SUBPROBLEM SOLVING AND MACHINE PRIORITIZATION IN THE SHIFTING BOTTLENECK PROCEDURE FOR WEIGHTED TARDINESS JOB SHOPS

Roland Braune

Institute for Production and Logistics Management
Johannes Kepler University
Altenberger Straße 69
4040 Linz, Austria

roland.braune@jku.at

ABSTRACT
In this paper we perform investigations on the Shifting Bottleneck Procedure for weighted tardiness job shop scheduling problems. We propose machine prioritization rules which explicitly consider the specific structure of the occurring subproblems and compare them with conventional criteria. We study their effects in combination with alternative subproblem solution methods. Furthermore, we analyze the role of machine backtracking as an advanced control structure. Computational results are presented based on a set of adapted benchmark problems.

Keywords: job shop scheduling, total weighted tardiness, shifting bottleneck, machine prioritization

1. INTRODUCTION

Job Shop Scheduling (French 1982) involves sequencing a set of jobs on multiple machines. In the simplest case, each job is processed on each machine exactly once, whereby the processing orders (routings) are predetermined (precedence constraints) and can be different for each job. A job hence consists of several operations having fixed processing times assigned to them. A machine can only process one operation at a time (capacity / disjunctive constraint) and no preemption is allowed, i.e. once started, the processing of an operation cannot be interrupted until it has finished.

The most common optimization objective in job shop scheduling is certainly the minimization of the makespan, i.e. the maximum completion time $C_{\max}$ of all jobs.

However, other measures are at least that important and closer to the real world. Those include the number of tardy jobs, total completion or flow time and total weighted tardiness (TWT) $\sum w_j T_j$ which is subject of investigation in our contribution.

The tardiness T_j of a job j with respect to its due date d_j is computed as $T_j = \max(0, C_j - d_j)$, where C_j denotes the completion time of job j.

The associated problem may also include release times r_j for jobs and is denoted as

$$Jm \mid r_j \mid \sum w_j T_j$$

in the three-field notation of Graham, Lawler, Lenstra and Rinnooy Kan (1979).

Contrary to the makespan objective, weighted tardiness job shops have not attracted much attention in scheduling research, only a few contributions are available in this area. Among them are dedicated priority dispatch rules (Vepsalainen and Morton 1987; Anderson and Nyirenda 1990), local search based approaches (Kreipl 2000; de Bontridder 2005) and genetic algorithms (Mattfeld and Bierwirth 2004). This paper deals with an additional approach, the so called *Shifting Bottleneck Procedure* (SBP), initially proposed for the minimum makespan problem (Adams, Balas and Zawack 1988) and later adapted to $Jm \mid r_j \mid \sum w_j T_j$ by Pinedo and Singer (1999). We perform a computational study concerning the application of different subproblem solution procedures and machine prioritization rules within the shifting bottleneck procedure. Comparable studies have been published for $C_{\max}$ and $L_{\max}$ (maximum lateness) problems (Holtsclaw and Uzsoy 1996; Demirkol, Mehta and Uzsoy 1997; Aytug, Kempf and Uzsoy 2002). We extend these investigations to weighted tardiness job shops by considering the special characteristics of this kind of problem. Our paper is organized as follows:

In Section 2 we introduce the Shifting Bottleneck Procedure for job shop scheduling in general. Sections 3-6 describe its main tasks in the TWT context, with a special focus on machine prioritization and subproblem solution. Section 7 gives a brief overview on the concept of machine backtracking. In Section 8 we present our experimental results. A conclusion and an outlook on future research are given in Section 9.

2. SHIFTING BOTTLENECK PROCEDURES FOR JOB SHOP SCHEDULING

Shifting Bottleneck Procedures belong to the most popular optimization methods in the area of job shop

scheduling. Established in the late 1980s by Adams, Balas and Zawack (1988), they have been successfully applied to a wide range of different problem setups both in theory and practice.

The main idea behind bottleneck scheduling is a decomposition of a job shop problem into several single machine problems. The single machine problems are solved separately, one after the other, always prioritizing the most critical machine with respect to a given bottleneck measure. Each single machine solution is inserted into a partial schedule for the enclosing job shop problem until all machines have been scheduled.

The procedure is based on the disjunctive graph representation of the underlying problem. During the progress of the SBP, the graph represents a partial solution to the job shop problem. Each time a subproblem is solved, the resulting sequence on the associated machine is inserted into the graph by orienting the (undirected) arcs between the operations on this machine. Let M be the set of all machines and M_0 the set of already scheduled machines. Then we can describe the flow of the shifting bottleneck procedure according to Figure 1.

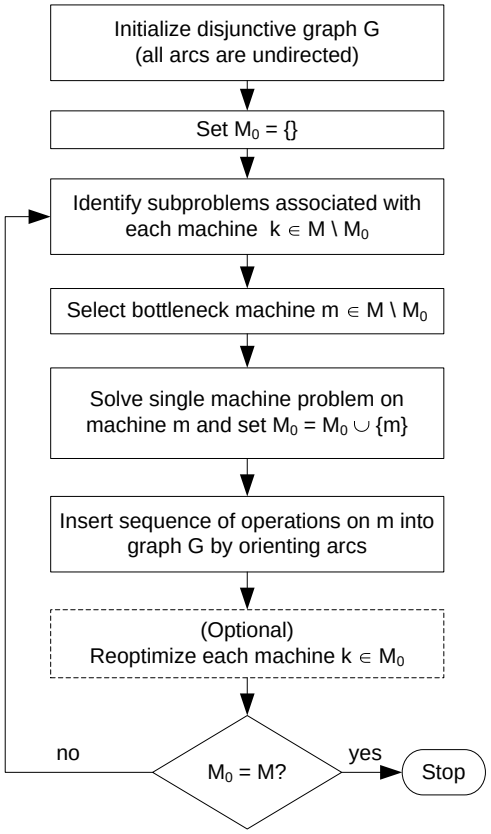

Figure 1: Outline of the Shifting Bottleneck Procedure

Given this outline, the following four main tasks can be identified:

1. Subproblem identification
2. Bottleneck selection (machine prioritization)
3. Subproblem solution
4. Reoptimization

Furthermore, the SBP can be embedded into an enumeration framework in order to examine different machine orders and determine the best one (Adams, Balas and Zawack 1988).

Due to the fact that the graph structure of tardiness job shops differs considerably from those of makespan or maximum lateness problems, the tasks subproblem identification, bottleneck solution and subproblem solution require a dedicated approach as described by Pinedo and Singer (1999). In the following sections we give an overview on the tasks enumerated above in the context of tardiness job shop scheduling. Particularly the bottleneck selection and subproblem solution processes receive specific attention, as they represent the basis of our investigations.

3. SUBPROBLEM IDENTIFICATION

Consider a tardiness job shop with m machines and n jobs. In the graph representation of this job shop, there are n sink nodes, one for each job. When isolating a single machine scheduling problem (SMSP) from the graph, each operation in the SMSP has exactly n different due dates resulting from the longest paths between the respective operation and the sink nodes.

According to Pinedo and Singer (1999), the single machine problems can be described as follows: Let (i, j) be an operation to be scheduled on machine i, then d_{ij}^k denotes the (local) due date of operation (i, j) with respect to job k. Given the completion time C_{ij} of operation (i, j), the tardiness of the operation concerning job k can be computed as $T_{ij}^k = \max(C_{ij} - d_{ij}^k, 0)$. Since all jobs on machine i have to be scheduled, the actual tardiness of job k is determined as $\max_{(i,j)\in N_i} T_{ij}^k$, where N_i denotes the set of all operations on machine i. The total increase in the objective function given a schedule on machine i is

$$\sum_{k=1}^{n} w_k \left(\max_{(i,j)\in N_i} T_{ij}^k \right)$$

Considering delayed precedence constraints (DPCs) due to potential paths in the graph between operations on a machine, the single machine subproblems to be solved can be described as

$$1 \mid r_j, DPC \mid \sum_k w_k (\max_j T_j^k)$$

4. BOTTLENECK SELECTION

The selection of a bottleneck machine is an intrinsic step in the SBP. The definition of a bottleneck is not unambiguous however. As for the makespan case, there are many different ways of determining whether a machine should be prioritized over others (Holtsclaw and Uzsoy 1996; Aytug, Kempf and Uzsoy 2002). After a short introduction on the conventional quality based prioritization, we propose bottleneck selection criteria which are dedicated to the subproblem structure arising

from tardiness job shops. The criteria are based on slack and problem infeasibility and need to be computed dynamically in each iteration of the SBP by analyzing the subproblems of all unscheduled machines.

4.1. Quality

The most common bottleneck selection criterion is oriented on the solution quality of the subproblems. In order to determine this measure, subproblems of all unscheduled machines have to be solved in advance and their resulting solution quality values are then ranked in descending order. The machine whose subproblem yields the highest TWT value is regarded as the bottleneck and its sequence gets inserted into the graph (Pinedo and Singer 1999).

4.2. Slack

The main idea behind this criterion is that the subproblem with the least slack with respect to its due dates may be regarded as the bottleneck. Due to the tighter due dates, it is more likely that the problem in question will increase the overall TWT. We distinguish two different types of slack: The slack of each operation with respect to its earliest local due date and the weighted minimum slack per job.

4.2.1. Earliest Due Date Slack (*SLCK*)

Consider an SMSP on machine i and let e_j be the earliest possible starting time of an operation j. Note that e_j is the maximum of the release time r_j and the earliest starting time of the operation due to potential delayed precedence constraints. Then for each operation j with processing time p_j a slack value can be computed as

$$slck_{ij} = \min_k(d_{ij}^k) - (e_j + p_j)$$

4.2.2. Weighted Minimum slack per job (*WSLCK*)

Contrary to *SLCK*, we compute the slack values not for each operation but for each job k of the superior TWT job shop problem. By this, we can include the job weights w_k into the calculation and therefore provide a more detailed estimate of the machine's criticality. The slack of an operation j regarding due date d_{ij}^k is defined as

$$slck_{ij}^k = \begin{cases} d_{ij}^k - (e_j + p_j) & \text{if } d_{ij}^k \text{ is defined} \\ \infty & \text{otherwise} \end{cases}$$

Since multiple operations may have a (local) due date regarding a job k, we are interested in the minimum slack value with respect to k. Additionally we want to consider job weights, hence we compute

$$wslck^k = 1/w_k \min_{(i,j)\in N_i}(slck_{ij}^k)$$

4.3. Infeasibility

The capacity (disjunctive) constraint stated in Section 1 enforces that no more than one operation at a time is processed on a machine. However, given time windows for operations defined by the release time r_j and a due date d_{ij}^k, it may occur that multiple operations require to be processed on a machine at the same time in order not to violate the due date(s). Such a situation can be detected by determining an infeasibility profile for the respective single machine problem (Aytug, Kempf and Uzsoy 2002).

We determine an infeasibility profile for the specific SMSPs described in Section 3 in the following way: For each operation j we define a time window ranging from the earliest starting time e_j to the earliest due date $\min_k(d_{ij}^k)$. The processing time p_j is distributed equally across the length of the time window, hence we obtain an average required capacity for any point within the time window.

By cumulating the time windows of all operations with their required capacities, we can create a capacity requirement profile for an SMSP as shown exemplarily in Figure 2.

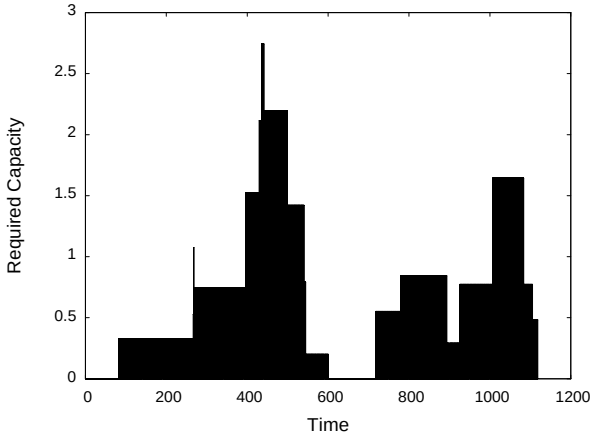

Figure 2: Example Infeasibility Profile

Such a profile can be referred to as an infeasibility profile in the context of disjunctive scheduling because cumulated average capacity requirements greater than 1 indicate a conflict situation. The machine is potentially required to process more than one operation at a time, which is infeasible due to the capacity constraint.

We use the infeasibility profile of a subproblem in order to determine two different bottleneck measures: The earliest due date infeasibility and the weighted average infeasibility per job.

4.3.1. Earliest due date infeasibility (*INFEAS*)

Given an infeasibility profile of an SMSP with respect to earliest due dates, we determine the area of the profile which exceeds the capacity limit. Let v_t be the (cumulated) capacity requirement at time t, which we refer to as the infeasibility value at time t. Based on this, we compute the earliest due date infeasibility bottleneck measure as

$$infeas = \sum_t \max(v_t - 1, 0)$$

4.3.2. Weighted avg. infeasibility / job (*WINFEAS*)

In order to take job weights into consideration we propose a further infeasibility related measure. For each job k of the superior tardiness job shop and each operation j we determine the time windows $[e_j, d_{ij}^k]$ and compute an average infeasibility value for each of these time windows:

$$infeas_{ij}^k = \begin{cases} (\sum_{t=e_j}^{d_{ij}^k} v_t)/(d_{ij}^k - e_j) & \text{if } d_{ij}^k \text{ is defined} \\ 0 & \text{otherwise} \end{cases}$$

We only keep the maximum value for each job k and multiply it with the associated weight w_k:

$$winfeas^k = w_k \max_{(i,j)\in N_i} (infeas_{ij}^k)^z$$

In order to amplify infeasibility values greater than one we actually square them ($z = 2$) before calculating the maximum.

5. SUBPROBLEM SOLVING

It is in every way important how well the occurring subproblems are solved: On the one hand, the solution quality may steer the bottleneck selection process itself as stated in Section 4.1. On the other hand, the resulting sequence on a bottleneck machine gets inserted into the overall partial solution and hence influences the whole subsequent optimization process.

Apart from solution quality, the computation time required for solving the SMSPs is an important factor in the context of bottleneck scheduling, especially when using machine backtracking and / or reoptimization.

Besides the beam search algorithm of Pinedo and Singer (1999), we use different local search methods for single machine optimization in our computational study. We apply a simple best improvement local search (BILS) procedure and a Tabu Search (TS) (Glover 1997) method. The approaches are conceptually similar to those presented in (Braune 2006). However, small modifications were necessary to respect the precedence constraints and the objective function of

$$1 \mid r_j, DPC \mid \sum_k w_k (\max_j T_j^k)$$

The local search methods rely on non-adjacent pairwise interchanges (NAPI) of jobs and use a modified ATC priority dispatch rule (Pinedo and Singer 1999) for generating the initial solution.

6. REOPTIMIZATION

Reoptimization may take place every time after a new single machine sequence has been inserted into the partial solution graph. According to the original idea proposed by Adams, Balas and Zawack (1988), a reoptimization cycle consists of the following steps:

1. Sort the set M_0 of already sequenced machines according to decreasing solution quality
2. For each machine $i \in M_0$ do the following:
   (a) Isolate the corresponding subproblem from the graph by removing all directed arcs between operations on machine i
   (b) Optimize the single machine problem and insert the resulting sequence into the graph
3. Unless the termination criterion is satisfied, goto step 1. Otherwise stop.

The reoptimization procedure terminates after a fixed number of reoptimization cycles or in case the most recent cycle did not yield a further improvement in overall solution quality. The latter principle is referred to as full reoptimization in the following.

7. BACKTRACKING

Backtracking or selective enumeration (Adams, Balas and Zawack 1988) can be used to examine various different machine sequences for a given problem. Each node in the search tree corresponds to a partial permutation of already scheduled machines. Branching is performed on the remaining unscheduled machines which are ranked according to the used bottleneck measure. The number of branches actually created at each level is controlled by an "aperture" parameter β (Pinedo and Singer 1999).

8. EXPERIMENTAL RESULTS

Our computational study is based on a set of 22 modified benchmark instances of size 10 x 10 taken from the OR-Library (Beasley 1990): ABZ5, ABZ6, LA16 – LA20, LA21 – LA24 (downsized by omitting the last 5 jobs), MT10 and ORB1 – ORB10. Originally intended for the makespan objective, these instances have been adapted by Pinedo and Singer (1999) in order to be able to use them for total weighted tardiness experiments. In fact they added a weight w_j and a due date d_j for each job. The due dates d_j were generated according to the following rule:

$$d_j = r_j + \left\lfloor f \cdot \sum_{i=1}^{10} p_{ij} \right\rfloor$$

where f denotes the tightness factor for due dates. We used the benchmark set with $f = 1.5$.

Since subproblem solution methods are also subject of our investigations, we first of all carried out a performance comparison on the single machine problem level. For this purpose, we sampled 30000 occurring SMSPs during the application of the SBP to the benchmark problem set. Optimal solutions are provided by the beam search method using the maximum aperture size. The solution quality obtained by the ATC priority dispatch rule (cf. Section 5) with $K = 2$ serves as a baseline. The local search methods (BILS, TS) have been parametrized according to Table 3.

The quality results are summarized in Table 1 in terms of mean percentage deviations from the optimal solution.

Table 1: Quality Deviations Obtained for Subproblems

	ATC	BILS	TS
Deviation	313,94 %	46,34 %	0,25%

Table 2: Computation Time for Subproblem Solution

	Beam Search	BILS	TS
Time (ms)	7953	250	239761

Table 3: Parametrization of LS Methods

	TS	BILS
Neighborhood	NAPI	NAPI
Initial solution	ATC	ATC
Neighborhood size	20	max
Tabu tenure	2	-
Max non-improving iterations	100	-

Furthermore, we compared the total running times of the four methods over all 30000 problems (cf. Table 2). Our investigations reveal that Tabu Search yields close to optimal solutions, but requires computation times which are orders of magnitude higher than for the (exact) Beam Search algorithm. Note that the running times of the ATC rule are negligibly small.

As a next step we applied the Shifting Bottleneck Procedure to the job shop benchmark set. We tested different combinations of bottleneck selection rules, subproblem solution methods and backtracking schemes. The resulting total weighted tardiness values are normalized using the weighted sum of job processing times (Lin, Goodman and Punch 1997):

$$(\sum w_j T_j)/(\sum w_j p_j)$$

The results are aggregated for the problem set, hence we report the sum of all individual TWT values.

For comparison purposes, Table 4 lists aggregated results from priority dispatch rules, Pinedo and Singer's SBP (SB-PS), their priority threshold backtracking heuristic (PTB) and their Branch and Bound algorithm which is able to solve the instances to optimality. Note that Singer and Pinedo imposed time limits for the applications of the SBPs, therefore results are not fully reproducible.

We run the SBP with and without backtracking (single pass) and always using full reoptimization. When backtracking is applied, we impose a limit on the number of solved SMSPs for each single benchmark instance. We believe that the computational effort can be better measured that way than in terms of CPU time. We compare the bottleneck selection criteria described in Section 4 with simple workload (total processing time) based and random prioritization. As for the subproblem solution methods, we apply Beam Search, BILS and Tabu Search when no backtracking is applied. Since TS is very time consuming we did not use it in the backtracking experiments.

The results obtained without backtracking, as summarized in Table 5, are moderate, yet most of them better than the PDR output. Quality and infeasibility based bottleneck selection rules obviously outperform the others, particularly the simple *INFEAS* rule performs very well.

Table 4: Results from PDRs and from Literature

Best PDR	PTB	SB-PS ($\beta = 2$)	SB-PS ($\beta = 3$)	Opt.
1,8189	1,3955	0,7426	0,6094	**0,5733**

Table 5: Single Pass Computational Results

Criterion	Beam Search	BILS	TS
quality	**1,3183**	1,6544	1,6013
SLCK (Σ)	1,7899	2,4005	1,7403
WSLCK (Σ)	1,6306	2,0908	1,7357
INFEAS	**1,1963**	1,6574	1,2730
WINFEAS (Σ)	**1,3809**	1,6904	1,4212
workload	1,6483	2,1408	1,7628
random	2,2178	2,2060	2,0005

Table 6: Results Obtained with Backtracking ($\beta = 2$) and a Limit of 100000 Solved SMSPs

Criterion	Beam Search	BILS
quality	0,6801	0,7910
SLCK (Σ)	0,6879	0,7808
WSLCK (Σ)	0,6864	0,7849
INFEAS	0,7284	0,8383
WINFEAS (Σ)	0,6973	0,7808
workload	0,7284	0,8839
random	0,7222	0,8392

Table 7: Results Obtained with Backtracking ($\beta = 3$) and a Limit of 300000 Solved SMSPs

Criterion	Beam Search	BILS
quality	**0,6237**	0,7059
SLCK (Σ)	0,6526	0,7429
WSLCK (Σ)	0,6578	0,7392
INFEAS	0,6519	0,7150
WINFEAS (Σ)	**0,6263**	0,7079
workload	0,6783	0,7741
random	0,6663	0,7801

As for the slack based measures, *WSLCK* exceeds *SLCK* which is possibly not meaningful enough to distinguish sharply between the machines. It further becomes clear that the choice of the subproblem solution approach significantly affects solution quality. While TS is only slightly worse than the Beam Search algorithm, the performance of the simple local search method considerably declines.

The incorporation of backtracking leads to a tremendous improvement in solution quality (cf. Table 6 and Table 7). In this context, it is striking that quality and *WINFEAS* perform almost equally well, with a slight though not significant advantage over the other

rules. Hence we conjecture that *WINFEAS* is a good indicator for the expected solution quality. In general, it can be observed that the differences between the bottleneck selection rules are quite small. Even the random rule is not clearly inferior. We presume that the probability of generating good solutions increases with the number of examined machine sequences. To verify this conjecture, we enumerated all possible machine sequences for selected problem instances and discovered that the number of machine sequences yielding near-optimal solutions was very large. This observation coincides with the findings of Aytug, Kempf and Uzsoy (2002) for the makespan objective.

Again, the BILS subproblem solution method performs definitely worse compared to the exact Beam Search algorithm. It seems that the exact solution of the occurring subproblems is an essential factor in TWT oriented shifting bottleneck scheduling.

9. CONCLUSION AND OUTLOOK

We have presented a computational study of the Shifting Bottleneck Procedure for weighted tardiness job shop scheduling. On the one hand, we have developed dedicated bottleneck selection criteria and compared them to conventional ones. Empirical results show that the dedicated criteria yield consistently good and partly equivalent performance compared to the conventional quality based prioritization. The advantage of the proposed rules is mainly the fact that the subproblems need not be solved in advance in order to rank them. This may not be critical for small problems, but possibly a key factor for medium size or even large-scale instances.

On the other hand, we have analyzed the effect of applying alternative subproblem solution methods. Our computational results clearly indicate that solving the subproblems to optimality or at least close to optimality is vital in the TWT context. However, preliminary tests revealed that the running time of Pinedo and Singer's Beam Search algorithm increases dramatically with the problem size. As a consequence, trying to find optimal solutions this way can be considered infeasible for subproblem instances with more than 15 jobs. For this reason, we are convinced that efficient heuristic subproblem solving will play a central role when encountering larger instances.

Our future research will be directed towards an effective application of the SBP to medium-size and large-scale TWT job shops. According to preliminary experiments (Braune, Wagner and Affenzeller 2007), we assume that the required computation time renders backtracking impractical for large instances, even when using heuristic SMSP solvers. Therefore we intend to gain deeper insight into subproblem interaction in order to make the single pass procedure more competitive.

REFERENCES

Adams, J., Balas, E., Zawack, D., 1988. The shifting bottleneck procedure for job shop scheduling. *Management Science*, 34 (3), 391-401.

Anderson, E.J., Nyirenda, J.C., 1990. Two new rules to minimize tardiness in a job shop. *International Journal of Prod. Research*, 28 (12), 2277-2292.

Aytug, H., Kempf, K., Uzsoy, R., 2002. Measures of subproblem criticality in decomposition algorithms for shop scheduling. *International Journal of Production Research*, 41 (5), 865-882.

Beasley, J.E., 1990. *OR-Library*. Available from: http://people.brunel.ac.uk/~mastjjb/jeb/info.html [Accessed 17th July 2008].

De Bontridder, K.M.J., 2005. Minimizing total weighted tardiness in a generalized job shop. *Journal of Scheduling*, 8, 479-496.

Braune, R., Affenzeller, M., Wagner, S., 2006. Efficient heuristic optimization in single machine scheduling. *Proceedings of the International Mediterranean Modelling Multiconference I3M 2006*, 499-504. Barcelona.

Braune, R., Wagner, S., Affenzeller, M., 2007. Optimization methods for large-scale production scheduling problems. In: R. Moreno-Diaz et al., eds. *EUROCAST 2007, LNCS 4739*. Springer Verlag Heidelberg, 812-819.

Demirkol, E., Mehta, S., Uzsoy, R., 1997. A computational study of shifting bottleneck procedures for shop scheduling problems. *Journal of Heuristics*, 3, 111-137.

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

Glover, F., 1997. *Tabu Search*. Kluwer Academic Publishers.

Graham, R.L., Lawler, E.L., Lenstra, J.K. and Rinnooy Kan, A.H.G., 1979. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Operations Research,* 5, 187-326.

Holtsclaw, H.H., Uzsoy, R., 1996. Machine criticality measures and subproblem solution procedures in shifting bottleneck methods. *Journal of the Operational Research Society*, 47, 666-677.

Kreipl, S., 2000. A large step random walk for minimizing total weighted tardiness in a job shop. *Journal of Scheduling*, 3, 125-138.

Lin, S.-C., Goodman, E.D., Punch, W.F., 1997. A genetic algorithm approach to dynamic job shop scheduling problem. *Proceedings of the 7th International Conference on Genetic Algorithms 1997*, 481-488, East Lansing, USA.

Mattfeld, D.C., Bierwirth, C., 2004. An efficient genetic algorithm for job shop scheduling with tardiness objectives. *European Journal of Operational Research*, 155 (3), 616-630.

Pinedo, M., Singer, M., 1999. A shifting bottleneck heuristic for minimizing the total weighted tardiness in a job shop. *Naval Research Logistics*, 46 (1), 1-17.

Vepsalainen, P.J., Morton, T.E., 1987. Priority rules for job shops with weighted tardiness costs. *Management Science,* 33 (8), 1035-1047.