ELSEVIER

Int. J. Production Economics 68 (2000) 199–213

international journal of production economics

www.elsevier.com/locate/dsw

A heuristic solution concept for a generalized machine sequencing problem with an application to radiator manufacturing

G. Zäpfel*, M. Wasner

Institute of Production Management, University of Linz, A-4045 Linz-Auhof, Austria

Received 16 April 1998; accepted 9 October 1998

Abstract

The job shop scheduling problem has received considerable attention from production and operations researchers since the early 1950s. However, scheduling practice in manufacturing systems up to now has tended to ignore the wealth of methods, techniques and results of scheduling theory. Relatively few articles have reported on successful applications of this theory. This paper considers a real problem in practice, namely a generalized job shop scheduling problem for a supplier for the automotive industry, which can be seen as typical for this line of business.The problem is described by a mixed integer programming model. Because an exact solution concept is not suited for the practical problem (the dimension of the real problem is too large) an approximative search-based algorithm is developed. Simulation results that show the qualitity level of the approximative scheme are reported. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Generalized job shop scheduling; Case study; Radiator industry; Local search method

1. Problem

Nowadays just-in-time delivery especially challenges the supplier industry. Just-in-time obligates the supplier to deliver the material to the customer when it is needed. Delivery times are determined by the current demand of the customer's production and assembly unit (e.g., the automotive industry). The supplier must guarantee that customer orders can be finished just in time; that is, all operations of the jobs resulting from customer orders are executable in the technological sequences on the machines within the planned and contracted delivery time and all necessary tools are available. For the supplier industry this is a challenging task because it poses a complex machine sequencing problem, as the following real case study shows for a supplier of the automotive industry.

We consider a supplier who produces car radiators. Altogether, the firm offers 40 different car radiators with an average of 11 operations per job. Examples include: bending the metals, assembly operations and packaging. The production department consists of 15 work centers, which are able to execute a typical process stage (e.g., packaging). The work centers partly comprise several identical machines, which means that an operation that is assigned to a certain work center can be executed

* Corresponding author.
E-mail address: guenther.zaepfel@jk.uni-linz.ac.at (G. Zäpfel).

on any of these parallel machines. Each car radiator requires a sequence of operations in a certain order (technological sequence). Each operation is assigned to exactly one predetermined work center. The technological sequence is different for each radiator. For certain radiator types some work centers can be omitted and it is possible that processing a radiator type necessitates visiting a work center more than once. Furthermore, certain tool restrictions apply; that is, processing a workpiece at a work center requires a certain type of tool, of which only a limited number are available (the number of available tools of a given tool type is less than the number of parallel machines). A fixed assignment of a certain tool type to a work center is given, but the tools of each tool type can be used by any machine belonging to this work center.

To determine the machine sequencing plan, the supplier also has to take set-up times into consideration. Set-up times are relevant for cases where an operation immediately follows another operation on the same machine and a *different* tool type must be used. Therefore sequence-dependent setup times must be included in the analysis.

Table 1 shows the dimensions of the machine sequencing problem for the real case.

In each planning period 15–20 different radiators must be produced. Of a certain radiator type, more than one unit can be required by the customer, whereby on average the lot size is about 10. This means that 150–200 jobs must be executed within a planning period, and the resulting machine sequencing problem has to assign 1650 ($= 150 \times 11$) to 2200 ($= 200 \times 11$) operations to machines and tools.

Table 1
Dimensions of the real problem (planning horizon = 1 week)

Average number of jobs per planning horizon	185
Average number of operations per job	11
Number of work centers	15
Number of total machines	65
Maximum number of parallel machines	10
Minimum number of parallel machines	1
Number of tool types	116
Number of total tools	134
Maximum number of identical tools	2
Minimum number of identical tools	1

The problem of the supplier can now be defined as follows: For given customer orders (variants and quantities of car radiators) for the planning horizon (in this case one week), a machine sequencing plan must be determined which minimizes the maximum flow (throughput) time for all the jobs (called the makespan) and fulfills the above restrictions. The objective function of makespan makes sense for this type of problem because we are searching for a solution that executes all customer orders on time within the planning horizon. If this fails, further courses of actions are taken, e.g., introducing overtime.

In the following we define more precisely the resulting problem of machine sequencing and develop a solution concept which is applicable for real cases.

2. A generalized job shop scheduling problem with parallel machines, sequence-dependent setup times and tool constraints

The generalized job shop problem can be characterized as follows: A set of operations $V = V_1 \cup V_2 \cup \cdots V_K \cup \{0, n\}$ is given, whereby V_k denotes the set of operations of job k, $k = 1, \ldots, K$, and "0" is the dummy operation "start" (the first operations of *all* jobs) and "n" the last operation of *all* jobs. There are precedence constraints among operations of the same job, i.e. for all $(i, j) \in A_k$ there are precedence relations, whereby A_k is the set of ordered pairs of operations constrained by the precedence relation of job k, where $k = 1, \ldots, K$. No precedence relations exist among operations of different jobs.

Each operation i must be processed on a predetermined work center $s(i)$, which consists of a set of machines. Any operation on a work center s can be assigned to any machine $m \in M_s$ (set of machines belonging to work center s). Each operation can be performed only on one machine at a time. Furthermore, to process any operation i, a certain type of tool $t(i)$ is necessary. For each type of tool t there can be a set of identical tools which we denote by W_t. Any operation i that requires tool type t can be assigned to any tool $w \in W_t$.

For a given technological sequence of the operations of each job on work centers, the problem is to find for every operation an assignment of machines and tools so that the maximum throughput time of all the jobs (makespan) is minimized.

We use the following additional symbols:

Variables:

t_i = starting time of operation i,

$$x_{im} = \begin{cases} 1, & \text{if operation } i \text{ is assigned on machine } m \\ 0, & \text{otherwise,} \end{cases}$$

$$y_{iw} = \begin{cases} 1, & \text{if operation } i \text{ is assigned to tool } w, \\ 0, & \text{otherwise,} \end{cases}$$

$$\delta_{ij} = \begin{cases} 1, & \text{if operation } i \text{ is assigned to the same machine or the same tool after operation } j, \\ 0, & \text{otherwise.} \end{cases}$$

Parameters:

a_i = processing time of the operation i,

r_{ij} = set up time between operation i and j, where[1]
$$\begin{cases} r_{ij} > 0 & \text{if } s(i) = s(j) \wedge t(i) \neq t(j), \\ r_{ij} = 0 & \text{otherwise,} \end{cases}$$

C_1, C_2 = very large constant.

Sets and indices:

E_s = set of all pairs of operations to be performed on work center s,

M = $\bigcup_{s=1}^{S} M_s$ = set of all machines where M_s is the set of machines belonging to work center s, $s = 1, \ldots, S$: indices of the work centers,

W_t = set of tools of tool type t.

The generalized job shop sequencing problem can be described by the following optimization problem:

$$\min t_n$$

$$t_j - t_i \geqslant a_i, \quad \forall (i,j) \in A_k, \ k = 1, \ldots, K, \tag{1}$$

$$C_1(1 - x_{im}x_{jm}) + C_2\delta_{ij} + t_j - t_i \geqslant a_i + r_{ij}, \tag{2a}$$

$$C_1(1 - x_{im}x_{jm}) + C_2(1 - \delta_{ij}) + t_i - t_j \geqslant a_j + r_{ji},$$
$$\forall \{i,j\} \in E_s, \ \forall m \in M_s, \quad s = 1, \ldots, S \tag{2b}$$

$$C_1(1 - y_{iw}y_{jw}) + C_2\delta_{ij} + t_j - t_i \geqslant a_i, \tag{3a}$$

$$C_1(1 - y_{iw}y_{jw}) + C_2(1 - \delta_{ij}) + t_i - t_j \geqslant a_j,$$
$$\forall \{i,j\} \in E_s, \ s = 1, \ldots, S, \forall w \in W_{t(i)} = W_{t(j)}, \tag{3b}$$

$$\sum_{m \in M_{s(i)}} x_{im} = 1, \quad \forall i \in V \backslash \{0, n\}, \tag{4}$$

$$\sum_{w \in W_{t(i)}} y_{iw} = 1, \quad \forall i \in V \backslash \{0, n\}, \tag{5}$$

$$t_i \geqslant 0, \quad \forall i \in V$$

$$x_{im} \in \{0, 1\}, \quad \forall i \in V \backslash \{0, n\}, \forall m \in M_{s(i)},$$

$$y_{iw} \in \{0, 1\}, \quad \forall i \in V \backslash \{0, n\}, \forall w \in W_{t(i)},$$

$$\delta_{ij} \in \{0, 1\}, \quad \forall \{i,j\} \in E_s, s = 1, \ldots, S. \tag{6}$$

The formulated machine sequencing problem is a generalized job shop problem. This can be seen as follows: If every work center consists of only one parallel machine (thus each operation is assignable a unique machine and tool), we get in the restrictions (2a), (2b), (3a) and (3b): $x_{im}x_{jm} = 1$ and $y_{iw}y_{jw} = 1$. Furthermore, if no setup time occurs ($r_{ij} = 0$, $\forall i,j \in E_s, s = 1, \ldots, S$) and E_m denotes the set of operations executable on machine $m \in M$, we get the classical job shop sequencing problem:

$$\min t_n$$

$$t_j - t_i \geqslant a_i, \quad \forall (i,j) \in A_k, k = 1, \ldots, K,$$

$$C\delta_{ij} + t_j - t_i \geqslant a_i, \quad \forall \{i,j\} \in E_m, m \in M,$$

$$C(1 - \delta_{ij}) + t_i - t_j \geqslant a_j, \quad \forall \{i,j\} \in E_m, m \in M,$$

$$t_i \geqslant 0, \quad \forall i \in V,$$

$$\delta_{ij} \in \{0, 1\}, \quad \forall \{i,j\} \in E_m, m \in M.$$

The presented machine sequencing problem is a generalization of the classical job-shop problem, which belongs to the most intractable computational problems considered [1,2,14,15]. A problem with 10 jobs and 10 machines which was formulated in 1963 [3] was solved optimally 26 years later [4]. It is well known that the classical job

[1] As described in Section 1, a fixed assignment of a certain tool type to a workcenter is given. If $s(i) \neq s(j)$ the appropriate tool types are different and thus a setup time does not exist; it is zero.

shop scheduling problem is NP-hard [5]. We have at least the same difficulty of finding an optimal solution for the generalized job-shop scheduling problem mentioned in Section 1. Exact methods, for example, the branch and bound algorithms (e.g., [6]), are not suited for the dimensions of a practical problem. Therefore we also propose an approximative method, based on a local search algorithm to solve the generalized job shop sequencing problem.

The literature presents other heuristics solution methods for generalized classical job shop problems. Especially, the machine sequencing problem has been studied for flexible manufacturing systems (e.g. [7]). A set of jobs, each consisting of a number of operations, have to be processed on a flexible manufacturing system. The production system encompasses different multi-purpose machines which can be equipped with different tools. An operation can be processed on every machine which is equipped with the specific tool. The goal is a feasible assignment of operations to machines such that the total schedule length is minimized. The solution strategy belongs to the local search concept in conjunction with the tabu search techniques. In comparison to our problem, no setup times are taken into consideration and the assumption is made that the multi-purpose machines in the system are already equipped with tools.

In the following we propose an approximation method based on a local search algorithm; our method is able to solve the generalized sequencing problem discussed in Section 2.

3. A solution concept for the generalized job shop scheduling problem

The proposed solution concept for the generalized job shop scheduling problem belongs to the class of approximation methods. The core of the concept consists of a fast problem-specific algorithm for the generalized job shop scheduling problem (called GJS algorithm) embedded in a local search method. The concept can be seen as an extension of the solution strategy of Storer et al. [8], which was developed for the classical job shop scheduling problem.

The GJS algorithm is a generalized version of the Giffler and Thompson algorithm [9].
We employ the following symbols:

V^*	$= V\backslash\{0, n\}$ set of all operations,
S	set of operations which can be assigned in the current iteration step (assignable operations),
M	set of all machines,
W	set of all tools,
α	set of all operations which must be executed on the first technological position of each job,
t_i^0	earliest possible starting time of operation i,
d_i^0	earliest possible completion time of operation i,
T_m	current time availability of machine $m \in M$,
Ta_w	current time availibilty of tool $w \in W$,
Δ_m	operation which was assigned last on machine m,
r_{ij}	setup time which results if operation j follows i; $r_{0i}{:} = 0, \forall i \in V^*$,
r_s	maximum setup time in work center s, i.e. $r_s{:} = \max\{r_{ij}{:}\{i, j\} \in E_s\}$,
$s(i)$	work center where operation i must be processed,
$n(k)$	operation which immediately follows operation k due to the technological order,
$\bar{E}_s$	operations which must be executed on work center s.

The GJS-algorithm can be described as follows:

Step 1 (Initialization)
$S := \alpha$,
$t_i^0 := 0, i \in S$,
$T_m := 0, m \in M$,
$Ta_w := 0, w \in W$,
$\Delta_m := 0, m \in M$.

Step 2 (Determination of the conflict set)
Among all operations $i \in S$ let k^* be the one with the smallest completion time; i.e., $d_{k^*}^0 = \min\{d_i^0 := t_i^0 + a_i, i \in S\}$. Let $s(k^*)$ denote the work center where k^* has to be processed.

Step 3 (Selection of an operation)
Randomly choose any operation k from the conflict set K; that is, $k \in K$ (the conflict set K will be derived below).

Step 4 (Selection of a machine and tool and update of time parameters)
$t_k := t_k^0$,
$s := s(k)$, $v := t(k)$,
Select any machine $m^* \in \{m \in M_s \wedge T_m + r_{\Delta_m,k} \leqslant t_k^0\}$,
Select any tool,
$w^* \in \{w \in W_v \wedge Ta_w \leqslant t_k^0\}$,
$T_{m^*} := t_k + a_k$,
$\Delta_{m^*} := k$,
$\mathrm{Ta}_{w^*} := t_k + a_k$,

$$t_i^0 := \max_{i \in S \cap \bar{E}_s \setminus \{k\}} \begin{cases} t_i^0, \\ \min\{T_m + r_{\Delta_m,i}: m \in M_{s(k)}\}, \\ \min\{\mathrm{Ta}_w: w \in W_{t(k)}\}. \end{cases}$$

$$t_i^0 := \max \begin{cases} t_k + a_k, \\ \min\{T_m + r_{\Delta_m,i}: m \in M_{s(i)}\}, \quad i := n(k), \\ \min\{\mathrm{Ta}_w: w \in W_{t(i)}\}. \end{cases}$$

Step 5 (Update set of assignable operations)
$S := S \setminus \{k\} \cup \{n(k)\}$.

Step 6 (Stop condition)
If $|S| > 0$ go to step 2.
Otherwise stop and print schedule.

In principle, the GJS algorithm is characterized by selecting in each iteration one operation from a conflict set K and assigning to this operation – denoted by k – a starting time, a machine and a tool. In continuation, the earliest starting times of all the other operations belonging to the current assignable operations as well as the earliest starting times of the immediate technological successors (as available) are determined. These steps must be carried out as long as assignable operations exist ($|S| > 0$). The steps are identical to a forward scheduling algorithm.

The determination of the conflict set K and the selection $k \in K$ in each iteration are the most cruical steps in the GJS algorithm. We are looking for a conflict set such that $K \subset S$ is as small as possible and which guarantees the optimum solution if we select the right operation in each iteration in step 3. The following theorem can be proved for the GJS algorithm (where $p(k)$ denotes the technological predecessor of operation k, if it exists):

Theorem. *If we select the right operation $k \in K$ in step* 3, *whereby*

$$K = \left\{ i : i \in S \wedge \left[\begin{array}{l} (s(i) = s(k^*) \wedge t_i^0 < d_{k^*}^0 + r_{ik^*}) \vee \\ (s' := s(i) \neq s(k^*) \wedge t_i^0 < d_{k^*}^0 + r_{s'} \wedge \\ \quad \exists_{\substack{j \in S \\ s(j) = s(i)}} \quad d_j^0 < d_{k^*}^0 + r_{s'}) \end{array} \right] \right\}$$

and for all operation $i \in Z = V_1 \cup V_2 \cup \cdots \cup V_K \setminus \alpha$ holds $a_{p(i)} > r_{s(i)}$ then the GJS algorithm always generates the optimum solution for every regular measure of performance. This means

- *the scheduling objective is to minimize a function of the set of job completion times, and*
- *the function can increase only if at least one of the completion times in the schedule increases.*

Proof. See Wasner [10] □

Remark. For the case of the *classical* job shop scheduling problem, we can simplify $\bar{E}_s := O_m$ where O_m is the set of operations which must be processed on machine m. The conflict set reduces to $K = \{i: i \in S \cap O_m \wedge t_i^0 < d_{k^*}^0\}$. This special conflict set is the same used by Giffler and Thompson [9].

Comparing the two conflict sets, we can deduce that they are equivalent if no parallel machines exist and all setup times are zero. The existence of setup times is the reason for the difficult construction of the above conflict set. We have to consider operations from different work centers (only because the setup time is not zero). In the classical job shop scheduling problem, all setup times are zero and therefore $a_{p(i)} > 0$ is always fullfilled.

According to the theorem, the GJS algorithm always generates the optimal solution if the right operation in step 3 is selected. But we have to keep in mind that the theorem only postulates an existence proof and we cannot deduce which operation k (if there is more than one) must be selected. To solve the conflict, priority rules are often used. For the simulation results of Section 4 priority rules given in Table 2 have been implemented.

For a selection of priority rules, the GJS algorithm generates a feasible solution for the generalized job shop sequencing problem; this solution must

Table 2

Rule	Description
Earliest completion time rule	Operation k is assigned the highest priority which can be completed earliest, i.e., $d_k^0 = \min\{d_i^0: i \in K\}$
Earliest starting time rule	Operation k with the earliest possible starting time is assigned the highest priority, i.e., $t_k^0 = \min\{t_i^0: i \in K\}$.
Waiting time rule	Operation k with the longest waiting time on the machine is assigned the highest priority, i.e., $t_k^0 - d_{p(k)} = \max\{t_i^0 - d_{p(i)}: i \in K\}$.
Longest remaining processing time rule	Operation $k \in K$ of a job is assigned the highest priority where the *remaining processing time is maximum.*
Longest total job time rule	Operation $k \in K$ of a job is assigned the highest priority where the sum *of all operation times of the job is maximum.*
Random rule	Operation k is chosen randomly.

not be the most favorable. To increase the chance of finding "good solutions" of the machine sequencing problem, the GJS algorithm is embedded in a *local search concept*: starting with a feasible solution (fulfilling the restrictions of the model in Section 2), the general idea is to modify the current solution by slightly perturbing it (searching a neighborhood solution). The neighborhood solution is compared with the current solution and accepted if an improved objective function results for the neighborhood solution (and becomes the new current solution). One of the most important tasks in constructing the local search concept is the definition of a neighbor. For our concept we divide the time axis into arbitrary intervals and denote these with the term time windows in accordance with the paper of Storer et al. [8]. Each time window is assigned exactly one priority rule, whereby some time windows can be assigned the same priority rule.

Neighborhood ***N*** can then be defined more precisely as follows:

N: A time-window priority assignment Z_1 is a neighbor of an assignment Z_2 iff Z_1, Z_2 are different in exactly one time-window priority assignment.

Based on the above definition, the following local search concept will be applied:

Initialization: Determine an arbitrary time-window priority assignment as well as a current (feasible) solution using the GJS algorithm.

Repeat the steps 1 to 5 n times (n must be chosen):

1. Select an arbitrary time window.
2. Select a neighbourhood (i.e., a priority rule different from the old one).
3. Determine the new solution using the GJS algorithm.
4. Compare the current solution with the new solution. Accept the new solution and the underlying time-window priority assignment if the objective function value improved, and update current solution := new solution. Otherwise current solution := (old) current solution.
5. Go to step 1.

Theoretically, the search space is given by P^W if we denote the number of different priority rules with P and the number of time windows with W.

4. Simulation experiments

4.1. Test problems

In the following we describe the real application of the job shop sequencing problem for a main supplier of the automative industry. As mentioned before, the enterprise produces 40 different types of radiators (for BWM, Mercedes, Ford, etc). Table 3 shows the technical sequence of 16 typical radiator types and their production times as well as the assignment to the work centers.

Table 3
16 radiator types and their operation times

Radiator type	Operations												
	1	2	3	4	5	6	7	8	9	10	11	12	13
12 Work center →	5	12	6	7	11	9	10	13					
Oper. time →	263	24	297	167	291	69	118	123					
14	1	3	2	5	15	6	7	11	9	10	13		
	138	60	45	165	12	360	223	276	69	154	150		
15	1	2	5	12	6	7	8	11	3	11	9	10	13
	135	38	112	16	270	167	50	57	36	93	81	118	95
16	1	2	5	3	12	6	7	11	9	10	13		
	108	38	143	26	24	216	95	105	69	118	95		
17	1	4	2	5	12	6	7	11	9	10	13		
	144	33	44	150	45	243	151	135	69	118	156		
19	1	3	2	5	3	12	6	7	8	9	10	13	
	102	26	38	84	30	17	240	167	75	60	75	95	
21	1	3	2	5	15	6	7	11	9	10	13		
	148	26	39	143	2	304	223	183	69	105	131		
22	1	4	2	5	12	6	7	8	9	10	13		
	147	33	30	83	17	297	167	60	65	72	95		
24	5	12	6	7	8	11	9	10	11				
	225	44	345	167	59	77	69	74	95				
26	1	2	5	3	12	6	7	11	9	10	13		
	141	38	130	38	17	348	216	284	56	75	95		
32	5	12	6	7	12	11	15	13					
	225	44	345	167	17	150	81	150					
33	5	12	6	7	11	9	13						
	471	44	345	167	140	75	150						
35	1	3	2	5	12	6	7	3	11	9	10	13	
	147	23	38	112	16	270	167	36	135	81	118	95	
36	1	3	2	5	12	6	7	11	9	10	13		
	147	36	38	112	16	270	167	93	81	118	95		
37	1	2	5	12	3	6	7	11	15	10	13		
	135	23	225	44	38	345	167	285	81	118	75		
40	5	12	6	7	12	11	9	13					
	225	44	345	240	17	150	75	150					

Table 4 presents the number of parallel machines which are available for the work centers and the maximum set up time arising in a center.

To make a statement about the quality of the proposed solution concept, simulation experiments were carried out. Altogether 100 different generalized machine scheduling problems had been solved, where the differences between the single machine scheduling problems consisted of different product mixes and quantities. The number of different car radiators, produced in one period, was chosen in a range between 11 and 24, and the mean value was 18. The lot size of one car radiator varied between 1 and 42, and the average was about 10. So the dimension of every machine scheduling problem corresponds to the dimension of the mentioned real case of the supplier.

4.2. *Measure of efficiency for the test problems*

We are not able to determine the optimum solutions due to the resulting dimension of the real case generalized machine sequencing problems.

Table 4
Number of parallel machines and (maximum) setup times

Work centers	Parallel machines	Maximum of the set up time in center (in min)
1	4	20
2	3	5
3	3	15
4	3	15
5	9	30
6	10	15
7	7	0
8	2	0
9	2	0
10	2	0
11	10	15
12	1	0
13	7	15
14	1	0
15	1	0

Therefore the simulation results are always related to the lower bound of the optimum solution, which means the deviation of the resulting makespan from the lower bound of each machine scheduling problem is determined as a measure of efficiency. This can be seen as a meaningful measure because the efficiency of the proposed procedure is oriented at the maximum possible deviation (worst case).

The lower bound LB for our generalized machine sequencing problem is derived as follows:

$$\mathrm{LB} = \max\{\mathrm{LB}_1, \mathrm{LB}_2; \mathrm{LB}_3\}$$

where

$$\mathrm{LB}_1 = \max_{s\in\{1,\dots,S\}} \left\{ \sum_{i\in\bar{E}_s} a_i/\bar{M}_s + \min_{k\in\{1,\dots,K\}} \left\{ \sum_{i\in V_k} a_i\alpha_{is} \right\} + \min_{k\in\{1,\dots,K\}} \left\{ \sum_{i\in V_k} a_i\beta_{is} \right\} \right\} \quad (7)$$

where

$s(i)$	=	work center where operation i has to be processed,
$\bar{M}_s$	=	number of parallel machines of work center s,
$\bar{E}_s$	=	set of all operations to be performed on work center s,
V_k	=	set of operations of job k,
$N(i)$	=	technological successors of operation i,
$P(i)$	=	technological predecessors of operation i,
α_{is}	=	$1 \Leftrightarrow s(j) \neq s, \forall j \in P(i)\cup\{i\}$ ($\alpha_{is} = 0$ otherwise),
β_{is}	=	$1 \Leftrightarrow s(j) \neq s, \forall j \in N(i)\cup\{i\}$ ($\beta_{is} = 0$ otherwise).

LB_1 is based on the estimation of the minimum necessary time of the machine load in every work center. The maximum of all these times of the work centers is taken. To estimate the minimum time for a work center s, we determine the machine load that is the sum of all processing times encompassing all jobs and operations belonging to this center and divide the sum by the number of parallel machines of this center if no delay, i.e., waiting time occurs (the first term of Eq. (1) on the right side). To this term we add the minimum time span that must pass to start an operation of a job on work center s (second term of Eq. (1) on the right) and add the minimum time span that must pass until the jobs are finished utilizing work centers $s \neq s(i)$ (third term of Eq. (1)).

$$\mathrm{LB}_2 = \max_{t\in T} \left\{ \sum_{i\in F_t} a_i/\bar{W}_t + \min_{k\in\{1,\dots,K\}} \left\{ \sum_{i\in V_k} a_i\gamma_{it} \right\} + \min_{k\in\{1,\dots,K\}} \left\{ \sum_{i\in V_k} a_i\delta_{it} \right\} \right\} \quad (8)$$

whereby

$t(i)$	=	tool type that operation i has to be produced on,
T	=	set of all tool types,
$\bar{W}_t$	=	number of parallel tools of tool type t,
F_t	=	set of all operations to be performed with tool type t,
γ_{it}	=	$1 \Leftrightarrow t(j) \neq t, \forall j \in P(i)\cup\{i\}$ ($\gamma_{it} = 0$ otherwise),
δ_{it}	=	$1 \Leftrightarrow t(j) \neq t, \forall j \in N(i)\cup\{i\}$ ($\delta_{it} = 0$ otherwise).

In analogy to LB_1, bound LB_2 is based on the minimum necessary utilization time for a tool type. Term 1 represents the minimum possible time of the tool load for the tool type (without delay). Term 2 takes into consideration the earliest time span in which tool type t can be used by an operation of the jobs, and term 3 expresses the minimum time span which must pass until the jobs are finished utilizing tool types $t \neq t(j)$.

$$\mathrm{LB}_3 = \max_{k \in \{1,\ldots,K\}} \left\{ \max_{\substack{i \in V_k \\ s := s(i) \\ t := t(i)}} \left\{ a_i \left\{ \frac{EQ(k)}{\min(\bar{W}_t, \bar{M}_s)} \right\}^* \right\} + \sum_{i \in V_k \setminus \{\hat{i}_k\}} a_i \right\} \tag{9}$$

where

$\{x\}^*$	=	smallest integer $\geqslant x$,
$\mathrm{EQ}(k)$	=	number of jobs of the production program equal to job k,
$\hat{i}_k$	=	operation $i \in V_k$, where $a_i \left\{ \frac{EQ(k)}{\min(\bar{W}_t, \bar{M}_s)} \right\}^*$ is maximum $(s := s(i); t := t(i))$.

LB_3 is based on the estimation of the maximum of the minimum necessary times of the jobs. This bound can be important if customer orders consist of several identical items. In this case we have the choice to handle the customer order in the machine sequencing problem as one lot or we can divide the quantity of the customer order in sublots. A sublot can be processed on a machine even if the other sublots of the customer order have not yet been processed on the upstream work centers (called *lot streaming or splitting*). If $\mathrm{EQ}(k) = 1$ for all $k \in \{1,\ldots,K\}$, LB_3 is equal to the sum of the processing times of the longest job, a simple lower bound.

We introduce the following example to illustrate the determination of the bound in the case splitting is used: Assume a customer order of 3 items of radiator type 12 and 2 items of type 14 (see Table 3), which is supposed to be equivalent to the production program of this period. If each item defines ist own job (largest possible splitting), we get 5 jobs ($K = 5$), whereby jobs 1, 2 and 3 are equal (representing radiator 12), just as job 4 and 5 (representing radiator 14).

Remark. Because a_i is defined as the processing time of an operation $i \in V_k$ for one item of the radiator type k (see Table 3), the processing time a_i must be multiplied if lot size of this job is larger than 1.

The lower bound LB_3 can be now determined as follows: If lot streaming is not realized, $\mathrm{EQ}(\text{job } \tilde{1}) = \mathrm{EQ}(\text{job } \tilde{2}) = 1$ then $\mathrm{LB}_3 = \max\{3(263 + 24 + 297 + 167 + 291 + 69 + 118 + 123), 2(138 + 60 + \cdots + 150)\} = \max\{(4056, 3304) = 4056\}$, which is equal to the processing time of the longest job.

If each item of a radiator type defines its own job, then the number of jobs of the production program equal to job 1 to job 5 is given by $\mathrm{EQ}(1) = \mathrm{EQ}(2) = \mathrm{EQ}(3) = 3$ and $\mathrm{EQ}(4) = \mathrm{EQ}(5) = 2$. Furthermore, assume $\bar{W}_t = 2$ for all tooltypes t then (see Tables 3 and 4):

$$\mathrm{LB}_3 = \max_{k \in \{1,2,\ldots,5\}} \left\{ \begin{array}{l} \mathrm{jb}_1 = 297 \left\{ \frac{3}{\min(2,10)} \right\} + 263 + 24 + 167 + 291 + 69 + 118 + 123; \mathrm{jb}_2; \mathrm{jb}_3, \\ \mathrm{jb}_4 = 360 \left\{ \frac{2}{\min(2,10)} \right\} + 138 + 60 + 45 + 165 + 12 + 223 + 276 + 69 + 154 + 150; \mathrm{jb}_5 \end{array} \right\}$$

where

$$\mathrm{jb}_k = \max_{\substack{i \in V_k \\ s := s(i) \\ t := t(i)}} \left\{ a_i \left\{ \frac{\mathrm{EQ}(k)}{\min(\bar{W}_t, \bar{M}_s)} \right\}^* \right\} + \sum_{i \in V_k \{\hat{i}_k\}} a_i.$$

Because $\mathrm{jb}_1 = \mathrm{jb}_2 = \mathrm{jb}_3$ and $\mathrm{jb}_4 = \mathrm{jb}_5$ we get $\mathrm{LB}_3 = \max\{1649; 1649; 1649; 1652; 1652\} = 1652$. The lower bound LB_3 is 40.7% lower for the case of splitting.

The advantage of a lot streaming policy can be seen in the fact that makespan decreases. If lot streaming is permitted, it can be shown that the lower bound $\mathrm{LB} = \max\{\mathrm{LB}_1, \mathrm{LB}_2, \mathrm{LB}_3\}$ for the real data of our problem is 70–78% lower than when lot streaming is excluded. We have to pay a price for lot streaming: The number of iterations increases and the computational effort (CPU time) escalates with an increasing number of jobs to be scheduled.

4.3. Simulation results

Fig. 1 shows the simulation results of a special, but typical weekly production program, where the 16 different radiators of Table 3 had to be processed – with lot sizes of 22, 7, 16, 21, 15, 2, 15, 34, 35, 2, 1, 6, 3, 5, 4 and 4. The number of jobs is 192 if each item of a lot (sublot) constitutes a distinct job. The best generated solution from the lower bound of the optimum solution is depicted in dependence on the number of iterations. The chosen number of time windows is 10.

For the 100 simulation experiments described in Section 4.1, Fig. 2 shows the average improvements of the solutions in relation to the number of iterations. More precisely, the deviations of the solutions for the proposed algorithm of Section 3 are compared with the lower bounds after 100, 300, 1000 and 3000 iterations. It can also be seen that the number of time windows hardly influences the quality of results and the mean value x as well as the standard deviation s continuously decrease. It is remarkable that the mean deviation from the lower bound of the optimum solution is less than 6% after 100 iterations.

Fig. 3 shows that the improvements of a solution after 500 iterations are quite small. Further iterations only improve the solution (received after 500 iterations) by not more than 6%.

4.4. Number of iterations and CPU time of the proposed algorithm

The simulation experiments were executed on a 486/33 MHz computer. The average number per minute is 165 iterations. To ensure an average deviation of the solution from the optimum solution of 3%; not more than 2000 iterations are necessary. In this case the simulation effort amount to 12 minutes. The production programm of the supplier in the planning periods are only dependent on the product mix and the lot sizes and therefore on the number of operations which must be planned. More precisely, the computational effort of the proposed solution concept is O(number of operations $\times$ number of jobs). However, for the supplier in total the loading is similar from period to period with a deviation of 10%. So we can conclude that the CPU time for the solution of the real case of the generalized machine sequencing problem is in accordance with the above estimation. Because a more modern PC, for example with a Pentium processor, is much faster at most few minutes are necessary to solve the real case problem of the resulting generalized machine scheduling problem.

4.5. Comparision of the proposed algorithm with simulated annealing

To compare the proposed solution algorithm with a further local search method, a concept of disjunctive graphs in conjunction with simulated annealing was developed to solve the generalized job shop problem [10]. In principle, this disjunctive graph-oriented concept consists of the following steps:

Initialization: Determine a current (feasible) solution described by a generalized disjunctive graph (see Fig. 4).

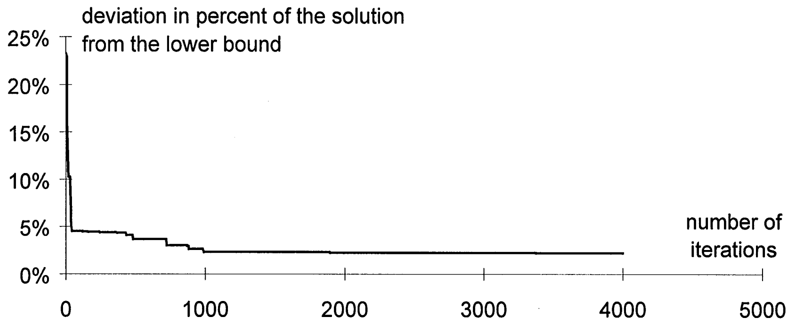

Fig. 1. Deviation of the solution from the lower bound (dependent on the number of iterations).

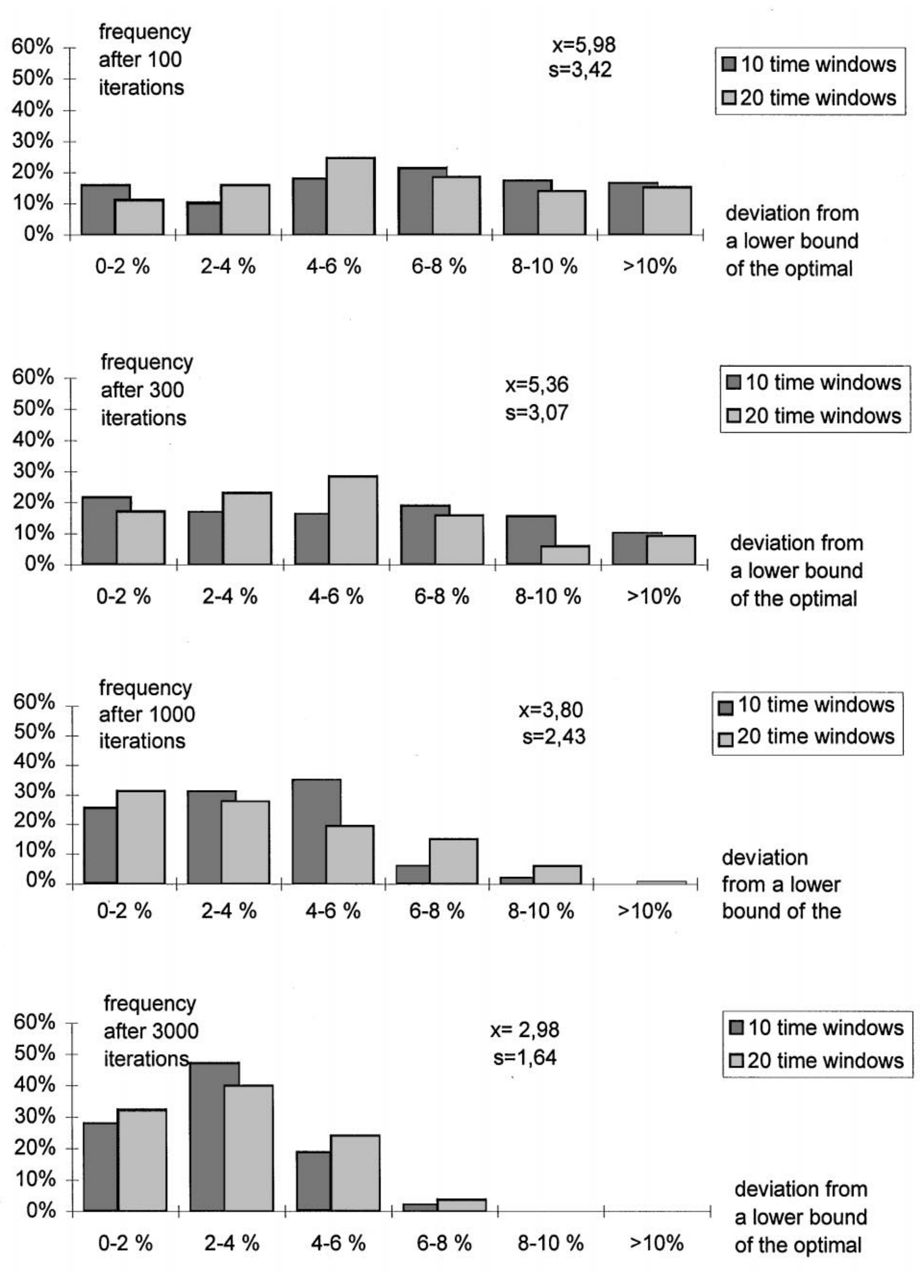

Fig. 2. Bar chart of the deviation of a solution after *n* iterations using 10 or 20 time windows.

Repeat *n* times the steps 1 to 4 (*n* must be chosen).

1. Select a neighborhood of the current solution.
2. Determine the new solution by computing the longest path through the new disjunctive graph.
3. Compare the current with the new solution. Accept the new solution according to the simulated annealing concept, which means:
   If the makespan of the new solution is smaller than the current solution then current solution := new solution.

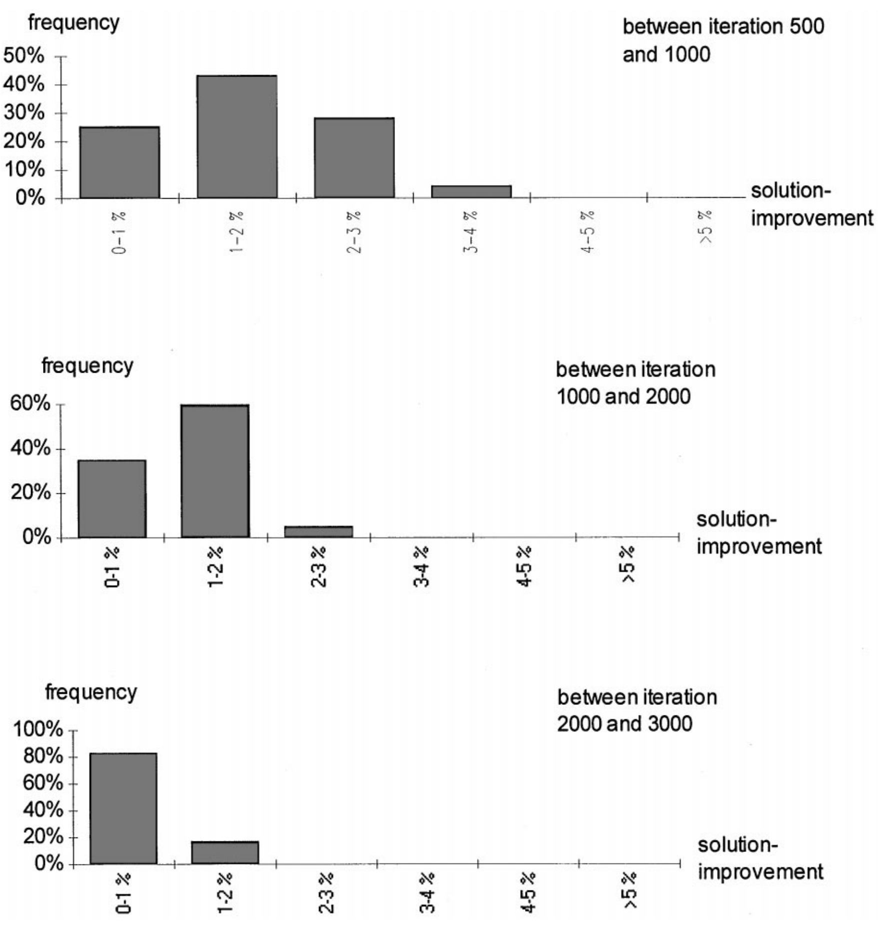

Fig. 3. Solution improvements with an increasing number of iterations.

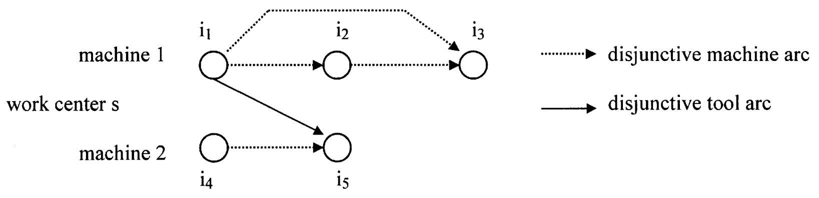

Fig. 4. Cutting of a disjunctive graph.

If the makespan of the new solution is larger, it will nevertheless be accepted with a predetermined probability $p = \min\{1, e^{-\text{delta}/T}\}$. Delta is proportional to the difference between the new and the old (current) solution and T is indirectly proportional to the number of iterations.

4. Goto 1.

The construction of the generalized disjunctive graph is explained in Fig. 4 by an example.

The (generalized) disjunctive graph consists of two kinds of disjunctive arcs (see also [11] for a similar construction). One kind represents the machines (arcs between operations $i_1 - i_2$, $i_1 - i_3$, $i_2 - i_3$, $i_4 - i_5$) and another represents the tools ($i_1 - i_5$). The direction of the arc denotes the technical sequence in which the operations are processed. In the example operation i_1 must be processed on before operations i_2 and i_3. The arc between i_5 and i_1 shows that the two operations use the same tool, where $i_1$is assigned before i_5. Each solution constructed with the GJS algorithm can be quite easily transformed into a disjunctive graph. The neighborhood search for a current solution is based on changing arcs along the critical path. (It can be proved that it is only possible to improve the solution if the longest path through the graph is shortened.) Thus the neighborhood in the disjunctive graph-based local search method is in principle determined by turning along the critical path

1. a disjunctive arc which corresponds to a change of machine assignment, or
2. a disjunctive arc which correspond to a change of tool assignment or by assigning along the critical arc,
3. an operation to another machine of the same work center, or
4. an operation to another tool of the same tool type and assuring that the resulting new graph contains no cycle.

To compare the GJS algorithm and the disjunctive graph-based local search method, three versions have been developed concerning the quality of solutions:

1. a GJS algorithm as described in Section 3,
2. a GJS algorithm embedded in a simulated annealing concept, which also means a solution of a neighborhood (makespan) which is larger than a current solution is accepted with a predetermined probability (see the above formula in step 3),
3. the disjunctive graph concept using simulated annealing as described by the above steps in this section.

To compare the quality of the three concepts, the simulation experiments corresponding to Section 4 were replicated. Fig. 5 shows the results of the comparison. From this diagram it can be seen that the GJS algorithm clearly outperforms the other two concepts using simulated annealing. The GJS algorithm, embedded in a simulated annealing concept, is less attractive in comparison to the GJS concept of Section 3 because the solution quality never is better for the same CPU time (curve 2). The disjunctive graph concept using simulated annealing fails to solve efficiently the generalized machine sequencing problem (curve 3). It brings about solutions that differ more than 30% from the lower bound of the optimum solution. The main reason is that the disjunctive graph concept using simulated annealing is more myopic than the GJS algorithm because it uses less structural information concerning chronological machine and tool assignments.

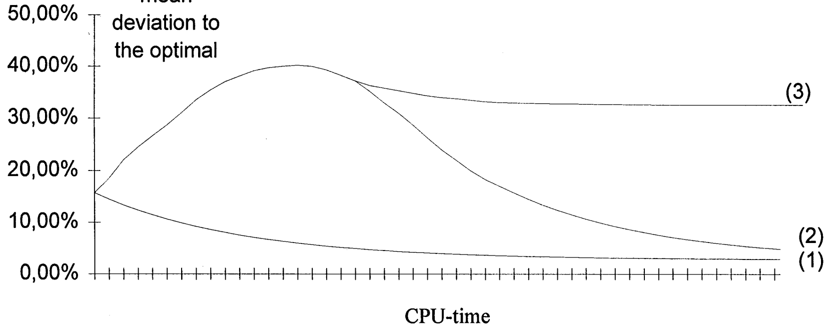

Fig. 5. Comparison of the quality of solutions with (1) GJS algorithm and (2) the GJS-algorithm embedded in simulated annealing versus (3) a disjunctive graph concept using simulated annealing.

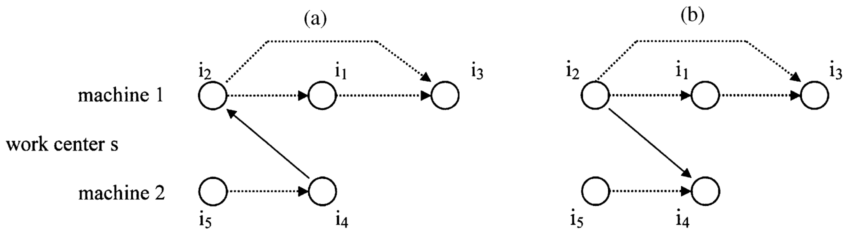

Fig. 6. Insufficient use of structural information can lead to a disjunctive graph represented in Fig. 6a. The GJS-algorithm would automatically generate Fig. 6b.

Fig. 6a shows the modified disjunctive graph of Fig. 4, with the machine arcs i_1–i_2 and i_4–i_5 inverted. Although the solution is feasible (no cycle) and realistic, it does not sufficiently consider simultaneous tool coordination. The GJS algorithm (which is a forward scheduling concerning job operations, machine- and tool assignments) would automatically create the solution of Fig. 6b, where the tool-arc is inverted, too.

Another reason for the worse results with the disjunctive graph model is that the GJS algorithm changes a priority rule to a time-window assignment in every iteration, whereby a priority rule can contain more structural information about "meaningful solutions" of the machine sequencing problem.

5. Summary

We have proposed a model for a generalized job shop machine sequencing problem and a local search procedure to solve the problem. Furthermore, it was demonstrated that the iterative procedure was able to solve the generalized machine sequencing problem within a few minutes and it yields a nearly global optimum for the real case. In summary, the following results can be reported:

- Already after 100 iterations a solution can be achieved which deviates in average less than 6% from the optimum solution (Fig. 2).
- Although the global optimum of the generalized machine sequencing problems could not be identified and the simulation results are related to the lower bound (the quality of this bound is also not known), in average the solutions of the new local search concept only deviate 3% after 3000 iterations (Fig. 2).
- After 2000 iterations solution improvements are hardly to realize (Fig. 3).
- The computational effort to achieve this results is low.
- In the context of machine sequencing lot streaming is an important means to shorten makespan (see also [12]).

It should be emphasized that the above static machine sequencing case can trivially be transformed into a dynamic one with comparable simulation results. Then realistic events like machine repairs during a planning horizon or a later machine or tool availability than at time zero can also be taken into consideration (for this version, see [10]). For example, a delayed machine or tool availability in a planning period for the new jobs results from the loading caused by the production of missing quantities of backorders in the current period.

A topic for future research is the embedding of the presented short-term scheduling concept in an enlarged hierarchical planning system. The hierarchical planning system consists of

- an upper-level planning unit,
- a lower-level planning unit.

The upper-level planning unit can be interpreted as the material and capacity planning system, which must guarantee in the short term that the right material is available in the right quantity and that the planned delivery time for the short-term machine sequencing problem is feasible. In the supplier industry a planning horizon of three months or more is usual and the demand forecast for this planning horizon is uncertain, whereby upper and lower bounds can be estimated.

In the short term, customer orders are known and the presented generalized machine sequencing problem must be solved to guarantee the execution of the orders on time and, if needed, to adapt the capacity of the resources. Material availability and planned delivery times are preconditions of the upper-level planning period.

One of the main problems to be solved in such a hierarchical planning system is the determination of a consistent aggregation and dissagregation process [13]. This is a challenging task for further research.

References

[1] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, Sequencing and scheduling: algorithms and complexity, in: S.C. Graves, A.H.G: Rinnooy Kan, P.H. Zipkin (Eds.), Handbooks in Operations Research and Management Science, vol. 4, Logistics of Production and Inventory, Elsevier, Amsterdam, 1993.

[2] J. Blazewicz, W. Domschke, E. Pesch, The job shop scheduling problem: Conventional and new solution techniques, European Journal of Operational Research 93 (1996) 1–33.

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

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

[5] J.K. Lenstra, A.H.G. Rinnooy Kan, Computational complexity of discrete optimization problems, Annals of Discrete Mathematics 4 (1979) 1212–1240.

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

[7] J. Hurink, B. Jurisch, M. Thole, Tabu search for the job-shop scheduling problem with multi-purpose machines, OR Spektrum 15 (1994) 205–215.

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

[9] B. Giffler, G.L. Thompson, Algorithms for solving production scheduling, Operations Research 8 (1960) 487–503.

[10] M. Wasner, Heuristiken für ein Hybrid-Job-Shop-Problem unter Berücksichtigung von reihenfolgeabhängigen Rüstzeiten und Werkzeugrestriktionen, Master Thesis, University of Linz, Austria 1996.

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

[12] S. Dauzere-Peres, J.B. Lasserre, Lot streaming in job-shop scheduling, Operations Research 45 (1997) 584–595.

[13] G. Zäpfel, Production planning in the case of uncertain individual demand. Extension for an MRP II concept, International Journal of Production Economics 46–47 (1996) 153–164.

[14] B.L. Maccarthy, J. Liu, Addressing the gap in scheduling research: A review of optimization and heuristic methods in production scheduling, International Journal of Production Research 31 (1993) 59–79.

[15] N. Nasr, E.A. Elsyyed, Job shop scheduling with alternative machines, International Journal of Production Research 28 (1990) 1595–1609.