Genetic algorithm with simulation for scheduling of a flow shop with simultaneously loaded stations

Professor Dr.-Ing. Frank Herrmann
Ostbayerische Technische Hochschule Regensburg – Technical University of Applied Sciences Regensburg
Innovation and Competence Centre for Production Logistics and Factory Planning (IPF)
PO box 120327, 93025 Regensburg, Germany
E-Mail: Frank.Herrmann@OTH-Regensburg.de

KEYWORDS
Simulation of restrictions, scheduling, flow-shop, no-buffer (blocking), no-wait, genetic algorithm, real world application.

ABSTRACT
In this study, a real world flow shop with a transportation restriction is regarded. This restriction reduces the set of feasible schedules even more than the no-buffer restrictions discussed in the literature in the case of limited storage. Still this problem is NP-hard. Since this scheduling problem is integrated in the usual hierarchical planning, the tardiness is minimised. Compared to even specific priority rule for this class of problems the suggested genetic algorithm delivers significant better results. The specific structure of this class of problems complicates the calculation of the performance criteria. This is solved by a simulation algorithm.

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. So, in recent years, a considerable amount of interest has arisen in no-buffer (blocking) scheduling problems and in no-wait scheduling problems, with makespan as objective criteria. Often these production systems deliver products for other systems as well. Due to the hierarchical planning which is implemented in enterprise resource planning systems (ERP system) (see e.g. Jacobs et al. 2010), the local completion times in one production system in many cases determine the earliest possible starting times in another production system. Thus, the delay of the operations in a production system has an impact on the effectivity of this coordination process. Therefore, tardiness is considered as objective criteria.

2. A REAL WORLD APPLICATION

The problem is a modification of a partly automated production line at Fiedler Andritz in Regensburg to produce filter (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 4 stations which are shown in Figure 1. Station 1 assembles 6 single batons (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. So, there is no buffer in the production line and each feasible schedule of jobs is a permutation of these jobs. 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,

Proceedings 28th European Conference on Modelling and Simulation ©ECMS Flaminio Squazzoni, Fabio Baronio, Claudia Archetti, Marco Castellani (Editors)
ISBN: 978-0-9564944-8-1 / ISBN: 978-0-9564944-9-8 (CD)

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.

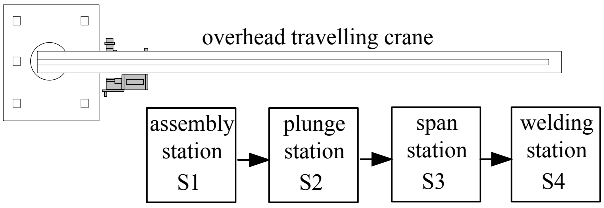

Figure 1: Structure of the production line

There are 10 part types. Their different heights or designs causes different processing times which are listed in Table 1.

Table 1: Processing times for all part types in minutes

Part type	Station				Sum of times
	S1	S2	S3	S4	
P1	100,5	50	53,5	9	213
P2	256,5	50	53,5	9	369
P3	122	135	90	75	422
P4	256,5	50	267	9	582,5
P5	182	200	135,5	140	657,5
P6	100,5	300	53,5	300	754
P7	223	250	196	220	889
P8	223	250	206,5	220	899,5
P9	100,5	300	267	300	967,5
P10	256,5	300	267	300	1123,5

At the company's production site, the jobs for filters are generated by an SAP system and produced filters are stored before they are assembled into other products or sold directly to customers. Therefore, all jobs with a release date after the beginning of a period are released at the beginning of this period. One period consists of one day with three 8 hour shifts. For this investigation, sequences of jobs of filter types with lot size 1 are randomly generated for each period t by an generating algorithm which has been designed in accordance with the proceeding in (Russel et al. 1987) and in (Engell et al. 1994): An additional filter type F, released in period t, consumes capacity on each station during the time between t and its due date; 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 on each station is below a maximal load level, otherwise it will be skipped to the next period. A maximal load level is an (intended) average load $\left(L_0(S)\right)$ 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.

In reality at the company, there are large numbers of periods with a low number of late jobs and large numbers of periods with a high (or even very high) number of late jobs. To achieve a comparable situation for this investigations, due dates are determined in a way so that scheduling with the FIFO rule (first-in-first-out) 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. As a result of the generating algorithm's calculations the mean difference between the due dates and the release dates are between 2 and 3,5 days with a standard deviation of 0,5 days. Andritz Fiedler has confirmed that such timeframes for processing jobs are representative.

The time needed for loading and unloading a station is negligible compared to the duration of the operation itself. In addition this task is independent from the allocation (or loading) of other stations and the required time 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 for release dates a_i $(1 \le i \le N)$ and due dates f_i $(1 \le i \le N)$ respectively. Also there is the duration $t_{i,j}$ of operation $\left(o_{i,j}\right)$ j $(1 \le j \le M)$ of job i $(1 \le i \le N)$, which is being worked on station j. Due to the reasons, said in the introduction, as performance criteria average tardiness $\left(T_{Mean}\right)$ and standard deviation of the tardiness $\left(T_\sigma\right)$ are primarily analysed.

The time between two consecutive executions of the load process is determined by the maximum of the duration of the operations (including setup time) 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.

The no-buffer condition means a relaxation of the scheduling problem with the (above) “load”-restriction. Scheduling problems with the no-buffer are proven to be NP-hard in the strong sense for more than two stations; see e.g. (Hall and Sriskandarajah 1996), which contains a good survey of such problems.

3. LITERATURE REVIEW

As mentioned earlier, the real application is close to the class of no-buffer (blocking) scheduling problems. Solutions for the no-buffer (blocking) scheduling problems are published in various papers. Most papers minimise makespan as the ones in the following review – later publications to minimise tardiness are reviewed. Thus, the following review explains a large spectrum of possible procedures. In (McCormick et al. 1989) a schedule is extended by a (unscheduled) job that leads to the minimum sum of blocking times on machines which is called profile fitting (PF). Often the starting point of an algorithm is the NEH algorithm presented in (Nawaz et al. 1983), as it is the best constructive heuristic to minimize the makespan in the flow shop with blocking according to many papers, e.g. (Framinan et al. 2003). Therefore, (Ronconi 2004) substituted the initial solution for the enumeration procedure of the NEH algorithm by a heuristic based on a makespan property proven in (Ronconi and Armentano 2001) as well as by the profile fitting (PF) of (McCormick et al. 1989). (Ronconi 2005) 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 makespan, (Grabowski and Pempera 2007) realised and analysed a tabu search algorithm. As an alternative approach, (Wang and Tang 2012) have developed a discrete particle swarm optimisation algorithm. In order to diversify the population, a random velocity perturbation for each particle is integrated according to a probability controlled by the diversity of the current population. Again, based on the NEH algorithm, (Wang et al. 2011) 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 makespan criterion. This leads to an initial solution with higher quality. With special settings as a result of the mechanism of a harmony search algorithm, better results are achieved. Also (Ribas et al. 2011) presented an improved NEH-based heuristic and uses 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. 2012). 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 profile fitting (PF) approach of (McCormick et al. 1989), (Pan and Wang 2012) addressed two simple constructive heuristics. Then, both heuristics and the profile fitting 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 (Taillard 1993), which is widely used as benchmark in the literature.

To the best of my knowledge, only a few studies investigate algorithms for the total tardiness objective (for flow shops with blocking). (Ronconi and Armentano 2001) have developed a lower bound which reduces the number of nodes in a branch-and-bound algorithm significantly. (Ronconi and Henriques 2009) 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 based (greedy randomized adaptive search procedure) search heuristic. There are just a few genetic algorithms with this performance criteria: for a no-wait flowshop scheduling problem one is published in (Chaudhry and Mahmood 2012) and for a flowshop with blocking one is published in (Januario et al. 2009).

3. GENETIC ALGORITHM

In the literature to scheduling problems each genetic algorithm has typically the following basic structure (s. e.g. (Werner 2013)):

0. Representation of a schedule by a chromosome
1. Initial population
2. Fitness of a (actual) population
3. Selection
4. Application of a genetic operation
5. Formation of the new population
6. If stopping criteria is not satisfied, then go to step 2
7. Selection of a best chromosome (i.e. schedule)

Due to the load condition each feasible schedule is a permutation of jobs. So, this permutation can be used as a chromosome. Above all, the load restriction determines the allocation of each operation on a station during the execution of the (above) permutation of the jobs. This is simulated by an algorithm. So, a performance criteria as tardiness can be calculated. (Note: This concept can be extended to further restrictions – which are not covered by the representation of a schedule.)

Initial populations are generated either by accident or by a priority rule as well as standard heuristics for flow shop problems with a relatively low runtime. Implemented are many rules and concrete settings are said in chapter "experimental calibration of the genetic algorithm".

The fitness of a chromosome i (F(i)) is the average tardiness of the jobs (i.e. the overall performance criteria) of the simulated permutation (schedule).

Possible selections are the fitness-proportional selection (or roulette wheel selection), the tournament and a specific percentage which is chosen accidently. With the fitness-proportional selection the probability P(i) of selecting the i-th chromosome is given by $\frac{F(i)}{\sum_{i \in P} F(i)}$ with population P. In the tournament selection,

n (a parameter) chromosomes of the actual population are accidently selected and the one with the best fitness comes in a new set P. This operation is repeated until a certain number of chromosomes are chosen. If m chromosomes have the best fitness, but just m' < m ones can put in the set than the m' ones are chosen by accident. A chromosome can be selected and chosen more than once. Then, P is the new population.

There are two classes of genetic operations: crossover and mutation. Crossover mixes two selected chromosomes of the current population to two chromosomes. It is applied with probability PC, which is usually high (> 0,6 is often recommended). Mutation changes the position of one or more orders (genes) in a single permutation (chromosome). It is applied with probability PM, which is usually small (often 0,01 < PM < 0,1 is recommended).

Some of the crossover operators described in (Werner 2013) are implemented, namely: order crossover (OX), cycle crossover (CX) and order based crossover (OBX). Also the mutation operators described in (Werner 2013) are implemented, namely: shift-mutation (also named as neighborhood or insertion neighbourhood), pairwise interchange neighborhood (also named as swap neighbourhood), API neighbourhood, and inversion neighbourhood; note: (Werner 2013) contains some theoretical results about mutation operators.

Formation of the new population is done by elite strategy: The best chromosome or a specific percentage of the best chromosomes is in the next population and all others are chosen by accident.

As stopping criteria a maximal number of iterations – i.e. generating of populations –, maximal runtime is available or after a percentage of the number of orders in a scheduling problem.

4. EXPERIMENTAL CALIBRATION OF THE GENETIC ALGORITHM

Prior to applying the genetic algorithm on the real world application, the parameters are chosen. For this, the genetic algorithm applied on some small generated test problems.

In order to use problems, which are comparable to the real world problem, 4 stations are being considered. The routings are created from a set of so called basis routings which are stated in table 2. The total net processing times of these basis routings covers the same range as the ones in the real world application. A concrete 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 effectively generate a part type sequence by the generating algorithm. Instead, the part types are generated by a uniform distribution, and the following three basic scenarios for the order release are randomly generated: all orders are realised in one, two or three periods, so that in each scenario the difference of the number of orders in the periods is at most one – note, that sometimes orders of previous periods are still in the production system. The number of orders vary between 8, 16 and 24.

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% – resulting in $3 \cdot 4 \cdot 3 = 36$ combinations. For each operation in the five basis routings (see table 2) 4 processing times are generated. This leads to $5 \cdot 4 \cdot 4 = 80$ routings. So, in total $36 \cdot 80 = 2880$ experiments are generated.

By using the basis routings and a uniform distribution of the parts, station 3 is the bottleneck station, because the sum of all operation times at station 3 is greater than these sums at the other stations. Because of the way alternative processing times are being generated for 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, about 60000 are generated. From those 2880 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).

Routing / part type	Station 1	Station 2	Station 3	Station 4
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	800	750	850	710

Table 2: Basis routings for the creation of scheduling problems in minutes

The parameters of the genetic algorithm are varied as follows:

- Population size: 5, 10, 20, 50, 100, 200, 400, 600, 800 and 1000.
- Selection: fitness-proportional selection, the tournament and accidently chosen a specific percentage. The parameter n in the tournament is 20%, 40% and 60% of the number of orders in a scheduling problem.
- Crossover: OX, CX and OBX.
- Crossover probability: 0.0, 0.1, 0.2, 0.3, 0.4 and 0.5.
- Mutation: shift-mutation, pairwise interchange neighborhood, API neighbourhood and inversion neighbourhood.
- Mutation probability: 0.0, 0.005, 0.01, 0.015, 0.02 and 0.03.
- Formation (elite strategy): the best chromosome or a specific percentage of the best chromosomes is in the next population and all others are chosen by accident. The percentage is: 5%, 10%, 15%, 20%, 30% and 40% of the number of orders in a scheduling problem.
- Stopping criteria: after 30%, 40%, 50% and 60% of the number of orders in a scheduling problem.

Initial population is generated either by accident or by one or more priority rules. Over the last decades, many priority rules are suggested and analysed and due to the dynamic environments in industrial practise, priority rule are still analysed in many studies on scheduling – one example of a recent one is (El-Bouri 2012) – and they are often used in industrial practise. An investigation about priority rules to this real world application is presented by the author in (Herrmann 2013). This investigation shows that the calculation of the processing time of an order, which is assigned to the flow shop next, is critical for the performance of the priority rule. The processing time depends on the next jobs on the flow shop. In (Herrmann 2013) it is shown that a calibration of such a tail is possible and with this constant tail the priority rules delivers often better results. Especially, the most successful priority rules benefits from this setting.

The following priority rules – as in (Engell et al. 1994) or (El-Bouri 2012) – are used. In the definition t is the current time, f_i the due date of job i, tt_i the total processing time of i, calculated with tail or as sum of operation times (net processing time) and a low value is always preferred – without the named exceptions:

- First in first out = t_i, t_i is the arrival time in queue in front of the flow shop and last in first out = t_i, where a high value is preferred.
- Shortest processing time = tt_i.
- Longest processing time = tt_i; here a high value is preferred.
- Earliest finishing time = $t_i + tt_i$.
- Earliest due date = f_i is here identical with earliest operational due date (because only the first station is scheduled) and modified earliest due date = $\max\{t + tt_i, f_i\}$.
- Slack = $f_i - t - tt_i$ is identical with slack per remaining number of operation = $\frac{f_i - t - tt_i}{M}$ and slack per remaining processing time = $\frac{f_i - t - tt_i}{tt_i}$.
- Truncated shortest processing time with parameter r = $\min\left\{tt_i + r, \frac{f_i - t - tt_i}{M}\right\}$.
- CR+SPT $= \begin{cases} \frac{f_i - t}{tt_i}, & f_i - t - tt_i > 0 \\ tt_i, & f_i - t - tt_i \le 0 \end{cases}$.
- CR = $\frac{f_i - t}{tt_i}$.
- RR = $(f_i - t - tt_i)\cdot e^{-\eta} + e^{\eta}\cdot tt_i$, (by (Raghu and Rajendran 1993)) with utilisation level η of the entire flow shop defined by $\eta = \frac{b}{b+j}$ with b being the busy time and j being the idle time of the entire flow shop.
- RM (by (Rachamadugu and Morton 1982)) with priority index: $\frac{1}{\pi_i}\cdot e^{-\frac{k}{t}\cdot\max\{f_i - t - tt_i, 0\}}$ with either local processing time costing $\pi_i^l = tt_i$ (called RM local) or global processing time costing $\pi_i^g = \sum_{i \in U_t} tt_i$, where U_t is the set of unfinished jobs in the pool of orders, excluding job i (called RM global).

Since mean tardiness is most important these small experiments are just evaluated by this criterion.

An analysis of the schedules determined by the genetic algorithm shows in some cases that an improvement is achieved by an empty station. Especially in a rolling scheduling there occur cycles of orders in which at least one order starts in the next period and another starts in the previous period and often it is beneficial to split this cycle in 2; so an empty station occurs. The procedure of priority rules cause in such cases an empty station and outperforms the genetic algorithm even if the sequences of orders calculated by a priority rule is in the initial population. Technically, an empty station in the genetic algorithm is achieved by an artificial order i whose duration time on each station is zero and whose release date is less than the release dates of all normal jobs, so i can start immediately. By a huge due date of i, no tardiness occur; thus, there is no effect on the objective function. Of course, such an artificial order could be not only beneficial at the end of a period, but also in between; even with high workloads (or time pressure, respectively), which occurs in the following experiments quite often. A preliminary study shows that 12 such artificial orders are sufficient for all experiments; also for the ones with the real word application.

The experiments shown that fitness-proportional selection outperforms tournament selection. In any case it is very beneficial that 20% of the best chromosomes survive by elite strategy. The population size has a large impact on the performance. An increase of the above named numbers until 400 causes a decrease, at the beginning a significant decrease, of the mean tardiness. A further increase (from 400) causes a (moderate) increase of the mean tardiness. An initial population of chromosomes generated by one or more priority rules is outperformed by a population of accidentally generated chromosomes. Such an initial population is improved by a mixture of both ways of determining an initial population. The best results are achieved by 15 copies of a chromosome generated by a single priority rule in the initial population. This is executed for each of the above mentioned priority rules. Calculating the processing time via a fix tail is better than using the net processing time. In addition, a sequence of orders is determined by the NEH heuristic with mean tardiness as objective function and for presorting the list of orders each of the above named priority rules are used. Then, the initial population is filled up with accidentally generated chromosomes.

A major impact has the probability of crossover and mutation. As reported in other publications as well a high crossover probability is beneficial. The best mean tardiness is achieved by a probability of 0,9; it may be noted that the results by a probability of 0,4 is 4,4% above and by a probability of 0,4 it is 13,4% above. Compared to this the impact of the mutation probability is less important. The best value is achieved by a low probability of 0,005 which is just 0,74% better than the one by a probability of 0,02. These values are measured for the order crossover operator and the inversion neighbourhood mutation operator. This combination of operators (for crossover and mutation) delivers also for other probabilities (of these two operators) the best mean tardiness.

In any case a large number of iterations is beneficial, because with the elite strategy very good solutions survive over the generations. A stopping after 30% of the number of orders in a scheduling problem delivers nearly in the all cases the smallest mean tardiness.

4. COMPUTATIONAL RESULTS

The real world application is simulated for the sequence of orders explained in section 2. If an assignment of an operation on the first station ends in the next period t, the orders of period (t+)1 is realised, so that the orders of the periods t and (t+1) are known. Average tardiness $\left(T_{Mean}\right)$ and standard deviation of the tardiness $\left(T_{\sigma}\right)$ reach a steady state by a simulation horizon of 5000 periods.

Due to the results published in (Herrmann 2013) genetic algorithm (GA) with the parameter setting due to the above calibration is compared to the results of the best priority rules, which are RR and RM local. The results shown in Table 3 are average objective values relative to the solutions of GA set to 100%; thus, e.g., the solutions generated by RR rule for time pressure 30% were about 51% above GA on the average.

Table 3: Relative performance measures of the best priority rules compared to the genetic algorithm (GA)

Rule	Time pressure			
T_{Mean}	30%	50%	70%	85%
GA	100	100	100	100
RR	1,51	1,41	1,33	1,21
RM local	1,75	1,69	1,51	1,4
RM global	2,93	2,51	2,2	1,99
T_{σ}	30%	50%	70%	85%
GA	100	100	100	100
RR	1,21	1,19	1,13	1,09
RM local	2,4	1,9	1,63	1,51
RM global	3,5	2,1	1,64	1,42

The genetic algorithm outperforms the priority rules significantly. At the company site the benefit would even much higher, because their scheduling procedure is much simpler (than these priority rules).

Without the exception of the priority rules for generating chromosomes for the initial population, just standard operators are used in the genetic algorithm. A first analysis of optimal schedules of the above test problem shows that, for example, outliers in the cycle times are avoided, but priority rules have them often. It could be beneficial to have chromosomes with such a property in a (initial) population and operators who use them.

5. CONCLUSIONS

This paper presents a real world flow shop scheduling problem with more restrictive restrictions than the ones normally regarded in literature. Despite of standard operators in the genetic algorithm of this publication, it outperforms the improved priority rules in (Herrmann 2013) substantially. A further improvement seems to be possible by integrating specific properties of very good schedules, especially in the operators.

More technical restrictions in companies occur by limited resources, like the available number of coils or assembly ground plates, and workers for the manual tasks causes other schedules to be optimal or at least very good. Such requirements are also left to future investigations.

REFERENCES

Januario, T.; J. Arroyo and M. Moreira. 2009. “Genetic Algorithm for Tardiness Minimization in Flowshop with Blocking”. In: N. Krasnogor, B. Melián, J. Moreno, J. Moreno-Vega, D. Pelta (Editors): “Nature Inspired Cooperative Strategies for Optimization (NICSO 2008)”. Studies in Computational Intelligence Volume 236, 2009, 153 – 164.

El-Bouri, A. 2012. “A cooperative dispatching approach for minimizing mean tardiness in a dynamic flowshop”. *Computers & Operations Research*, Volume 39, Issue 7 (July), 1305 – 1314.

Chaudhry, I.; S. Mahmood. 2012. “No-wait Flowshop Scheduling Using Genetic Algorithm”. In Proceedings of the World Congress on Engineering 2012 Vol III, WCE 2012, July 4 - 6, 2012, London, U.K..

Engell, S.; F. Herrmann; and M. Moser. 1994. “Priority rules and predictive control algorithms for on-line scheduling of FMS”. In *Computer Control of Flexible Manufacturing Systems*, S.B. Joshi and J.S. Smith (Eds.). Chapman & Hall, London, 75 – 107.

Framinan, J.M.; R. Leisten; and C. Rajendran. 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 – 148.

Grabowski, J. and J. Pempera. 2007. "The permutation flow shop problem with blocking. A tabu search approach.". *Omega*, 35 (3), 302 – 311.

Hall, N.G. and C. Sriskandarajah. 1996. "A survey of machine scheduling problems with blocking and no-wait in process". *Operations Research* 44 (3), 510–525.

Herrmann, F. 2013. "Simulation based priority rules for scheduling of a flow shop with simultaneously loaded stations". In: Proceedings of the 27th EUROPEAN Conference on Modeling and Simulation, May 27th – 30th, 2013, Ålesund, Norway.

Jacobs, F.R.; W. Berry; D. Whybark; T. Vollmann. 2010. "Manufacturing Planning and Control for Supply Chain Management". McGraw-Hill/Irwin (New York), 6 edition.

Lawrence, S. and T. Morton. 1993. "Resource-constrained multi-project scheduling with tardy costs: Comparing myopic, bottleneck, and resource pricing heuristics.". *European Journal of Operational Research* 64, 168 – 187.

McCormick, S.T.; M.L. Pinedo; S. Shenker; and B. Wolf. 1989. "Sequencing in an assembly line with blocking to minimize cycle time". *Operations Research*, 37 (6), 925 – 935.

Nawaz, M.; E.E. Enscore; and I. Ham. 1983. "A heuristic algorithm for the m-machine, n-job flow sequencing problem". *Omega*, 11(1), 91 – 95.

Pan, Q.; and L. Wang. 2012. "Effective heuristics for the blocking flowshop scheduling problem with makespan minimization". *Omega*, 40 (2), 218 – 229.

Rachamadugu, R.M.V. 1987. "Technical Note –A Note on the Weighted Tardiness Problem". *Operations Research*, 35, 450 – 452.

Rachamadugu, R.V. and T.E. Morton. 1982. "Myopic heuristics for the single machine weighted tardiness problem". Working Paper No. 28-81-82, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA.

Raghu, T.S. and C. Rajendran. 1993. "An efficient dynamic dispatching rule for scheduling in a job shop". *International Journal of Production Economics* 32, 301 – 313.

Rajendran, C and O. Holthaus. 1999. "A comparative study of dispatching rules in dynamic flowshops and job shops". *European Journal of Operational Research*, 116 (1), 156 – 170.

Ribas, I.; R. Companys; and X. Tort-Martorell. 2011. "An iterated greedy algorithm for the flowshop scheduling problem with blocking". *Omega*, 39, 293 – 301.

Ronconi, D.P. and V.A. Armentano. 2001. "Lower Bounding Schemes for Flowshops with Blocking In-Process". *Journal of the Operational Research Society*, 52 (11), 1289 – 1297.

Ronconi, D.P. 2004. "A note on constructive heuristics for the flow-shop problem with blocking". *International Journal of Production Economics*, 39 – 48.

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.

Ronconi, D. and L. Henrique. 2009. "Some heuristic algorithms for total tardiness minimization in a flow shop with blocking". *Omega*, 37 (2), 272 – 281.

Russel, R.S.; E.M. Dar-El; and B.W. Taylor. 1987. "A comparative analysis of the COVERT job sequencing rule using various shop performance measures". *International Journal of Production Research*, 25 (10), 1523 – 1540.

Taillard, E. 1993. "Benchmarks for basic scheduling problems". *European Journal of Operational Research*, 64 (2), 278 – 285.

Vepsatainen, A.P. and T.E. Morton. 1987. "Priority rules for job shops with weighted tardiness costs". *Management Science* 33/8, 95 – 103.

Voß, S. and A. Witt. 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 – 458.

Wang, X. and L. Tang. 2012. "A discrete particle swarm optimization algorithm with self-adaptive diversity control for the permutation flow shop problem with blocking". *Applied Soft Computing*, (12, 2), 652 – 662.

Wang, L.; Q.-K. Pan; and M.F. Tasgetiren. 2011. "A hybrid harmony search algorithm for the blocking permutation flow shop scheduling problem". *Computers & Industrial Engineering*, 61 (1), 76 – 83.

Wang, C.; S. Song, S.; J.N.D. Gupta; and C. Wu. 2012. "A three-phase algorithm for flowshop scheduling with blocking to minimize makespan". *Computers & Operations Research*, 39 (11), 2880 – 2887.

Werner, F. 2013. "A survey of genetic algorithms for shop scheduling problems". In: Heuristics. - New York, Nova Publishers, 133 – 160.

AUTHOR BIOGRAPHY

Frank Herrmann was born in Münster, Germany and went to the RWTH Aachen, where he studied computer science and obtained his degree in 1989. During his time with the Fraunhofer Institute IITB in Karlsruhe he obtained his PhD in 1996 about scheduling problems. From 1996 until 2003 he worked for SAP AG on various jobs, at the last as director. In 2003 he became Professor for Production Logistics at the University of Applied Sciences in Regensburg. His research topics are planning algorithms and simulation for operative production planning and control. His e-mail address is Frank.Herrmann@OTH-Regensburg.de and his Web-page can be found at www.hs-regensburg.de/frank.herrmann