math.*, 1994, *49*, p. 107-127.
9. P. Brucker and O. Thiele, A Branch and Bound Method for the General-Shop Problem with Sequence-Dependent Setup-Times, *OR Spektrum*, 1996 (to appear).
10. J. Carlier and E. Pinson, An Algorithm for Solving the Job-Shop Problem, *Management Sci.*, 1989, *35*, p. 164-176.
11. G. Dobson, U. S. Kamarkar and J. L. Rummel, Batching to Minimize Flow Times on One Machine, *Management Sci.*, 1987, *33*, p. 784-799.
12. M. R. Garey R. Tarjan and G. Wilfong, One Machine Processor Scheduling with Symmetric Earliness and Tardiness Penalties, *Mathematics of Operations Research*, 1988, *13*, p. 330-348.
13. J. N. D. Gupta, Single Facility Scheduling with Multiple Job Classes, *European J. Oper. Res.*, 1988, *33*, p. 42-45.
14. W. S. Gere, Heuristics in Job Shop Scheduling, *Management Sci.*, 1996, *13*, p. 167-190.
15. R. E. Graham, E. L. Lawler and J. K. Lenstra, A. H. G. Rinnooy Kan, Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey, *Annals Discr. Math.*, 1979, *5*, p. 287-326.
16. R. Haupt, A Survey of Scheduling Rules, *Operations Research*, 1989, *25*, p. 45-61.
17. N. Ishii and J. T. Talavage, A Mixed Dispatching Rule Approach in FMS Scheduling, *International J. Flexible Manufacturing Systems*, 1994, p. 69-87.
18. U. Kleinau, Two-Machine Shop Scheduling Problems with Batch Processing, *Math. Comput. Modelling*, 1993, *17*, p. 55-63.
19. R. Kuik, M. Salomon and L. Van Wassenhove, Batching Decisions: Structure and Models, *European J. Oper. Res.*, 1994, *75*, p. 243-263.
20. P. J. M. Van Laarhoven E. H. L. Aarts and J. K. Lenstra, Job Shop Scheduling by Simulated Annealing, *Operations Res.*, 1992, *40*, p. 113-125.
21. J. K. Lenstra, A. H. G. Rinnooy Kan and P. Brucker, Complexity of Machine Scheduling Problems, *Annals of Discrete Mathematics*, 1977, *1*, p. 343-362.
22. B. L. MacCarthy and J. Liu, Adressing the Gap in Scheduling Research: A Review of Optimization and Heuristic Methods in Production Research, *International J. Production Research*, 1993, *31*, p. 59-79.

23. C. L. MONMA, C. N. POTTS, On the Complexity of Scheduling with Batch Setup Times, *Operations Research,* 1989, *37*, p. 788-803.
24. J. F. MUTH and G. L. THOMPSON, Industrial Scheduling, Prentice Hall, Englewood Cliffs, NJ, 1963.
25. N. NADDEF and C. SANTOS, One-Pass Batching Algorithms for the One-Machine Problem, *Discrete Appl. Math.,* 1988, *21*, p. 133-145.
26. M. NAWAZ, E. E. ENSCORE and I. HAM, A Heuristic Algorithm for the m-Machine, n-Job Flow Shop Sequencing Problem, *OMEGA*, 1983, *11*, p. 91-95.
27. E. NOWICKI and C. SMUTNICKI, A Fast Taboo Search Algorithm for the Job Shop Problem, TU Wroclav, Preprint 8/93, 1993.
28. P. S. OW and T. E. MORTON, The Single Machine Early/Tardy Problem, *Management Sci.,* 1989, *35*, p. 177-191.
29. E. PESCH, Machine Learning by Schedule Decomposition, University of Limburg, 1993.
30. S. S. PANWALKAR and W. ISKANDER, A Survey of Scheduling Rules, *Operations Res.,* 1977, *25*, p. 25-61.
31. C. SANTOS and M. MAGAZINE, Batching in Single Operation Manufacturing Systems, *Operations Research Letters,* 1985, *4*, p. 99-103.
32. Y. N. SOTSKOV, T. TAUTENHAHN and F. WERNER, Heuristics for Permutation Flow Shop Scheduling with Batch Setup Times, *OR Spektrum*, 1996, *18*, p. 67-80.
33. E. TAILLARD, Benchmarks for Basic Scheduling Problems, *European J. Oper. Res.*, 1993, *64*, p. 278-285.
34. A. J. VAKHARIA and Y.-L. CHANG, A Simulated Annealing Approach to Scheduling a Manufacturing Cell, *Naval Res. Log. Quart.,* 1990, *37*, p. 559-577.
35. L. Van WASSENHOVE and C. N. POTTS, Integrating Scheduling with Batching and Lot Sizing: A Review of Algorithms and Complexity, *Journal. Oper. Res. Soc.,* 1992, *43*, p. 395-406.
36. S. WEBSTER and K. R. BAKER, Scheduling Groups of Jobs on a Single Machine, *Operations Research,* 1995, *4*, p. 692-703.
37. F. WERNER and A. WINKLER, Insertion Techniques for the Heuristic Solution of the Job Shop Problem, *Discrete Appl. Math.,* 1995, *58*, p. 191-211.