Scheduling a Flow Shop with Simultaneously Loaded Stations: A Real World Application

Frank Herrmann

OTH Regensburg, University of Applied Sciences Regensburg, Innovation and Competence Centre for Production Logistics and Factory Planning, Regensburg 93025, Germany

Abstract: Companies use special designed flow shops, in order to satisfy specific demands. Products need to be transported (from one station to the next station) by a crane and the way of working of this crane excludes the intermediate storage (of a work piece). In addition, the way of working restricts the set of feasible schedules even more than the no-buffer restriction discussed in the literature in the case of limited storage. Since this scheduling problem is integrated in the usual hierarchical planning, the tardiness is minimised. A linear optimisation model is presented to provide a formal description of this NP-hard problem. It is also used to explain the performance of priority rule based heuristic solutions on small test problems. In detail, priority rules as well as a priority rule based branch and bound procedure are analysed; priority rules are regarded, because a priority rule is still the standard procedure for on-line scheduling in industrial practise. Out of successful priority rules in the literature the best one is identified by an extensive simulative investigation. An improved look-ahead is realised by a restricted search over all possible schedules.

Key words: Scheduling, real world flow shop with restrictions, no buffer and no wait, minimising tardiness, optimisation model, priority rules, priority rule based branch and bound procedure.

1. Introduction

Specific products are produced by special machines which are often grouped in a flow shop. They have to produce small batches with short response times, so scheduling algorithms are needed to ensure that under the constraint of a high average load of the flow shop, the due dates of the production orders are met. Nowadays, such special designed flow shops often have technological restrictions, which complicate the scheduling. For example in cell manufacturing, buffer could be non-existent due to limited space and storage facilities. Therefore, in recent years, a considerable amount of research for no-buffer (blocking) scheduling problems and for no-wait scheduling problems are published. But real world problems have often more restrictions for example a limited amount of production aids.

Often these production systems deliver products for other systems as well. Due to the hierarchical planning developed by Hax et al. [11], which is implemented in ERP (enterprise resource planning) system [14] and in a comparable way at the authors institute in the commercial simulation tool “Plant Simulation”, the local completion times in one production system in many cases determine the earliest possible starting times in another production system. Therefore, the delay of the operations in a production system has an impact on the effectivity of this coordination process. This impact is much more severely if a significant fraction of the jobs is finished with large delays than if almost all jobs are finished with an approximately equal delay [14]. Consequently, the width of the tardiness distribution is very important and not only the average tardiness.

This paper bases on the technological restrictions in a real word problem which is explained in Section 2 and it can be seen as a test problem for a class of scheduling problems. A mathematical model is described in Section 3 and a literature review is given in Section 4. In order to ensure on-line scheduling,
priority rules as well as a restricted search over all possible schedules are used in Section 5 and computational results are presented in Section 6. Some conclusions are given in Section 7.

2. A Real World Application

The problem is a modification of a partly automated production line at Fiedler Andritz in Regensburg to produce filters (baskets) with a lot size of 1. All filters have unified constructions. They differ in varying heights of the baskets and there exist different designs.

The production line consists of four stations which are shown in Fig. 1. Station 1 assembles six single batons which called consoles on an assembly ground plate to a skeleton of a filter basket. Baton profiles are assembled into the provided slots of the filter basket skeletons. At the plunge station a wire coil is contrived in the device of a lining machine. The lining machine straightens the wire and inserts batons into the slots. To ensure stability, the span station installs span kernels in the case of outflow filter baskets and span belts in the case of inflow filter baskets. Then, the filter basket is lifted from the assembly ground plate and is transported to the welding station, at which the baton profiles are welded on the filter basket skeletons. The accomplished filter basket leaves the production line. Prior to this, the span medium is removed. An overhead travelling crane lifts a filter basket out of a station, transports it to the next station and inserts it directly in this station. This is just possible if this station is free. Therefore, there is no buffer in the production line. Due to other operational issues the crane can just be moved if all stations are inactive. Since an operation cannot be interrupted, the transport has to be performed after the completion of all operations on the stations in the flow shop. Due to further operational issues this restriction has to be applied also for the first and the last station; note, that the crane loads S1 and unloads S4 as well. In summary, all stations are loaded and unloaded with filters during a common process and this process starts with the last station S4, followed by station S3, S2, until station S1 is reached. It is allowed that a station is empty; then this station is skipped (may be partially) in this process.

There are 10 filter types. They differ in the height of the basket and there processing times are listed in Table 1. One half of the parts are inflow-filters and the other half are outflow-filters, so that two filter types have the same processing times on station S1, S2 and S4, but slightly different times on station S3.

Produced filters are stored before they are assembled in other products or sold to customers directly. So, all jobs are released at the beginning of a period; in accordance with the reality at the company site one

![Fig. 1  Structure of the production line.](image)

| Height | Station 1 | Station 2 | Station 3 by IF | Station 3 by OF | Station 4 |
|--------|-----------|-----------|----------------|----------------|-----------|
| 300 mm | 100.5 min | 50 min    | 51 min         | 48.5 min       | 9 min     |
| 800 mm | 97 min    | 148.5 min | 99.5 min       | 100.5 min      | 101.5 min |
| 1,200 mm | 151 min  | 199 min   | 103 min        | 101.5 min      | 148 min   |
| 1,500 mm | 202.5 min | 251.5 min | 198.5 min      | 200 min        | 202 min   |
| 1,800 mm | 252 min  | 300.5 min | 250 min        | 249 min        | 299 min   |
period consists of seven days with three shifts about eight hours. Sequences of jobs of filter types with lot size 1 are randomly generated for each period t by a generating algorithm in Ref. [5], it is in accordance with the proceeding in Ref. [37] and which proceeds in principle as follows: An additional filter type F, released in period t, consumes capacity on each station during the time between now and his due date on each station; the calculation for the capacity just uses the net processing time and does not regard the dependencies between the jobs (released so far). F is accepted as long as this consumed capacity is below a maximal load level on each station, else it is skipped to the next period. A maximal load level is an (intended) average load \( L_0(S) \) plus \( 0, -30\% \) and \( +30\% \) of \( L_0(S) \). Over the first 5, 10, 15 etc. consecutive periods, the load level variations average to zero.

At the company site there are large numbers of periods with a low number of late jobs and large numbers of periods with a high (or even higher) number of late jobs. Due dates are determined so that scheduling with the FIFO (first in first out) rule causes a specific percentage of late jobs. The company confirmed that job sequences with 30%, 50%, 70% and 85% of late jobs by scheduling with the FIFO rule (called time pressure) are comparable to the ones which occurred in the real operation. Due to the result of the generating algorithm, the difference between the due dates and the release dates is between 14 and 30 days. The company confirmed that such large intervals for processing jobs occurred in the real application.

The duration of loading (or unloading) a station is negligible compared to the duration of the operation as well as it is independent of the allocation (or loading) of the other stations and it is included in the duration of the operation.

The general scheduling problem consists of M stations and a pool of N jobs, which may change at any time, with known earliest possible starting times respectively release dates \( a_i \) \((1 \leq i \leq N)\) and due dates \( f_i \) \((1 \leq i \leq N)\) as well as duration \( t_{i,j} \) of operation \((o_{ij})\, j(1 \leq j \leq M)\) of job \((1 \leq i \leq N)\), which is worked on station \( j \). Two performance criteria average tardiness \( T_{\text{Mean}} \) and root mean square of tardiness \( T_{\text{RMS}} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} T_i^2} \) ( \( T_{\text{RMS}} \) is similar to standard deviation of the tardiness but easier to calculate) are primarily analysed.

The time between two consecutive executions of the load process is determined by the maximum of the duration of the operations on the stations in the flow shop; this is called cycle time. This “load”-restriction, the no-buffer condition and the capacity of the stations are the main restrictions. Set-up times are relatively small compared to operation times and they are included in operation times. It may be noted that for some years scheduling problems with set-up times or costs are discussed frequently, for a relative recent surveys [1].

The no-buffer condition means a relaxation of the scheduling problem with the above “load”-restriction. A good survey of scheduling problems with the no-buffer condition was given by Hall and Sriskantharajah [12]. It is proved that this problem is NP-hard in the strong sense for more than two stations. Thus, the problem class with this real world application as example is NP-hard in the strong sense as well.

3. A Linear Optimisation Model

There are two main approaches in Ref. [38]: sequence-base models (a sequence of jobs is determined) and discrete time models (the discrete periods, in which the jobs are completed, are determined). In many cases, the modelling of restrictions, especially for a limited storage capacity, requires for sequence-base models more complicated constraints than for discrete time models. On the other
hand, sequence-base models have normally shorter run times. As can be seen below, for the class of problems regarded in this paper restrictions can be formulated by quite simple constraints, so that a sequence-base model seems to be the best choice. Nevertheless, in many papers, for example in Ref. [9], an integer model with discrete time periods is used. If general precedence constraints occur, then flow shop problems are modelled by the well-known RCPSP (resource constrained scheduling problem), sometimes referred to as a project shop [18]; an example is in Ref. [42].

Already the no-buffer restriction means that all feasible schedules are a permutation of the N jobs of the scheduling problem.

Permutation flow-shops can be modelled as an assignment problem with additional restrictions. So, a decision variable can be defined which assigns each job exactly to one position in a permutation:

\[ x_{i,p} = \begin{cases} 1, & \text{job } i \text{ is at position } p \text{ in the permutation} \\ 0, & \text{otherwise} \end{cases} \]

where, \( 1 \leq i, p \leq N \).

Each job \( i \) and each position \( p \) is handled once:

1. \[ \sum_{p=1}^{N} x_{i,p} = 1 \quad \text{for all} \quad 1 \leq i \leq N \]  
   \[ \sum_{i=1}^{N} x_{i,p} = 1 \quad \text{for all} \quad 1 \leq p \leq N \]

With FT_{p,j} = an upper bound of the realized finish time of the job at position \( p \) in a permutation on station \( j \), the capacity restriction is ensured by:

\[ \text{FT}_{p,j} + \sum_{j=1}^{N} t_{i,j} \cdot x_{i,p} \leq \text{FT}_{p,j} \]

where, \( 1 \leq p \leq N \), \( 1 \leq j \leq M \);

\[ \text{FT}_{p,0} \] is the release date of the job at position \( p \).

Restriction (5) ensures that the orders of jobs on each station are identical.

\[ \text{FT}_{p,-1,j} + \sum_{i=1}^{N} t_{i,j} \cdot x_{i,p} \leq \text{FT}_{p,j} \]

where, \( 1 \leq p \leq N \), \( 1 \leq j \leq M \)

\( \text{FT}_{0,j} \) is the availability of station \( j \) for all \( 1 \leq j \leq M \).

The “load” restriction is ensured by identical finish times of the jobs on the M stations:

\[ \text{FT}_{p,j-1,M} + \text{FT}_{p,j-1} = \text{FT}_{p,j,M-1} \]

where, \( 1 \leq p \leq N - M + 1 \), \( 1 \leq j \leq M - 1 \).

For the jobs at the end of a permutation, \( N - M + 1 < p \leq N \), not every station is occupied. Let \( p \) be \( N - M + 1 + k \), \( 1 \leq k \leq M - 2 \). Then just the stations from \( k + 1 \) until \( M \) can be regarded. Thus, the same restriction with other parameter limitations is needed:

\[ \text{FT}_{p,j-1,M} + \text{FT}_{p,j-1} = \text{FT}_{p,j,M-1} \]

where, \( N - M + 1 < p < N \), \( 1 \leq j \leq N - p \).

\[ x_{i,p} \in \{0,1\} \]

Restrictions (9), (10) ensure upper bounds for the variable tardiness \( T_p \).

\[ T_p \geq 0 \]

\[ T_p \geq \text{FT}_{p,M} - \sum_{i=1}^{N} t_{i} \cdot x_{i,p} \]

where, \( 1 \leq p \leq N \).

Then

\[ \text{Minimise} \left( \sum_{p=1}^{N} T_p \right) \]

minimises the tardiness.

An empty station is achieved by an artificial job \( i \) whose duration time on each station is zero and whose release date is less than the release dates of all normal jobs and \( i > N \), so that \( i \) is not regarded in the objective function.

4. Literature Review

Since the scheduling problem is NP hard (in the strong sense), exact methods like branch and bound approaches can just be applied to problems with a small number of stations. For example, Kim [15]
described a branch and bound algorithm for
minimising mean tardiness in two machine flow shops.
As shown in the literature to algorithms for scheduling
problems, see Ref. [2], the complexity depends on the
performance criteria and in this sense minimising
make span is the easiest problem; in Ref. [13] such
complexity relations between performance criteria are
described and proven. So, for example, the two
machine flow shop problem with minimising make
span is solved in polynomial time by the Johnson
algorithm but the two machine flow shop problem
with minimising mean tardiness is NP hard. This may
be the reason why the research on the (weighted)
tardiness objective is small compared to the extensive
research on minimising make span.

As shown earlier, the real application is close to the
class of no-buffer (blocking) scheduling problems.
Solutions for the no-buffer (blocking) scheduling
problems are presented in various papers already. In
Ref. [20] a schedule is extended by an unscheduled
job that leads to the minimum sum of blocking times
on machines which is called PF (profile fitting). Often
the starting point of an algorithm is the NEH
algorithm presented by Nawaz, Enscore, and Ham in
Ref. [22], since it is the best constructive heuristic to
minimize the make span in the flow shop with
blocking according to Refs. [7, 8, 17, 23]. Thus,
Ronconi [33] substituted the initial solution for the
enumeration procedure of the NEH algorithm by a
heuristic based on a make span property proven in Ref.
[35] as well as by the PF mentioned in Ref. [20] above.
Ronconi [34] used an elaborated lower bound to
realise a branch-and-bound algorithm which becomes
a heuristic since the CPU time of a run is limited. Also
for minimising make span [10] realised and analysed a
tabu search algorithm. As an alternative approach [43]
developed a discrete particle swarm optimisation
algorithm. In order to diversify the population, a
random perturbation to the velocity of each particle
according to a probability controlled by the diversity
of the current population is integrated. Again, based
on the NEH algorithm, Wang et al. [44] described a
harmony search algorithm. First the jobs (i.e. a
harmony vector) are ordered by their non-increasing
position value in the harmony vector, called largest
position value, to obtain a job permutation. A new
NEH heuristic is developed on the reasonable premise,
that the jobs with less total processing times should be
given higher priorities for the blocking flow shop
scheduling with make span criterion. This leads to an
initial solution with higher quality. With special
settings due to the mechanism of a harmony search
algorithm better results are achieved. Also Ribas et al.
[32] presented an improved NEH-based heuristic and
used this as the initial solution procedure for their
iterated greedy algorithm. A modified simulated
annealing algorithm with a local search procedure is
proposed by Wang et al. [45]. For this, an
approximate solution is generated using a priority rule
specific to the nature of the blocking and a variant of
the NEH-insert procedure. Again based on the PF
approach of Ref. [20], Pan and Wang [25] addressed
two simple constructive heuristics. Then, both
heuristics and the profile are combined with the NEH
heuristic to three improved constructive heuristics.
Their solutions are further improved by an
insertion-based local search method. The resulting
three composite heuristics are tested on the
well-known flow shop benchmark of Ref. [40] which
is used in nearly all of the pervious mentioned papers
as well [10, 23, 32-34, 43, 44].

To the best of my knowledge, just a few studies
investigate algorithms for the total tardiness objective
(for flow shops with blocking). Ronconi [35]
developed a lower bound which reduces the number of
nodes in a branch-and-bound algorithm significantly.
Ronconi [36] described several versions of a local
search. First, with the NEH algorithm they explore
specific characteristics of the problem. A more
comprehensive local search is developed by a GRASP
(greedy randomized adaptive search procedure) based
search heuristic.
5. Heuristic Solution with Priority Rules

The real application operates in dynamic environments where real time events like station failure, tool breakage, arrival of new jobs with high priority, changes of due dates etc. may turn a feasible schedule into an infeasible one. Ouelhadj and Petrovic [24] presented a review about such real time events. A priority rule is an ordering of the queues in front of the stations according to a criterion which is calculated from the information available about the respective operation or job. A well known example is: select the job with the EDD (earliest due date) rule. So, with priority rules (or dispatching rules) a station can be assigned by a job in a very short time. Since the users in the real application expect that such an assignment is done immediately if such a real time event occurs, the company decided to use priority rules; this assessment is customary practise in industry, so that priority rules are still analysed in many studies on scheduling as in this one, see Refs. [3, 4, 6, 14, 19, 30, 31, 39]. Solving the optimisation model to the real world application with CPLEX (as part of the ILOG software) on a 2.83 GHz PC shows that for around 20 jobs, the runtime is below two hours and already with 30 jobs, the runtime is normally two days or even significantly larger. Tests with a realistic number of jobs (details are in section computational results) need runtimes which are significantly over three weeks, so this is not an alternative, the priority rules in this investigation (see below) need runtimes around one milli-second. Compared with optimisation techniques, priority rules are easier to implement and to control at the shop floor level in real life situations (see Ref. [26]), it is expected that users can understand priority rules better and therefore they have more confidence in the resulting schedule.

Since each feasible schedule is a permutation, the sequence of jobs on the first station determines the permutation. As in the real world scheduling problem it cannot be assumed that even a perfect scheduling of the first station causes a good schedule for the entire flow shop. Using another station does not solve this issue. An exception is the case, in which there is one bottleneck. Operationally, bottleneck and non-bottleneck stations can be distinguished by the concept of station dominance, which is used in Ref. [21] to characterize efficient solvable flow shop problems. Such station dominance does not occur in the real world scheduling problem. Significant better results are achieved in studies by the author, if the flow shop is regarded as one station.

Due to the “load”-restriction the processing time of a job A on the flow shop depends on the other jobs processed on the flow shop at the same time. Therefore, its processing time can be significant larger than the sum of the processing times of its single operations \( t_\lambda \), called net processing time of job A. A realistic processing time for job A is achieved, if the processing on the flow shop is simulated with respect to the jobs processed on this flow shop at the same time. After four cycles \( \tau_1, \ldots, \tau_4 \), job A leaves the flow shop. So, these cycle times depend on the three jobs (X1, X2, and X3) on the flow shop, as A is assigned to station 1, and the three jobs (B1, B2, and B3) following A in the sequence (tail of A); this is illustrated in Fig. 2. So, \( \tau_1 + \tau_2 + \tau_3 + \tau_4 \) is the total processing time of A \( t_\lambda \). This processing time of job A is only correct, if the tail is structurally identical with the three jobs which follow job A in the final permutation. Since these are not known a deviation normally occurs between this calculated processing time and the processing time which will really occur if job A is chosen. Therefore different tails are analysed, see the section computational results.

In a preliminary analysis a large number of priority rules are analysed. For this, job shop problems (via routings) are generated randomly by a tool. These priority rules are applied on these job shop problems by simulation experiments described below—with small simulation horizons as well as large ones. These experiments show that for many job shop problems the priority rules which are precisely defined below...
Fig. 2 Calculation the processing time of a job A.

CR+SPT, ODD and SL/OPN are Pareto optimal to the average, the variance and the maximum tardiness, see Ref. [5].

For the class of problems regarded here, the slack for job \( i \) is defined by \( f_i - t - t_{i}\), where \( t \) is the current time and \( f_i \) is the due date of job \( i \). Then,

\[
\text{CR+SPT} = \begin{cases} 
    \frac{f_i - t}{t_{i}}, & f_i - t - t_{i} > 0 \\
    \frac{t_{i}}{t}, & f_i - t - t_{i} \leq 0
\end{cases}
\]

(shortest processing time) rule, ODD is identical with the EDD-rule for the class of problems regarded here, and

\[
\text{SL/OPN} = \frac{f_i - t - t_{i}}{M}
\]

(where a low value is always preferred). This explains why SL/OPN and CR+SPT are often used as benchmark; according to Ref. [29], other combinations deliver worse results for flow shop problems. In addition, newer rules are regarded. One is RR, which seeks to minimize both mean flow time and mean tardiness of jobs. Originally, see Ref. [29], it is defined for job shop problems. For the adaptation the remaining work content is \( t_{i} \) and the probable waiting time of the successor of this operation (in job \( i \)) at the next station, which seems to be less effective in flow shops for minimizing mean flow time due to Ref. [30], is integrated in the calculation of the cycle times. Thus,

\[
(f_i - t - t_{i}) \cdot e^{-\eta} + e^{\eta} \cdot \frac{t_{i}}{M}
\]

is the busy time and \( j \) is the idle time of the entire flow shop, is the priority index and the job with the lowest priority index is processed next. Using properties of an optimal solution to the single-station weighted tardiness scheduling problem, Rachamadugu and Morton [28] developed a weighted slack-based scheduling rule RM. According to the investigation of this rule in Ref. [28], it can be assumed that it delivers near optimal results to the one station problem. Ref. [27] shows that this rule can be used as an effective pruning device in enumerative methods. Modifications of this rule deliver very good results for flow shop and job shop problems with weighted tardiness criteria (see Ref. [41]). The rule was successfully adapted to RCPSP in Ref. [42]. The priority index for an operation \( j \) of a job \( i \) is:

\[
\frac{w_{i,j}}{t_{i,j}} \cdot e^{\frac{k}{s_{i,j}}},
\]

where \( s_{i,j} \) is the slack, \( w_{i,j} \) is a weight, \( k \) is an empirically determined “look-ahead” parameter and \( T \) is the average duration of all jobs processed at the same station. The job with the highest priority index is processed by the flow shop next. For
the problem regarded here, there is no weight, so that
\( w_i = 1 \) for each job \( i \), \( \bar{T} \) is \( \frac{1}{N} \sum_{i=1}^{N} t_{i} \) and the
slack is \( f_i - t - t_{i} \) as usual. Especially, due to the
relatively recent paper [42], it is motivated to integrate
resource costs \( \pi_i \), so that in total the priority index is:
\[
\frac{1}{\pi_i} \cdot e^{k(f_i-t-n_i)}. \]
As local processing time costing
\( \pi_i^l = t_{i} \) is used, called RM local, and as global
processing time \( \pi_i^g = \sum_{i \in U_i} t_{i} \), where \( U_i \) is the set
of unfinished jobs in the pool of orders, including job \( i \),
is used, called to RM global. In Ref. [16] additional
resource costs are regarded which do not fit to the
problem regarded here or are already integrated in
other parameters; e.g. a static bottleneck resource cost
is not applicable because it is assumed that each
station is temporary a bottleneck.

As an alternative to strict search over all possible
permutations is developed, all possible permutations
can be represented by a decision tree. Each edge or
branch corresponds to the assignment of a job on the
flow shop and each node represents a partial
permutation which is determined by the sequence of
edges that lead from the root to this node. From each
node there are many edges which represent all
possible next assignments of jobs to the flow shop. So,
the construction of the full tree corresponds to a full
enumeration of all possible permutations. With a
branch-and-bound algorithm a partial decision tree is
constructed which contains the optimal solution. The
tree is reduced by cutting off those branches which
will only lead to solutions which are worse than the
best found so far or can be excluded by a bound of the
cost functional at the optimum which can be estimated
from the solutions obtained so far. For minimising
tardiness, especially in dynamic situations, no
efficient bounds are known. The idea is to generate
branches which are preferred by priority rules and to
restrict the depth of the tree to ensure a real time
scheduling. In detail the algorithm performs the
following steps:

(1) Continuation
A partial permutation \( P \), which may be the empty
one, is given, and \( n \) jobs are not scheduled. Thus,
there are \( n \) possible edges. They are reduced by
disregarding each job \( A \) whose release date is more
than \( D \) time units larger than the actual time which is
the end of the last cycle to the execution of \( P \); i.e. \( A \)
would lead to an idling period of more than \( D \) time
units if \( A \) is scheduled on the first station next. All
these possible edges are sorted by one or more priority
rules. The best \( s \) edges are chosen, so that \( s \) new nodes
are constructed. This process is iterated until all
generated continuations have a certain length.

(2) Evaluation
The processing of each continuation on the flow
shop is simulated. The resulting schedules are
evaluated according to the criterion weighted loss of
slack as follows: Let \( t_{i}^0 \) be the slack of the job \( i \)
according to the initial schedule and \( t_{i}(n) \) be the
slack of the job \( i \) after the continuation number \( n \).
Then the cost of the continuation is:
\[
Q(n)=\sum_{i=1}^{N} e^{-t_{i}(n)}-e^{-t_{i}}. \]

(3) Selection
Of the \( p \) best continuations \( q \) (first) jobs are
scheduled. This gives \( p \) new partial permutations.
Then step 1 is repeated.

After the generation of a number of new partial
permutations, \( r \) (first) jobs of the best continuation
found so far are definitively scheduled.

To ensure a reasonable run time the parameter \( p \)
was set to 40 and \( q \) was set to the length of each
continuation. The three steps are executed twice, so
that 1,600 different continuations were constructed.
Therefore, this algorithm realises a look-ahead of (a
mixture of) priority rules, and is called B & B
(branch-and-bound) algorithm with heuristics.
6. Computational Results

A large number of preliminary studies with priority rules (many more than the ones mentioned above) show that the use of simulated processing times instead of net processing times improves the priority rules by around 16% (for both performance criteria $T_{\text{Mean}}$ and $T_{\text{RMS}}$). These experiments show that this is caused by a higher number of cycles whose operations have significantly different processing times. This effect is increased by an arbitrary mixture of small and large processing times for the operations in a routing, which is partially the case for the real world application and small test problems. This also explains why the SPT rule often prefers a job A versus another job B, although in the real production process the processing time of A is larger than the one of B. Thus, the rules often do not work as expected according to other investigations, this affects the SPT rule more than the other rules. Experiments show that parameter $k$ in both versions of the RM rule has a negligible impact. Then, the experiments are restricted on the Pareto optimal rules in earlier investigations (i.e. CR+SPT, ODD and SL/OPN); some sample simulations show that this is justified.

The B & B algorithm with heuristics is tested with the following parameter settings. To ensure long continuations a small $s$ is used; concrete an $s$ between 2 and 4 delivered the best results for the small test problems and the real world application. In the continuation step all priority rules, including combinations, mentioned above are used to determine the edges in order to continue a partial permutation. In most of the small test problems and the real world application SL/OPN turned out to be most effective; again the simulated processing time is used instead of the net processing time. The best results achieved by a rather small constant $c$, around 0.0001 in the evaluation step.

The priority rules mentioned above were tested on the real world application as well as on some small test problems which are generated and for which an optimal solution is calculated. To achieve acceptable run times for the optimal solutions just 20 jobs are generated.

In order to use problems, which are comparable to the real world problem, four stations are regarded. The routings are created from a set of so called basis routings which are stated in Table 2. The total net processing time due to these basis routings covers the same range as the ones in the real world application. In detail a routing is created from one basis routing $R$, as routing 3 for example. The processing time for a station $S$, as station 2 for example, is created by a normal distribution whose mean value is the processing of $S$ in $R$, also 200 minutes in the example, and the deviation is either 20% of the mean value (small deviation) or 75% of the mean value (large deviation); of course negative processing times are excluded in both cases. The pool of orders is too small to generate a part type sequence by the generating algorithm effectively. Instead, the part types are generated by a uniform distribution, and the following four basic scenarios for the order release are generated accidently:

- in 5 successive periods 4 orders are realised;
- in 4 successive periods 5 orders are realised;
- in 2 successive periods 10 orders are realised;
- all (20) orders are realised in one period.

| Table 2 Basis routings for the creation of scheduling problems in minutes. |
|-----------------------------------------------|
| Routing/part type | Station 1 (min) | Station 2 (min) | Station 3 (min) | Station 4 (min) |
|-------------------|----------------|----------------|----------------|----------------|
| 1                 | 100            | 50             | 50             | 10             |
| 2                 | 150            | 100            | 100            | 150            |
| 3                 | 100            | 200            | 150            | 200            |
| 4                 | 200            | 150            | 300            | 150            |
| 5                 | 250            | 250            | 250            | 200            |
| Sum of operation times at station | 800        | 750            | 850            | 710            |
The due dates were generated by a fix flow factor, so that under scheduling with FIFO the percentage of late jobs is 30%, 50%, 70% or 85%; so there are 16 combinations. For each operation in the five basis routings 4 processing times are generated. This leads to $5 \times 4 \times 4 = 80$ routings. So, in total $16 \times 80 = 1,280$ experiments are generated.

By using the basis routings and a uniform distribution of the parts, station 2 is the bottleneck station, because the sum of all operation times at station 2 is greater than these sums at the other stations. Due to the generation procedure of alternative processing times at the stations, the sequence of stations due to this criterion (the sum of all operation times at a station) can change. In the real application each station is a bottleneck in a significant portion of the periods over a large horizon, because the products are non-uniform distributed in the demand over a large horizon. In order to ensure a comparable situation, much more than 1,280 experiments are generated (namely around 60,000) and from these 1,280 experiments are chosen, so that in around 15% of the experiments station 1 is a bottleneck station, in around 30% of the experiments station 2 is a bottleneck station, in around 25% of the experiments station 3 is a bottleneck station, and in around 30% of the experiments station 4 is a bottleneck station (and of course, the other conditions are still fulfilled).

For the application of the priority rules a tail for the simulations of the processing times has to be defined. One possibility consists of jobs with low processing times. Let us assume that there are two jobs with the same net processing time. In job A the first operations have high processing times and the last ones have low processing times. In job B it is the other way round. Then the simulated processing time of A is often smaller than the one of B; of course, with some allocations of jobs (already) on the flow shop the simulation calculates a smaller processing time for B than for A. So, there is a tendency to prefer jobs with small processing times on the stations at the end of the flow shop. Now, let us regard the other extreme, so that the tail consists of jobs with high processing times. Then the jobs in the tail determine the simulated processing time. This causes sometimes similar simulated processing times for different jobs, so that with respect to the intended priority criterion wrong decisions are made. In order to analyse different tails sub permutations of seven jobs out of the optimal solutions to these small test problems are regarded. The first three jobs serve as a realistic allocation of the flow shop for a simulation of the processing time and the last three jobs serve as a realistic tail. Then, for different jobs on the fourth position its processing time is simulated. The same is done with a static tail. On average a tail of the jobs X, Y and Z, where X has a small net processing time, Y a very large net processing time and Z has a medium net processing time, delivers the best results. In the case of the heuristic branch-and-bound long continuations should reduce the impact of the tail. Indeed no impact can be observed if a relative large $s$ (compared to the number of possible branches) and a relative long continuation (compared to the length of a full permutation) is possible as with very small test problems (less than 20 jobs), in which a nearly full enumeration is executed (with an appropriate parameter setting). If $s$ is reduced to a small value, as normally used in the experiments, an impact can be observed, because jobs with the highest priorities tend to be too similar especially if their priorities are comparable. Therefore, the priority rules and the heuristic branch-and-bound use the same tail.

Table 3 contains the results for the three performance criteria. They are average objective values relative to the optimum solution which is set to 100%. So, for example, the solution resulted by CR+SPT was about 176.11% above the optimum on the average. The heuristic branch-and-bound delivers on average the best values with $s = 2$ and $c = 0.0002$. The depth of its decision tree, which is evaluated in each step, causes...
Table 3  Results for the test problem instances; B&B with s = 2, c = 0.0002 (and SL/OPN).

| Optimum for | CR+SPT (%) | ODD (%) | SL/OPN (%) | RR (%) | RM local (%) | RM global (%) | B&B (%) |
|-------------|------------|---------|------------|--------|--------------|--------------|---------|
| $T_{\text{Mean}}$ | 100 | 176.11 | 220.41 | 190.98 | 158.16 | 168.08 | 151.60 | 131.43 |
| $T_{\text{RMS}}$ | 100 | 178.60 | 300.70 | 131.86 | 141.51 | 122.67 | 107.79 | 98.19 |
| $T_{\text{Max}}$ | 100 | 252.18 | 212.37 | 210.01 | 164.97 | 165.56 | 148.42 | 157.22 |

that the heuristic branch-and-bound delivers better results than the priority rules. But its look ahead is too short to achieve optimal solutions. (Note: In the case of a small number of jobs optimal solutions are achieved.) According to the sequence in the performance criteria of the priority rules, the results are in accordance to the ones in Refs. [5, 14, 16, 29, 30]. Especially in Refs. [29, 30], the relative distances between the performance criteria of the priority rules are significantly larger than the ones in Table 3 and these ones are much larger than the ones in Ref. [42]. According to the optimal solution, the gap is much larger than that in the results of Ref. [42].

In the case of jobs which are inevitable finished late (called scenario 1 in the following), the results are in better accordance with the mentioned results from Ref. [42] (compared to the optimal solution; they are shown in Table 4). (Note: For this, special experiments are generated. If such a pool of jobs is scheduled by FIFO nearly each job is late and in some experiments the optimal mean tardiness is even (relative) small.) In this case the improvement by the heuristic branch-and-bound is smaller. Responsible for the large decline (of the figures for the priority rules in Table 3 versus the ones in Table 4) are problem instances where a large amount of jobs have sufficient slack (called scenario 2 in the following), there results are in Table 5; in this scenario $T_{\text{mean}} = 49$ min, for the optimal mean tardiness, which is small compared to the net processing time of the jobs (644 min is the average net processing time of the jobs).

The optimal solutions in each of these two extreme scenarios have specific structural characteristics. In scenario 1, small cycle times ($\tau$) are achieved at the expense of a markedly number of large cycle times, $\tau_{\text{Mean}} = 185$ and $\tau_{\sigma^2} = 6213$. Some rules have even a much higher variance of cycle times but they do not use it to reduce the average cycle times as in the optimal solution; this is shown in Table 4. SPT based rules tend to a small $\tau_{\text{Mean}}$ at the expense of large $\tau_{\sigma^2}$ and slack based rules tend to a small $\tau_{\sigma^2}$ at the expense of a high $\tau_{\text{Mean}}$. The values in Table 4 confirm this statement; note that additional experiments show that the SPT rule delivers even slightly better values than the rules listed in Table 4; this explains that CR+SPT delivers the best results. Generally in the rules RR, RM local and RM global, the slack component is dominant, so that the rules try to avoid a high variance of the cycle times. Because the slack has, due to the processing time costing, a larger impact on the global version of the RM rule than on the local version (due to the processing time costing), the local version of the RM rule is slightly better (for $T_{\text{Mean}}$) than the global one. This explains the results for RR also. The depth of the decision tree in the heuristic branch-and-bound compensates the high impact of the slack, because SL/OPN is used in the continuation step, but it is too small to deliver the best results; with CR+SPT and a low look ahead by $s=4$ in the continuation step the best results for $T_{\text{Mean}}$ are delivered at the expense of a higher $T_{\text{RMS}}$.

A permutation with a small difference in the cycle times of two succeeding cycles is normally delivered by the optimal solutions in scenario 2. This causes a smaller variance of the cycle times at the expense of higher mean cycle times; compare $\tau_{\text{Mean}} = 215$ and
Table 4  Results for the test problem instances (scenario 1) in which all jobs are inevitable finished late; B&B with s = 2, c = 0.0002 (and SL/OPN).

| Optimum for | CR+SPT (%) | ODD (%) | SL/OPN (%) | RR (%) | RM local (%) | RM global (%) | B&B (%) |
|-------------|------------|---------|------------|--------|--------------|---------------|---------|
| $T_{Mean}$  | 100        | 125.45  | 158.49     | 156.75 | 134.40       | 138.20        | 139.16  | 129.07  |
| $T_{RMS}$   | 100        | 168.96  | 102.45     | 97.08  | 113.21       | 99.53         | 103.3   | 99.03   |
| $T_{Max}$   | 100        | 128.07  | 132.3      | 126.59 | 124.66       | 126.6         | 115.37  | 121.84  |
| $\tau_{Mean}$ | 100    | 112.17  | 132.43     | 126.22 | 121.62       | 120.54        | 118.14  | 109.55  |
| $\tau_{\sigma^2}$ | 100 | 153.94  | 138.79     | 90.6   | 73.23        | 86.26         | 62.36   | 55.63   |
| $\tau_{Max}$ | 100       | 130     | 140.33     | 150.00 | 133.33       | 131           | 124.33  | 131.22  |

Table 5  Results for the test problem instances (scenario 2) in which a large amount of jobs have sufficient slack; number of outliers are listed in brackets (remember there are 1280 experiments) in row to $\tau_{Max}$ ; B&B with s = 2, c = 0.0002 (and SL/OPN).

| Optimum for | CR+SPT (%) | ODD (%) | SL/OPN (%) | RR (%) | RM local (%) | RM global (%) | B&B (%) |
|-------------|------------|---------|------------|--------|--------------|---------------|---------|
| $T_{Mean}$  | 100        | 358.71  | 398.06     | 368.42 | 315.54       | 323.09        | 305.25  | 265.67  |
| $T_{RMS}$   | 100        | 343.33  | 440.56     | 509.44 | 340.83       | 543.06        | 590.83  | 570.18  |
| $T_{Max}$   | 100        | 324.4   | 353.77     | 277.14 | 336.06       | 343.37        | 130.34  | 120.34  |
| $\tau_{Mean}$ | 100    | 108.37  | 113.02     | 120.23 | 123.26       | 114.88        | 106.98  | 104.19  |
| $\tau_{\sigma^2}$ | 100 | 193.21  | 148.18     | 124.44 | 140.44       | 163.98        | 118.64  | 119.43  |
| $\tau_{Max}$ | 100       | 340     | 237.14     | 180    | 300.29       | 248.86        | 163.14  | 160.08  |

$\tau_{\sigma^2} = 2275$ with these values in scenario 1 for the optimal solution. Although the rules tend to achieve the same behaviour, the mean tardiness is significantly worse. A more detailed analysis showed that there occurred a few (normally just one) outliers in the cycle times, which are avoided by the optimal solutions usually; this is shown in Table 5 by the maximum cycle times, i.e. $\tau_{Max}$ and the number of outliers (about all experiments). The smaller outliers in the cycle times by the global RM rule compared with the local variant seem to be caused by the look ahead in the global RM rule. This causes smaller values of $T_{Mean}$. Furthermore, the look ahead (due to the consideration of all unfinished jobs) in the global RM rule seems still to be too short, so that the global RM rule just avoids outliers partially (in terms of number and, even more important, of size). The relative large look ahead in the heuristic branch-and-bound causes an even smaller number of outliers, which causes a significant better $T_{Mean}$.

The flow shop in the real world application is running for quite a while. An analysis of the demand over a large horizon shows that the products are non-uniform distributed. In detail, the distribution of the five routings in the sequence of jobs is: 20% of the jobs have routing 1, 20% of the jobs have routing 2, 40% of the jobs have routing 3, 10% of the jobs have routing 4 and 10% of the jobs have routing 5. Then, the generating algorithm for job sequences causes a work load distribution on the four stations that in 22% of the periods the first station has the largest work load, in 31% of the periods the second station has the largest work load, in 23% of the periods the third station has the largest work load, in 23% of the periods the fourth station has the largest work load; so, each station is a bottleneck for a significant number of
periods. Due to these temporary bottlenecks scheduling cannot focus on a station, so that a good permutation for this station also fits for all other stations, which would be beneficial for a priority rule.

At the company site the jobs are scheduled manually by a foreman who mixes simple criteria as earliest due date, estimated slack, operation time and some kind of personal experience with the scheduling of other jobs to the same filter type especially regarding that the operations in each cycle have similar processing times.

So, the procedure at the company site cannot be compared with the investigated priority rules in this paper. But, the company has records about due dates and completion times of some large job sequences. The company allowed to publish that the following results are significant better than the figures to the different measures of tardiness calculated out of these records.

As explained earlier, the workload covers the demand of one week with 7 days and three shifts about 8 hours per day. Due to the creation of jobs, the mean number of jobs per week is 53.25 with a standard deviation of 12.62 (the numbers are in Refs. [33, 78]). In order to manage the impact of the jobs in the next period, especially on the decisions at the end of the actual period, at least a significant portion of the jobs in the next period has to be regarded. In the simulations the mean number of jobs is 73.23 with a standard deviation of 14.85 (and they are in Refs. [43, 97]).

The generating procedure generates sequences of jobs so that on average the time-dependent course of the work load on the stations and the urgency of the jobs are identical; so, similar to a stationary stochastically process. Therefore, both performance criteria average tardiness ($T_{\text{mean}}$) and root mean square of tardiness ($T_{\text{rms}}$) reach a steady state by a simulation horizon of 350 periods (i.e. around 14,000 jobs) for each time pressure and scheduling with FIFO.

Therefore, a simulation horizon of 350 periods is used. To avoid errors from startup and rundown, the first and the last 10 periods are disregarded.

As for small problem instances a static tail for calculating $t_i^t$ (for a job $i$) is used. The heights of the 3 part types of each tail are 300 mm, 1,800 mm, and 1,200 mm. Experiments with some alternative static tails did not deliver better values, as expected due to the analysis of the small problem instances.

The results by the priority rules and the heuristic branch-and-bound are shown in Fig. 3. The values for $T_{\text{mean}}$ determine a sequence of rules which is identical with the one for the small test problems. So, this is in accordance with the results in literature like Refs. [5, 16, 29, 30, 42].

In accordance with the expectation (see Ref. [14]) CR+SPT delivers a small $T_{\text{mean}}$ at the expense of a higher $T_{\text{rms}}$ and the opposite is true for SL/OPN. As said earlier, the first behaviour is typical for a SPT based rule and the second for a slack based rule. The other three rules RR, local RM and global RM avoid large $T_{\text{mean}}$ and $T_{\text{rms}}$ respectively, which is in accordance with the results in the literature, but they deliver not always the best value. Nevertheless, the rule RR combines the rules SPT and slack better than the CR+SPT rule. Further experiments indicate that the application of the exponential function on the slack in the (two versions of the) RM rule, partially modified by constants, uses the slack more effective than the other slack based rules and delivers good results even if it is beneficial to prefer the job with the smallest processing time. But this is not enough to ensure that both RM rules deliver better results than RR—indeed half of the figures for RR are better than the ones for RM local. The better processing time costing of RM global compared to RM local causes that RM global delivers always the best results. The results of the heuristic branch-and-bound are considerably better than the ones from the priority rules. A larger $s$, compared to the setting of 2, causes an increase of $T_{\text{mean}}$ and $T_{\text{rms}}$. Increasing values of $c$, compared to the setting of 0.0002, lead to smaller values of $T_{\text{rms}}$ but larger mean values. The maximal tardiness is hardly affected by a relative small change of both parameters.
Compared to the small test problems the relative distances between the values for $T_{\text{Mean}}$, primarily for the priority rules, are significant larger. As with the small test problems these relative distances are larger than the ones in Ref. [42] but still smaller than the ones in Refs. [29, 30]. But partially these relative distances are comparable with the ones in Ref. [30]. The results to the priority rules in Ref. [30] and in Ref. [29] are significantly impacted by the station utilisation levels, which are primarily 80% and 95%; in addition Ref. [30] uses an allowance factor and Ref. [29] uses the due date tightness, which both have a smaller impact than the station utilisation level. On the sequence of the priority rules these parameters have just a low impact. In this investigation the time pressure is only increased by tighter due dates while the load on the stations remains unchanged. Thus, it can be expected that the time pressure here has a similar effect as the second parameter in Ref. [30] or in Ref. [29]. A much larger impact is caused by the fact that the load oscillates significantly in the periods; especially the load is much higher than 95% and much
lower than 80%. Thus a tighter due date has a more significant effect in periods with relative very high load than in periods with a relative very small load. In total, this seems to cause different relative distances; partially this causes also the similar effect for small test problems.

Due to the analysis for the small test problems, there are pools of jobs thus a large variance of the cycle times is beneficial and pools of jobs thus a small variance delivers better results. Just as with the small test problems, a SPT based rule produces a large variance of the cycle times and a slack based rule produces a small variance. Experiments show that outliers occur more often than with the small test problems, so that their negative impact on the performance is greater. Just as with the small test problems the outliers are higher by a combination of SPT and slack as by the SPT rule or the slack rule. This is caused by jobs with a large processing time which are becoming “suddenly” very urgent due to its slack. The company confirmed that this effect occurred quite often in the manual scheduling and the best three priority rules avoid this significantly. Due to a relative large look ahead, the heuristic branch-and-bound avoids this even better.

Both reasons explain why the relative distances in the performance between the rules are even larger than the ones in the studies by Refs. [5, 14, 16, 42]. An increase of the time pressure reduces the positive effect of a slack, thus these relative distances decrease with an increase of the time pressure; see Fig. 3. In the case of the highest time pressure these relative distances are even smaller than the ones in Ref. [42].

Again the heuristic branch-and-bound profits from the depth of the decision tree, which explains that the use of SL/OPN outperforms their alternatives and that this B&B outperforms the best priority rule by around 10-15%. The improvements of B&B compared with the priority rules are caused by the same reasons as with the small test problems, thus the relative improvements are comparable.

### 7. Conclusions

Real world flow shop scheduling problems have sometimes individual and specific restrictions which are not covered by the restrictions in the classification of scheduling problems used in the literature normally, see e.g. Ref. [2]. An example is the real world application in this investigation. The paper contains an optimisation model for a class (C) of scheduling problems which covers this real world application. Therefore, this paper seems to be a new enhancement for the scheduling of flow shops with special restrictions.

The paper contains a survey of existing priority rules for no-buffer flow shop scheduling problems. It is explained why these priority rules cannot be applied directly on the real world application or the class C respectively, and an extension is suggested. Some of those priority rules, which are analysed to be very effective in the literature, are extended by this way. The resulting priority rules are applied on a large set of small test problems for which optimal solutions are calculated. In general, the performance of the priority rules is in accordance with the ones in the literature but the relative distance between the performance criteria differ from the ones in some papers; as a trend the distances seem to be larger. The comparison between the schedules created by these priority rules and the optimal schedule (for mean tardiness) shows that priority rules do not recognise when to prefer large variances of cycle times or small ones, respectively. In addition, priority rules cause outliers in the cycle times—compared to the optimal solution. With a considerably number of simulation experiments it is shown that these behaviours of priority rules cause the relative performance of them in the real world application as well. A significant improvement about around 10-15% compared to the best priority rule is achieved by strict search over all possible schedules due to a heuristic B&B, since a relative large look ahead compared to the one in
priority rules reduces mistakes of priority rules.

References

[1] Allahverdi, A., Ng, C. T., Cheng, T. C. E., and Kovalyov, M. Y. 2008. “A Survey of Scheduling Problems with Setup Times or Costs.” European Journal of Operational Research 187: 985-1032.

[2] Brucker, P. 2010. Scheduling Algorithms. Springer Verlag, Osnabrück.

[3] Chiang, T. C., and Fu, L. C. 2009. “Using a Family of Critical Ratio-Based Approaches to Minimize the Number of Tardy Jobs in the Job Shop with Sequence Dependent Setup Times.” European Journal of Operational Research 196 (1): 78-92.

[4] Chiang, T. C., and Fu, L. C. 2012. “Rule-Based Scheduling in Wafer Fabrication with Due Date-Based Objectives.” Computers & Operations Research 39: 2820-35.

[5] Engell, S., Herrmann, F., and Moser, M. 1994. “Priority Rules and Predictive Control Algorithms for On-line Scheduling of FMS.” In Computer Control of Flexible Manufacturing Systems, edited by Joshi, S. B., and Smith, J. S. Chapman & Hall, London, 75-107.

[6] El-Bouri, A. 2012. “A Cooperative Dispatching Approach for Minimizing Mean Tardiness in a Dynamic Flowshop.” Computers & Operations Research 39 (7): 1305-14.

[7] Framinan, J. M., and Leisten, R. 2003. “An Effective Constructive Heuristic for Flowtime Minimisation in Permutation Flow Shops.” Omega: International Journal of Management Science 31: 311-7.

[8] Framinan, J. M., Leisten, R., and Rajendran, C. 2003. “Different Initial Sequences for the Heuristic of Nawaz, Enscore and Ham to Minimize Makespan, Idletime or Flowtime in the Static Permutation Flowshop Sequencing Problem.” International Journal of Production Research 41: 121-48.

[9] Gicquel, C., Hege, L., Minoux, M., and van Canneyt, W. 2012. “A Discrete Time Exact Solution Approach for a Complex Hybrid Flow-Shop Scheduling Problem with Limited-Wait Constraints.” Computers & Operations Research 39: 629-36.

[10] Grabowski, J., and Pempera, J. 2007. “The Permutation Flow Shop Problem with Blocking. A Tabu Search Approach.” Omega 35 (3): 302-11.

[11] Hax, A. C., and Meal, H. C. 1975. “Hierarchical Integration of Production Planning and Scheduling.” In Studies in Management Sciences, edited by Geisler, M. A. Vol. 1. Logistics. North-Holland, Amsterdam.

[12] Hall, N. G., and Sriskandarajah, C. 1996. “A Survey of Machine Scheduling Problems with Blocking and No-Wait in Process.” Operations Research 44 (3): 510-25.

[13] Herrmann, F. 2009. Logik der Produktionslogistik. Oldenbourg Verlag, Regensburg.

[14] Herrmann, F. 2011. Operative Planung in IT-Systems for Production Planning and Control. Vieweg + Teubner Verlag, Regensburg.

[15] Kim, Y. D. 1993. “A New Branch and Bound Algorithm for Minimizing Mean Tardiness in Two-Machine Flowshops.” Computers and Operations Research 20 (4): 391-401.

[16] Lawrence, S., and Morton, T. 1993. “Resource-Constrained Multi-Project Scheduling with Tardy Costs: Comparing Myopic, Bottleneck, and Resource Pricing Heuristics.” European Journal of Operational Research 64: 168-87.

[17] Leisten, R. 1990. “Flowshop Sequencing with Limited Buffer Storage.” International Journal of Production Research (28): 2085-100.

[18] Morton, T., and Pentico, D. 1993. Heuristic Scheduling Systems: With Applications to Production Systems and Project Management. New York: Wiley.

[19] Mouelhi, C. W., and Pierreval, H., 2010. “Training a Neural Network to Select Dispatching Rules in Real Time.” Computers and Industrial Engineering 58 (2): 249-256.

[20] McCormick, S. T., Pinedo, M. L., Shenker, S., and Wolf, B. 1989. “Sequencing in an Assembly Line with Blocking to Minimize Cycle Time.” Operations Research 37 (6): 925-35.

[21] Monma, C. L., and RinnooyKan, A. H. G. 1983. “A Concise Survey of Efficiently Solvable Special Cases of the Permutation Flow-Shop Problem.” RAIRO: Operations Research 17: 105-19.

[22] Nawaz, M., Enscore, E. E., and Ham, I. 1983. “A Heuristic Algorithm for the m-Machine, n-Job Flow Sequencing Problem.”Omega: International Journal of Management Science 11 (1): 91-5.

[23] Nowicki, E. 1999. “The Permutation Flow Shop with Buffers: A Tabu Search Approach.” European Journal of Operational Research 116: 205-19.

[24] Ouelhadj, D., and Petrovic, S. 2009. “A Survey of Dynamic Scheduling in Manufacturing.” Journal of Scheduling 12 (4): 417-31.

[25] Pan, Q., and Wang, L. 2012. “Effective Heuristics for the Blocking Flowshop Scheduling Problem with Makespan Minimization.” Omega: International Journal of Management Science 40 (2): 218-29.

[26] Pinedo, M. L. 2005. Planning and Scheduling in Manufacturing and Services. New York: Springer.

[27] Rachamadugu, R. M. V. 1987. “Technical Note—A Note on the Weighted Tardiness Problem.” Operations Research 35: 450-2.
Scheduling a Flow Shop with Simultaneously Loaded Stations: A Real World Application

[28] Rachamadugu, R. V., and Morton, T. E. 1982. “Myopic Heuristics for the Single Machine Weighted Tardiness Problem.” Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA.

[29] Raghu, T. S., and Rajendran, C. 1993. “An Efficient Dynamic Dispatching Rule for Scheduling in a Job Shop.” International Journal of Production Economics 32: 301-13.

[30] Rajendran, C., and Holthaus, O. 1999. “A Comparative Study of Dispatching Rules in Dynamic Flowshops and Jobshops.” European Journal of Operational Research 116 (1): 156-70.

[31] Rajendran, C., and Alicke, K. 2007. “Dispatching in Flowshops with Bottleneck Machines.” Computers and Industrial Engineering 52 (1): 89-106.

[32] Ribas, I., Companys, R., and Tort-Martorell, X. 2011. “An Iterated Greedy Algorithm for the Flowshop Scheduling Problem with Blocking.” Omega: International Journal of Management Science 39: 293-301.

[33] Ronconi, D. P. 2004. “A Note on Constructive Heuristics for the Flow-Shop Problem with Blocking.” International Journal of Production Economics: 39-48.

[34] Ronconi, D. P. 2005. “A Branch-and-Bound Algorithm to Minimize the Makespan in a Flowshop with Blocking.” Annals of Operations Research (138): 53-65.

[35] Ronconi, D. P., and Armentano, V. A. 2001. “Lower Bounding Schemes for Flowshops with Blocking In-Process.” Journal of the Operational Research Society 52 (11): 1289-97.

[36] Ronconi, D., and Henrique, L. 2009. “Some Heuristic Algorithms for Total Tardiness Minimization in a Flowshop with Blocking.” Omega: International Journal of Management Science 37 (2): 272-81.

[37] Russel, R. S., Dar-El, E. M., and Taylor, B. W. 1987. “A Comparative Analysis of the Covert Job Sequencing Rule Using Various Shop Performance Measures.” International Journal of Production Research 25 (10): 1523-40.

[38] Sundaramoorthy, A., and Maravelias, C. T. 2008. “Modeling of Storage in Batching and Scheduling of Multi Stage Processes.” Industrial and Engineering Chemistry Research 47 (17): 6648-60.

[39] Swaminathan, R., Pfund, M. E., Fowler, J. W., Mason, S. J., and Keha, A. 2007. “Impact of Permutation Enforcement When Minimizing Total Weighted Tardiness in Dynamic Flowshops with Uncertain Processing Times.” Computers and Operations Research 34 (10): 3055-68.

[40] Taillard, E. 1993. “Benchmarks for Basic Scheduling Problems.” European Journal of Operational Research 64 (2): 278-85.

[41] Vepsäläinen, A. P., and Morton, T. E. 1987. “Priority Rules for Job Shops with Weighted Tardiness Costs.” Management Science 33 (8): 95-103.

[42] Voß, S., and Witt, A. 2007. “Hybrid Flow Shop Scheduling as a Multi-mode Multi-project Scheduling Problem with Batching Requirements: A Real-World Application.” International Journal of Production Economics 105: 445-58.

[43] Wang, X., and Tang, L. 2012. “A Discrete Particle Swarm Optimization Algorithm with Self-adaptive Diversity Control for the Permutation Flowshop Problem with Blocking.” Applied Soft Computing 12 (2): 652-62.

[44] Wang, L., Pan, Q. K., and Tasgetiren, M. F. 2011. “A Hybrid Harmony Search Algorithm for the Blocking Permutation Flow Shop Scheduling Problem.” Computers & Industrial Engineering 61 (1): 76-83.

[45] Wang, C., Song, S., Gupta, J. N. D., and Wu, C. 2012. “A Three-Phase Algorithm for Flowshop Scheduling with Blocking to Minimize Makespan.” Computers and Operations Research 39 (11): 2880-7.