European Journal of Operational Research 207 (2010) 1235–1243

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

On heuristic search for the single machine total weighted tardiness problem – Some theoretical insights and their empirical verification

Martin Josef Geiger *

Helmut Schmidt University – University of the Federal Armed Forces Hamburg, Holstenhofweg 85, 22041 Hamburg, Germany

ARTICLE INFO

Article history:
Received 4 October 2009
Accepted 21 June 2010
Available online 1 July 2010

Keywords:
Metaheuristics
Scheduling
Neighborhood operators

ABSTRACT

The article presents theoretical and experimental investigations of computational intelligence techniques for machine sequencing problems. Contrary to other approaches, which are experimentally driven only, our work is motivated by gaining insights in the underlying principles of heuristic search for this particular problem. We therefore first theoretically analyze local search neighborhoods, deriving expectations about their relative performance. An empirical study on benchmark data follows, verifying the initial propositions.

In result, we may conclude theoretically and empirically on the relative performance of neighborhood search operators for the single machine total weighted tardiness problem. The results are useful for the proposition of heuristic search procedures based on local search, as they lead to an order of neighborhood structures with respect to their relative performance. The obtained insights are verified by investigating the effectiveness of a (multi-operator) Variable Neighborhood Search approach for the problem at hand. We are able to show that most known benchmark instances are reliably solved to optimality, leaving an overall average gap of around 1% above the optimum.

1. Introduction

Machine scheduling plays an important role in modern manufacturing. After having agreed upon particular delivery dates with the customers, the production and delivery of goods needs to be organized such that orders can be fulfilled within the given time boundaries, thus avoiding tardiness of orders as much as possible. One aspect in this context is the already mentioned *machine scheduling* of the job floor, where starting times have to be found for the operations with respect to a set of side constraints and one or several optimality criteria. The side constraints usually ensure the technical feasibility of the schedule, such as the common circumstance that a machine may only process a single operation at a time. The optimality criteria allow an evaluation of a particular production schedule, expressing its quality from the perspective of a decision maker (production planner).

Research in machine scheduling and related fields is particularly active and has led to the development of numerous exact optimization approaches, many of which involve e.g. branch-and-bound techniques. Most problems unfortunately are however $\mathcal{NP}$-hard. Consequently, heuristic approximation approaches have been developed for a variety of scheduling problems. These approaches range from problem specific heuristics such as dispatching rules to problem-independent metaheuristics/local search algorithms. The latter class of search algorithms is based on *neighborhood search*, a search principle which tries to improve alternatives through a process of successive improving and/or non-improving steps. In each step of the algorithm, one or several alternatives are modified with the ultimate goal of identifying the optimal solution. A transition from one alternative x to x' is referred to as a *move*, and the computational procedures used to map x into x' are known as *neighborhood operators*.

In this sense, local search may be seen as an iterative walk through a neighborhood graph $G = (V,A)$. While the vertex set V in G is given by the set of alternatives admissible to the heuristic search procedure, the set of arcs A is induced by the particular neighborhood structure. It becomes clear, that the appropriate choice of these neighborhood operators highly influences the performance of the search algorithm as it defines the possible way how the algorithm may progress in the search space. Therefore, reliable information on a qualitatively good neighborhood structure is beneficial when designing heuristic search algorithms.

In the current article we present an investigation of local search neighborhoods for the single machine total weighted tardiness scheduling problem (SMTWTP). The SMTWTP is a well-known planning problem from operations research, engineering and computer science. It is characterized by the assignment of starting times for a given set of jobs on a single processor, minimizing the (weighted) tardy completion of the jobs with respect to given

* Tel.: +49 40 6541 2591.
E-mail address: m.j.geiger@hsu-hh.de

0377-2217/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2010.06.031

due dates. In this sense, the tardiness is given an economical interpretation, referring to the consideration of costs which consequently have to be avoided as much as possible.

Although most practical problems involve multiple resources (e.g. machines), many problems can be decomposed into a subsequently solved series of single machine problems, as for example done in the famous shifting bottleneck heuristic [2]. The effective solution of each single machine subproblem is therefore a relevant aspect for solving more complex models.

Besides the problem's relevance with respect to the applicability in real-world situations, the SMTWTP is computationally challenging, as it has been proven to be strongly $\mathcal{NP}$-hard [17].

While some exact methods are available, many successful solution approaches are based on heuristics [1]. More recently, several different metaheuristics have been developed for the SMTWTP, successfully solving benchmark instances from the scientific literature. Important work includes simple local search [19], Evolutionary Algorithms [11], Ant Colony Optimization [5,13,20]. Iterated Local Search [14], and Simulated Annealing [21]. A particularly successful neighborhood search technique for the SMTWTP is *Iterated Dynasearch* [10], which uses dynamic programming to determine an optimal series of moves to be executed simultaneously.

While it has been pointed out that available benchmark instances are comparably easily solvable by local search [14], several recent publications aim to push the scientific knowledge even further by proposing more refined metaheuristics [3,7,9,18,22,24]. Common to these approaches is the fact that they lead to rather impressive results within the chosen experimental settings. On the other hand, comparably little has been learned about suitable neighborhood search operators for this problem, or about the underlying principles of why local search successfully solved known benchmark instances.

Instead of identifying appropriate neighborhoods in a rigorous way, some investigations rely on multi-operator search that combines different search strategies in an additive way, without giving priority to promising moves first. A key ingredient of such algorithms therefore is the availability of sufficient computational capacity. On the other hand, several approaches identify promising moves, successfully avoiding inferior ones and thus improving the effectiveness of search. A prominent example consists of the previously mentioned Dynasearch approach [10,15], which allows the searching of an exponential size neighborhood in polynomial time by exploiting problem specific structures. Other problem specific improvements have been obtained by a block-based tabu search approach [8], which, on the basis of either tardy or early blocks of jobs, refines the set of promising neighborhood moves.

Instead of presenting a slight modification/improvement of some existing search algorithm, we aim to address the fundamental question of what neighborhood yields good results in the SMTWTP, and why. Knowing the answer to this challenging research question will then help to improve local search for the given problem on a sound theoretical and empirical basis.

From this perspective, and taking into account the remarks of [14], we also need to raise the question whether recent results are so surprising after all. Or is it possible that, despite the problem's complexity, the SMTWTP is a 'solved' problem for metaheuristics? Are there 'difficult' instances, and if so, how can they be described in terms of their technical properties? Can we draw conclusions on how to organize future research?

With respect to the research questions, the article is organized as follows. In Section 2, the single machine total weighted tardiness scheduling problem is given and a simplifying representation for the schedules is introduced. The following Section 3 presents neighborhood search operators for the problem at hand. A theoretical discussion of their performance is given, and expectations about their relative performance are derived. Experiments are carried out in Section 4, investigating, and thus verifying/falsifying the hypotheses of the previous section. As a synthesis, we then combine different neighborhood search operators in a Variable Neighborhood Search algorithm, and investigate the effectiveness of this approach for benchmark instances taken from the literature. Conclusions are given in Section 6.

2. Problem description

2.1. Notations

The single machine total weighted tardiness problem (SMTWTP) can be stated as follows. A set of jobs $\mathcal{J} = \{J_1, \ldots, J_n\}$ needs to be processed on a single machine. Each job J_j consists of a single operation only, involving a processing time $p_j > 0, j = 1, \ldots, n$. The relative importance of the jobs is expressed via a nonnegative weight $w_j > 0, j = 1, \ldots, n$. Processing on the machine is only possible for a single job at a time, excluding parallel processing of jobs. Each job J_j is supposed to be finished before its due date D_j. If this is not the case, a tardiness T_j occurs, measured as $T_j = \max\{s_j + p_j - D_j; 0\}$. The overall objective of the problem is to find a feasible schedule x minimizing the total weighted tardiness as given in Expression (1)

$$\min \; TWT = \sum_{j=1}^{n} w_j T_j. \tag{1}$$

A schedule for a particular problem can be interpreted as a vector of starting times of the jobs, $x = (s_1, \ldots, s_n)$. We assume that processing starts at time 0, thus $s_j \geqslant 0, \; j = 1, \ldots, n$. A possible overlapping of jobs on the machine is avoided by the formulation of disjunctive side constraints given in Expression (2)

$$s_j \geqslant s_k + p_k \dot{\vee} s_j + p_j \leqslant s_k \quad \forall j, k = 1, \ldots, n, \; j \neq k. \tag{2}$$

2.2. Solution representation

As the objective function of the single machine total weighted tardiness problem is a *regular* function [4], it is known that at least one active schedule exists which is also optimal. The problem of finding an optimal schedule may therefore be reduced to the problem of finding an optimal sequence of jobs. A given sequence is represented by a permutation $\pi = \{\pi_1, \ldots, \pi_n\}$ of the job indices. Each element π_i in π stores the index of the job which is to be processed as the ith job in the processing sequence. The permutation of indices is then 'decoded' into a schedule by the following Expressions (3)

$$\begin{aligned} s_{\pi_1} &= 0, \\ s_{\pi_i} &= s_{\pi_{i-1}} + p_{\pi_{i-1}} \quad \forall i = 2, \ldots, n. \end{aligned} \tag{3}$$

Obviously, this leads to an active schedule without any waiting times between jobs.

2.3. Benchmark instances from the literature

Optimization approaches for the SMTWTP are commonly verified using the benchmark instances of [11]. The authors presented 375 data sets with varying characteristics. For values of $n = 40$, $n = 50$, and $n = 100$, 125 instances are each proposed. The computation of the processing times p_j is randomly drawn from a uniform distribution [1,100], while the weights are taken from [1,10]. Depending on the relative range of due dates *RDD* and the average tardiness factor *TF*, the due dates are randomly computed as integer values within $\left[P\left(1 - TF - \frac{RDD}{2}\right), P\left(1 - TF + \frac{RDD}{2}\right)\right]$, where $P = \sum_{j=1}^{n} p_j$. Five instances have been computed for each combination

of *RDD* and *TF*: $RDD \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$, $TF \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$.

All data sets are available in the OR-Library under http://people.brunel.ac.uk/~mastjjb/jeb/info.html. For the ones with $n = 40$ and $n = 50$, optimal solutions are known, while for the ones of size $n = 100$, at least to our knowledge, only best known solutions are reported in the literature, lacking their final proof of optimality. It should be noticed however, that the results for the larger data sets are commonly assumed to be optimal, as, despite rather active research in this area, there has not been any improvement of the best known upper bounds within past ten years.

3. An a priori view on search operators

3.1. Local search neighborhoods

Computational intelligence methods aim to solve the SMTWTP by altering the job sequence π until an optimal solution is identified or at least no further improvement is possible. Apart from genetic crossover operators, which stochastically recombine the permutations of two solutions, simple local search operators are commonly used for this problem [14]:

- Exchange (EX) of the positions of two jobs, placed at position i and j, $i \neq j$, sometimes referred to as 'interchange'. In the following example, the job at position $i = 3$ is exchanged with the one at position $j = 7$, and a modified permutation of jobs π' is obtained

$$\begin{array}{lcccccccccc} \pi = \{ & 1, & 3, & 4, & 2, & 7, & 5, & 8, & 6 & \} \\ & & & i & & & & j & & \\ \pi' = \{ & 1, & 3, & 8, & 2, & 7, & 5, & 4, & 6 & \} \end{array}.$$

- Shift of a job from its previous position i to another position j, $i \neq j$, also called 'insert'.

While the exchange operator leads to $\frac{n(n-1)}{2}$ moves, the shift neighborhood is of cardinality $n(n-1)$. With respect to the following analysis, a distinction will be made between the following two different types of shift operators [23]:

- Forward shift (FSH), removing a job from position i and reinserting it at position j, $i < j$. As an example, a job at position $i = 3$ may be removed and reinserted at position $j = 7$.

$$\begin{array}{lcccccccccc} \pi = \{ & 1, & 3, & 4, & 2, & 7, & 5, & 8, & 6 & \} \\ & & & i & & & & j & & \\ \pi' = \{ & 1, & 3, & 2, & 7, & 5, & 8, & 4, & 6 & \} \end{array}.$$

- Backward shift (BSH), removing a job from position j and reinserting it at position i, $i < j$. The following example illustrates such a move with $i = 3$ and $j = 7$.

$$\begin{array}{lcccccccccc} \pi = \{ & 1, & 3, & 4, & 2, & 7, & 5, & 8, & 6 & \} \\ & & & i & & & & j & & \\ \pi' = \{ & 1, & 3, & 8, & 4, & 2, & 7, & 5, & 6 & \} \end{array}.$$

Both forward and backward shift lead to $\frac{n(n-1)}{2}$ possible moves and therefore are, also in comparison to the exchange neighborhood, of identical cardinality.

It should be noticed that the described neighborhoods are not disjunct, meaning that a true subset of the neighbors is generated by all operators. This is the case for an exchange of adjacent jobs, and for a shift of jobs from position i to j with $|i - j| = 1$. In the work of [12], a further distinction between the neighborhoods is introduced which eliminates the common neighbors, assigning them to a separate 'adjacent pairwise exchange' (aEX) neighborhood. Consequently, the following operators are derived:

- Adjacent exchange (aEX), exchanging all jobs at positions i and j with $j = i + 1$.
- Non-adjacent exchange (naEX), consisting of an EX operator without the neighboring solutions of aEX. It is therefore required that $|i - j| > 1$.
- Non-adjacent forward shift (naFSH), being defined as FSH without aEX.
- Non-adjacent backward shift (naBSH), representing a neighborhood identical to the one of BSH without the elements of aEX.

Contrary to the initially introduced operators EX, FSH and BSH, the neighborhoods as given in [12] are disjunct. While the cardinality of aEX is $n - 1$, the others (naEX, naFSH, naBSH) are $\frac{n(n-1)}{2} - (n-1)$ and therefore slightly smaller than the ones of EX, FSH and BSH.

3.2. Elimination of moves

When analyzing alternatives and their job sequences for the SMTWTP, an observation can be made that may be used to add some problem specific structure to the previously presented neighborhoods.

For any given permutation of jobs $\pi = \{\pi_1, \ldots, \pi_n\}$, let t be the lowest position index of a tardy job, $1 \leqslant t \leqslant n + 1$, where we assume a value of $t = n + 1$ if there is no tardy job. Using t, a separation of π into two disjunctive subsequences π^a and π^b may be made such that $\pi^a = \{\pi_1, \ldots, \pi_{t-1}\}$ and $\pi^b = \{\pi_t, \ldots, \pi_n\}$.

Knowing that none of the jobs in π^a is tardy, we can conclude that no modification to π^a alone exists which improves the overall evaluation of the solution. As a corollary, any improving local search move is required to include at least one of the elements of π^b. This could mean that local search is performed in π_b only, or that jobs from π_a and π_b are involved. In the both cases, an improvement is possible if the reduction in the weighted tardiness outweighs the increase.

When integrating this separation into the neighborhoods EX, FSH and BSH, the cardinality of the neighborhoods increases with n and decreases with t such that an overall number of $\frac{n(n-1)}{2} - \frac{(t-1)(t-2)}{2}$ possible moves can be computed for EX, FSH and BSH. Similarly, aEX contains $n - t + 1$ elements, and naEX, naFSH and naBSH lead to $\frac{n(n-1)}{2} - \frac{(t-1)(t-2)}{2} - (n - t + 1)$ neighboring solutions.

The neighborhoods naturally become empty for $t = n + 1$. In this special case, $\pi^b = \emptyset$, and thus there is no tardy job that can be improved ($TWT = 0$).

It should be noticed, that similar considerations are used in the block-based tabu search approach [8]. Here, the permutation of jobs is decomposed into disjunct subsets of two categories: subsequent tardy ('T-blocks') and subsequent non-tardy ('E-blocks') jobs. In brief, this allows to accelerate search by reducing the cardinality of promising neighborhood moves. In relation to this more general approach, we only consider (at most) two disjunct subsets of jobs.

3.3. Neighborhoods and their effect on the tardiness

3.3.1. Introductory considerations

The commonly discussed principle of local search stresses the modification of alternatives such that an improvement is obtained. With a modification on one hand, some aspects of the alternatives are implicitly kept intact on the other hand. Successful neighborhood search techniques consequently keep a balance between maintaining certain structures of the alternatives while improving others at the same time. In this sense, local search is not only about

modifications by means of some operator, but equally about constance of favorable attributes.

The relative performance of neighborhood search techniques is often analyzed on the basis of experimental investigations. A priori results of the underlying principles of the used operators are however comparably scarce. For the here studied SMTWTP, some insights may be gained by analyzing the effect of the operators on the job sequences. In the a priori investigation of EX, FSH, BSH, aEX, naEX, naFSH and naBSH we are going to refer to changes in the permutation by means of two positions i, j with $i < j$. Starting times prior to the application of a move are denoted as s_j, the ones after the move as s'_j.

- Exchange
  Applying the EX operator, we obtain starting times of $s'_{\pi_j} = s_{\pi_i},\ s'_{\pi_i} = s_{\pi_j} + (p_{\pi_j} - p_{\pi_i})$, and $s'_{\pi_k} = s_{\pi_k} + (p_{\pi_j} - p_{\pi_i}),\ i < k < j$. The operator may improve the tardiness of the job at position $j, T'_{\pi_j} \leqslant T_{\pi_j}$, lead to a worse tardiness for the job at position $i, T'_{\pi_i} \geqslant T_{\pi_i}$ and have an effect on the jobs at the positions k, $i < k < j$ depending on the value of $(p_{\pi_j} - p_{\pi_i})$. For $(p_{\pi_j} - p_{\pi_i}) > 0$ we obtain $T'_{\pi_k} \geqslant T_{\pi_k}$, for $(p_{\pi_j} - p_{\pi_i}) < 0$ we get $T'_{\pi_k} \leqslant T_{\pi_k}$, and if $(p_{\pi_j} - p_{\pi_i}) = 0$ we have $T'_{\pi_k} = T_{\pi_k}$.
  This observation equally holds for EX as well as for naEX. The special case of aEX only affects exactly two jobs, the one at position i and the one at j. While the computation of the effects for job π_i and π_j remain as described above, no intermediate jobs at the positions k, $i < k < j$ exist.
- Forward shift
  The application of FSH and naFSH leads to a starting time $s'_{\pi_i} = s_{\pi_j} + (p_{\pi_j} - p_{\pi_i})$, whereas the jobs starting from position $i+1$ to j obtain values of $s'_{\pi_k} = s_{\pi_k} - p_{\pi_i},\ i < k \leqslant j$. As $j > i$, we know that $s_{\pi_j} \geqslant s_{\pi_i} + p_{\pi_i}$ and therefore obtain $s'_{\pi_i} \geqslant s_{\pi_i} + p_{\pi_j}$ and finally $T'_{\pi_i} \geqslant T_{\pi_i}$. For the jobs starting from position $i+1$ to j, values of $T'_{\pi_k} \leqslant T_{\pi_k},\ i < k \leqslant j$ are derived.
- Backward shift
  BSH and naBSH leads to a reduction of the starting time for the job at position $j, s'_{\pi_j} = s_{\pi_i}$ whereas the jobs from position i to $j-1$ are delayed such that $s'_{\pi_k} = s_{\pi_k} + p_{\pi_j}$. While it holds for the tardiness of the job at position j that $T'_{\pi_j} \leqslant T_{\pi_j}$, the tardiness of the jobs from position i to $j-1$ are $T'_{\pi_k} \geqslant T_{\pi_k},\ i \leqslant k < j$.

The described impact of the operators is summarized in Tables 1 and 2.

3.3.2. On the influence of t and the move intensity

It follows from Section 3.2 that $t \leqslant j$. The value of i on the other hand may be $t \leqslant i$ as well as $t > i$. The latter case has an influence on the positive effect of the forward shift operators, namely FSH and naFSH. Here, for the jobs at the positions k, $i < k < t$, the initial tardiness is $T_{\pi_k} = 0$ and therefore the relation $T'_{\pi_k} \leqslant T_{\pi_k}$ may be replaced with $T'_{\pi_k} = T_{\pi_k} = 0$, making an improvement of the jobs at these position impossible.

Table 1
Effect of the operators on the tardiness.

Operator	Positive effect	Negative effect
aEX	$T'_{\pi_j} \leqslant T_{\pi_j}$	$T'_{\pi_i} \geqslant T_{\pi_i}$
EX/naEX: $(p_{\pi_j} - p_{\pi_i}) = 0$	$T'_{\pi_j} \leqslant T_{\pi_j}$	$T'_{\pi_i} \geqslant T_{\pi_i}$
EX/naEX: $(p_{\pi_j} - p_{\pi_i}) > 0$	$T'_{\pi_j} \leqslant T_{\pi_j}$	$T'_{\pi_i} \geqslant T_{\pi_i}$, $T'_{\pi_k} \geqslant T_{\pi_k},\ i < k < j$
EX/naEX: $(p_{\pi_j} - p_{\pi_i}) < 0$	$T'_{\pi_j} \leqslant T_{\pi_j}$, $T'_{\pi_k} \leqslant T_{\pi_k},\ i < k < j$	$T'_{\pi_i} \geqslant T_{\pi_i}$
FSH/naFSH	$T'_{\pi_k} \leqslant T_{\pi_k},\ i < k \leqslant j$	$T'_{\pi_i} \geqslant T_{\pi_i}$
BSH/naBSH	$T'_{\pi_j} \leqslant T_{\pi_j}$	$T'_{\pi_k} \geqslant T_{\pi_k},\ i \leqslant k < j$

Table 2
Effect of the operators for each job.

Operator	i	$i < k < j$	j
aEX	$T'_{\pi_i} \geqslant T_{\pi_i}$		$T'_{\pi_j} \leqslant T_{\pi_j}$
EX/naEX: $(p_{\pi_j} - p_{\pi_i}) = 0$	$T'_{\pi_i} \geqslant T_{\pi_i}$	$T'_{\pi_k} = T_{\pi_k}$	$T'_{\pi_j} \leqslant T_{\pi_j}$
EX/naEX: $(p_{\pi_j} - p_{\pi_i}) > 0$	$T'_{\pi_i} \geqslant T_{\pi_i}$	$T'_{\pi_k} \geqslant T_{\pi_k}$	$T'_{\pi_j} \leqslant T_{\pi_j}$
EX/naEX: $(p_{\pi_j} - p_{\pi_i}) < 0$	$T'_{\pi_i} \geqslant T_{\pi_i}$	$T'_{\pi_k} \leqslant T_{\pi_k}$	$T'_{\pi_j} \leqslant T_{\pi_j}$
FSH/naFSH	$T'_{\pi_i} \geqslant T_{\pi_i}$	$T'_{\pi_k} \leqslant T_{\pi_k}$	$T'_{\pi_j} \leqslant T_{\pi_j}$
BSH/naBSH	$T'_{\pi_i} \geqslant T_{\pi_i}$	$T'_{\pi_k} \geqslant T_{\pi_k}$	$T'_{\pi_j} \leqslant T_{\pi_j}$

While the same observation holds for EX and naEX iff $(p_{\pi_j} - p_{\pi_i}) \leqslant 0$, it does not hold for the BSH and the naBSH operator, as well as EX and naEX in case of a $(p_{\pi_j} - p_{\pi_i}) > 0$. In brief, this indicates that there might be a difference between the relative performance of the neighborhood operators depending on the value of t. As t will depend on the actual properties of the actual data set, this can be expected to be a problem specific behavior.

Besides these general effects of the operators on the tardiness, the weights of the jobs affected by the moves play an important role as well. It is generally possible to suspect that jobs J_j having relatively large weights w_j should be scheduled at the beginning of the sequence. Unfortunately, this can only be considered as a general tendency, and counterexamples to this observation are likely to exist.

At least for existing benchmark instances from the literature however, we may suspect that the influence of the weights of the jobs is rather small. Recalling that the weights have been randomly drawn from the rather narrow uniform distribution [1,10], the relative importance of a job can only become at most ten times the one of another. In average however, the difference between weights will be even smaller. Consequently, moving a block of several jobs is likely to influence the tardiness more than a single job only, and we expect that the above described neighborhoods work in this sense on the data sets from the literature. This however implies that for benchmark data with other properties, e.g. ones with correlated weights and due dates, the weights w_j should not be left aside in the analysis.

3.3.3. Formulation of working hypotheses

First of all, it is possible to observe that the aEX operator improves the tardiness of a single job only while worsening the one of another. Besides, the neighborhood has by far the smallest cardinality. Also, its cardinality only grows linear with n, as opposed to a quadratic growth for the other neighborhoods. On the basis of these observations, we suspect that aEX is relatively less powerful when trying to improve a given alternative, and we may formulate this in Hypothesis 1. It should be noticed however that this expected relative operator performance has already been experimentally investigated for other scheduling problems, such as the permutation flow shop scheduling problem [23].

Hypothesis 1. aEX performs worse than the other neighborhoods.

It becomes obvious that BSH and naBSH have a negative effect on the tardiness values of several jobs while improving the situation for a single job only, the one at π_j. FSH and naFSH on the other hand improve a set of jobs while worsening the situation for a single one. This tendency can be suspected to be a general behavior of the two operators, and the effects also depend on the distance $j - i$. With increasing $j - i$, the number of forward shifted jobs in BSH and naBSH and the number of backward shifted jobs in FSH and naFSH increases. As $j - i$ is bounded by $n - 1$, the gap between FSH/naFSH and BSH/naBSH in terms of their relative performance should increase with increasing number of jobs n. This observation leads to the proposition of the following Hypotheses 2 and 3 on the relative performance of BSH/naBSH vs. FSH/naFSH.

Hypothesis 2. BSH/naBSH are inferior to FSH/naFSH.

Hypothesis 3. The effect of Hypothesis 2 increases with increasing number of jobs n.

Similar considerations can be made for EX/naEX vs. BSH/naBSH. EX/naEX exchange two jobs, possibly improving the tardiness of the one at position j while possibly worsening the situation for the job from position i. The potential tardiness changes of jobs in between positions i and j depend on the processing time difference $(p_{\pi_j} - p_{\pi_i})$. In direct comparison to BSH/naBSH, this leads to the observation that EX/naEX keep the starting times comparably more stable, resulting in a relatively superior performance as suspected in the following Hypothesis 4. Again, this effect is expected to become more obvious with increasing number of jobs n, expressed in Hypothesis 5.

Hypothesis 4. BSH/naBSH are inferior to EX/naEX.

Hypothesis 5. The effect of Hypothesis 4 increases with increasing number of jobs n.

When comparing FSH/naFSH and EX/naEX, different aspects can be observed that undermine a superior relative performance of either of the two operator principles. For FSH/naFSH, it is the ability to improve a block of jobs from position $i+1$ to j. For EX/naEX, it is the behavior of maintaining relatively stable starting times for the jobs from $i+1$ to $j-1$. Based on the investigations carried out above, it does not appear to be possible to formulate justified hypotheses about the relative performance of the neighborhood operators. We expect this to depend on problem specific structures, possibly even on instance specific data. Instead, we are going to resolve this in an experimental investigation which is also to verify or falsify the hypotheses given above.

4. Experiments with hillclimbing

4.1. Experimental setup

The theoretical insights and hypotheses gathered above have been tested in an experimental setting. We used the benchmark instances given in the OR-library [6], available online under http://people.brunel.ac.uk/~mastjjb/jeb/info.html and described in the previous Section 2.3.

It has been pointed out already, that some of the instances, particularly the ones with $n = 40$ and $n = 50$ can be solved rather easily with simple local search [14], and therefore do not present a challenge for state-of-the-art heuristics. As we on the other hand aim to concentrate on the relative performance of neighborhoods, and thus on the relations among them, we do not expect this to lead to problems or a bias in the analysis.

The relative performance of the operators has been tested using a simple hillclimbing algorithm. During search, both a best-improving-move and a first-improving-move strategy are implemented, replacing the current solution x with the *overall best* or the *first identified improving* neighboring solution x'. An initial solution is randomly computed, consisting of a random permutation of jobs. Random restarts are made once a locally optimal alternative has been found. The definition of the experimental setting aims to minimize the potential bias of problem specific heuristics such as constructive approaches e.g. based on the Earliest Due Date rule. While it is known that integrating specific heuristic knowledge improves the quality of the solutions considerable [5], it equally could create a bias towards a certain subset of the search space.

Experiments are run executing 100 random restarts for each neighborhood operator/problem instance-combination The obtained local optima are stored so that average and best values of the weighted total tardiness of the solutions can be computed.

4.2. Identification of optimal alternatives

We first analyzed the number of instances for which the optimal solutions have been found by the different operators, keeping in mind that for the instances with $n = 100$, the best known solutions are not yet proven to be optimal. Table 3 gives these numbers. A possible maximum value is 125 due to the available 125 instances for each class of instance sizes.

The results reported in Table 3 investigate the neighborhoods for two generic acceptance strategies. On the one hand, the *best-improvement* strategy searches the entire neighborhood of the alternatives, finally accepting the best one in the end. On the other hand, the *first-improvement* strategy starts searching the neighborhood and accepts the first-improving-move, thus excluding the remaining alternatives in the neighborhood from being considered.

It can be seen, that with increasing number of jobs n, the number of instances in which an optimal solution has been identified decreases. This is to be expected as the growth of the search space is not polynomially bounded and the instances become more difficult to solve using this simplistic approach. With respect to the here carried out investigation however, an interesting pattern of the relative performance of the operators can be seen. Obviously, EX is able to identify the optimal/best known solutions in considerable more cases than FSH, while BSH leads to the worst results.

With respect to the acceptance strategy of best-versus first-improving move, the numbers differ, the overall conclusions on the relative comparison of the different neighborhood structures are however the same.

A more detailed analysis is given in Table 4. Here, the influence of the parameters *RDD* and *TF* is investigated. As the numbers are aggregated over all instances with $n = 40$, $n = 50$, and $n = 100$, the maximum possible value in each cell is 15.

The results indicate that medium values of *TF* lead to more difficult instances, while instances being generated using small (0.2) and high (1.0) values are easier to solve.

Another comparison of the neighborhoods is based on the minimum (best) values of the total weighted tardiness obtained during all test runs. We count the number of instances for which the direct comparison of two operators, based on the best found objective values over all test runs, holds the relations given in Table 5. As the results for the first-improvement-strategy are rather similar, we chose to omit the corresponding data.

Comparing the values of BSH and FSH leads to the observation that FSH leads in significantly more instances to better results than

Table 3
Number of instances for which optimal/best known solutions have been found in the test runs. Results are reported for both the best-improvement and the first-improvement-strategy.

Acceptance strategy	Operator	$n = 40$	$n = 50$	$n = 100$
Best-imp.	EX	108	85	48
	FSH	86	65	31
	BSH	48	35	13
	aEX	8	4	0
	naEX	30	28	23
	naFSH	63	44	26
	naBSH	25	21	12
First-imp.	EX	118	99	45
	FSH	85	64	29
	BSH	46	31	23
	aEX	13	6	0
	naEX	40	31	27
	naFSH	34	29	24
	naBSH	19	19	18

Table 4
Solved instances depending on *RDD* and *TF* (best-improvement-strategy).

		EX					FSH					BSH				
	TF:	0.2	0.4	0.6	0.8	1.0	0.2	0.4	0.6	0.8	1.0	0.2	0.4	0.6	0.8	1.0
RDD:	0.2	11	13	7	10	15	5	6	3	7	13	0	0	0	2	12
	0.4	11	9	6	11	15	12	8	8	6	9	3	0	0	4	9
	0.6	15	6	3	9	11	15	4	4	3	9	12	0	0	5	6
	0.8	15	4	4	7	12	15	6	5	2	6	13	0	0	1	7
	1.0	15	8	5	8	11	15	11	3	1	6	13	4	0	1	4

		aEX					naEX				
	TF:	0.2	0.4	0.6	0.8	1.0	0.2	0.4	0.6	0.8	1.0
RDD:	0.2	0	0	0	0	5	8	1	0	0	0
	0.4	0	0	0	0	3	11	2	0	0	0
	0.6	0	0	0	0	0	15	3	0	0	0
	0.8	3	0	0	0	1	15	4	0	0	0
	1.0	0	0	0	0	0	15	7	0	0	0

		naFSH					naBSH				
RDD:	0.2	5	5	3	3	7	0	0	0	1	5
	0.4	12	8	7	1	6	3	0	0	0	4
	0.6	15	4	3	0	1	12	0	0	1	1
	0.8	15	3	2	2	4	14	0	0	0	1
	1.0	15	10	1	0	1	13	3	0	0	0

Table 5
Number of instances for which the comparison of minimum values holds (best-improvement-strategy).

Operator comparison		$n = 40$	$n = 50$	$n = 100$
EX vs. FSH:	min (EX) < min (FSH)	37	43	84
	min (EX) > min (FSH)	6	21	16
	min (EX) = min (FSH)	82	61	25
FSH vs. BSH:	min (FSH) < min (BSH)	71	80	105
	min (FSH) > min (BSH)	6	12	7
	min (FSH) = min (BSH)	48	33	13
EX vs. BSH:	min (EX) < min (BSH)	77	89	112
	min (EX) > min (BSH)	1	0	0
	min (EX) = min (BSH)	47	36	13
naEX vs. naFSH:	min (naEX) < min (naFSH)	18	28	70
	min (naEX) > min (naFSH)	78	70	33
	min (naEX) = min (naFSH)	29	27	22
naFSH vs. naBSH:	min (naFSH) < min (naBSH)	90	94	108
	min (naFSH) > min (naBSH)	12	12	5
	min (naFSH) = min (naBSH)	23	19	12
naEX vs. naBSH:	min (naEX) < min (naBSH)	69	81	100
	min (naEX) > min (naBSH)	40	27	13
	min (naEX) = min (naBSH)	16	17	12

BSH. Also, the performance gap of the two operators increases with increasing *n*. It has to be mentioned however, that the observation of an increasingly better performing FSH relative to BSH is partially based on the fact that the number of instances in which both operators identified the optimal/best known solution decreases with increasing *n*.

Similar observations can be made when comparing BSH and EX. Moreover, the difference between the operators is even bigger in favor of EX.

The comparison of EX vs. FSH shows that the exchange neighborhood leads in more instances to better results than vice versa. While based on the theoretical comparison of the two operators, no clear hypothesis could have been formulated, the experimental investigations reveal a rather clear advantage of EX over FSH. There are however still some instances in which the opposite relation holds, and the results are therefore less obvious.

Somewhat surprisingly, a different picture arises when looking at the results of the disjunct neighborhoods. Here, naFSH performs best. Especially for instances which have been generated using a $TF \in \{0.6, 0.8, 1.0\}$, the naEX neighborhood fails to identify an optimal solution. On the other hand, the rather weak operator aEX successfully solves some of these data sets. This indicates that adjacent moves are an important ingredient in EX, contributing to the overall performance of the neighborhood operator.

4.3. Average quality of local optima

In addition to the minimum values for the total weighted tardiness we computed and compared average values. It can be suggested that this analysis leads to a clearer distinction between the different operators, as only few instances can be expected in which two neighborhoods lead in average to the same results. Tables 6 and 7 give the obtained values for the best-improvement and the first-improvement strategies.

The comparison of average values leads to the same observations as comparing minimum values. Again, BSH turns out to lead to inferior results in significantly more instances than FSH and EX. Also, the performance gap grows with increasing *n*. Identical to the observation made on the basis of minimum values, EX outperforms FSH in more instances than FSH does EX. The relation in terms of the relative performance also gets stronger with increasing *n*.

Contrary to the relative performance when looking for the best results, naEX leads to, in average, better results than naFSH. Also, this relation becomes stronger with increasing *n*.

Due to the stochastic nature of generating the initial solution, some variance in the obtained results is present. We therefore computed (counted) the number of instances for which a particular neighborhood lead to significant better results than the other. The maximum accepted error has been set to 1%, and the results are shown in Tables 6 and 7. While for some few instances, the measured differences are not significant, for the vast majority of instances they are.

In the light of the experimental results, we accept Hypotheses 2–5.

5. Variable neighborhood descent

5.1. Configuration

On the basis of past studies of local search for the SMTWTP [14], and following the outcomes of the previous section, we investigate

Table 6
Number of instances for which the comparison of average values holds (*best-improvement-strategy*). Besides the total number of such instances (column '#'), 125 being the maximum for each value of n, the number of instances for which the difference is significant (error $\leqslant 1\%$) is given in columns '#sig.'

Operator comparison		$n = 40$		$n = 50$		$n = 100$	
		#	#sig.	#	#sig.	#	#sig.
EX vs. FSH:	aver (EX) < aver (FSH)	93	88	95	93	101	99
	aver (EX) > aver (FSH)	19	13	19	9	8	6
	aver (EX) = aver (FSH)	13		11		16	
FSH vs. BSH:	aver (FSH) < aver (BSH)	123	116	122	117	125	124
	aver (FSH) > aver (BSH)	2	1	3	1	0	0
	aver (FSH) = aver (BSH)	0		0		0	
EX vs. BSH:	aver (EX) < aver (BSH)	124	121	125	124	125	125
	aver (EX) > aver (BSH)	1	0	0	0	0	0
	aver (EX) = aver (BSH)	0		0		0	
naEX vs. naFSH:	aver (naEX) < aver (naFSH)	81	70	87	82	99	97
	aver (naEX) > aver (naFSH)	33	24	27	23	11	8
	aver (naEX) = aver (naFSH)	11		11		15	
naFSH vs. naBSH:	aver (naFSH) < aver (naBSH)	117	113	123	119	124	122
	aver (naFSH) > aver (naBSH)	8	1	2	1	1	0
	aver (naFSH) = aver (naBSH)	0		0		0	
naEX vs. naBSH:	aver (naEX) < aver (naBSH)	117	115	118	117	125	125
	aver (naEX) > aver (naBSH)	8	7	7	4	0	0
	aver (naEX) = aver (naBSH)	0		0		0	

Table 7
Number of instances for which the comparison of average values holds (*first-improvement-strategy*). Again, significance test are carried out (error $\leqslant 1\%$).

Operator comparison		$n = 40$		$n = 50$		$n = 100$	
		#	#sig.	#	#sig.	#	#sig.
EX vs. FSH:	aver (EX) < aver (FSH)	104	99	104	103	105	105
	aver (EX) > aver (FSH)	5	4	6	2	1	0
	aver (EX) = aver (FSH)	6		5		19	
FSH vs. BSH:	aver (FSH) < aver (BSH)	106	89	107	89	90	73
	aver (FSH) > aver (BSH)	19	6	18	7	35	16
	aver (FSH) = aver (BSH)	0		0		0	
EX vs. BSH:	aver (EX) < aver (BSH)	124	122	124	124	125	125
	aver (EX) > aver (BSH)	1	0	1	1	0	0
	aver (EX) = aver (BSH)	0		0		0	
naEX vs. naFSH:	aver (naEX) < aver (naFSH)	98	92	103	100	106	106
	aver (naEX) > aver (naFSH)	14	8	7	5	2	0
	aver (naEX) = aver (naFSH)	13		15		17	
naFSH vs. naBSH:	aver (naFSH) < aver (naBSH)	102	90	101	80	82	59
	aver (naFSH) > aver (naBSH)	23	10	24	8	43	27
	aver (naFSH) = aver (naBSH)	0		0		0	
naEX vs. naBSH:	aver (naEX) < aver (naBSH)	118	114	123	120	125	124
	aver (naEX) > aver (naBSH)	7	4	2	0	0	0
	aver (naEX) = aver (naBSH)	0		0		0	

two different configurations of Variable Neighborhood Descent (VND), a simplified variant of Variable Neighborhood Search (VNS) [16]. VNS is based on the observation, that an alternative which is locally optimal with respect to a particular operator is not necessarily locally optimal w.r.t. some other neighborhood. On the other hand, the global optimum is, by definition, locally optimum w.r.t. any operator. The VND/VNS metaheuristic thus makes use of multiple neighborhoods, changing the operator when some momentarily investigated neighborhood fails to improve the current best solution.

The first proposed VND algorithm, VND-1, applies the basic neighborhood operators in order of EX → FSH → BSH, while the second, VND-2, implements the reverse order BSH → FSH → EX. Knowing that there appears to be a relative order of neighborhood operators, it will be interesting to further investigate the effects of the different configurations of the VND algorithms.

All neighborhoods are searched in a best-move-fashion, thus searching the entire neighborhood. As known from VNS, the neighborhood is switched to the succeeding operator if the active one fails to improve the current solution. Search terminates with the identification of a local optimum, which is, in case of VND/VNS, an alternative being locally optimal with respect to all considered neighborhoods. Obviously, the implemented VND algorithm here deviates from the general concept of Variable Neighborhood Search, which also makes use of escape strategies such as 'shaking'-moves. This however is intentional, as we aim to study the quality of local optima that have been obtained by neighborhood search only.

Again, 100 test runs have been carried out for each benchmark instance/VND-configuration, each starting from a (different) random job permutation.

In comparison to other metaheuristics, such as Simulated Annealing or Tabu Search, the here presented VND algorithms come with the feature of providing results on the basis of (local) neighborhood search only, without employing escape strategies that allow a continuation once a local optimum is reached. In this

sense, we may expect that the obtained insights will be based on the 'power' of the neighborhood operators, free from the influence of additional control structures (which are helpful but not within the scope of this study).

5.2. Results

Table 8 shows the number of instances which have successfully been solved by the VND algorithms in at least a single out of the 100 test runs.

We can see that VND-2 is able to solve more instances than VND-1. It can be suspected that this behavior originates from the relative order of neighborhood operators. VND-2 starts with the weak operator BSH, and then continues to relatively better neighborhoods. Therefore, an improvement of solutions being locally optimal to the initially searched neighborhoods appears to be more likely as in the case of VND-1, where search is organized starting with relatively good operators.

Again, the number of solved instances is decreasing with increasing n. However, the results remain on a rather high level despite the growing difficulty of the benchmark instances. Only few instances remain unsolved.

More detailed results are given in Table 9, which shows the number of solved instances with respect to the choices of *RDD* and *TF*. In comparison to the results obtained by hillclimbing, a similar pattern arises in the investigation of VND. Again, both small and high values of *TF* lead to easy instances. In the light of the overall impressive results of VND however, this pattern is less obvious. Knowing that VND-2 successfully solved 97.6% of the benchmark instances, only few instances are left that can be considered to be 'difficult'.

Besides the best case behavior of the algorithms, the average quality of the local optima over all 100 test runs is interesting. Table 10 gives the results with respect to this.

Also on an average level, VND-2 leads to better results than VND-1. Moreover, the average deviation from the optimal/best known solutions is rather small, which indicates that VND is indeed an effective approach for most instances from the literature.

Unfortunately however, a considerable dispersion can be identified within these results. As shown in Table 11, the average results deviate for some few instances significantly more than in the majority of data sets. Even for some smaller instances, i.e. $n = 40$ and $n = 50$, considerable deviations can be found in some cases.

Combining the results of Tables 10 and 11, an interesting observation emerges. While VND-2 successfully solves the instances with $TF = 0.2/RDD = 0.2$, $TF = 0.4/RDD = 0.6$, $TF = 0.4/RDD = 0.8$ in the *best case*, the results show a relative high deviation from the optimal/best known results in the *average case*. On the other hand, instances that are not solved in the best case, i.e. the ones with $TF = 0.6/RDD = 0.8$, are approximated very nicely in average.

5.3. Computational effort

The following Table 12 gives an overview about the required computational effort for finding a local optimum, depending on the neighborhood operator/metaheuristic and the problem size n. While the generation of the neighboring solution (by means of modifying some job positions) as such is not very time consuming, the evaluation of the newly generated schedule is, simply as the starting times of various jobs need to be re-computed. Independent from the computer hardware in use, this element of the local search heuristics is consequently going to influence the computational effort the most. The number of evaluated solutions until

Table 8
Number of instances for which optimal/best known solutions have been found by VND.

Algorithm	$n = 40$	$n = 50$	$n = 100$
VND-1 (EX → FSH → BSH)	121	114	108
VND-2 (BSH → FSH → EX)	125	124	117

Table 9
Number of instances solved to optimality by VND depending on *RDD* and *TF*.

		VND-1					VND-2				
	TF:	0.2	0.4	0.6	0.8	1.0	0.2	0.4	0.6	0.8	1.0
RDD:	0.2	15	14	11	13	15	15	15	15	15	15
	0.4	13	14	14	12	15	15	15	14	14	15
	0.6	15	11	11	15	15	15	15	15	15	14
	0.8	15	11	13	15	15	15	15	12	14	15
	1.0	15	14	12	15	15	15	14	15	14	15

Table 10
Average deviation from the optimal/best known solution.

Algorithm	$n = 40$ (%)	$n = 50$ (%)	$n = 100$ (%)
VND-1 (EX → FSH → BSH)	2.59	2.36	2.10
VND-2 (BSH → FSH → EX)	0.99	1.45	0.98

Table 12
Average number of evaluations for finding a local optimum (best-improvement-strategy).

	$n = 40$	$n = 50$	$n = 100$
EX	23,147	46,355	416,920
FSH	24,866	51,471	513,764
BSH	29,563	62,637	693,206
aEX	5,720	10,774	78,742
naEX	18,313	37,552	352,326
naFSH	23,084	48,191	489,338
naBSH	27,486	58,744	669,395
VND-1	26,389	52,692	471,406
VND-2	39,847	85,420	950,904

Table 11
Average deviation of VND-2 from the optimal/best known solutions given in percent.

		$n = 40$					$n = 50$					$n = 100$				
	TF:	0.2	0.4	0.6	0.8	1.0	0.2	0.4	0.6	0.8	1.0	0.2	0.4	0.6	0.8	1.0
RDD:	0.2	6.7	0.4	0.2	0.2	0.0	1.5	0.4	0.4	0.1	0.0	0.4	0.2	0.2	0.2	0.0
	0.4	0.0	0.3	0.6	0.1	0.0	13.9	1.2	0.4	0.1	0.0	1.1	1.1	0.4	0.1	0.0
	0.6	0.0	4.6	0.9	0.0	0.0	0.0	3.1	1.8	0.2	0.0	0.0	2.5	0.6	0.1	0.0
	0.8	0.0	9.7	0.3	0.0	0.0	0.0	10.3	0.8	0.2	0.0	0.0	15.9	1.1	0.1	0.0
	1.0	0.0	0.0	0.6	0.1	0.0	0.0	0.0	1.9	0.1	0.1	0.0	0.2	0.3	0.1	0.0

identifying a local optimum has therefore been chosen for assessing the computational effort of the different hillclimbers.

Clearly, the length of the hillclimbing runs significantly increases with n, and it becomes clear that for large values of n, the identification of even a local optimum becomes intractable. On the other hand, the basic operators EX, FSH and BSH do not show bigger differences. On the contrary, the relatively best-performing neighborhood EX requires less computations for finding (better) local optima. Consequently, the effort for the execution of VND-1 is much smaller in comparison to VND-2, as the latter implementation first starts with the computationally demanding identification of qualitatively inferior local optima for BSH.

When analyzing the number of evaluations needed for the decomposed neighborhoods aEX, naEX, naFSH and naBSH, we are able to show that each of them converges faster to a local optimum than the combined counterparts. This is not surprising, as the neighborhoods are of smaller cardinality. Then however, the obtained results are of lower quality, which balances out the observed phenomenon.

Besides the number of evaluations required for identifying a local optimum, the computing time for the generation and evaluation of a single alternative influences the overall computational effort. On an Intel Xeon X5550 processor, running at 2.67 GHz, we have been able to compute and evaluate a single alternative for $n = 40$ in 0.0111 milliseconds, while making use of a single core only. For $n = 50$, the average time spent on an alternative is 0.0138 milliseconds, and for $n = 100$, it is 0.0300 milliseconds. This implies that the computational effort for finding a local optimum is heavier influenced by the number of evaluations, than the time required for evaluating a single alternative.

6. Summary and conclusions

An investigation of local search neighborhoods for the single machine total weighted tardiness scheduling problem has been presented. We first theoretically analyzed the possible effect of the operators on the tardiness values, leading to the proposition of four hypotheses on the relative performance of the neighborhoods.

Experimental investigations have been carried out on benchmark instances taken from the literature. We observed that, especially for small problem instances, numerous optimal solutions have been found despite the simplistic heuristic search framework. In conclusion, and based on the numerical results, the proposed hypotheses have been accepted.

The exchange of jobs turns out to be a superior local search operator to forward shift, and both are clearly superior to the backward shift neighborhood. The results are, with few counterexamples, clear and hold for instances of different sizes. We suggest to integrate these relations of operators in the future proposition of local search heuristics for the SMTWTP. In multi-operator search, EX should be given priority over FSH, and FSH over BSH.

When combining the elementary neighborhoods in a Variable Neighborhood Search framework, surprisingly good results are obtained. Almost all benchmark instances reported in the literature are solved to optimality, and the average gap to the optima is rather small. Our experiments therefore support the claim that existing benchmark data does not present a challenge to state-of-the-art metaheuristics, especially as rather simple local search already leads to good results. Future research in the SMTWTP must therefore make use of considerable larger instances, which preferably are generated with a tardiness factor TF around 0.6.

References

[1] T.S. Abdul-Razaq, C.N. Potts, L.N. Van Wassenhove, A survey of algorithms for the single machine total weighted tardiness scheduling problem, Discrete Applied Mathematics 26 (1990) 235–253.

[2] Joseph Adams, Egon Balas, Daniel Zawack, The shifting bottleneck procedure for job shop scheduling, Management Science 34 (3) (1988) 391–401.

[3] Arife Burcu Colka Altunc, Ahmet Burak Keha, Interval-indexed formulation based heuristics for single machine total weighted tardiness problem, Computers and Operations Research 36 (6) (2009) 2122–2131.

[4] Kenneth R. Baker, Introduction to Sequencing and Scheduling, John Wiley & Sons, New York, London, Sydney, Toronto, 1974.

[5] Andreas Bauer, Bernd Bullnheimer, Richard F. Hartl, Christine Strauss, An ant colony optimization approach for the single machine total tardiness problem, Proceedings of the 1999 Congress on Evolutionary Computation CEC99, vol. 2, IEEE Service Center, Piscataway, NJ, 1999, pp. 1445–1450.

[6] J.E. Beasley, OR-library: Distributing test problems by electronic mail, Journal of the Operational Research Society 41 (11) (1990) 1069–1072.

[7] Ümit Bilge, Müjde Kurtulan, Furkan Kıraç, A tabu search algorithm for the single machine total weighted tardiness problem, European Journal of Operational Research 176 (2007) 1423–1435.

[8] Wojciech Bożejko, Józef Grabowski, Mieczynsław Wodecki, Block approach – Tabu search algorithm for single machine total weighted tardiness problem, Computers and Industrial Engineering 50 (2006) 1–14.

[9] Fuh-Der Chou, An experienced learning genetic algorithm to solve the single machine total weighted tardiness scheduling problem, Expert Systems with Applications 36 (2009) 3857–3865.

[10] Richard K. Congram, Chris N. Potts, Steef L. van de Velde, An iterated dynasearch algorithm for the single-machine total weighted tardiness problem, INFORMS Journal on Computing 14 (1) (2002) 52–67.

[11] H.A.J. Crauwels, C.N. Potts, L.N. Van Wassenhove, Local search heuristics for the single machine total weighted tardiness scheduling problem, INFORMS Journal on Computing 10 (3) (1998) 341–350.

[12] Federico Della Croce, Generalized pairwise interchanges and machine scheduling, European Journal of Operational Research 83 (1995) 310–319.

[13] Matthijs den Besten, Thomas Stützle, Marco Dorigo, Ant colony optimization for the total weighted tardiness problem, in: M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J.J. Merelo, H.-P. Schwefel (Eds.), Parallel Problem Solving from Nature-PPSN VI, Lecture Notes in Computer Science, vol. 1917, Springer Verlag, Berlin, Heidelberg, New York, 2000, pp. 611–620.

[14] Matthijs den Besten, Thomas Stützle, Marco Dorigo, Design of iterated local search algorithms – An example application to the single machine total weighted tardiness problem, in: E.J.W. Boers, J. Gottlieb, P.L. Lanzi, R.E. Smith, S. Cagnoni, E. Hart, G.R. Raidl, H. Tijink (Eds.), Applications of Evolutionary Computing, Lecture Notes in Computer Science, vol. 2037, Springer Verlag, Berlin, Heidelberg, New York, 2001, pp. 441–451.

[15] A. Grosso, F. Della Groce, R. Tadei, An enhanced dynasearch neighborhood for the single-machine total weighted tardiness scheduling problem, Operations Research Letters 32 (2004) 67–72.

[16] Pierre Hansen, Nenad Mladenović, Developments of variable neighborhood search, in: Celso C. Ribeiro, Pierre Hansen (Eds.), Essays and Surveys in Metaheuristics, Kluwer Academic Publishers, Boston, Dordrecht, London, 2002, pp. 415–439.

[17] J.K. Lenstra, A.H.G. Rinnooy Kan, P. Brucker, Complexity of machine scheduling problem, in: P.L. Hammer et al. (Eds.), Studies in Integer Programming, Annals of Discrete Mathematics, vol. 1, North Holland, Amsterdam, The Netherlands, 1977, pp. 343–362.

[18] R. Maheswaran, S.G. Ponnambalam, An intensive search evolutionary algorithm for single-machine total-weighted-tardiness scheduling problems, International Journal of Advanced Manufacturing Technology 26 (2005) 1150–1156.

[19] R. Maheswaran, S.G. Ponnambalan, An investigation on single machine total weighted tardiness scheduling problems, The International Journal of Advanced Manufacturing Technology 22 (3–4) (2003) 243–248.

[20] Daniel Merkle, Martin Middendorf, An ant algorithm with a new pheromone evaluation rule for total tardiness problems, in: S. Cagnoni, R. Poli, G.D. Smith, D. Corne, M. Oates, E. Hart, P.L. Lanzi, E.J. Willem, Y. Li, B. Paechter, T.C. Fogarty (Eds.), Real-World Applications of Evolutionary Computing, Lecture Notes in Computer Science, vol. 1903, Springer Verlag, Berlin, Heidelberg, New York, 2000, pp. 287–296.

[21] A.C. Nearchou, Solving the single machine total weighted tardiness scheduling problem using a hybrid simulated annealing algorithm, in: Second IEEE International Conference on Industrial Informatics INDIN'04, Berlin, Germany, June 2004, pp. 513–516.

[22] R. Panneerselvam, Simple heuristic to minimize total tardiness in a single machine scheduling problem, International Journal of Advanced Manufacturing Technology 30 (2006) 722–726.

[23] Colin R. Reeves, Landscapes, operators and heuristic search, Annals of Operations Research 86 (1999) 473–490.

[24] Xianpeng Wang, Lixin Tang, A population-based variable neighborhood search for the single machine total weighted tardiness problem, Computers and Operations Research 36 (6) (2009) 2105–2110.