Computers & Operations Research 66 (2016) 44–57

Contents lists available at ScienceDirect

Computers & Operations Research

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

A study on local search neighborhoods for the job shop scheduling problem with total weighted tardiness objective

J. Kuhpfahl*, C. Bierwirth

Martin-Luther-University Halle-Wittenberg, School of Economics and Business, Große Steinstraße 73, D-06108 Halle, Germany

ARTICLE INFO

Available online 3 August 2015

Keywords:
Job shop scheduling
Total weighted tardiness
Disjunctive graph model
Neighborhoods
Heuristics

ABSTRACT

In this paper we consider the job shop scheduling problem with total weighted tardiness objective (JSPTWT). This objective reflects the goal to achieve a high service level which is of increasing importance in many branches of industry. The paper concentrates on a class of baseline heuristics for this problem, known as neighborhood search techniques. An approach based on disjunctive graphs is developed to capture the general structure of neighborhoods for the JSPTWT. Existing as well as newly designed neighborhoods are formulated and analyzed. The performance and search ability of the operators (as well as combinations thereof) are compared in a computational study. Although no dominant operator is identified, a transpose-based perturbation on multiple machines turns out as a promising choice if applied as the only operator. Combining operators improves the schedule quality only slightly. But, the implementation of operators within a meta-heuristic enables to produce a higher schedule quality. A structural classification of neighborhood operators and some new analytical results are presented as well.

1. Introduction

The vast majority of research in job shop scheduling is dealing with problems where finding a schedule with a minimum makespan is pursued [2]. In recent years, however, also tardiness objectives have received increasing attention because on-time order fulfillment is of predominant importance in modern pull-oriented supply chain systems. Keeping job due dates is a prerequisite for serving customers within the promised delivery time and avoiding out-of-stock costs. In a situation where not all job due dates can be met, the minimization of the total (weighted) tardiness of the jobs turns out as an appropriate objective for machine scheduling.

In this paper we study basic neighborhood search techniques, originally developed to solve classic job shop problems (JSP). We analyze their performance in the context of the total weighted tardiness objective. The considered neighborhood search techniques constitute a class of baseline heuristics for solving complex machine scheduling problems in general. At its core, a neighborhood search technique performs a particular perturbation of a given schedule where the new schedule replaces the former schedule provided it is of better quality. The way in which a perturbation is executed defines a set of neighboring schedules, which is briefly referred to as a neighborhood definition. Different neighborhood definitions have been proposed for the JSP and tested in combination with meta-heuristics like Simulated Annealing and Tabu Search, see e.g. [9,23]. Later, further neighborhood definitions have been proposed and also implemented for solving the job shop scheduling problem with total weighted tardiness objective (JSPTWT), see e.g. [6,26].

The aim of this paper is manifold. First we present a generic approach which allows us to describe all known neighborhood definitions for scheduling problems in a systematic fashion. The approach is based on the well known disjunctive graph model which is adapted to the requirements of the JSPTWT. New neighborhood definitions are derived from our generic approach by identifying schedule perturbations which have not been considered so far. Finally, we investigate key characteristics for the proposed neighborhood definitions which are expected to indicate differences in their behavior and search quality. The intention is to provide support on selecting and implementing neighborhood search operators within powerful scheduling systems.

The paper is organized as follows: the next section introduces the problem under consideration and reviews the relevant literature. Section 3 presents the modified disjunctive graph model forming a framework for the different neighborhood operators described in Section 4. The feasibility guarantee and the connectivity property of neighborhoods are investigated analytically in Section 5. In Section 6, the search quality and performance of the

* Corresponding author.
E-mail addresses: jens.kuhpfahl@wiwi.uni-halle.de (J. Kuhpfahl), christian.bierwirth@wiwi.uni-halle.de (C. Bierwirth).

http://dx.doi.org/10.1016/j.cor.2015.07.011
0305-0548/© 2015 Elsevier Ltd. All rights reserved.

various neighborhoods are assessed within a comprehensive computational study where the operators are tested by implementing them in a steepest descent algorithm as well as a simple simulated annealing algorithm. Section 7 concludes the paper.

2. Problem definition and literature review

In the standard JSP we are given n jobs that have to be processed on m machines. Each job consists of m operations that have to be processed in a given technological sequence, i.e. the processing of an operation is restricted to a preassigned machine. Preemption during the execution of an operation as well as overlapping is not allowed. The processing time of the i-th operation of job j is denoted by p_{ij}. The standard JSP is to find a schedule that minimizes the completion time of the latest finished job, also referred to as the makespan of a schedule.

The JSPTWT is an extension of the standard JSP where the processing of job j cannot start before a given ready time r_j and should be finished no later than a given due date d_j. The objective function value of a schedule is defined as $TWT = \sum_{j=1}^{n} w_j \cdot \max\{0, c_j - d_j\}$, where c_j denotes the completion time of job j and w_j denotes its weight.

In contrast to the standard JSP, the JSPTWT has received much less attention in the literature. Beside a Branch-and-Bound approach proposed by Singer and Pinedo [20], research mainly concentrates on the development of heuristics. Specific priority rules, e.g. the Apparent Tardiness Cost rule (ATC) [24], have been proposed for constructing schedules such that due dates are taken into account. A corresponding survey of due date related dispatching rules is provided by Jayamohan and Rajendran [11]. A Shifting Bottleneck Procedure for the JSPTWT is presented by Pinedo and Singer [19] where single machine subproblems are solved by a due date related dispatching rule. A tabu search approach to the JSPTWT with unit job weights is proposed by Armentano and Scrich [3]. For local search, the critical transpose neighborhood operator proposed by van Laarhoven [22] is used. Local search is also employed in the large step random walk procedure of Kreipl [13]. In the intensification phase, a local optimum is determined using the neighborhood operator of Matsuo et al. [16]. In the diversification phase, a new feasible start solution is generated by perturbing the local optimum doing a number of random neighborhood moves. Zhang and Wu [26] propose a simulated annealing algorithm which incorporates a neighborhood operator relying on a block property. A hybrid heuristic has been developed for the JSPTWT by Essafi et al. [10] who combine a genetic algorithm with local search using the neighborhood operator of van Laarhoven [22], called genetic local search. Another hybrid approach is presented by Bülbül [6] who integrates a tabu search with a shifting bottleneck procedure.

The above review indicates that heuristic approaches to the JSPTWT are often based on local search mechanisms although local search is time-consuming due to the need of calculating the objective function value of a newly produced schedule. Therefore, Mati et al. [15] have developed a fast estimation method for a lower bound on the objective function value which drastically accelerates the evaluation of moves performed with the neighborhood operator of van Laarhoven. Similarly, Braune et al. [7] have developed an evaluation method for schedules generated with the neighborhood operator of Dell'Amico and Trubian [9]. Based on neighborhood search techniques, a lot of progress has been made in solving large-scale JSPTWT instances over the last decade. Still, there is a lack of knowledge regarding the impact of different neighborhood operators incorporated in the scheduling procedures.

3. Disjunctive graph model

The disjunctive graph model is a fundamental problem representation form for the standard JSP, see Adams et al. [1]. In this section, we briefly revisit the transformation of the disjunctive graph model for the JSPTWT as proposed by Pinedo and Singer [19] as well as Kreipl [13]. Furthermore, we propose a slight modification of their graph model which allows us to gather the total tardiness value of a solution directly from its graph representation.

The standard JSP is represented by a disjunctive graph $G=(V,A,E)$, where the i-th operation of job j is denoted by node $(i/j) \in V$. The set of nodes is completed by two dummy nodes, namely the start node 0 and the finishing node 1. The set of operations of job j is connected by directed arcs $[(i/j),(i+1/j)] \in A$ representing its technological machine sequence. For every job j, there is one directed arc connecting the start node with the first operation node, $[0,(1/j)] \in A$, and one further directed arc connecting the last operation node with the finishing node, $[(m/j),1] \in A$. A weight is given for every arc in A which represents the processing time of an operation. Arcs $[0,k] \in A$ have a weight of 0. The precedence relation between operations $k, l \in V$ of different jobs being processed on the same machine is represented by pairs of disjunctive arcs $[k,l],[l,k] \in E$. By selecting one arc of each pair of disjunctive arcs we obtain a subset E' of E which describes a feasible schedule if and only if the corresponding graph $G'=(V,A,E')$ is acyclic. The length of the longest path from 0 to 1 in G' determines the makespan of the represented schedule.

For the JSPTWT, [13,19] have modified the disjunctive graph model by introducing an individual sink node B_j for every job j (instead of the common node 1) which measures the completion time of the job. In addition to this, we introduce a new sink node denoted F_j for every job j which is simply appended to B_j. Furthermore, arcs $[0,F_j]$ and $[B_j,F_j]$

Table 1
Data to the example 3 × 3 instance.

Job data			Op. 1		Op. 2		Op. 3		
j	w_j	r_j	Mch.	p_{1j}	Mch.	p_{2j}	Mch.	p_{3j}	d_j
1	1	0	1	3	2	4	3	4	12
2	2	2	3	2	1	3	2	4	15
3	1	2	1	5	3	5	2	2	18

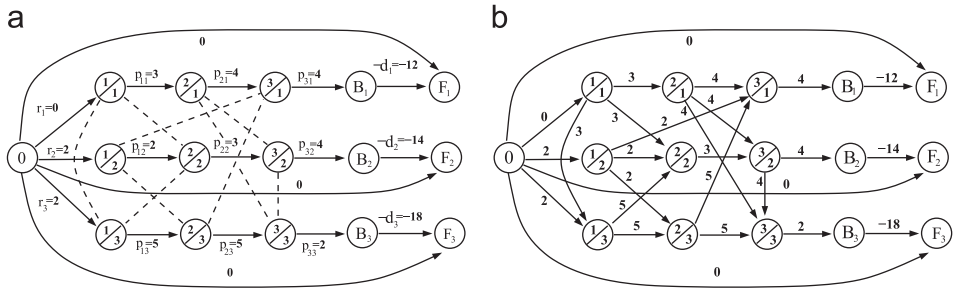


Fig. 1. JSPTWT: (a) Disjunctive graph G (dashed line = pair of disjunctive arcs), (b) feasible solution as directed graph G'.

are added to the directed arc set A. Weights 0 and $-d_j$ are assigned to these arcs to compute the tardiness of job j as the non-negative distance between its completion time and due date. Additionally, the weight of arc $[0,k] \in A$ represents the ready time of the job with k as its first operation. Fig. 1(a) shows the disjunctive graph G for a small instance of the JSPTWT, defined in Table 1. We can represent a feasible schedule of the JSPTWT by selecting one arc of each pair of disjunctive arcs leading to a directed acyclic graph G'. The graph representation is shown for a randomly chosen solution for the JSPTWT instance in Fig. 1(b). To compute the total weighted tardiness of this schedule, we have to determine for every job j a longest path starting at 0 and ending at the sink node F_j. The length of the longest path from 0 to F_j corresponds to the tardiness of job j. For example, considering job 1 in Fig. 1(b), the longest path leading from 0 to its completion node B_1 is $[0,(1/1),(1/3),(2/3),(3/1),B_1]$ which is of length $c_1 = 17$. The tardiness of job 1 is determined by the length of the longest path from 0 to F_1, given by $t_1 = \max\{0, c_1 - d_1\} = \max\{0, 17-12\} = 5$. Similarly, the completion of job 3 is determined by the longest path $[0,(1/1),(1/3),(2,2),(3/2),(3/3),B_3]$ from node 0 to B_3 which is of length $c_3 = 17$. For job 3, however, the longest path leading from 0 to node F_3 consists just of arc $[0,F_3]$ because $t_3 = \max\{0, c_3 - d_3\} = \max\{0, 17-18\} = 0$. In other words, job 3 is on time.

Following Nowicki and Smutnicki [18], a longest path is decomposable into critical arcs and critical blocks. We refer to an arc of a longest path as a critical arc if it connects two operations that are processed consecutively on the same machine. A critical block is defined as a sequence of operations connected by a maximum chain of critical arcs belonging to one longest path. Note that while assessing the total weighted tardiness of schedule G', all critical arcs and blocks are identified during the construction of the n longest paths.

4. Neighborhoods for job shop scheduling

The concept of disjunctive graphs is fundamental to the definition of local search neighborhoods. A neighborhood definition typically provides a small set of similar solutions that enable the move from a current feasible solution to a new and hopefully better feasible solution [2]. Such nearby solutions are generated according to a defined perturbation scheme which is applied by a neighborhood operator. The most basic perturbation is the reversal of one critical arc in the disjunctive graph representation. This perturbation can potentially improve the schedule quality but it cannot destroy the schedule's feasibility. More complex perturbation schemes promise to find even larger improvements of a schedule but they often cannot guarantee schedule feasibility. The approach adopted in this paper is to detect cycles by iteratively determining the starting times of all operations from their predecessors. The run-time complexity of this test is $\mathcal{O}(nm)$, see Taillard [21]. Usually either the best improving feasible solution or the first improving feasible solution found in the neighborhood of a schedule replaces this schedule. Continuing this procedure ends up in a local optimum, which might be the global optimum in rare cases.

4.1. Old neighborhoods

Various neighborhood definitions have been proposed for machine scheduling problems, see Table 2. In this section we briefly reflect the corresponding schedule perturbation schemes using the disjunctive graph approach.

CT: The *critical transpose* neighborhood of van Laarhoven [22] defines the elementary perturbation of a schedule which is a reversal of one critical arc, also known as interchange [16] or swap [15] of two adjacent operations. van Laarhoven et al. [23] have shown that the resulting schedule is always feasible with respect to the perturbation scheme of the operator. Moreover, the CT operator can only reduce the length of a longest path in case that the reversed critical arc is positioned at the beginning or end of a critical block. Otherwise, when the reversed arc is inside the critical block, it does not change the starting time of the succeeding operation in the critical block, and thus, it does not reduce the length of this path. This observation motivates a simple modification of the CT to yield a dominating performance. The modified operator, called CET, is described below in detail. It substitutes the CT operator in the computational investigations.

CET: Since the reversal of critical arcs inside a critical block cannot yield an improvement, Nowicki and Smutnicki [18] have proposed a reduced CT neighborhood, called *critical end transpose* neighborhood, which reverses only critical arcs at the beginning or at the end of a critical block. Fig. 2 shows an example with a critical block $[u_1, u_2, u_3, u_4]$ consisting of three critical arcs. The CET operator reverses either arc $[u_1, u_2]$ or $[u_3, u_4]$ as it is depicted by the gray colored arc in Fig. 2. Since the CET neighborhood defines an efficient subset of the CT neighborhood, it always produces feasible schedules.

CET+2MT: Matsuo et al. [16] have enhanced the idea of CET by reversing additional arcs related to predecessors and successors in the machine sequences, see Fig. 3. Based on the reversal of the critical arc $[u_1, u_2]$, two further machine arcs might be reversed simultaneously provided that the following conditions hold. The machine arc $[x_1, x_2]$ (with x_1 being the job successor operation of u_1) is reversed if and only if (i) x_1 is started before the completion of u_2 and (ii) x_1 is completed directly before the start of x_2. The

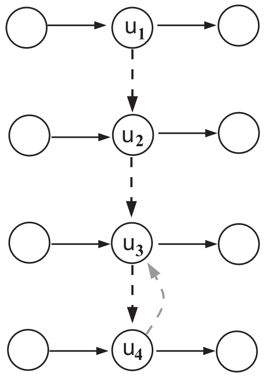


Fig. 2. CET move.

Table 2
List of neighborhood operators proposed in the literature. The naming of the neighborhoods refers to the denotation introduced by Anderson et al. [2].

Abbrev.	Name	Authors	Ref.
CT	Critical Transpose	van Laarhoven	[23]
CET	Critical End Transpose	Nowicki, Smutnicki	[18]
CET+2MT	Critical End Transpose + 2-Machine Transpose	Matsuo et al.	[16]
CE3P	Critical End 3-Permutation	Dell'Amico, Trubian	[9]
CEI	Critical End Insert	Dell'Amico, Trubian	[9]
CSR	Critical Sequence Reverse	Zhang, Wu	[26]

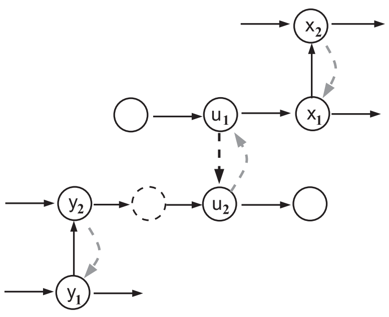


Fig. 3. CET+2MT move.

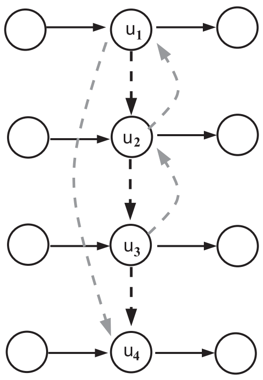


Fig. 4. CE3P move.

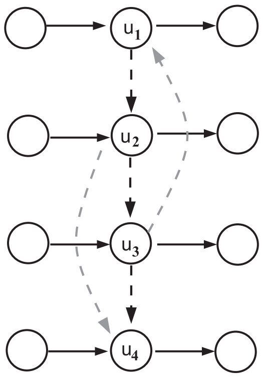


Fig. 5. CEI move.

machine arc $[y_1, y_2]$ (with y_2 being one of the job predecessor operations of u_2) is reversed if and only if (i) the direct job predecessor of u_2 is not completed before the start of u_1, (ii) y_2 is started immediately after the completion of y_1, and (iii) all operations between y_2 and u_2 are processed without idle time. These accompanying arc reversals attempt to break up blockings in the longest path which result from the reversal of the critical arc $[u_1, u_2]$. Accordingly, this operator is called *critical end transpose + 2-machine transpose* operator. A feasibility guarantee also holds for this operator which is shown in the Theorem 1 (see Section 5).

CE3P: The reversal of one critical arc can be viewed as a 2-permutation of two adjacent operations. Dell'Amico and Trubian have generalized this idea by the *critical end 3-permutation* operator. Additional to the two operations at the beginning (or the end) of a critical block, the directly preceding (or directly succeeding) operation on the machine is taken into account, even it does not belong to a longest path. A neighboring schedule is generated by changing the order of three consecutive operations by means of a 3-permutation. However, only 3 of the 6 existing 3-permutation are taken into consideration, namely those which reverse the two operations at the beginning (or at the end) of the critical block. For example, the order of the consecutive operations u_1, u_2 and u_3 at the beginning of a critical block can be changed by means of one of the 3-permutations $[u_2, u_1, u_3]$, $[u_2, u_3, u_1]$ or $[u_3, u_2, u_1]$. The last perturbation is shown in Fig. 4. As this operator sometimes produces infeasible solutions, the feasibility test is executed after the schedule generation. In case of an infeasible neighboring schedule, the performed perturbation is discarded.

CEI: A further neighborhood operator proposed by Dell'Amico and Trubian [9] attempts to shift an operation within a critical block. The new position of this operation in the block is determined as the most distanced insert position that still yields a feasible schedule. We call this operator the *critical end insert* operator. In order to ensure that the length of the longest path is effected by a perturbation, operations inside a critical block are only shifted to the first or last position of the block. In the example of Fig. 5, the operation u_3 is shifted to the front of the critical block, resulting in the new sequence $[u_3, u_1, u_2, u_4]$. Note that CEI already includes a feasibility test to ensure only feasible schedules can be generated. However, the perturbation scheme of the operator basically enables generating infeasible schedules which is why we classify it as an operator without feasibility guarantee.

A variant of CEI has been proposed by Balas and Vazacopoulos [4]. Instead of searching the most distanced feasible insert position for an operation, they determine the most distanced insert position for which feasibility is granted with regard to path conditions for cyclic aviodance. This variant is faster but can fail in finding the most improving CEI move. Therefore, we omit it in this study.

CSR: The idea to invert a connected sequence of operations in a critical block has been proposed by Zhang and Wu [26]. In the example of Fig. 6, four operations of the critical block $u_1, \ldots u_4$ are considered as the sequence to be inverted. The new sequence $[u_4, u_3, u_2, u_1]$ establishes the reversal of all critical arcs between u_1 and u_4. The described operator is called *critical sequence reverse* operator. Obviously, CSR guarantees the generation of feasible schedules for sequences of only two operations which corresponds to an ordinary CT perturbation [25]. Applying the operator to longer sequences requires a feasibility test to be executed in each step, i.e., after a single arc has been reversed.

4.2. New neighborhoods

We introduce six new perturbation schemes, which extend and supplement the above described concepts. Table 3 presents an overview of these neighborhoods.

ECET: Instead of inverting either the first or the last arc of a critical block (CET), the *extended critical end transpose* neighborhood inverts the first or the last arc or both simultaneously, see Fig. 7. Note that such a perturbation is identical with a CET move for blocks consisting of less than three critical arcs. The resulting schedule is always feasible as will be shown in the Section 5.

ICT: Large critical blocks basically determine the length of a longest path. The idea of the *iterative critical transpose* neighborhood is to break up long critical blocks by inverting every second arc, starting with the first arc of the block. Fig. 9 shows an ICT move for a block of length 5, inverting the first, the third and the fifth arc. The feasibility guarantee for ICT is shown in the next section as well.

CE4P: Like CE3P, this perturbation scheme considers either the first or the last arc of a critical block. But different to CE3P, both, the directly preceding and the directly succeeding operations of the critical arc are involved in the perturbation which leads to the *critical end 4-permutation* neighborhood. Consider four such consecutive operations $u_1, \ldots, u_4$, as shown in Fig. 8. To ensure that at

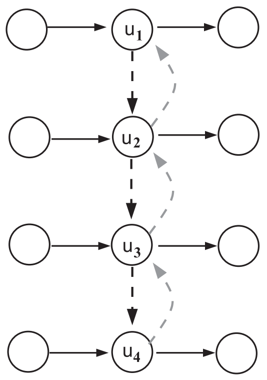


Fig. 6. CSR move.

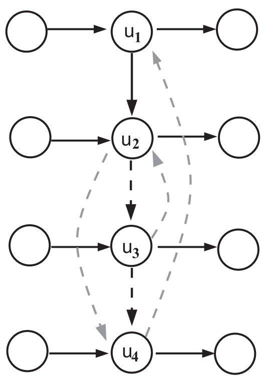


Fig. 8. CE4P move.

Table 3
List of new neighborhood operators.

Abbrev.	Name	Derived from
ECET	Extended Critical End Transpose	CET
ICT	Iterative Critical Transpose	CET
CE4P	Critical End 4-Permutation	CE3P
DOCEI	Double Outwards Critical End Insert	CEI
DICEI	Double Inwards Critical End Insert	CEI
CEI+2MT	Critical End Insert + 2-Machine Transpose	CEI, CET+2MT

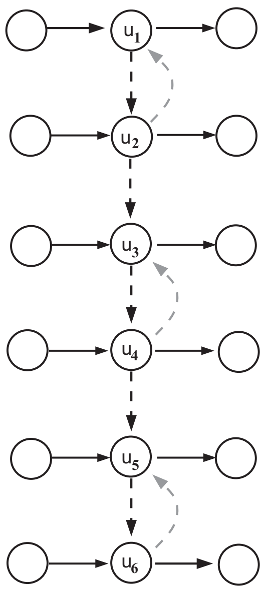


Fig. 9. ICT move.

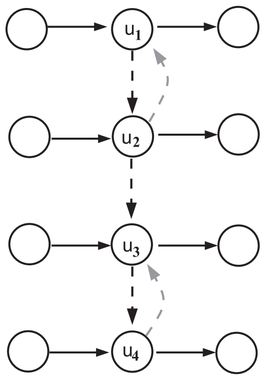


Fig. 7. ECET move.

least the order of the two inner operations u_2 and u_3 is inverted (which represents the initial critical arc), the operator can generate a total of 12 different 4-permutations. Of course, like CE3P, CE4P possibly produces infeasible schedules. Therefore, only those permutations are taken into consideration which pass the feasibility test.

DOCEI: The *double outwards critical end insert* neighborhood selects two consecutive operations in a critical block and inserts the former operation at the end of the critical block and the latter operation at the beginning of the critical block. Note that this perturbation scheme extends the concept of the CEI operator. Infeasible schedules are omitted. An example is shown in Fig. 10 for a block of length 5, where the consecutive operations u_3 and u_4 are first excluded and then reinserted at the end and the beginning of the critical block, which leads to the new operation sequence $u_4, u_1, u_2, u_5, u_6, u_3$. Being an extension of CEI, the feasibility guarantee does not hold for DOCEI.

DICEI: The perturbation by the *double inwards critical end insert* operator works like DOCEI with the exception that the selected consecutive operations are not shifted outwards to the border of the critical block, but the border operations are shifted inwards between the selected operations. Fig. 11 shows the same example considered for DOCEI before, now leading to the new operation sequence $u_2, u_3, u_6, u_1, u_4, u_5$. Being an extension of CEI, the feasibility guarantee does not hold for DICEI.

CEI+2MT: The *critical end insert + 2-machine transpose* neighborhood operator, first introduced in [14], combines a CEI move with a two machine transpose move (like in CET+2MT). In other words, the perturbation scheme joins a shift of an operation with two additional arc reversals. An example is illustrated in Fig. 12. Here, operation u_2 is shifted to the end of the critical block in CEI manner. Furthermore, to break up blockings in the schedule, two additional arcs are reversed which belong to the job successor of the shifted operation u_2 and the job predecessor of its succeeding operation u_3 in the critical block. Computational experiments have shown that shifting an operation to the beginning of the block rarely leads to improvements. A possible explanation for this could be that a shift of an operation to the beginning may cause multiple critical operations to start later while shifting an operation to the end may allow these operations to start earlier. Therefore, the operator is designed by shifting operations always to the end of the block. A feasibility guarantee does not hold for CEI+2MT like for CEI and its other extensions.

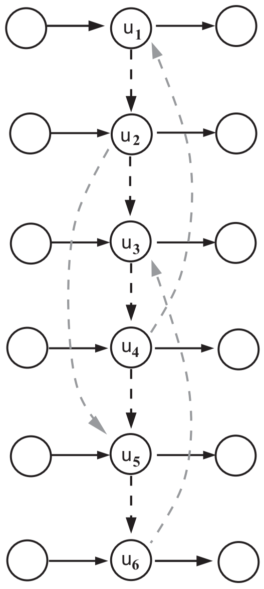


Fig. 10. DOCEI move.

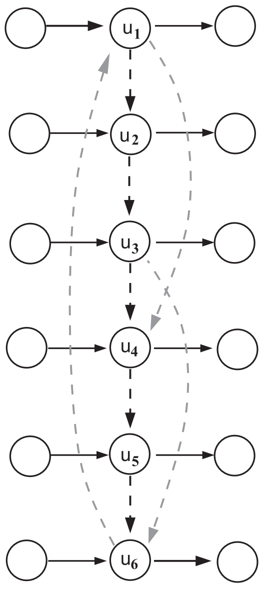


Fig. 11. DICEI move.

Based on the above definitions, the following inclusions of the neighborhood operators are obvious:

CET ⊆ CT ⊆ CSR
CET ⊆ CE3P ⊆ CE4P
CET ⊆ ECET
CET ⊆ CEI

A classification of the proposed neighborhoods is provided by the underlying principle of generating new schedules using transpose-based, insertion-based and sequence-based perturbations. Moreover, changing the order of operations of a schedule on a single or on multiple machines can be distinguished. The corresponding scheme is shown in Table 4. Note that the operators CE3P and CE4P appear in the class of sequence-based operators because their generation scheme is based on sequences of three and four operations, respectively.

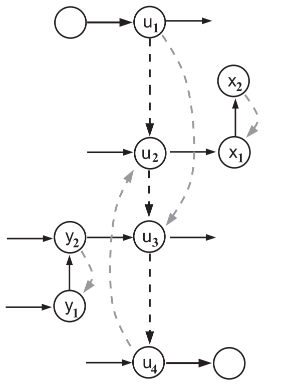


Fig. 12. CEI+2MT move.

Table 4
Classification of neighborhood operators based on perturbation schemes

	Transpose-based	Insertion-based	Sequence-based
On single machine	CT/CET ECET ICT	CEI DOCEI DICEI	CE3P CE4P CSR
On multiple machine	CET + 2MT	CEI+2MT	

5. Theoretical properties of neighborhood operators

Essential characteristics of neighborhood definitions are (i) the feasibility guarantee of the new solution, (ii) the connectivity of the underlying solution space, (iii) the size of the neighborhood, and (iv) the improvement rate observed for the obtained neighboring solutions [17]. While properties (i) and (ii) can be investigated analytically, properties (iii) and (iv) require empirical evidence taken from computational experiments. In this section, we review the existing knowledge about feasibility guarantees and the connectivity properties and further exhibit some new analytical results for the neighborhood definitions introduced in Sections 4.1 and 4.2.

5.1. Feasibility guarantee

The most basic perturbation of a schedule is performed by CT which simply reverses two adjacent critical operations (i.e. one critical arc). As proved by van Laarhoven et al. [23], a CT perturbation always results in a feasible schedule. Lemma 1 extends this result towards the reversals of adjacent non-critical operations, provided the operations are immediately started on the same machine without idle time. We use Lemma 1 to prove Lemma 2 and Theorem 1. Theorem 1 proves the feasibility guarantee for CET+2MT. Based on Lemma 2, feasibility guarantees are derived for ECET and ICT. The second Lemma claims that two simultaneous reversals of machine arcs (belonging to the same or different machines) as defined by two pairs of adjacent operations that are all distinct, will also lead to a feasible schedule, provided a path exists which connects the two machine arcs.

Lemma 1. *Let G' be a directed graph representing a feasible active schedule. Furthermore, let v and w be two not necessarily critical, adjacent operations on a machine that are processed immediately after each other without intermediate machine idle time. Then, no path other than $[v,w]$ can exist from v to w in G', and the reversal of arc $[v,w]$ always produces a feasible solution.*

Proof. Assume that there is a path P leading from v to w. This path must contain at least two or more other operations, denoted by

$o_1, \ldots, o_r$ ($r \geq 2$), because it runs over a different machine. Due to precedence relations given in path $P = [v, o_1, \ldots, o_r, w]$, the processing of w can start at the earliest after the completion of o_r. Since the processing of o_r starts after the completion of v, $v \neq o_r$, and $p_{o_r} > 0$, the assumption of no intermediate machine idle time between the processing of v and w is violated. Hence, no further path can exist between v and w, and reversing arc $[v, w]$ cannot create a cycle.□

Lemma 2. *Let G' be a directed graph representing a feasible active schedule. Furthermore, let $[v_1, w_1]$ and $[v_2, w_2]$ be two not necessarily critical arcs in G' that represent two precedence relations on arbitrary machines with v_1, w_1, v_2, w_2 all distinct. Assume that there is no intermediate idle time between the processing of v_1 and w_1 as well as v_2 and w_2 on their machines, and there exists a path $P = [w_1, \ldots, v_2]$ leading from w_1 to v_2. Then, reversing both arcs simultaneously always produces a feasible solution.*

Proof. Suppose that reversing both arcs $[v_1, w_1]$ and $[v_2, w_2]$ creates a cycle $\mathcal{C}$. This cycle has to contain both reversed arcs, because otherwise, the exclusive reversal of only one arc already produces a cycle which is precluded by Lemma 1. Let $\mathcal{C} = [w_1, v_1, P_1, w_2, v_2, P_2, w_1]$ denote the cycle with P_1 and P_2 being paths connecting v_1 with w_2 and v_2 with w_1, respectively. Note that both these paths must already exist in G', i.e. before $[v_1, w_1]$ and $[v_2, w_2]$ have been reversed. Therefore, due to the existence of $P = [w_1, \ldots, v_2]$ in G', also path $[w_1, \ldots, v_2, P_2, w_1]$ must exist in G' violating the assumption that G' contains no cycle.□

It is important to mention that the perturbation considered in Lemma 2 cannot be analyzed by two separate arc reversals as it requires the two arcs to be connected by a path. From Lemma 2 it is clear that a feasibility guarantee holds for the neighborhood operator ECET. The argumentation of Lemma 2 can also be extended to a chain of more than two arcs connected by a path that are reversed in a new schedule. In this way, we also derive a feasibility guarantee for the neighborhood operator ICT.

Theorem 1. *Let G' be a directed graph representing a feasible active schedule. Then, neighborhood operator CET+2MT always produces feasible solutions.*

Proof. Without loss of generality, we suppose that exactly three arcs are reversed by the perturbation. Let $d = [x_1, x_2]$, $e = [u_1, u_2]$, and $f = [y_1, y_2]$ denote the relevant arcs in the considered perturbation (see Fig. 3), and let d', e' and f' denote the corresponding reversed arcs in the new schedule. Again, we assume that the new schedule contains a cycle $\mathcal{C}$. In the following, we analyze how this cycle could look like.

Case 1: Cycle $\mathcal{C}$ contains none of the reversed arcs d', e' and f'. Therefore, $\mathcal{C}$ must already exist in G' which contradicts the feasibility of the origin schedule.

Case 2: Cycle $\mathcal{C}$ contains exactly one of the reversed arcs. Since there is no immediate machine idle time in the processing of the adjacent operations corresponding to the three arcs, this would mean a violation of Lemma 1.

Case 3: Cycle $\mathcal{C}$ contains exactly two of the reversed arcs. Let, for instance, $\mathcal{C}$ contain d' and e' (see Fig. 13, left side). Then, the cycle must have the form $\mathcal{C} = [e', P_1, d', P_2, e']$. Since, P_1 and P_2 already exist in G' and are thus not effected by the perturbation, the path $[u_1, x_1, P_2]$ is longer than the path leading from u_1 to u_2 in G'. Hence, arc e cannot be a critical arc in G'. This argumentation is the same in case that e' and f' or d' and f' are involved in the cycle.

Case 4: Cycle $\mathcal{C}$ contains all three reversed arcs. It can have only one of the two forms $\mathcal{C} = [f', P_1, e', P_2, d', P_3, f']$ (see Fig. 13, right side), or $\mathcal{C} = [f', P_1, d', P_2, e', P_3, f']$. In either cases, paths P_1, P_2 and P_3 exist in G' since they are not effected by the perturbation. In the former case, the length of path $[u_1, x_1, P_3, y_2, \ldots, u_2]$ exceeds the length of arc $e = [u_1, u_2]$. In the latter case, the length of path $[u_1, x_1, P_2, u_2]$ exceeds the length of arc $e = [u_1, u_2]$. Anyway, in neither case arc e is a critical arc in G'.

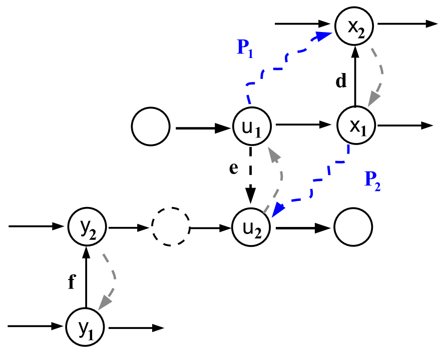
With the above contradictions, we obtain the feasibility guarantee of the new schedule generated by the neighborhood operator CET+2MT.□

The above findings on the feasibility guarantee of the neighborhood operators are summarized in Table 6. We recognize from the table that the feasibility guarantee holds only for the transpose-based operators CT, CET, ECET, ICT and CET+2MT.

5.2. Connectivity property

A neighborhood NB is called connected if any schedule can be transferred into any other schedule by a finite sequence of NB moves. With respect to optimization, a slightly weaker definition is sufficient to assess the capabilities of a neighborhood [8]. A neighborhood NB is called opt-connected if any schedule s_0 can be transferred into an optimal schedule s^* by a finite sequence of NB moves, i.e. a sequence of solutions $s_0, s_1, \ldots, s_k$ exists with $s_t \in NB(s_{t-1})$ for $t = 1, \ldots, k$ and $s_k = s^*$. For example, the neighborhood CT, which performs the reversal of one critical arc, is opt-connected for the standard JSP [23] as well as for the JSPTWT [10,12]. Since CSR includes CT, also CSR is opt-connected. Dell'Amico and Trubian [9] have shown that CEI is opt-connected for the standard JSP. Their proof can be transferred directly to the JSPTWT.

In the following, we demonstrate by a counterexample that CET, ECET, ICT, CET+2MT, CE3P, and CE4P are all not opt-connected. For this, consider the problem instance in Table 5



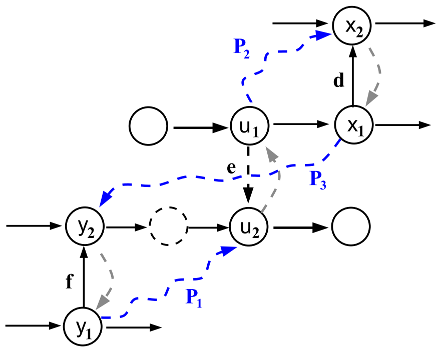


Fig. 13. Left: $d', e' \in \mathcal{C}$ (Case 3a), Right: $d', e', f' \in \mathcal{C}$ (Case 4a).

consisting of four jobs and two machines. Note that the jobs have a common due date and follow the same technological machine sequence (we actually consider a flow shop problem). An optimal schedule s^* is obtained from Johnson's rule, shown on the left side of Fig. 14. Note that there exist multiple optimal schedules to this problem instance, where job 4 is always the one processed last on machine 1. Given the schedule s_0, shown on the right side of Fig. 14, it is clear that it cannot be transferred into s^* by means of CET moves because CET merely allows reversing the two leftmost and the two rightmost operations of the critical block $[(1/1),(1/4),(1/3),(1/2)]$. Therefore, operation $(1/4)$ will never take the last position in the block meaning that job 4 cannot be processed as the last job on machine 1. This, however, is a necessary condition for schedule optimality in this problem instance.

Taking a look at the schedule perturbation schemes performed by ECET, ICT and CET+2MT, it becomes obvious that the above example disproves the opt-connectivity property for these neighborhood definitions as well. For the CE3P and CE4P neighborhoods, a similar counterexample is obtained by simply adding two further jobs with processing time $p_{1j}=4$ and $p_{2j}=2$ to the above flow shop instance and increasing the common due date respectively. Sequencing these jobs on the machines in between of job 4 and job 3 in s_0, it is clear that the optimal schedule cannot be reached from the considered solution. Note that the argument is always that job 4 cannot enter the last position (not even a position in the latter half) of machine 1.

The counterexample presented above is not applicable to the remaining neighborhood operators DOCEI, DICEI and CEI+2MT. Therefore, we classify the opt-connectivity property as "open" for these operators. The analytical results regarding the feasibility guarantee and connectivity property of neighborhoods are summarized in Table 6. It is interesting to note that only the CT neighborhood is opt-connected and provides a feasibility guarantee at the same time.

6. Computational study

In this section we compare the performance of the proposed neighborhood operators within a comprehensive computational study. First, we describe the used test suite and then we report and analyze the outcome of local search algorithms employing a single operator as well as a combination of different operators.

Table 5
Data to the example 4 × 2 instance.

Job j	w_j	r_j	Op. 1		Op. 2		
			Mch.	p_{1j}	Mch.	p_{2j}	d_j
1	1	0	1	4	2	2	16
2	1	0	1	4	2	3	16
3	1	0	1	4	2	3	16
4	1	0	1	4	2	1	16

6.1. Test suite

The computational tests are done using 53 benchmark instances from Beasley's OR-Library [5]. The test suite contains 18 instances with 10 jobs and 10 machines each (ABZ05-ABZ06, LA16-LA20, MT10 and ORB01-ORB10). The suite is supplemented by 35 instances with 10 to 30 jobs and 5 to 15 machines (LA01-LA15 and LA21-LA40), where the job-machine-ratio varies from 1:1, 3:2, 2:1, 3:1 to 4:1. Due dates and job weights are computed for all instances according to the procedure of [20]. In our study we exclusively use the due date factor $f=1.3$ which creates relatively tight due dates using the formula below:

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

The job weights are set in the instances as follows: the first 20% of the jobs receive a weight of 4, the next 60% of jobs receive a weight of 2, and the final 20% of the jobs get a weight of 1. The objective function value of the so far best known solutions to all 53 test instances is reported by Essafi et al. [10]. Note that none of these instances has been solved with a total weighted tardiness of zero so far. Less tight due date factors of $f=1.5$ and $f=1.6$, which are also considered in [10,20], are ignored in our study. In this way at least one job of a feasible schedule gets tardy which enables a reliable assessment of weak and strong performing neighborhood operators.

6.2. Local search with a single neighborhood operator

The eleven neighborhood operators proposed in Section 4 are assessed in the following way. First, we compute a sufficiently large sequence $s_i = (s_i^1, s_i^2, \ldots)$ of random start solutions for each of the 53 benchmark instances $i=1,\ldots,53$. The neighborhood operators $j=1,\ldots,11$ are incorporated in a steepest descent algorithm. Given a start solution, this algorithm computes the objective function value of all neighboring solutions with respect to the used neighborhood operator. Provided that the best neighboring solution shows a lower total weighted tardiness than the start solution, the best neighboring solution replaces the start solution and the search is continued from the new solution. Otherwise, if no improved solution is found in the neighborhood of the start solution, the steepest descent algorithm terminates and a new steepest descent algorithm is started with the next random start solution taken from the sequence s_i. The entire procedure stops as soon as one of the two alternative termination criteria is reached. With the first termination criterion, the steepest descent algorithm is repeated for neighborhood $j=1,\ldots,11$ and instance $i=1,\ldots,53$ until a fixed number of local optima is generated. With the second termination criterion, the algorithm stops after a fixed number of neighboring schedules is evaluated for every operator and every problem instance.

Using the described schedule generation procedure, we perform two experiments on our test suite. In both experiments, the same sequences s_i of start solutions are used for the problem instances by each of the neighborhood operators. In the first experiment, the performance quality of the neighborhood operators is assessed without considering how many evaluations a

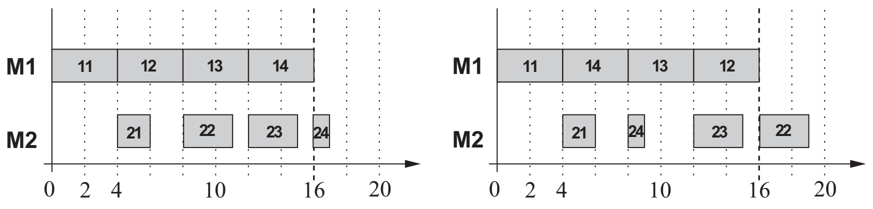


Fig. 14. Gantt-Charts of the optimal schedule s^* (left) and the initial schedule s_0 (right).

Table 6
Classification of neighborhood operators based on feasibility and connectivity.

	Opt-connected	Not opt-connected	Open
Feasibility guarantee	CT	CET	
		ECET	–
		ICT	
		CET+2MT	
No feasibility guarantee			CEI+2MT
	CEI	CE3P	DICEI
	CSR	CE4P	DOCEI

Table 7
Results of the computational study using a single neighborhood operator.

Operator	Experiment 1				Experiment 2				
	Gap	# Eval	# Imp	Rank	Gap	# LOpt	# Imp	Rank	Sec
CEI	73.69	26.85	17.57	1	71.14	6575	12.14	4	280
CET+2MT	73.98	3.07	17.13	2	60.17	27,169	14.35	1	271
CE4P	75.34	31.71	15.81	3	84.09	2331	12.63	11	332
CEI+2MT	76.70	11.42	13.06	4	69.26	12,509	9.55	3	690
CSR	78.44	75.50	16.05	5	80.90	4781	10.41	9	486
CE3P	81.83	14.40	12.29	6	78.53	6023	10.44	7	231
CET	81.84	2.68	15.10	7	67.03	28,947	13.43	2	156
ECET	83.32	4.70	14.50	8	72.45	18,657	12.40	6	138
DOCEI	96.13	3.37	4.57	9	81.18	28,266	4.42	10	351
DICEI	96.70	3.10	4.30	10	80.63	28,918	4.30	8	306
ICT	100.02	0.53	3.39	11	72.29	108,700	3.79	5	487

steepest descent walk takes. This is achieved by terminating the procedure after a fixed number of steepest descent walks (termination criterion 1). In the second experiment, the performance quality of the neighborhood operators is assessed for a certain level of computational effort. This is reached by terminating the procedure after a fixed number of schedule evaluations have been carried out (termination criterion 2). The number of evaluations is chosen with regard to the size of an instance, i.e. the number of jobs and machines involved. Note that the number of produced local optima usually differs for the competing neighborhood operators. This kind of comparison allows assessing neighborhoods under a common limitation of the computational effort.

Let $BFS(i,j)$ denote the best solutions found by neighborhood operator j for problem instance i. Furthermore, let $BKS(i)$ denote the objective function value of the best known solution for problem instance i. In order to assess the solution quality achieved by a neighborhood operator, we compute the average gap to the best known solutions:

$$\text{Gap}(j) = \frac{1}{53}\sum_{i=1}^{53}\frac{BFS(i,j)-BKS(i)}{BKS(i)}[\cdot 100\%] \quad \text{for } j = 1, \ldots, 11$$

In the experiments described below, the measure of average gap is used to assess and to compare the performance of neighborhood operators. As an alternative measure of performance, we have considered the relative improvement rate obtained by operators against an initial solution. In our computations, it turned out that both measures of performance deliver no significant differences of the outcome in terms of the resulting ranking.

Experiment 1. For every neighborhood operator and every problem instance, a total of 100 local optimal schedules is generated. Hence, over the entire test suite, a total of 5,300 local optimal schedules is produced by every neighborhood operator. With the common limit of generated local optima, we measure the absolute performances of the respective neighborhood operators. Based on the best schedule found for each problem instance, the average gap value is computed and reported in the left part of Table 7. The results indicate that the operators perform very different on the test suite. Five of the operators (CEI, CET+2MT, CE4P, CEI+2MT and CSR) show a gap between 73% and 78%. This group is followed by three operators (CE3P, CET and ECET) with a gap of approximately 82%. The remaining three operators (DOCEI, DICEI and ICT) form a group which performs with a gap above 95%. The rank achieved by the operators is also reported in the table. As one might expect, the rank correlates quite closely with the average number of improving steps (# Imp) that the operators have realized during the steepest descent walks. The coefficient of correlation is -0.86. This is also illustrated by the dominant downward slope of the curve belonging to Experiment 1 in the left diagram of Fig. 15. It means that the more improving steps an operator executes, the better the produced solution quality is. Considering the total number of schedule evaluations # Eval (in millions) made by the operators, there is no clear correlation with the rank obtained. For example, operator CE3P, which belongs to the medium performing group of operators, consumes more than four times of the evaluations needed by the second best performing operator CET+2MT. The very most schedule evaluations (75.5×10^6) are made by CSR which takes rank 5 among all operators.

Experiment 2. This experiment is motivated by the observation that the number of schedule evaluations varies strongly for the operators in Experiment 1. Therefore, in Experiment 2, we introduce a common limit on the number of schedule evaluations available for the neighborhood operators. The idea of the experiment is to measure the relative performances of the neighborhood operators with regard to the computational effort. Since the size of the instances of the test suite varies, we impose a size-dependent limit of $10 \cdot n^2 \cdot m^2$ schedule evaluations. This means that the total number of evaluations (# Eval) is set constant for every test instance in this experiment. E.g., for an instance with $n=10$ jobs and $m=10$ machines, a total of 100,000 schedules are evaluated by each of the eleven operators. A total of 12.86 million schedule evaluations are made by each operator on the entire test suite. This allows us to compare the computational effort of the operators in terms of run-time. The time needed by an operator to evaluate 12.86 million schedules on our machine (Intel-Core i7-2600 (3.4 GHz), Linux openSUSE 11.4 (x86_64), using C++) is shown in the last column (Sec) of Table 7. The observed computation times indicate that the computational effort of the operators is within a reasonable range. It differs at most by a factor of 5 as is observed between the most demanding (CEI+2MT) and modest (ECET) operators.

Like in Experiment 1, the assessment and comparison of an operator is based in this experiment on the average gap, the associated rank and the average number of improving steps. Furthermore, the total number of local optima (# LOpt) generated by the operators for the entire test suite is counted. The corresponding results are reported in the right part of Table 7. Like in Experiment 1, the operators perform very different for the test suite but the gained results also allow us to sort them into three groups. The best performing operators (CET+2MT, CET, CEI+2MT) show a gap between 60% and 69%. This group is followed by four operators (CEI, ICT, ECET and CE3P) with a gap between 71% and 78%. The group of weaker operators contains four operators (DICEI, CSR, DOCEI, CE4P) with a gap above 80%. As shown in the left diagram of Fig. 15 by the curve belonging to Experiment 2, the correlation observed between the rank and the average number of improving steps realized by the operators is much weaker than in the previous experiment. The column # LOpt indicates that the

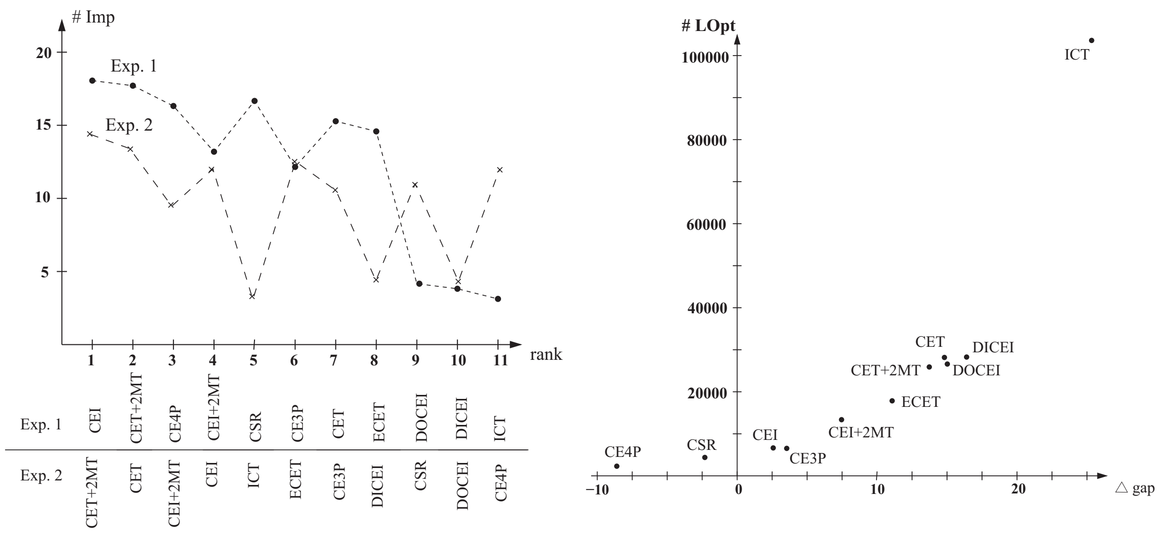


Fig. 15. Correlation between rank and # Imp (left), correlation between # LOpt and Δ gap (right).

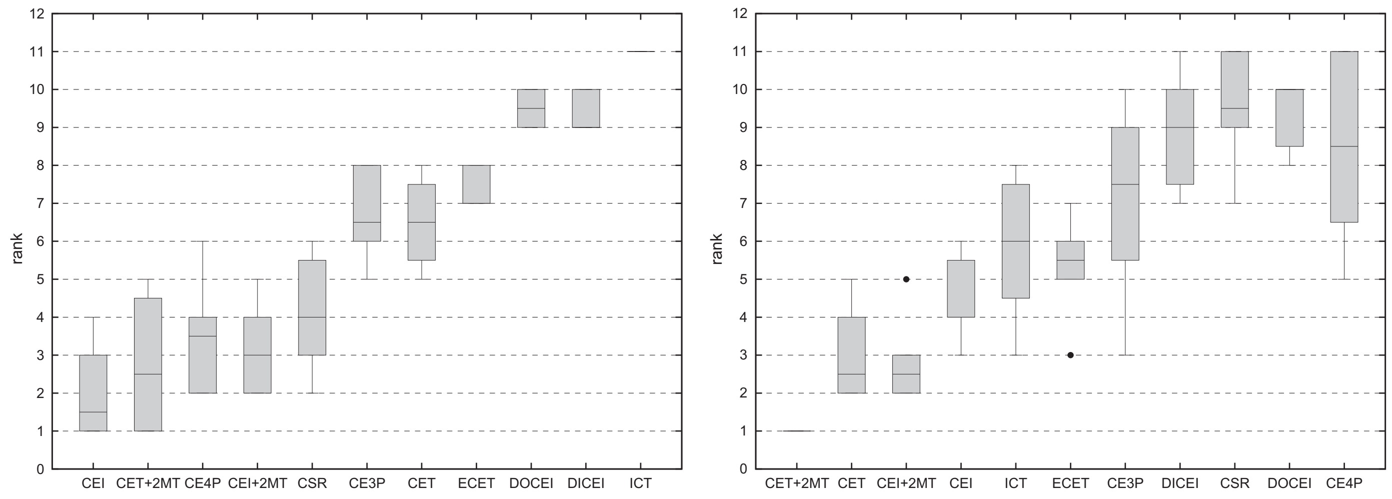


Fig. 16. Ranks of neighborhood operators according to the eight instance classes in Experiment 1 (left) and Experiment 2 (right).

operators exploit the available capacity for schedule evaluations very differently. While, for example, operator CEI produces about 6.600 local optimal schedules, operator ICT comes along with more than the tenfold. Still, both operators differ only by one rank.

Comparing the outcome of both experiments, it is striking that nearly all operators have reached a better gap in Experiment 2. With the exception of CSR and CE4P, the operators have also generated more local optima in this experiment than in Experiment 1. Remember that a fixed limit of 5,300 local optimal schedules is prescribed in Experiment 1. While CSR and CE4P have performed less steepest descent walks in Experiment 2 than in Experiment 1, the other operators have actually produced more local optimal schedules here (see column # LOpt). The larger number of restarts performed by the vast majority of operators in Experiment 2 explains the better gap which is observed in Experiment 2. We can also see from Table 7 and the right diagram of Fig. 15 that the operators benefit quite consistently from an increased number of restarts. For example, operators CET+2MT, CET, DOCEI and DICEI make about 27,000 and 29,000 restarts in Experiment 2 which respectively leads to an improvement (Δ gap) of 14% to 16% against Experiment 1.

The solution quality achieved by the operators in the two experiments is not perfectly consistent. It could be possible that the operator performance varies on the benchmark set because instances range from 10 to 30 jobs and 5 to 15 machines. This fact might create different conditions for the operators. For example, a poor performing operator like DOCEI may not be able to show its true potential when solving small 10×10 instances since critical blocks are typically very short in these instances. Therefore, to analyze the sensitivity of the operators, we reconsider the computational results of Experiments 1 and 2 on the background of instance sizes ($n \times m$) met in the test suite. Actually, with sizes of 10×5, 15×5, 20×5, 10×10, 15×10, 20×10, 30×10, and 15×15, we observe eight different instance classes in the test suite. The corresponding outcomes of the operators in terms of achieved average gaps, the number of schedule evaluations made in Experiment 1 as well as the generated local optima in Experiment 2 are shown in the Appendix. For the instance class with instance size 10×5, the average gap of the operators ranges between 18% and 50%, see Tables A1 and A3. But the performance quality worsens significantly if larger problem instances have to be solved. The poorest results are observed for the instances 15×15. The gap achieved for this instance class never falls below 100%. The ranks achieved by the operators in Experiment 1 and 2 within each of the eight instance classes are shown by box plots in Fig. 16. The bar within the boxes represents the median of the ranks obtained by an operator on the instance classes, while the boxes represent the 50% range of ranks closest to the median, known as interquartile range. The whiskers appearing at some of the boxes indicate the 1.5-times interquartile range. Note that the abscissa of the left (right) plot is arranged according to the total ranks achieved by the operators in Experiment 1 (Experiment 2). For Experiment 1, the top and the bottom of the boxes are mostly stretched over only two or three ranks. Furthermore, the deterioration of the medians correlates quite well with the total ranks of the operators. We observe that the weakest operator ICT also achieves the last rank in each of the instance classes. Turning to Experiment 2, the heights of the boxes and the distance of the whiskers vary little stronger. Here, operators behave more sensitive regarding different instance classes, but a correlation is clearly observable.

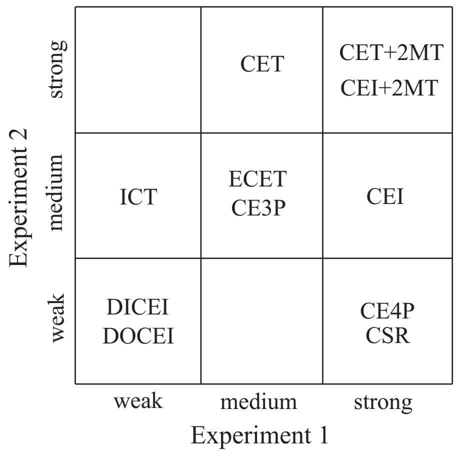


Fig. 17. Representation of the groups as portfolio.

The size $n \times m$ of the instances influences the number of schedule evaluations made in Experiment 1 as well as the number of generated local optima in Experiment 2. For example, in Experiment 1, 56% of the schedule evaluations made by CEI are carried out for instances belonging to class 30×10 (value 15.06, see Table A2). In Experiment 2, the number of schedule evaluations done by CEI are limited to $10 \cdot 30^2 \cdot 10^2 = 9 \times 10^5$ for instances of size 30×10. Therefore, in Experiment 2, only 174 local optima are generated by CEI for the entire instance class (see Table A4). As the class consists of five instances, a total of 500 local optima is generated by an operator for the class in Experiment 1. Consequently, the average gap produced by CEI in Experiment 1 (66.19, Table A1) is better than in Experiment 1 (69.13, Table A3). Contrary outcomes are e.g. observed for CEI with regard to instance classes 10×10 and 15×15. Obviously, some instances require a very large number of schedule evaluations for reaching

Table 8
Results of pairs of neighborhood operators applied jointly.

Operator 1	Operator 2	**Experiment 3**				**Experiment 4**			
		Gap	# Eval	# Imp	Rank	Gap	# LOpt	# Imp	Rank
CET+2MT	CE4P	62.64	28.45	20.45	1	70.03	2965	17.55	21
CEI	CET+2MT	63.43	19.90	22.12	2	61.65	7464	15.87	3
CEI	CEI+2MT	64.31	28.97	21.84	3	62.84	5652	15.49	7
CE4P	CEI+2MT	64.49	39.81	21.13	4	69.48	2427	17.33	19
CET+2MT	CE3P	64.65	13.82	20.23	5	62.13	6084	17.52	5
CET+2MT	CSR	65.15	54.93	20.86	6	65.14	5684	13.63	10
CEI+2MT	CE3P	65.40	21.89	20.12	7	72.04	4696	16.22	24
CEI+2MT	CSR	65.47	64.15	21.30	8	66.64	4639	13.82	13
CEI+2MT	CET	67.33	12.16	21.32	9	61.78	10,617	15.84	4
CET+2MT	CEI+2MT	67.65	11.76	20.14	10	60.43	11,213	14.52	1
CET+2MT	CET	68.85	3.70	20.06	11	60.88	19,506	17.58	2
CEI+2MT	ECET	68.93	13.48	20.79	12	63.68	9268	15.72	8
CET+2MT	ECET	69.58	5.83	19.64	13	62.46	15,548	16.79	6
CET+2MT	DOCEI	70.33	13.23	19.42	14	66.66	11,694	14.24	14
CEI+2MT	DICEI	70.56	17.47	16.70	15	65.42	8556	12.69	11
CEI+2MT	DOCEI	70.71	17.18	16.83	16	68.04	8548	12.77	17
CET+2MT	DICEI	71.27	13.40	19.14	17	67.57	11,750	14.07	16
CEI	CE3P	71.74	26.84	18.80	18	78.26	4369	14.39	34
CET+2MT	ICT	71.84	4.22	19.11	19	63.69	20,788	16.13	9
CEI	CE4P	72.55	39.28	18.23	20	77.49	2711	14.29	32
CEI	CSR	72.70	60.29	18.46	21	73.95	4830	11.77	28
CEI+2MT	ICT	72.93	11.02	16.93	22	67.53	11,912	13.05	15
CE4P	DOCEI	73.57	36.42	16.73	23	80.95	2634	14.16	41
CEI	ECET	73.76	17.62	18.09	24	69.12	8456	12.96	18
CE4P	ICT	73.94	30.04	16.31	25	80.82	2922	14.19	40
CSR	DICEI	74.20	67.05	17.22	26	76.50	4976	11.48	30
CE3P	ECET	74.32	5.11	16.28	27	73.17	6298	14.69	25
CEI	ICT	74.50	22.91	19.12	28	69.83	7813	13.21	20
CE4P	DICEI	74.56	36.39	16.50	29	81.26	2636	13.97	42
CSR	DOCEI	74.76	68.05	17.54	30	77.77	4976	11.54	33
CE4P	ECET	75.52	23.98	15.75	31	78.50	3397	14.06	36
CSR	ICT	76.74	60.12	17.08	32	79.11	6027	11.13	39
CSR	ECET	78.55	42.45	16.11	33	75.58	6753	10.83	29
CET	DOCEI	78.72	10.68	16.53	34	71.14	12,920	12.54	22
ECET	DOCEI	79.12	11.92	16.10	35	73.21	11,041	12.44	26
ECET	DICEI	79.16	11.83	15.68	36	73.30	11,072	12.27	27
CET	DICEI	79.60	10.51	16.05	37	71.54	12,962	12.37	23
CE3P	DOCEI	80.44	16.92	13.76	38	78.35	6556	10.81	35
CE3P	DICEI	80.61	16.86	13.42	39	66.33	6573	10.65	12
CE3P	ICT	81.09	11.28	13.24	40	76.74	7826	11.22	31
DOCEI	ICT	93.55	2.78	5.52	41	79.09	30,370	5.50	38
DICEI	ICT	93.82	2.55	5.18	42	78.85	31,258	5.31	37

those 100 local optima in Experiment 1, without being able to reduce the gap. In Experiment 2, perhaps nothing changes for these instances, while for other instances, much more than 100 local optima are potentially generated due to the $10 \cdot n^2 \cdot m^2$ schedule limit, effecting a significant reduction of the average gap compared to Experiment 1.

Nevertheless, the above analysis justifies dividing the set of operators into weak, medium and strong performing ones on the basis of their ranks achieved in Experiments 1 and 2. With respect to these experiments, we obtain two different groupings which are shown in Fig. 17. Merely CET+2MT and CEI+2MT perform strong in both experiments. Two operators perform on a medium level and two operators on a weak level in both experiments. The rest of the operators perform differently in the two experiments.

Nevertheless, even the best performing operators produce an average gap above 60%, meaning that the total weighted tardiness of the best found solutions deviate significantly from the total weighted tardiness of the best known solution. In other words, local search with a single operator generates a relatively weak solution quality. Unfortunately, this is not alleviated by simply increasing the run-time of the local search. For example, doubling the number of schedule evaluations in Experiment 2 yields a 7.0% reduction of the gap, and triplication yields a 8.8% reduction. Obviously, multiplying the computational effort does not pay off.

6.3. *Local search with multiple neighborhood operators*

Taking as granted that different operators possess individual capabilities, it might be profitable to combine them in a local search procedure. In this section, we search for pairs of operators which complement one another in a favorable fashion. For this purpose, the above experiments are repeated with the modification of jointly applying two neighborhood operators in the steepest descent search algorithm instead of just a single one. The neighborhood operators are used alternately until neither of them can improve the current schedule anymore. This means that the algorithm switches the two operators between two iterations, independently of whether an improving schedule has been generated or not. Hence, both operators are able to contribute to the progress made even if one of the operators produces much stronger improvements than the other one. Finally, the algorithm ends up with a schedule which is local optimal with respect to both neighborhood definitions. In the experiments we consider only combinations of operators which appear compatible. Two operators are assumed incompatible if one operator includes the other (like CSR includes CET) or if both operators belong to the same perturbation class (like CET and ICT, see Table 4). In total, we investigate 42 combinations of the eleven neighborhood operators. The results achieved by these combinations of operators are shown in Table 8. Here, the operator which has performed better in Experiment 1 is named as Operator 1. Note that Experiments 3 and 4 are designed just like Experiments 1 and 2 above and that all performance indicators are computed in the same way.

Experiment 3. The gap observed for the considered operator combinations vary in a range from 62% to 94%. While the worst operator combinations are composed from the two worst performing single operators (DICEI and ICT), the best operator combination is formed by CET+2MT and CE4P which achieve rank 2 and 3 alone. Only the second best operator combination applies the two best single operators CEI and CET+2MT. In the next two positions we observe very similar combinations with only CEI+2MT taking the role of CET+2MT. It is also worth to mention that these two operators are always involved in the top 17 operator combinations. Apparently, perturbing schedules on multiple machines (+2MT) is advantageous in combinations with all other operators considered. But applying the two operators together yields merely rank 10. We further observe that combining operators leads to a significant increase in the number of improving steps made until a local optimum is reached. The operator combination which has made the largest number of improving steps has achieved the second best gap. The best combination (CET+2MT and CE4P) improves the gap of the best single operator CEI by over 10%. The following 22 combinations also achieve gaps better than CEI alone. However, we also observe that not all tested operator combinations are really compatible. Some combinations obtain a gap which improves the gap achieved by the better of the two operators only marginally. In other cases, combinations perform even worse (like the combination of CEI and ICT).

Experiment 4. The outcome of this experiment is quite similar to the outcome of Experiment 3. The gap achieved by the operator combinations is in most cases a little better than in Experiment 3. If all operator combinations make the same number of schedule

Table 9
Performance comparison of steepest descent algorithm and simulated annealing.

Operator	Experiment 2		Experiment 5			
			SA (10 runs)		SA (100 runs)	
	Gap	Rank	Gap	Rank	Gap	Rank
CET+2MT	60.17	1	34.49	1	20.81	1
CET	67.03	2	48.91	3	30.74	3
CEI+2MT	69.26	3	53.79	5	35.74	5
CEI	71.14	4	47.32	2	30.91	4
ICT	72.29	5	107.63	11	85.87	11
ECET	72.45	6	50.04	4	30.00	2
CE3P	78.53	7	78.82	7	48.07	7
DICEI	80.63	8	90.45	10	63.68	10
CSR	80.90	9	64.68	6	42.93	6
DOCEI	81.18	10	86.55	9	61.31	9
CE4P	84.09	11	84.63	8	52.61	8

Table A1
Average gap in Experiment 1 according to the instance classes.

	All	Rk	10 × 5	15 × 5	20 × 5	10 × 10	15 × 10	20 × 10	30 × 10	15 × 15
CEI	73.69	1	28.40	45.25	53.79	79.07	83.81	86.38	66.19	132.68
CET+2MT	73.98	2	23.35	50.76	51.54	78.07	75.51	89.50	68.87	143.61
CE4P	75.34	3	25.95	50.47	54.19	78.66	87.64	87.18	68.82	141.15
CEI+2MT	76.70	4	28.26	48.36	54.45	83.74	85.82	89.10	67.40	138.20
CSR	78.44	5	29.74	50.22	53.58	85.02	85.80	91.24	68.62	146.19
CE3P	81.83	6	28.89	53.68	55.70	88.68	92.67	96.72	71.78	148.70
CET	81.84	7	34.26	56.12	62.37	86.92	86.73	95.52	73.53	146.10
ECET	83.32	8	32.81	56.09	62.95	89.49	92.31	95.72	74.72	146.41
DOCEI	96.13	9	42.42	69.00	71.21	102.10	104.84	106.12	82.08	175.78
DICEI	96.70	10	42.42	68.87	72.69	102.73	109.99	105.83	81.98	173.43
ICT	100.02	11	46.62	71.21	73.89	105.48	113.97	109.03	84.47	181.30

Table A2
# Eval in Experiment 1 according to the instance classes (in millions).

	All	10 × 5	15 × 5	20 × 5	10 × 10	15 × 10	20 × 10	30 × 10	15 × 15
CEI	26.85	0.30	1.38	4.00	0.91	1.10	3.25	15.06	0.85
CET+2MT	3.07	0.07	0.18	0.36	0.26	0.22	0.47	1.29	0.22
CE4P	31.71	0.68	1.95	4.02	3.11	2.90	6.37	9.76	2.93
CEI+2MT	11.42	0.16	0.66	1.81	0.52	0.51	1.39	5.92	0.43
CSR	75.50	0.46	3.01	11.27	1.15	1.91	6.65	49.83	1.22
CE3P	14.40	0.28	0.80	1.69	1.15	1.03	2.23	6.20	1.01
CET	2.68	0.06	0.16	0.31	0.25	0.22	0.42	1.06	0.21
ECET	4.70	0.10	0.31	0.63	0.38	0.35	0.71	1.91	0.32
DOCEI	3.37	0.06	0.22	0.50	0.25	0.21	0.46	1.46	0.21
DICEI	3.10	0.06	0.19	0.42	0.25	0.20	0.44	1.32	0.21
ICT	0.53	0.01	0.03	0.04	0.09	0.05	0.08	0.15	0.07

Table A3
Average gap in Experiment 2 according to the instance classes.

	All	Rk	10 × 5	15 × 5	20 × 5	10 × 10	15 × 10	20 × 10	30 × 10	15 × 15
CEI	71.14	4	35.88	56.11	62.47	67.89	83.81	86.93	69.13	115.30
CET+2MT	60.17	1	18.41	50.68	50.70	55.73	65.63	81.38	61.74	108.64
CE4P	84.09	11	50.48	63.76	62.11	86.74	97.09	93.94	70.57	141.15
CEI+2MT	69.26	3	28.26	55.24	57.98	69.07	80.74	85.30	67.54	110.41
CSR	80.90	9	48.00	69.36	66.37	76.95	87.55	101.65	79.65	127.92
CE3P	78.53	7	31.45	61.73	59.52	79.15	92.67	96.72	72.72	132.64
CET	67.03	2	25.28	56.12	61.41	63.11	74.37	87.76	67.25	111.10
ECET	72.45	6	31.72	56.09	62.95	70.86	84.11	88.69	70.06	119.30
DOCEI	81.18	10	40.61	69.00	71.21	76.20	87.22	98.85	77.91	141.34
DICEI	80.63	8	40.19	68.87	72.69	75.29	84.61	98.06	77.51	141.72
ICT	72.29	5	38.85	58.87	65.47	66.76	82.04	90.00	73.83	116.83

Table A4
# LOpt in Experiment 2 according to the instance classes.

	All	10 × 5	15 × 5	20 × 5	10 × 10	15 × 10	20 × 10	30 × 10	15 × 15
CEI	6575	319	127	69	3595	630	354	174	1307
CET+2MT	27,169	1344	740	526	13,503	2793	1822	1176	5265
CE4P	2331	82	37	25	1254	192	116	73	552
CEI+2MT	12,509	559	259	150	6447	1299	813	434	2548
CSR	4781	217	75	35	2821	433	204	86	910
CE3P	6023	234	135	100	2862	580	424	311	1377
CET	28,947	1464	1009	856	12,620	3000	2494	2203	5301
ECET	18,657	878	531	420	8643	1885	1495	1240	3565
DOCEI	28,266	1304	981	903	11,317	2939	2811	2941	5070
DICEI	28,918	1343	1161	1200	10,927	2958	2969	3342	5018
ICT	1,08,700	5027	4897	5593	36,390	10,820	12,539	15,797	17,637

evaluations the combination of CET+2MT with CEI+2MT performs best with a gap of about 60%. Six of the Top 10 combinations of Experiment 3 are among the Top 10 combinations of Experiment 4. However, no operator combination of Experiment 4 achieve a better gap than the best performing single operator CET+2MT in Experiment 4. Obviously, the number of operator combinations which fail in this experiment is much higher than in Experiment 3. This is explained by the observation that the weaker operator often consumes the vast majority of schedule evaluations and hinders the better operator to unfold its full potential.

To summarize, combining two adequate neighborhood operators has a significant impact on the gained absolute performance. However, the achievable quality is not yet satisfying. Moreover, the relative performance deteriorates in some cases. The best achieved gap in all experiments is reported on the single application of CET+2MT in Experiment 2. Therefore, the computational results obtained with more than two neighborhood operators are not included in this study.

6.4. Local search with a meta-heuristic

In order to test the capabilities of the different neighborhood operators when embedded in a local search based meta-heuristic, we have implemented a simple simulated annealing algorithm (SA). The algorithm starts with a random solution, randomly perturbs this solution by one of the eleven neighborhood operators, and accepts the new solution as the current solution in case it is improving. In case of deteriorations, the new solution is accepted as the current solution with a probability of $e^{-\Delta/T}$. The term Δ measures the relative deterioration of the new solution against the current solution, and T denotes the temperature. Initially, SA starts with a temperature $T_0 = 7.5$ which is reduced according to a geometric cooling scheme $T_{t+1} = T_t^{0.9}$ to almost zero ($T_{100} \approx 0$) within 100 cooling steps. The parameters are set such that a relative deterioration of $\Delta = 10\%$ is accepted after 50 cooling steps with a probability close to 0.1.

Experiment 5. For every neighborhood operator and for every problem instance of the test suite, a total of ten SA runs are executed. In each run, a number of $n^2 \cdot m^2$ schedule evaluations is performed with respect to the size of the problem instance. In other words, the temperature used by SA is cooled down by one step after $(n^2 \cdot m^2)/100$ evaluations have been carried out. In this way, the same computational effort is spent for the test suite as in Experiment 2. The outcome in this experiment is ten local optima, each one returned as the best found solution in one SA run. In a repetition of this experiment, a total of 100 SA runs are executed consuming exactly ten times the effort spent in Experiment 2. The average gap for both SA computations is shown in Table 9 and compared with the outcome of Experiment 2. Under the same computational amount as in Experiment 2 (SA, 10 runs) the average gap achieved by six of the eleven neighborhoods is improved significantly. The superior neighborhood CET+2MT which holds rank 1 in both experiments can almost halve the average gap when embedded in a meta-heuristic. Five neighborhoods, however, lead to a poorer solution when using SA. In particular ICT fails completely. While it holds rank 5 in Experiment 2, it achieves only the last rank in this experiment with an average gap worsened by 50%. Finally, the performance of the neighborhoods turns out to be fairly different under the meta-heuristic. Nevertheless, the best four neighborhoods of Experiment 2 (CET+2MT, CET, CEI+2MT, and CEI) appear in the top 5 of Experiment 5 as well. It is also interesting to aware that the meta-heuristic can take strong benefits from a longer run-time in terms of a higher number of restarts. In contrast to the steepest descent algorithm, SA continuously reduces the average gap achieved by all neighborhoods as it is demonstrated in the 100 runs computations (SA, 100 runs)). Unfortunately, this effect diminishes very quickly when the number of restarts is increased further on. A substantial further improvement requires to enhance the SA algorithm (e.g. by reheating) which is beyond the scope of the paper.

7. Conclusion

This paper provides a reformulation of five well known schedule perturbation schemes, all based on the disjunctive graph model, and develops six new ones. It further provides some new results on feasibility guarantees and connectivity properties of neighborhood definitions. To investigate the size and the improvement rate of the neighborhoods, we have conducted a thorough computational study based on a broad set of challenging benchmark instances. The study compares the quality of schedules achieved through a set of neighborhood search operators in a steepest descent walk. Based on the computational results, a group of five neighborhood operators performs efficiently in terms of the absolute performance observed for the total weighted tardiness measure. But only the operator CET+2MT is really convincing because of its excellent trade-off between computational effort and solution quality. Obtaining a better solution quality through local search can be achieved by combining the exploration strategies of at least two different neighborhood operators. The selection of operators, however, has to be done carefully. Numerous combinations of operators investigated in this study have ended up in a very disappointing schedule quality. This is because their individual principles may interfere with each other. Other combinations of operators support each other by leading to schedule quality which none of them can generate on its own. Although, there is no dominant pair of operators identified in the tests, a good choice is combining a single machine perturbation operator with a multiple machine perturbation operator. Anyway, the obtainable solution quality is limited when compared to the best known solutions. Therefore, the operators have been embedded into a local-search based meta-heuristic to analyze their behavior in a more sophisticated optimization algorithm. Using the neighborhoods within a simple simulated annealing algorithm indicates their different potential. Some of the considered neighborhoods allow extending the gained solution quality far beyond the level obtained by local search walks. Anyway, the most successful stand-alone operators turn out again to be the most promising ones under the control of a meta-heuristic.

Appendix A. Detailed results of Experiments 1 and 2

Tables A1, A2, A3, and A4.

References

[1] Adams J, Balas E, Zawack. D. The shifting bottleneck procedure for job shop scheduling. Manag Sci 1988;34(3):391–401.
[2] Anderson EJ, Glass CA, Potts CN. Machine scheduling. In: Aarts EHL, Lenstra JK, editors. Local search in combinatorial optimization. John Wiley & Sons Ltd; 1997. p. 361–414.
[3] Armentano VA, Scrich CR. Tabu search for minimizing total tardiness in a job shop. Int J Prod Econ 2000;63(2):131–40.
[4] Balas E, Vazacopoulos A. Guided local search and the shifting bottleneck for job shop scheduling. Manag Sci 1998;44(2):262–75.
[5] Beasley JE. OR-Library. ⟨http://people.brunel.ac.uk/mastjjb/jeb/info.html⟩; 2014.
[6] Bülbül K. A hybrid shifting bottleneck-tabu search heuristic for the job shop total weighted tardiness problem. Comput Oper Res 2011;38(6):967–83.
[7] Braune R, Zäpfel G, Affenzeller M. Enhancing local search algorithms for solving job shops with min-sum objectives by approximate move evaluation. J Sched 2013;16(5):495–518.
[8] Brucker P, Knust S. Complex scheduling. Berlin: Springer; 2006.
[9] Dell' Amico M, Trubian M. Applying tabu search to the job-shop scheduling problem. Ann Oper Res 1993;41(3):231–52.
[10] Essafi I, Mati Y, Dauzère-Pérès S. A genetic local search algorithm for minimizing total weighted tardiness in the job-shop scheduling problem. Comput Oper Res 2008;35(8):2599–616.
[11] Jayamohan MS, Rajendran. C. New dispatching rules for job shop scheduling: a step forward. Int J Prod Res 2000;38(3):563–86.
[12] Kreipl S. Werkstattsteuerung bei alternativen Arbeitsplänen. PhD thesis, University Passau; 1998.
[13] Kreipl S. A large step random walk for minimizing total weighted tardiness in a job shop. J Sched 2000;3(3):125–38.
[14] Kuhpfahl J, Bierwirth C. A new neighbourhood operator for the job shop scheduling problem with total weighted tardiness objective. In: Proceedings of international conference on applied mathematical optimization and modelling; 2012. p. 204–9.
[15] Mati Y, Dauzère-Pérès S, Lahlou C. A general approach for optimizing regular criteria in the job-shop scheduling problem. Eur J Oper Res 2011;212(1):33–42.
[16] Matsuo H, Suh CJ, Sullivan RS. A controlled search simulated annealing method for the general jobshop scheduling problem. Working paper 03-04-88. University Texas; 1988.
[17] Mattfeld DC. Evolutionary search and the job shop: investigations on genetic algorithms for production. PhD thesis, University of Bremen; 1996.
[18] Nowicki E, Smutnicki. C. A fast taboo search algorithm for the job shop problem. Manag Sci 1996;42(6):797–813.
[19] Pinedo ML, Singer. M. A shifting bottleneck heuristic for minimizing the total weighted tardiness in a job shop. Nav Res Logist 1999;46(1):1–17.
[20] Singer M, Pinedo ML. Computational study of branch and bound techniques for minimizing the total weighted tardiness in job shops. IIE Trans 1998;29:109–19.
[21] Taillard ED. Benchmarks for basic scheduling problems. Eur J Oper Res 1993;64(2):278–85.
[22] Van Laarhoven PJM. Theoretical and computational aspects of simulated annealing. PhD thesis, Erasmus University Rotterdam; 1988.
[23] van Laarhoven PJM, Aarts EHL, Lenstra JK. Job shop scheduling by simulated annealing. Oper Res 1992;40(1):113–25.
[24] Vepsalainen APJ. Priority rules for job shops with weighted tardiness costs. Manag Sci 1987;33(8):1035–47.
[25] Zhang R, Kuhpfahl J. Corrigendum to "A simulated annealing algorithm based on block properties for the job shop scheduling problem with total weighted tardiness objective" [Computers & Operations Research 38 (2011) 854-867]. Comput Oper Res 2013;40(11):2816.
[26] Zhang R, Wu. C. A simulated annealing algorithm based on block properties for the job shop scheduling problem with total weighted tardiness objective. Comput Oper Res 2011;38(5):854–67.