Les Cahiers du GERAD ISSN: 0711–2440

A neighborhood for complex job shop scheduling problems with regular objective

R. Bürgy

G–2016–34

May 2016

La publication de ces rapports de recherche est rendue possible grâce au soutien de HEC Montréal, Polytechnique Montréal, Université McGill, Université du Québec à Montréal, ainsi que du Fonds de recherche du Québec – Nature et technologies.

The publication of these research reports is made possible thanks to the support of HEC Montréal, Polytechnique Montréal, McGill University, Université du Québec à Montréal, as well as the Fonds de recherche du Québec – Nature et technologies.

Dépôt légal – Bibliothèque et Archives nationales du Québec, 2016
– Bibliothèque et Archives Canada, 2016

Legal deposit – Bibliothèque et Archives nationales du Québec, 2016
– Library and Archives Canada, 2016

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

A neighborhood for complex job shop scheduling problems with regular objective

Reinhard Bürgy

GERAD & Department of Mathematics and Industrial Engineering, Polytechnique Montréal, Montréal (Québec) Canada, H3C 3A7

reinhard.burgy@gerad.ca

May 2016

Les Cahiers du GERAD

G–2016–34

Abstract: Due to the limited applicability of the classical job shop scheduling problem in practice, many researchers have been addressing more complex versions of this problem by including additional process features such as time lags, setup times, and buffer limitations and pursuing objectives that are more relevant in practice than the makespan such as total flow time and total weighted tardiness. However, most solution approaches are tailored to the specific scheduling problem studied and are not applicable to more general settings. This article proposes a neighborhood that can be applied to a large class of job shop scheduling problems with regular objective. Feasible neighbors are generated by extracting a job from a given solution and reinserting it in a neighbor position. This neighbor generation extends the simple swapping of critical arcs in some sense, a mechanism widely used in the classical job shop but not applicable in more complex job shop problems. The neighborhood is used in a tabu search, and its performance is evaluated with an extensive experimental study using three well-known job shop scheduling problems, the (classical) job shop, the job shop with sequence-dependent setup times, and the blocking job shop, with the following five regular objectives: makespan, total flow time, total squared flow time, total tardiness, and total weighted tardiness. The obtained results support the validity of the approach.

Key Words: Complex job shop, general regular objective, scheduling, job insertion, neighborhood, tabu search

Acknowledgments: Part of the work of the author was done during a stay at GERAD as a postdoctoral fellow at Polytechnique Montréal. Reinhard Bürgy gratefully acknowledges the support of the Swiss National Science Foundation Grant P2FRP2_161720.

1 Introduction

The classical job shop scheduling problem has attracted the attention of a large number of researchers and has earned the reputation of being one of the most computationally stubborn combinatorial optimization problem (Applegate and Cook, 1991). This standard scheduling problem is, however, of limited applicability in practice as real-world scheduling problems possess process features that are not captured by the classical job shop model such as sequence-dependent setup times, minimum and maximum time lags between operations, limited buffer capacity, routing flexibility, and transportation tasks. Also, performance measures such as total (weighted) flow time, total squared tardiness, or other (possibly nonlinear) measures of earliness and tardiness are much more relevant in practice than the makespan criterion pursued in the classical job shop. This circumstance gave rise to active research on job shop scheduling models that are better capturing the real-world scheduling problems. Some of these models have established themselves over the years as standard problems in the literature. Among them are the job shop with total weighted tardiness objective, the job shop with sequence-dependent setup times, the no-wait job shop, and the blocking job shop.

With respect to the general scheduling research domain, Potts and Strusevich (2009) mention that many researchers were motivated during the last decade by the need to create scheduling models capturing more features that arise in practice, but conclude: "In spite of the vast body of research being produced, a large gap still remains between theory and practice". A similar statement can be found in (Pinedo, 2012, 431–435). This gap has also been widely acknowledged in scheduling problems of the job shop type. A main reason for this gap is, in our view, the fact that the vast majority of the articles addresses specific variants of the classical job shop capturing only a small number of additional process features and focusing on specific objectives. Only few articles deal with generic job shop scheduling problems. As a result, highly specialized solution approaches have been developed and job shop scheduling research has become more fragmented. Bülbül and Kaminsky (2013) are of a similar opinion, also pointing out that as a negative consequence, job shop scheduling software is highly specialized and customized.

Among the few contributions addressing more general job shop scheduling problems are the following. Mati et al. (2011) considered the job shop problem with general regular objective. This class of objectives comprises all functions that are monotone non-decreasing in the completion times. They modeled the problem in a disjunctive graph and developed a local search where neighbors are built by swapping a critical arc. As future research, they proposed to investigate more complex job shop scheduling problems with general regular objective. Bülbül and Kaminsky (2013) developed a decomposition heuristic for the job shop problem applicable to all linear functions of the start (and end) times of the operations. This class of objectives allows, for example, to consider intermediate holding costs, which is a relevant performance measure in practice. They modeled the problem in a disjunctive graph and developed a heuristic that is similar to the shifting bottleneck approach. An interesting and novel component of their approach is the integration of dual values from the timing problems (of partial schedules) into the single machine subproblems. Grimes and Hebrard (2015) recently considered a generic disjunctive machine scheduling problem, in which various processing features and objectives can be modeled. They propose a constraint programming approach combining a number of generic search techniques (restarts, adaptive heuristics, and solution-guided branching). While the basic approach is generic, some specific knowledge is incorporated to improve the effectiveness of their approach in certain problem types. Their numerical tests on standard job shop problems show that the approach works well in small and medium-sized instances, but scalability may be an issue. In addition, they observe that good initial upper bounds are improving the performance of the approach.

In view of the excellent (hybrid) local search approaches for specific job shop scheduling problems in the current state of the art (see e.g., Nowicki and Smutnicki, 2005; Zhang et al., 2008; González et al., 2012a), it seems at first glance surprising that there are almost no local search approaches proposed in a more general job shop scheduling setting. While the applied meta-heuristics are always of a generic nature, a main component of the local search, the neighborhood, is in most cases only applicable in a rather narrow setting. A major problem actually is the generation of *feasible* neighbors. Indeed, the simple swap of a critical arc generates a feasible neighbor with certainty in the (classical) job shop but might lead to infeasible neighbors in other variants. For the job shop with sequence-dependent setup times, necessary conditions for feasibility of a swap have been established (see e.g., Vela et al., 2010), and a critical arc is only swapped if such a condition is fulfilled. However, in more complex job shop problems, such as the blocking job shop, swapping a single arc mostly leads to an infeasible solution. Hence, another approach is needed for these complex problems. In this paper, we develop a neighborhood that is applicable to a large class of job shop problems and any regular objective. The neighborhood is in some sense extending the idea of swapping a critical arc and is based on a neighbor generation scheme that extracts a job from a given solution and reinserts it in a feasible neighbor position. The neighbor generation scheme relies on the job insertion theory developed by Gröflin and Klinkert (2007), and was already successfully applied in various job shop scheduling problems with makespan objective, namely in the (flexible) blocking job shop problem (Gröflin and Klinkert, 2009; Gröflin et al., 2011), in the blocking job shop with rail-bound transportation (Bürgy and Gröflin, 2016), and in a more general job shop model (Bürgy, 2014). This article casts the neighbor generation scheme in a unifying framework and extends its use to any regular objective. Note that we abstained from dealing with machine flexibility in this work and refer to (Gröflin et al., 2011; Bürgy, 2014) for a possible inclusion of this feature.

The remainder of the article is organized as follows. The next section proposes a generic job shop scheduling problem called the complex job shop with regular objective (CJS-R). Section 3 introduces the job insertion problem, which is an important subproblem of the CJS-R problem. The feasible solutions of this subproblem are characterized as the stable sets (with a prescribed cardinality) of a conflict graph, and structural properties of this conflict graph are established. Based on these results, Section 4 provides a generic scheme to derive feasible neighbor solutions and a specific neighborhood is proposed. In order to validate this neighborhood, Section 5 casts it in a tabu search, and its performance is evaluated with an extensive experimental study on well-known job shop scheduling problems. Some proofs and detailed numerical results are provided in the Appendix.

The introduction is concluded with some notation and terminology. Unless otherwise stated, the graphs will be directed and simple, and the following standard notation will be used. In a graph $G = (V,E)$, an arc $e = (v,w) \in E$ has a *tail* $t(e) = v$ and a *head* $h(e) = w$. For any set $V' \subseteq V$, the set of its ingoing arcs is $\delta^-(V') = \{e \in E : t(e) \in V \setminus V'$ and $h(e) \in V'\}$, the set of its outgoing arcs is $\delta^+(V') = \{e \in E : t(e) \in V'$ and $h(e) \in V \setminus V'\}$, the set of arcs in its cut is $\delta(V') = \delta^-(V') \cup \delta^+(V')$, and the set of arcs with both ends in V' is $\gamma(V') = \{e \in E : t(e) \in V'$ and $h(e) \in V'\}$. If an arc length vector $c \in \mathbb{R}^E$ is given, G will be denoted by the triplet $G = (V,E,c)$. In $G = (V,E,c)$, a cycle is called *positive* if its length is positive.

2 The complex job shop scheduling problem with regular objective

We first propose a generic job shop scheduling problem called the complex job shop with regular objective (CJS-R). The CJS-R problem is based on the scheduling model presented in (Gröflin and Klinkert, 2007), which is adapted to incorporate jobs and regular objectives. The CJS-R problem is then formulated as a combinatorial problem in a disjunctive graph. Finally, two standard job shop scheduling problems, the job shop problem with sequence dependent setup times and regular objective and the blocking job shop with transfer times, setup times, and regular objective, are specified as CJS-R problems.

2.1 Notation, data, and a problem formulation

Let V be a finite set of events (e.g. start or end of operations) and $\sigma \in V$ a dummy start event, $V^- = V \setminus \{\sigma\}$, and 2^{V^-} is the power set of V^-. Let $\mathscr{J} \subseteq 2^{V^-}$ be a set of jobs such that $\mathscr{J}$ forms a partition of V^-, i.e., a job $J \in \mathscr{J}$ consists of a set of events and each event $v \in V^-$ belongs to exactly one job. For each job $J \in \mathscr{J}$ denote by the set $V_J \subseteq V$ all events that belong to J. For ease of notation, introduce function $J : V \setminus \{\sigma\} \rightarrow \mathscr{J}$ mapping any event to its job, i.e., if event v belongs to job K then $J(v) = K$.

Furthermore, let $A \subseteq V \times V^-$ and $E \subseteq V^- \times V^-$ be two disjoint sets of precedence constraints and $d \in \mathbb{R}^{A \cup E}$ a weight function. A precedence constraint $(v,w) \in A \cup E$ with weight d_{vw} states that event v must occur at least d_{vw} time units before event w. Elements of set A and E are called *conjunctive* and *disjunctive precedence constraints*, respectively. Each conjunctive precedence constraint must be fulfilled in any feasible solution while a disjunctive precedence constraints must hold if some other disjunctive constraint in E is violated. To formalize this disjunctive structure, define a family of disjunctive sets $\mathscr{E} \subseteq 2^E$ where $\mathscr{E}$ is a partition of E into pairs. Then, for each (unordered) pair $\{(v,w),(v',w')\} \in \mathscr{E}$, either precedence constraint (v,w) or (v',w') must be fulfilled in any feasible solution. Note that a general disjunctive pair is sometimes denoted as $\{e,\bar{e}\}$ and $\bar{e}$ is called the mate of e.

Let $\alpha = (\alpha_v \in \mathbb{R} : v \in V)$ be a vector of the times α_v at which events $v \in V$ occur. Vector α specifies a *schedule*. The objective function $f : \mathbb{R}^V \rightarrow \mathbb{R}$ considered here satisfies for all vectors $\alpha, \alpha' \in \mathbb{R}^V$:

$$\alpha \leq \alpha' \text{ implies } f(\alpha) \leq f(\alpha'). \tag{1}$$

Objective functions satisfying (1) are called *regular* (see e.g., Pinedo, 2012, p. 19).

Assumption 1 *The following is assumed.*

(i) For each event $v \in V^-$, there exists $(\sigma, v) \in A$ with weight $d_{\sigma v} \geq 0$ and $(v,\sigma) \notin A$.

(ii) The solution space $\{\alpha \in \mathbb{R}^V : \alpha_\sigma = 0,\ \alpha_w - \alpha_v \geq d_{vw}$ for all $(v,w) \in A\} \neq \emptyset$.

(iii) For all conjunctive precedence constraints $(v,w) \in A \setminus \delta(\{\sigma\})$: $J(v) = J(w)$.

(iv) For all pairs of disjunctive precedence constraints $\{(v,w),(v',w')\} \in \mathscr{E}$: $J(v) = J(w')$, $J(w) = J(v')$, $J(v) \neq J(w)$, and the solution space $\{\alpha \in \mathbb{R}^V : \alpha_\sigma = 0,\ \alpha_q - \alpha_p \geq d_{pq}$ for all $(p,q) \in A \cup \{(v,w),(v',w')\}\} = \emptyset$.

Assumption (i) states that the dummy start event occurs not later than each other event, (ii) ensures that there exists a schedule that satisfies all conjunctive precedence constraints. Assumption (iii) says that any conjunctive precedence constraint, which does not involve the dummy start, considers two events of the same job. Hence, the conjunctive precedence constraints allow to define a job structure. Assumption (iv) states that a pair of disjunctive precedence constraints considers events that belong to exactly two distinct jobs, and that it is not possible to simultaneously satisfy both precedence constraints of such a pair.

The CJS-R problem $(V, \mathscr{J}, A, E, \mathscr{E}, d, f)$ can then be formulated as the following disjunctive programming problem:

$$\text{minimize } f(\alpha), \alpha \in \mathbb{R}^V \tag{2}$$

subject to:

$$\alpha_w - \alpha_v \geq d_{vw} \text{ for all } (v,w) \in A, \tag{3}$$

$$\alpha_w - \alpha_v \geq d_{vw} \text{ or}$$

$$\alpha_{w'} - \alpha_{v'} \geq d_{v'w'} \text{ for all } \{(v,w),(v',w')\} \in \mathscr{E}, \tag{4}$$

$$\alpha_\sigma = 0. \tag{5}$$

Any feasible solution $\alpha \in \mathbb{R}^V$ of the CJS-R specifies times α_v for all events $v \in V$ so that all conjunctive precedence constraints are satisfied (3), at least one disjunctive precedence constraint of each disjunctive pair is satisfied (4), and the start event occurs at time 0 (5). The objective is to minimize the value $f(\alpha)$ (2).

Some remarks are in order. While there is no mention of resources in the CJS-R scheduling problem, the most common applications are scheduling problems with machines that have unit capacities. Then, the capacity constraints can be easily captured by the disjunctive constraints (4). Also, there is no prescribed job structure. In many cases, a job is just a sequence of operations and for each operation its start event is considered. But the CJS-R problem allows also to specify other type of structures such as assembly or disassembly jobs.

2.2 A formulation in a disjunctive graph

The CJS-R problem $(V, \mathscr{J}, A, E, \mathscr{E}, d, f)$ is now formulated in an associated disjunctive graph $G = (V, A, E, \mathscr{E}, d)$ that is defined as follows. Each event $v \in V$ is represented by a node and we identify a node with the event it represents. Each precedence constraint $(v,w) \in A \cup E$ is represented by an arc with the corresponding length d_{vw}. Consequently, A is the set of conjunctive arcs, E is the set of disjunctive arcs, $\mathscr{E}$ is the family of disjunctive arcs pairs, and d is the arc length vector. Note that by Assumption 1 (ii), the conjunctive part (V, A, d) of G contains no positive cycle.

In order to capture solutions in graph G, we introduce selections of disjunctive arcs.

Definition 1 *Any subset of disjunctive arcs $S \subseteq E$ is called a selection. A selection S is positive acyclic if its associated graph $(V, A \cup S, d)$ contains no positive cycle, and is positive cyclic otherwise. A selection S is complete if $S \cap D \neq \emptyset$ for all $D \in \mathscr{E}$.*

Given a complete selection $S \subseteq E$, we have the following *timing problem* at hand. The space of the feasible times of the events is

$$\Omega(S) = \{\alpha \in \mathbb{R}^V : \alpha_\sigma = 0,\ \alpha_w - \alpha_v \geq d_{vw} \text{ for all } (v,w) \in A \cup S\}.$$

Clearly, $\Omega(S) \neq \emptyset$ if and only if S is positive acyclic. Consequently, a selection S is called *feasible* if S is complete and positive acyclic. Observe that in a feasible selection, exactly one disjunctive arc is picked for each pair of arcs. Indeed, for all $D \in \mathscr{E}$, graph $(V, A \cup D, d)$ is positive cyclic by Assumption 1 (iv), hence $|D \cap S| \leq 1$ if S is positive acyclic, therefore $|D \cap S| = 1$ if S is feasible.

For any feasible selection S, *earliest times* $\alpha(S) = (\alpha_v(S) : v \in V)$ can be efficiently calculated by determining for each node $v \in V$ the length $\alpha_v(S)$ of a longest path from σ to v in graph $(V, A \cup S, d)$. Then, $\alpha(S) \leq \alpha'$ for all $\alpha' \in \Omega(S)$. As the objective function f is regular,

$$f(\alpha(S)) = \min\{f(\alpha) : \alpha \in \Omega(S)\},$$

i.e., the earliest time schedule $\alpha(S)$ is an optimal solution in $\Omega(S)$.

It is then easy to see that the CJS-R problem $(V, \mathscr{J}, A, E, \mathscr{E}, d, f)$ can be formulated as the following combinatorial problem in disjunctive graph G: "Among all feasible selections, find a selection S minimizing $f(\alpha(S))$."

The following two comments are in order. First, in the sequel of the paper, CJS-R problems are specified by directly constructing their disjunctive graph $G = (V, A, E, \mathscr{E}, d)$. Second, an arc (σ, v) should be present for each $v \in V$ in order to satisfy Assumption 1 (i). However, (σ, v) may be omitted if there exists a non-negative length path in the conjunctive part (V, A, d) of G from σ to v, ensuring that v is executed not earlier than σ.

2.3 Examples

A large class of job shop problems can be formulated as CJS-R problems. We give two specific examples, which will be used in the sequel of the paper.

2.3.1 The job shop with sequence-dependent setup times and regular objective

The job shop with sequence-dependent setup times and regular objective (JSS) is a version of the classical job shop including sequence-dependent setup times and allowing all regular objectives. The problem can be formally described as follows. Given a set of operations I and a set of machines M, each operation $i \in I$ needs a specific machine, say $m_i \in M$, for its non-preemptive execution of duration $p_i > 0$. The set of operations is structured into jobs: given is a set of jobs $\mathscr{J}' \subseteq 2^I$ such that $\mathscr{J}'$ is a partition of I. For each job $J \in \mathscr{J}'$, its set of operations $\{i : i \in J\}$ is ordered $(J_1, J_2, \ldots, J_{|J|})$ specifying in which sequence the operations of J must be executed, hereby J_r denotes the r-th operation of job J. Two operations i, j of some job J are said to be consecutive if $i = J_r$ and $j = J_{r+1}$ for some $r, 1 \leq r < |J|$.

We allow for sequence-dependent setup times. If two operations i and j are using the same machine m and j follows i *directly* on machine m, then a setup of duration $s_{ij} \geq 0$ occurs between the completion of i and the start of j. We assume that the setup durations satisfy the so-called weak triangle inequality (see e.g., Brucker and Knust, 2011, p. 11), i.e., for any distinct operations i, j, k using a same machine $s_{ij} + p_j + s_{jk} \geq s_{ik}$ must hold. (Otherwise setup times between non-adjacent operations on a machine might become active in the disjunctive programming formulation.)

We also allow to specify a release time $s_J^{\mathrm{r}} \geq 0$ and a due date $s_J^{\mathrm{d}} \geq 0$ for each job $J \in \mathscr{J}'$. While the release times must be satisfied in any feasible solution, violations of the due date are allowed but can be penalized in the objective function.

The problem consists in finding starting times for all operations $i \in I$ so that each machine processes at most one operation at any time and some regular objective is minimized. In the three field notation introduced by Graham et al. (1979), the JSS is specified as $J|r_j, s_{ij}|\mathrm{reg}$.

The JSS can be described as a CJS-R problem in disjunctive graph $G = (V, A, E, \mathscr{E}, d)$ as follows (see also the example below which is illustrated in Figure 1). For each operation $i \in I$, introduce a node $v_i \in V$ indicating the start of operation i. For each job $J' \in \mathscr{J}'$, introduce a job J in $\mathscr{J}$ containing all nodes of the operations belonging to J'.

The set of conjunctive arcs A is built as follows: a) For each job $J \in \mathscr{J}'$, add an arc (σ, v_{J_1}) with length s_J^{r} reflecting the release time of J, b) for each job $J \in \mathscr{J}'$ and each pair of consecutive operations $i, j \in J$, add an arc (v_i, v_j) with length p_i reflecting the processing order within a job, and c) for each job $J \in \mathscr{J}'$ and each pair of operations

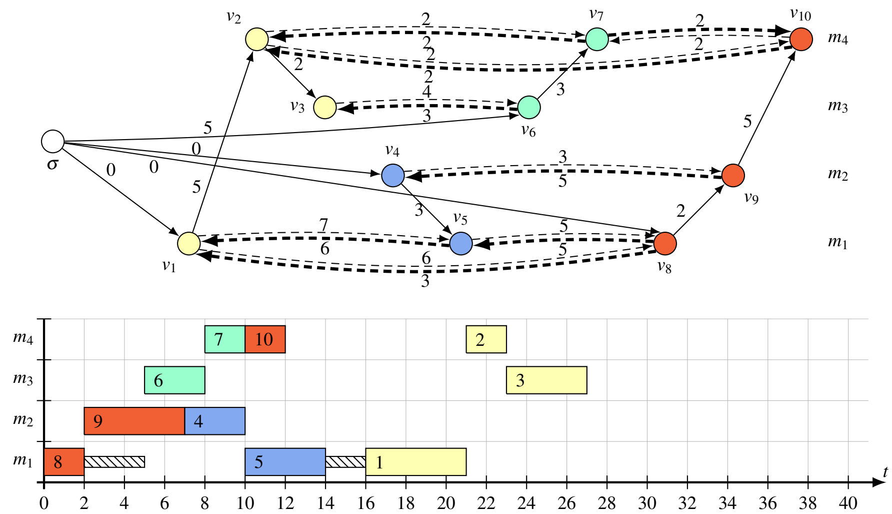

Figure 1: A JSS instance with four jobs and four machines. (top) Its associated disjunctive graph G. Conjunctive and disjunctive arcs are depicted by solid and dashed arcs, respectively. (bottom) A Gantt chart of the earliest time schedule $\alpha(S)$ of selection S consisting of the bold, dotted disjunctive arcs of G. The objective values of the five introduced objectives are $f^{\mathrm{m}}(\alpha(S)) = 27$, $f^{\mathrm{f}}(\alpha(S)) = 63$, $f^{\mathrm{sf}}(\alpha(S)) = 1169$, $f^{\mathrm{t}}(\alpha(S)) = 7$, and $f^{\mathrm{wt}}(\alpha(S)) = 21$. In the Gantt chart, each operation is depicted by one bar, and the narrow, hatched bars display setups.

$i = J_r, j = J_{r'}$, with $r < r'$ and $m_i = m_j$, add an arc (v_i, v_j) with length $p_i + s_{ij}$ reflecting the setup between operations of a same job.

The set of disjunctive arcs E is built by adding for each pair of operations $i, j \in I$ from distinct jobs using a same machine an arc (v_i, v_j) with length $p_i + s_{ij}$ and an arc (v_j, v_i) with length $p_j + s_{ji}$. The pairs of disjunctive arcs $\{(v_i, v_j), (v_j, v_i)\}$ form the disjunctive sets of $\mathscr{E}$.

For ease of notation, let $\alpha_J = \alpha_{J_{|J|}} + p_{J_{|J|}}$ be the time at which job $J \in \mathscr{J}$ is finished. The following five objective functions are considered in the sequel of the paper: makespan $f^{\mathrm{m}}(\alpha) = \max_{J\in\mathscr{J}} \alpha_J$, total flow time $f^{\mathrm{f}}(\alpha) = \sum_{J\in\mathscr{J}} \alpha_J$, total squared flow time $f^{\mathrm{sf}}(\alpha) = \sum_{J\in\mathscr{J}} \alpha_J^2$, total tardiness $f^{\mathrm{t}}(\alpha) = \sum_{J\in\mathscr{J}} \max\{0, \alpha_J - s_J^{\mathrm{d}}\}$, and total weighted tardiness $f^{\mathrm{wt}}(\alpha) = \sum_{J\in\mathscr{J}} w_J \cdot \max\{0, \alpha_J - s_J^{\mathrm{d}}\}$ where w_J is a given non-negative weight of job J. It can be easily verified that these objectives are regular.

A small example of a JSS instance is illustrated in Figure 1. It consists of ten operations $I = \{1, \ldots, 10\}$ and four jobs $J = (1,2,3)$, $K = (4,5)$, $L = (6,7)$, and $N = (8,9,10)$. The processing data can be directly read in the solution depicted in the Gantt chart of Figure 1. For example, operation 2 is executed on machine m_4 and has duration 2. The setup times between the operations on machine m_1 are $s_{1,5} = 2$, $s_{1,8} = 1$, $s_{5,1} = 2$, $s_{5,8} = 1$, $s_{8,1} = 1$, $s_{8,5} = 3$, and all other setup times are 0. The release time of job L is 5, all other release times are 0, the due date is 20, 30, 20, and 20 for job J, K, L, and N, respectively, and the weight of job J is 3 and all other job weights are 1.

2.3.2 The blocking job shop with transfer times, setup times, and regular objective

The blocking job shop problem with transfer times, setup times, and regular objective (BJS) is a version of the JSS introduced above characterized by the absence of buffer capacity. This additional feature implies that after completing an operation, a job waits on its current machine, thus blocking it, until it is transferred to its next machine. While transferring a job, both involved machines are simultaneously occupied. Each operation of a job is then best described by four steps: (i) a take-over step, where the job is taken over from its previous machine, (ii) a processing step, (iii) a possible waiting step on its current machine, (iv) a hand-over step where the job is transferred to its next machine. For the first operation of a job, the take-over step is more appropriately called loading step, and similarly for the last operation of a job, the hand-over step is called unloading step. We refer to (Gröflin and Klinkert, 2009; Gröflin et al., 2011) for a more detailed description of the BJS. In the three field notation, the BJS is specified as $J|\text{blocking}, r_j, s_{ij}|\text{reg}$.

In order to formally define the BJS problem, the following notation and data is used in addition to those given for the JSS in the previous section. Let t_{ij} be the time needed to transfer the job of operation i from machine m_i to its next machine m_j where i and j are consecutive operations in a job. Also, let t^{l}_J and t^{u}_J be the time needed to load job $J \in \mathscr{J}'$ on its first machine and unload it from its last machine, respectively. Denote by $I^{\mathrm{first}} \subseteq I$ and $I^{\mathrm{last}} \subseteq I$ the subset of all first and last operations of jobs, respectively.

The BJS can then be described as a CJS-R problem in graph $G = (V, A, E, \mathscr{E}, d)$ as follows (see also the example below which is illustrated in Figure 2). For each operation $i \in I$, add four nodes to V: v^1_i (start of take-over), v^2_i (end of take-over), v^3_i (start of hand-over), and v^4_i (end of hand-over). Note that the end of the take-over step is also the start of the processing step. For each job $J' \in \mathscr{J}'$, introduce a job J in $\mathscr{J}$ containing all nodes of the operations belonging to J'.

The set of conjunctive arcs A is built as follows:

1. For each job $J \in \mathscr{J}'$, add an arc $(\sigma, v^1_{J_1})$ with length s^{r}_J reflecting the release time of job J.
2. For each operation $i \in I$, add three arcs: a take-over arc (v^1_i, v^2_i) with length $t^{\mathrm{l}}_{J(i)}$ if $i \in I^{\mathrm{first}}$ and 0 otherwise, a processing arc (v^2_i, v^3_i) with length p_i, and a hand-over arc (v^3_i, v^4_i) with length $t^{\mathrm{u}}_{J(i)}$ if $i \in I^{\mathrm{last}}$ and 0 otherwise.
3. For any two consecutive operations i, j of some job $J \in \mathscr{J}'$, add two pairs of arcs $(v^3_i, v^1_j), (v^1_j, v^3_i)$ and $(v^4_i, v^2_j), (v^2_j, v^4_i)$ of length 0 synchronizing the start and end of the hand-over step of i with the start and end of the take-over step of j, and add an arc (v^3_i, v^2_j) of length t_{ij} reflecting the transfer of the job.
4. For each job $J \in \mathscr{J}'$ and each pair of operations $i = J_r, j = J_{r'}$ with $r < r'$ and $m_i = m_j$, add an arc (v^4_i, v^1_j) with length s_{ij} reflecting the setup between operations of a same job.

The set of disjunctive arcs E is built by adding for each pair of operations $i, j \in I$ from distinct jobs using a same machine an arc (v^4_i, v^1_j) with length s_{ij} and an arc (v^4_j, v^1_i) with length s_{ji}. The pairs of disjunctive arcs $\{(v^4_i, v^1_j), (v^4_j, v^1_i)\}$ form the disjunctive sets of $\mathscr{E}$.

We consider the same five regular objective functions as in the JSS. Note that the time α_J at which job J is finished is $\alpha_J = \alpha_{v^4_i}$ with $i = J_{|J|}$.

As a small example, we reconsider the JSS example introduced in the previous section as BJS instance, and let the duration of all transfer (take-over and hand-over), loading, and unloading steps be 1. Figure 2 depicts the disjunctive graph G of this example and a Gantt chart of a feasible schedule.

Note that the given formulation of the BJS is not the most compact form. It is for instance easy to see that nodes occurring at the same time can be represented by a single node. The given form is however more readable than, for example, the so-called alternative graph formulation presented in (Mascis and Pacciarelli, 2002).

3 A special subproblem: the job insertion problem

We first describe and formulate the job insertion problem and then present some structural properties. These are based on results of Gröflin and Klinkert (2007) and specialized to the problem under study. In consequence, proofs that are similar as in the referenced work are given in the appendix.

3.1 The job insertion problem

Informally, job insertion can be thought of as the following problem. Given a job and a feasible selection of all other jobs, insert the job in such a way that the resulting schedule is feasible, and if an optimal insertion is sought, find a feasible insertion with minimum objective value.

Specifically, given a CJS-R problem $(V, \mathscr{J}, A, E, \mathscr{E}, d, f)$ and a job $J \in \mathscr{J}$, a feasible selection of all other jobs is specified in disjunctive graph $G = (V, A, E, \mathscr{E}, d)$ by a selection R that is positive acyclic and "complete", i.e., $D \cap R \neq \emptyset$ holds for each pair $D = \{(v, w), (v', w')\} \in \mathscr{E}$ with $\{v, w\} \cap V_J = \emptyset$. We are interested in finding a feasible selection $S = T \cup R$, where T is more appropriately called a feasible *insertion* of job J.

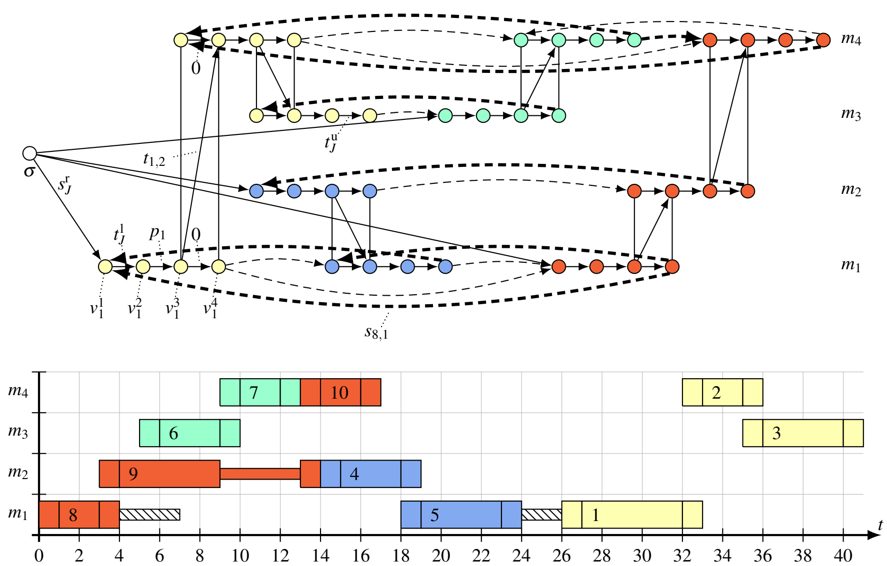

Figure 2: The BJS example. (top) Its associated disjunctive graph G. (bottom) A Gantt chart of the earliest time schedule $\alpha(S)$ of selection S consisting of the bold, dotted disjunctive arcs of G. Its objective values are: $f^{\mathrm{m}}(\alpha(S)) = 41$, $f^{\mathrm{f}}(\alpha(S)) = 95$, $f^{\mathrm{sf}}(\alpha(S)) = 2715$, $f^{\mathrm{t}}(\alpha(S)) = 21$, and $f^{\mathrm{wt}}(\alpha(S)) = 63$. In the Gantt chart, the take-over, processing, and hand-over steps are depicted by thick bars. The narrow bars stand for the setups (hatched) and the waiting steps (solid).

We study the job insertion problem in the *insertion graph* $G^J = (V, A^J, E^J, \mathscr{E}^J, d)$ of job J derived from G as follows. Set A^J of conjunctive arcs is obtained by adding to A the set R, and set E^J is obtained by deleting from E all disjunctive arcs that are not incident to job J. Formally, $A^J = A \cup R$, $E^J = E \cap \delta(V_J)$, and $\mathscr{E}^J = \{D \in \mathscr{E} : D \subseteq E^J\}$. Note that by Assumption 1 (iv), if $e \in E^J$ then also $\bar{e} \in E^J$. Clearly, $T \subseteq E^J$ is a (positive acyclic, complete, feasible) insertion in G^J if and only if $S = T \cup R$ is a (positive acyclic, complete, feasible) selection in G.

3.2 Structural properties of job insertion

Given is a job insertion problem of job J in its insertion graph $G^J = (V, A^J, E^J, \mathscr{E}^J, d)$. We examine cycles in graph $(V, A^J \cup E^J, d)$. The fictive node σ is never part of a cycle as $(V, A^J \cup E, d)$ contains no arcs entering σ. By Assumption 1 (iii), all conjunctive arcs incident to job J are of the type (σ, v), hence any cycle Z in $(V, A^J \cup E^J, d)$ enters job J through disjunctive arcs in $\delta^-(V_J)$ and leaves J through disjunctive arcs in $\delta^+(V_J)$, and the number of times Z enters and leaves J is the same, i.e., $|Z \cap \delta^-(V_J)| = |Z \cap \delta^+(V_J)| = k$ for some $k \geq 1$. Number k can be seen as the number of times Z visits job J.

Considering any positive cycle Z in $(V, A^J \cup E^J, d)$, it is easy to see that Z must visit J at least once. Indeed, the conjunctive part (V, A^J, d) of G^J contains no positive cycles since $A^J = A \cup R$ and R is a positive acyclic selection in G. We now capture all positive cycles visiting job J exactly once in a so-called conflict graph.

Definition 2 *The conflict graph of job insertion graph $G^J = (V, A^J, E^J, \mathscr{E}^J, d)$ is the undirected graph $H = (E^J, U)$ where for any distinct disjunctive arcs $e, f \in E^J$, edge $\{e, f\} \in U$ if insertion $\{e, f\}$ is positive cyclic.*

Based on the previous discussion and by Assumption 1 (iv), it is easy to see that the conflict graph H has the following structural properties.

Proposition 1 *Let $G^J = (V, A^J, E^J, \mathscr{E}^J, d)$ be a job insertion graph and $H = (E^J, U)$ its associated conflict graph.*

(i) $\{e, \bar{e}\} \in H$ for each disjunctive arc pair $\{e, \bar{e}\} \in \mathscr{E}^J$.
(ii) The conflict graph is bipartite with partitions $E^{J-} = E^J \cap \delta^-(V_J)$ and $E^{J+} = E^J \cap \delta^+(V_J)$.

Figure 3 depicts the job insertion graph and its associated conflict graph for the JSS and BJS examples.

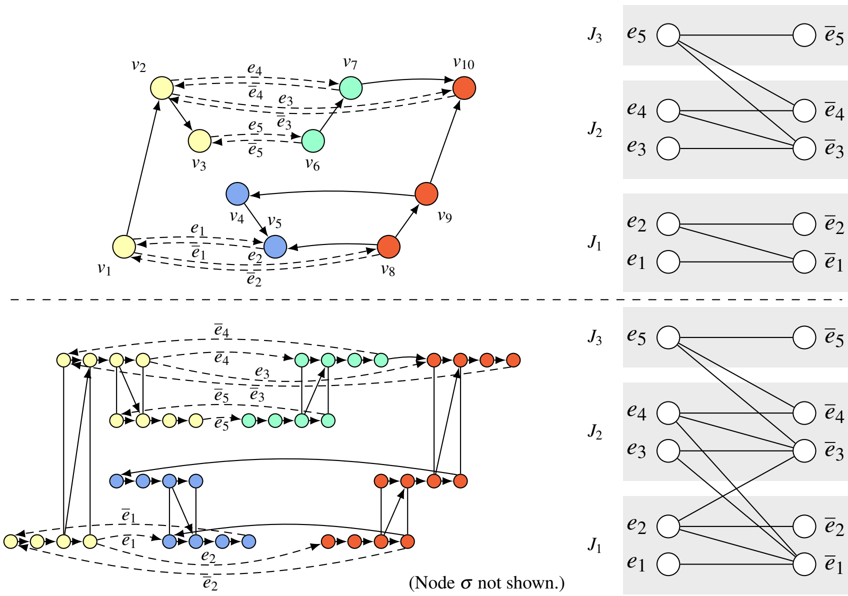

Figure 3: The job insertion graph and its associated conflict graph for the JSS example (above) and the BJS example (below). In both cases, the job insertion problem of job J (consisting of the operations 1, 2, and 3) is considered starting with the feasible selections of all other jobs given in Figure 1 and 2.

Proposition 2 *Let $G^J = (V, A^J, E^J, \mathscr{E}^J, d)$ be a job insertion graph and $H = (E^J, U)$ its associated conflict graph.*

(i) Any feasible insertion in G^J corresponds to a stable set in conflict graph H of size $|E^J|/2$.

(ii) Any stable set in H of size $|E^J|/2$ corresponds to a complete insertion.

Proof. See appendix. □

Unfortunately, an insertion T that corresponds to a stable sets of size $|E^J|/2$ can be positive cyclic as cycles visiting job J more than once may be present in graph $(V, A^J \cup T, d)$. We introduce a structural property of the job insertion graph preventing the existence of these cycles. It is called the short cycle property (SCP) and was introduced in (Gröflin and Klinkert, 2007) for general disjunctive graphs and is used here for job insertion graphs.

Definition 3 *A job insertion graph G^J has the SCP if for any positive cycle with arc set Z' in $(V, A^J \cup E^J, d)$, there exists a short positive cycle Z in $(V, A^J \cup E^J, d)$ with $Z \cap E^J \subseteq Z' \cap E^J$ and $|Z \cap E^J| = 2$.*

In other words, a job insertion graph has the SCP if for any positive cycle Z' visiting J more than once, there exists a positive cycle Z visiting J exactly once entering and leaving J through disjunctive arcs that are contained in Z'.

We say that a CJS-R problem has the *SCP for job insertion* if for all jobs $J \in \mathscr{J}$ and all feasible selections R of the other jobs, G^J has the SCP.

Theorem 1 *Let G^J be a job insertion graph with the SCP. There is a one-to-one correspondence between the feasible insertions in G^J and the stable sets T in conflict graph H with $|T| = |E^J|/2$.*

Proof. See appendix. □

Hence, if a CJS-R problem has the SCP for job insertion, then for all job insertion subproblems, a nice characterization of all feasible insertions as stable sets in an associated bipartite conflict graph is at hand.

3.3 The short cycle property in specific job shop problems

Unfortunately not all, but a large class of CJS-R problems have the SCP for job insertion. We show here that the JSS and BJS problems have the SCP for job insertion. We give a similar proof as in Lemma 8 of Gröflin and Klinkert (2007), and refer the reader to this work for general structural results concerning the SCP for insertion problems.

Proposition 3 *Let $G^J = (V, A^J, E^J, \mathscr{E}^J, d)$ be a job insertion graph of a JSS instance for some job $J \in \mathscr{J}$. Then G^J has the SCP.*

Proof. Given any positive cycle Z', let Z be a "shortest" positive cycle in graph $(V, A^J \cup E^J, d)$ with $Z \cap E^J \subseteq Z' \cap E^J$ where the length is measured by the number of disjunctive arcs $|Z \cap E^J|$. If $|Z \cap E^J| = 2$, we are done.

Assume $|Z \cap E^J| > 2$, and say Z visits job J $k = |Z \cap E^J|/2$ times. Describe Z by the concatenation $Z = (e_1, P_1, e'_1, Q_1, e_2, P_2, e'_2, Q_2, \ldots, e_k, P_k, e'_k, Q_k)$ where the e_i's and e'_i's are disjunctive arcs entering and leaving J, respectively, the P_i's are paths in $(V, \gamma(V_J), d)$ and the Q_i's are paths in $(V, \gamma(V \setminus V_J), d)$.

The following two simple properties of the subgraph $(V, \delta(A^J), d)$ are used. For any pairs of distinct operations J_r, J_s: i) if $r < s$, there exists a path from node v_r to node v_s and such as path is of positive length. Indeed, it must contain arc (v_r, v_{r+1}), which is of length $p_{J_r} > 0$, and no arc is of negative length. And ii) if $r > s$, there exists no path from v_r to v_s.

Consider P_1 and P_2 of cycle Z. P_1 starts at node v_{J_r} of some operation J_r and ends at node v_{J_s} of some J_s. Similarly, P_2 starts at some $v_{J_{r'}}$ and ends at some $v_{J_{s'}}$. By ii), $r \le s$ and $r' \le s'$ hold, and as Z is a cycle, $r \ne r'$ (a cycle does not enter the same node more than once).

Consider the following two cases. a) If $r < r'$, then let concatenation $W = (e_1, P^*, e'_2, Q_2, \ldots, e_k, P_k, e'_k, Q_k)$ where P^* is a path from node v_r to node $v_{s'}$ in $(V, \gamma(V_J), d)$. By i), path P^* exists and is of positive length as $r < r'$ and $r' \le s'$ implies $r < s'$. Then W is of positive length as there are no negative length arcs in G^J. Clearly, W is a closed walk visiting J less than k times. Therefore, W can be decomposed into cycles (cycle decomposition of integer circulations) where each cycle visits J at least once and less than k times, and as W is of positive length also one of these cycles, say Z'', is of positive length.

b) If $r > r'$ then consider the cycle $Z'' = (e_2, P^*, e'_1, Q_1)$, where where P^* is a path from node $v_{r'}$ to node v_s in $(V, \gamma(V_J), d)$. By i), path P^* exists and is of positive length as $r' < r$ and $r \le s$ implies $r' < s$. Then Z'' is of positive length.

In both cases, Z'' is a positive cycle with $Z'' \cap E^J \subseteq Z' \cap E^J$ visiting J less than k times, contradicting the choice of Z as being the shortest. □

Proposition 4 *Let $G^J = (V, A^J, E^J, \mathscr{E}^J, d)$ be a job insertion graph of a BJS instance for some job $J \in \mathscr{J}$. Then G^J has the SCP.*

Proof. The proof is similar to the proof given above. The only differences are the properties of subgraph $(V_J, \delta(A^J), d)$, which are used, namely: i) For any pair of operations J_r, J_s with $r \le s$, there exists a path from node v_r^1 (of operation J_r) to node v_s^4 (of J_s) and it is of positive length as it must contain the processing arc (v_r^2, v_r^3) and no arc is of negative length. And ii) for any pair of operations J_r, J_s with $r > s+1$, there exists no path from node v_r^1 (of operation J_r) to node v_s^4 (of J_s).

Then construct a positive cycle Z'' visiting J less than k times in the similar way as above. □

Hence, independent of the choice of the job J and the feasible selection R of all other jobs, G^J has the SCP in JSS and BJS problems. Therefore, JSS and BJS problems have the SCP for job insertion.

4 A neighborhood for the CJS-R

In this section, we develop a neighborhood that is applicable to all CJS-R problems that possess the SCP for job insertion.

4.1 A neighbor generation scheme

Given a CJS-R problem $(V,\mathscr{J},A,E,\mathscr{E},d,f)$ with the SCP for job insertion, let S be some feasible selection. We are interested in extracting some job $J\in\mathscr{J}$ and reinserting J in a position that is "close" to its position described by selection S. For this purpose, we consider the job insertion graph $G^J=(V,A^J,E^J,\mathscr{E}^J,d)$ where the feasible selection of all other jobs R is taken from S, i.e. $R=S\cap(E\setminus E^J)$. We give ourself a disjunctive arc $g\in E^J\setminus S$ that is forced to be in the neighbor insertion, and ask to construct a feasible insertion T_g with $g\in T$ that is "close" to the current insertion $T_S=S\cap E^J$, i.e., we seek to find a feasible insertion T_g maximizing $|T_g\cap T_S|$.The neighbor selection S_g is then obtained by $S_g=T_g\cup R$.

Consider the neighbor construction in the conflict graph $H=(E^J,U)$ associated to G^J. By Theorem 1, as the current insertion T_S is feasible, T_S is stable in H with size $|E^J|/2$, and any feasible insertion T_g is stable in H with size $|E^J|/2$. We now sketch the idea, how to iteratively construct T_g. The procedure is based on the following observation. For any $f\in T_g$, if there exists an edge $\{f,e\}\in U$, then e cannot be picked, hence its mate $\overline{e}$ must be part of any feasible insertion containing f. We say f implies $\overline{e}$. Hence, start with $T_g=\{g\}$. For each $f\in T_g$ and edge $\{f,e\}\in U$ with $\overline{e}\notin T_g$, add $\overline{e}$ to T_g. Repeat this step until T_g is "closed", i.e., for each node $f\in T_g$ there exists no edge $\{f,e\}\in U$ with $\overline{e}\notin T_g$. Finally, for each "uncovered" pair $D\in\mathscr{E}$, i.e., $D\cap T_g=\emptyset$, add $e\in D\cap T_S$ from the current insertion T_S to T_g.

We formalize this idea by introducing a closure operator and then prove feasibility of the constructed neighbor insertion.

Definition 4 *For any $e,f\in E^J$ in conflict graph $H=(E^J,U)$, let $e\rightarrow f$ if $\{e,\overline{f}\}\in U$. A sequence $P=(e_0,e_1,\ldots,e_n)$, $n\geq 0$, of distinct nodes in $H=(E^J,U)$ is an alternating path from e_0 to e_n if $e_i\rightarrow e_{i+1}$ for all $0\leq i<n$. For any two nodes $e,f\in E^J$, write $e\rightsquigarrow f$ if there exists an alternating path from e to f in H.*

Definition 5 *For any $Q\in E^J$, the closure of Q is the set*

$$\Phi(Q)=\{g\in E^J: e\rightsquigarrow g \text{ for some } e\in Q\}. \tag{6}$$

$Q\subseteq E^J$ is said to be closed if $Q=\Phi(Q)$.

Observe that $e\rightsquigarrow e$ holds for all $e\in E^J$ as $\{e,\overline{e}\}\in U$ by Proposition 1(i). Moreover, $\Phi(Q)$ can be rewritten as $\Phi(Q)=\bigcup_{e\in Q}\Phi(\{e\})$. Then, it is easy to see that Φ is a well-defined closure operator as: i) $Q\subseteq\Phi(Q)$, ii) $\Phi(\Phi(Q))=\Phi(Q)$, and iii) $Q\subseteq R$ implies $\Phi(Q)\subseteq\Phi(R)$.

For any subset of disjunctive arcs $T\in E^J$, let the span $[T]$ of T contain all arcs of T together with all their mates, formally, $[T]=\{e\in E^J:\{e,\overline{e}\}\cap T\neq\emptyset\}$.

Theorem 2 *Given any job insertion graph $G^J=(V,A^J,E^J,\mathscr{E}^J,d)$ with the SCP and a disjunctive arc $g\in E^J$, insertion*

$$T_g=\Phi(\{g\})\cup(T^S\setminus[\Phi(\{g\})]) \tag{7}$$

is feasible.

Proof. See appendix. □

It is easy to see that among all neighbor insertions T containing g, the constructed neighbor T_g is the "closest" to current neighbor T^S, i.e. maximizing $|T_g\cap T_S|$.

Theorem 2 implies feasibility of the corresponding selection, stated as follows.

Corollary 1 *Given a CJS-R problem $(V,\mathscr{J},A,E,\mathscr{E},d,f)$ with the SCP for job insertion, for any feasible selection S, job $J\in\mathscr{J}$, and disjunctive arc $g\in E^J$, the neighbor selection*

$$S_g^J=T_g\cup(S\setminus E^J) \tag{8}$$

is feasible.

We illustrate the neighbor generation scheme in our JSS and BJS examples. Starting with the selections given in Figure 1 and 2, we choose job J to be extracted and reinserted, obtaining the job insertion graphs and conflict graphs depicted in Figure 3. In both examples the current insertion T^S is composed of $T^S = \{\overline{e}_1, \overline{e}_2, \overline{e}_3, \overline{e}_4, \overline{e}_5\}$ placing job J after all other jobs. We decide to force e_5 to be part of the neighbor insertion. In the JSS example, we obtain $\Phi(\{e_5\}) = \{e_3, e_4, e_5\}$ and $T^S \setminus [\Phi(\{g\})] = \{\overline{e}_1, \overline{e}_2\}$, so the neighbor insertion is $T_g = \{\overline{e}_1, \overline{e}_2, e_3, e_4, e_5\}$. In the BJS example, we obtain $\Phi(\{e_5\}) = \{e_1, e_3, e_4, e_5\}$ and $T^S \setminus [\Phi(\{g\})] = \{\overline{e}_2\}$, so the neighbor insertion is $T_g = \{e_1, \overline{e}_2, e_3, e_4, e_5\}$. The corresponding earliest time schedules are illustrated in Figure 4.

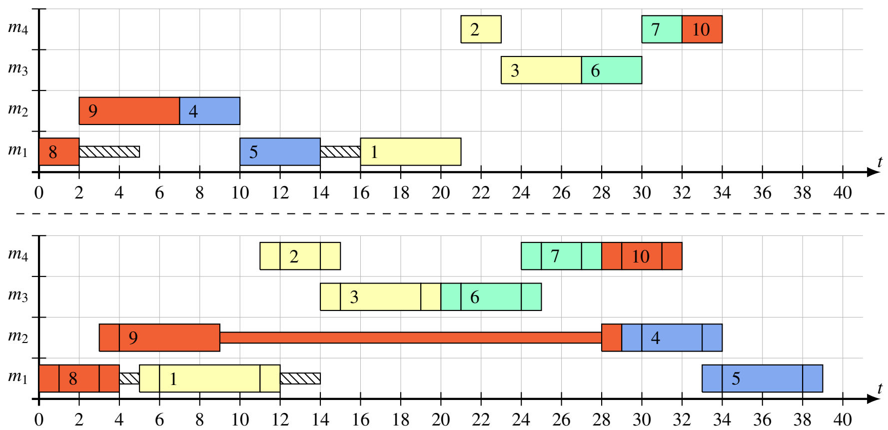

Figure 4: The neighbor schedules obtained by extracting job J (consisting of the operations 1, 2, and 3) from the schedule of Figure 1 and 2 in the JSS (above) and BJS example (below), respectively, and reinserting this job by forcing operation 3 to be executed before operation 6 on machine m_3.

Note that if an instance of the classical job shop scheduling problem is given and $\overline{g}$ is a critical arc incident to job J in some feasible selection S, then $\Phi(\{g\}) = \{g\}$, and the neighbor $S_g^J = \{g\} \cup (S \setminus \{\overline{g}\})$, hence S_g^J is obtained by replacing the critical arc $\overline{g}$ by its mate g.

4.2 A job insertion based neighborhood

Corollary 1 provides us with a general tool to derive a large set of neighbors for any feasible selection given the possible choices of the job and the "forced" disjunctive arc. In order to generate neighbors that are potentially improving the current solution, we use and extend the standard concept of critical arcs as follows.

Given is some CJS-R problem $(V, \mathscr{J}, A, E, \mathscr{E}, d, f)$ with the SCP for job insertion and some feasible selection S. For each node $v \in V$, let P_v be the arc set of a longest path from σ to v in graph $(V, A \cup S, d)$. Usually, the objective value $f(\alpha(S))$ is determined by a subset of (times of) nodes. For instance, in the BJS example in Figure 2, the makespan $f^m(\alpha(S))$ is determined by the start (and the duration) of operation 3, and the flow time $f^f(\alpha(S))$ is determined by the start of the operations 3, 5, 7, and 10. Let V^{contr} be a (setwise) minimal subset of nodes in V determining the objective value for selection S. For each $v \in V^{\text{contr}}$, path P_v is called a *critical path*. The set of *critical arcs* C_S of S is given by $C_S = \{e \in S : e \in P_v \text{ for some } v \in V^{\text{contr}}\}$. A job $J \in \mathscr{J}$ is said to be *critical* if $v \in V_J$ or $w \in V_J$ for some $(v, w) \in C_S$.

In order to move to a better neighbor, we need to replace some arc of C_S. Indeed, by construction of C_S, $\alpha(S') \geq \alpha(S)$ holds for any feasible selection S' with $C_S \subseteq S'$, hence $f(\alpha(S')) \geq f(\alpha(S))$ as function f is regular. In our neighborhood, we build a neighbor selection S_g^J by (8) for each critical job J and each critical arc $\overline{g} \in C_S \cap \delta(V_J)$.

We illustrate the neighborhood in our BJS example given the current selection S that corresponds to the solution depicted in Figure 4 and using the total flow time objective. All end nodes of the jobs contribute to the objective value hence $V^{\text{contr}} = \{v_3^4, v_5^4, v_7^4, v_{10}^4\}$. In Figure 5, graph $(V, A \cup S, d)$ of selection S is depicted, and the bold arcs form the set of arcs belonging to a longest path from σ to v for some $v \in V^{\text{contr}}$. The set of critical arcs is $C_S = \{\overline{a}, \overline{b}, \overline{c}, \overline{d}\}$ (bold dashed arcs in Figure 5), and the following eight neighbors are build: S_a^N, S_a^J, S_b^J, S_b^L, S_c^L, S_c^N, S_d^N, S_d^K. The earliest time schedules that correspond to these neighbors are illustrated in Figure 6. It can be seen, that for arc a, b, and c, the two neighbors generated with the same forced arc result in the same selection, while for arc d the two neighbors are different.

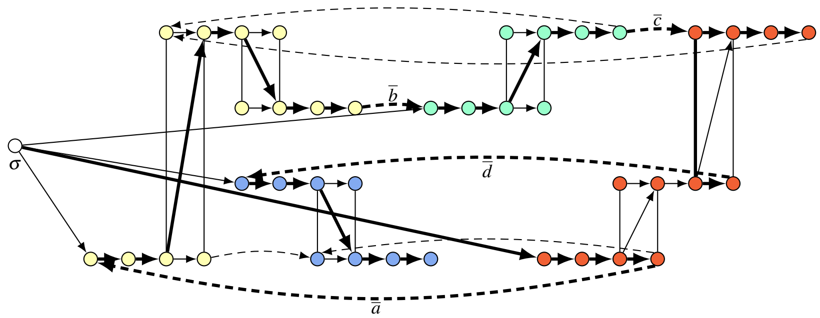

Figure 5: Graph $(V, A \cup S, d)$ of selection S that corresponds to the BJS solution of Figure 4 (below). The arcs of set A and S are depicted by solid and dashed lines, respectively.

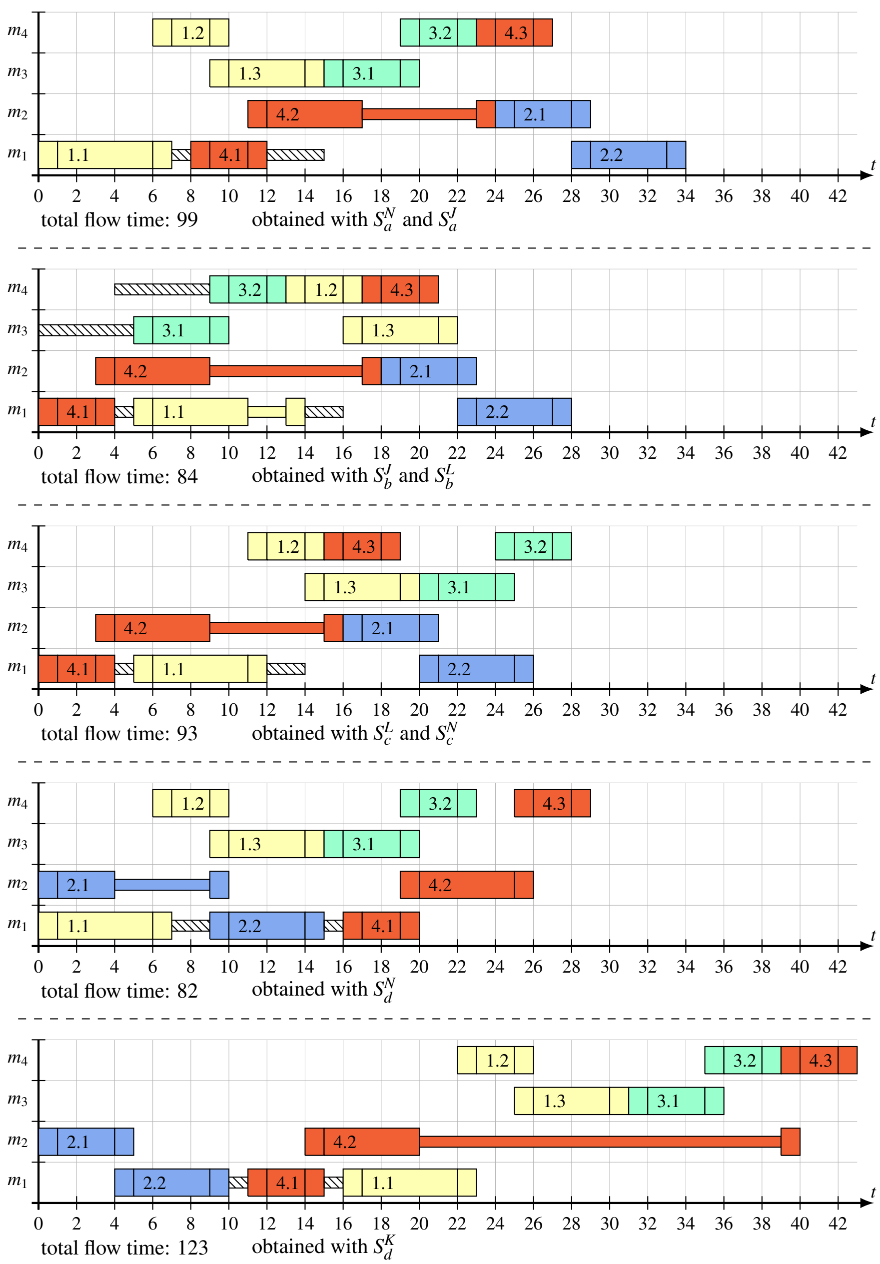

Figure 6: An illustration of the neighborhood in our BJS example. For each attained earliest time schedule, we indicate the objective value and the neighbors leading to this schedule.

5 Computational results

In order to evaluate the validity of the proposed neighborhood, we casted it in a tabu search and conducted an extensive experimental study using well-known job shop scheduling problems. This section first sketches the tabu search, then describes the experimental setting, and finally discusses the obtained results.

5.1 A tabu search

The described neighborhood can be used, in principle, in any local search scheme. We apply a tabu search using some generic features that proved useful in various local search approaches for job shop scheduling problems. It is similar to the approach taken in (Gröflin et al., 2011; Bürgy, 2014) and will be called *JILS* (job insertion based local search). We now describe its main ingredients and refer the reader to Glover and Laguna (1997) for a comprehensive description of tabu search.

A tabu list L is used for storing entries of the last $maxL$ iterations. Initially, list L is empty. In an iteration, i.e., after moving from a selection S to a neighbor S_g, arc $\overline{g}$ is added to L at first position and the oldest entry is deleted from L if $|L| > maxL$. A neighbor S' is called tabu if $S' \cap L \neq \emptyset$.

The choice of the move to be executed from the current selection S is based on the evaluation of the entire set of neighbors. First, the objective value $f(S')$ is computed for each neighbor S'. Some of these values are then "corrected" using the tabu list as follows: if a neighbor selection S' is tabu and $f(S')$ is not lower than the objective value of the best selection found so far, then a penalty of $(maxL - k) \times B$ is added to the objective value, where k is the position of the first entry in L making the move tabu and B is a large constant. We move then to a neighbor with lowest (corrected) objective value.

In order to improve the search, we implemented the following two additional long term memory structures, which were also used by Nowicki and Smutnicki (1996).

As the tabu search does not prevent being trapped in long cycles, we use a list C that stores the sequence of objective values obtained during the search. Cycles are detected by scanning C for repeated subsequences. Specifically, the search is said to be cycling at iteration k if there exists a period $\delta > maxL$ such that $C[k] = C[k - a\delta]$ for $a = 1, \ldots, maxR$, where $C[j]$ is the objective value in iteration j, $maxC$ reflects the maximum length of a cycle and $maxR$ the number of repetitions we search for.

A list E of so-called elite selections is maintained to diversify the search. Initially, E is empty and a new selection S is added at first position to E if its objective value is lower than the value of the best selection found so far. If the search runs for a given number $maxI$ of iterations without improving the best selection or if a selection has no neighbors or if a cycle is detected, then the current search path is terminated, list C is cleared, and the search is resumed form the first selection in list E. For this purpose, an elite selection S is stored together with its tabu list and the best (w.r.t. the objective value) $maxN$ neighbors that have not yet been directly visited from S. An elite selection is deleted from E if its set of neighbors is empty.

An initial selection is constructed as follows. First, generate randomly a permutation of all jobs. Then, according to this permutation, insert one by one a job into the current (partial) selection S^{part}. For the insertion of a job J into S^{part}, we use our job insertion procedure to generate a set of feasible selections $\mathscr{S}_J$, and we choose the best selection among them. Specifically, a first selection S_J is obtained by inserting J after all other jobs, i.e. $S_J = S^{\text{part}} \cup E^{J-}$. Store S_J in $\mathscr{S}_J$. While $E^{J-} \cap S_J \neq \emptyset$, execute the following three steps:

1. Determine a disjunctive arc $f \in E^{J-} \cap S_J$ so that f is on a longest path P_v for some $v \in V_J$.
2. If no such f exists stop, else build neighbor selection $S_{\overline{f}} = T_{\overline{f}} \cup (S \setminus E^J)$ according to (8).
3. Store $S_{\overline{f}}$ in $\mathscr{S}_J$ and set S_J to $S_{\overline{f}}$.

Finally, among all selections in set $\mathscr{S}_J$, choose selection S with lowest objective value and update S^{part} to S.

In order to further diversify the search, we make use of the parallel computing capabilities by starting and running in parallel $maxS$ independent search paths, each with its own initial selection, tabu list and list C. The $maxS$ search paths share list E of elite selections and the storage of the best selection found so far.

5.2 Experimental setting

JILS was implemented in Java and run on a PC with 3.3. GHz Intel Core i5-4590 processor (4 threads) and 16 GB memory.

Extensive test were performed to evaluate JILS using the job shop problems introduced in Sections 2.3.1 and 2.3.2 with various regular objectives. Specifically, we considered the job shop without setup times (JS), the job shop with sequence-dependent setup times (JSS), and the blocking job shop (BJS) with the five objective functions introduced in the examples: makespan, total flow time, total squared flow time, total tardiness, and total weighted tardiness.

We used standard benchmark instances from the literature: for the JS la01-40 introduced by Lawrence (1984), orb01-010 by Applegate and Cook (1991), and ta01-50 by Taillard (1994); for the JSS t2-psXY, $XY \in \{01,\ldots,15\}$ by Brucker and Thiele (1996), t2-laXYsdst, $XY \in \{21,24,25,27,29,38,40\}$, and t2-abz$Z$sdst, $Z \in \{7,8,9\}$, by Vela et al. (2010); and for the BJS la01-40 by Lawrence (1984) setting all transfer and setup times to 0 (obtaining so-called with-swap BJS instances).

In all problem types and instances, the release times are set to 0, and for the three tardiness objectives, the due date s^{d}_J of each job $J \in \mathscr{J}$ is set according to the popular rule introduced by Eilon and Hodgson (1967): $s^{\mathrm{d}}_J = \lfloor f * \sum_{i\in J} p_i \rfloor$ where f is referred to as the due date tightness factor. f is set to 1.3 for all JS instances, 1.6 for all JSS instances, and 1.8 for all BJS instances.

The input parameter settings of JILS were carefully analyzed in preliminary tests and set as follows: $maxC = 4$, $maxI = 50000$, $maxN = 10$, $maxS = 8$, and the tabu list size $maxL$ is set to 10 for all instances with makespan objective and 16 for all other objectives. We observed that JILS is quite robust with respect to different parameter settings.

Five independent runs of JILS were executed for each problem type, objective, and instance, and the computation time limit of each run was set to 1800 seconds. In order to evaluate the approach also with shorter run times, we recorded the objective value of the best solution found so far by JILS after 60 and 300 seconds. The appendix contains detailed results in Table 4, 5, and 6 for the JS, in Table 7 and 8 for the JSS, and in Table 9 and 10 for the BJS.

While we tried to keep the JILS as generic as possible, we incorporated a slight modification of the neighborhood for the JS with makespan objective. We observed in preliminary tests that better results can be obtained for this problem type if JILS considers only the critical arcs at the border of so-called critical blocks. Indeed, it is well-known that swapping a critical arc in the "inner part" of a critical block does not improve the makespan. Therefore, we used this smaller neighborhood, called N5 in (Blazewicz et al., 1996), for the JS with makespan objective.

5.3 Comparison with benchmarks

In order to assess the performance of JILS, we compare the attained results with current benchmarks from the literature when available. As these benchmarks were obtained using different computational settings, especially computing power and run times, we try to make a somewhat fair comparison by comparing the average results of JILS (over the five runs) after 60, 300, and 1800 seconds with the benchmark values. Note that our aim is to show that JILS provides good results for a large class of CJS-R problems, and it is not our goal to give a ranking of the different approaches nor to compete in a horse race fashion with other methods.

A simple performance measure is used to compare the results of JILS with benchmarks. For each benchmark result, we compute the relative gap of the average result of JILS over the five runs ($JILS$) with the result of the benchmark ($bench$), i.e., $(JILS - bench)/bench$. These gaps are determined for the results of JILS after a run time of 60, 300, and 1800 seconds. An advantage of this performance measure is its simple interpretation and variants of it are widely used. However, some care should be given to the numbers as large gaps might be obtained if the benchmark value is low even if the absolute difference is small. We use small multiples (see e.g., Tufte, 2001) to illustrate the relative gaps in a compact way.

We first consider the JS, then the JSS, and conclude with the BJS. In all three problem types, we first deal with the makespan objective, and then with all other objectives.

5.3.1 JS with makespan objective

There is a large collection of benchmark results available for the JS with makespan objective. We consider the following benchmarks, briefly describing the approaches and the basic computational settings.

As mentioned in Section 1, Grimes and Hebrard (2015) recently proposed a method that is applicable to a broad class of job shop scheduling problems with makespan objective. We consider their average results (over 10 runs) for the JS obtained with their general light weighted approach and abbreviate these results by *GH*. The authors used a Intel Pentium IV 3.0 GHz processor and a time limit of 3600 seconds per run.

Zhang et al. (2008) developed a heuristic for the JS with makespan objective combining simulated annealing and tabu search strategies. They apply a neighborhood based on swapping critical operations. We consider their average results over 10 runs (see Zhang et al., 2008, Table 6, p. 292) and abbreviate these results by *ZLRG*. The authors used a Intel Xeon 2.66 GHz processor. The run time varies between 11 and 923 second as the stopping criterion is based on the number of iterations.

We first discuss the results obtained in the instances la01 to la40 and orb01 to orb10. The optimal values are known for these instances (see Brucker et al., 1994; Perregaard and Clausen, 1998; Grimes and Hebrard, 2015). Similar as other state of the art heuristics, JILS finds optimal or near-optimal solutions for these instances within a short computation time. Indeed, the overall average relative gap to the optimal values is 0.14%, 0.06%, and 0.03% for the results of JILS obtained after 60, 300, and 1800 seconds, respectively. Also, JILS finds an optimal solution in all the five runs in 41 instances (out of 50), and in at least one of the five runs in 47 instances.

The results in the Taillard instances are now compared with GH. The relative gaps of JILS to GH are illustrated in Figure 7 (above) using small multiples as follows. The first, second, and third line depicts the relative gaps for the average results of JILS after 60, 300, and 1800 seconds, respectively. The time limit of JILS and the relative gap averaged over all instances is provided on the left. The range of the attained gaps is partitioned into seven intervals: $[a\%;-6.0\%]$, $]-6.0\%;-2.0\%]$, $]-2.0\%;-0.5\%]$, $]-0.5\%;0.5\%]$, $]0.5\%;2.0\%]$, $]2.0\%;6.0\%]$, and $]6.0\%;b\%]$ where a is the minimum of -10.0% and the minimum attained gap and similarly b is the maximum of 10.0% and the maximum attained gap. For each time limit, the relative gap of each instance is illustrated by drawing a small rectangle above the interval where the relative gap belongs to. In order to distinguish the instances, the rectangles are always drawn at the same relative position in each interval. For this purpose, we specify an ordering of the instances. We here order the instances according to their number in the instance name. We then represent the first instance (ta01) at the top left (first row, first column) in each interval, then ta02 on its right (first row, second column), and so forth, so the last instance (ta50) is represented on the bottom right (fifth row, 10th column). As an example, consider the instance ta42. As 10 instances are depicted per line, its position is at the fifth row in the second column. The benchmark value of GH is 2027.5, and the average value obtained by JILS is 2033, 1990, and 1982 for 60, 300, and 1800 seconds, respectively (see Table 4), giving respective relative gaps of 0.3%, -1.8%, and -2.2%. Hence, in the first, second, and third line of Figure 7, the rectangle of ta42 is drawn in the interval $]-0.5\%;0.5\%]$, $]-2.0\%;-0.5\%]$, and $]-6.0\%;-2.0\%]$, respectively.

Considering the resulting small multiples in Figure 7 (above), it can be observed that JILS is competitive with GH. Indeed, already after 60 seconds, the quality of JILS is comparable to GH, and after 1800 seconds, JILS is considerably lower than GH in the larger instances ta31 to ta50.

The following can be observed when comparing in the same fashion the relative gaps of JILS and ZLRG in Figure 7 (below). JILS needs some time to match the results of ZLRG. After 60 seconds, the ZLRG values are lower, especially in the large instances. But quite similar results are attained after 1800 seconds. Indeed, the overall average gap is then just 0.5%. Only in the largest instances, ZLRG produces results that are slightly lower than the results of JILS. It can be concluded that JILS is quite competitive with ZLRG.

Altogether, we conclude that JILS performs well in the JS with makespan objective.

5.3.2 JS with other objectives

The majority of the papers dealing with the JS considers the makespan objective. Consequently, there are less benchmark results available for the other objectives.

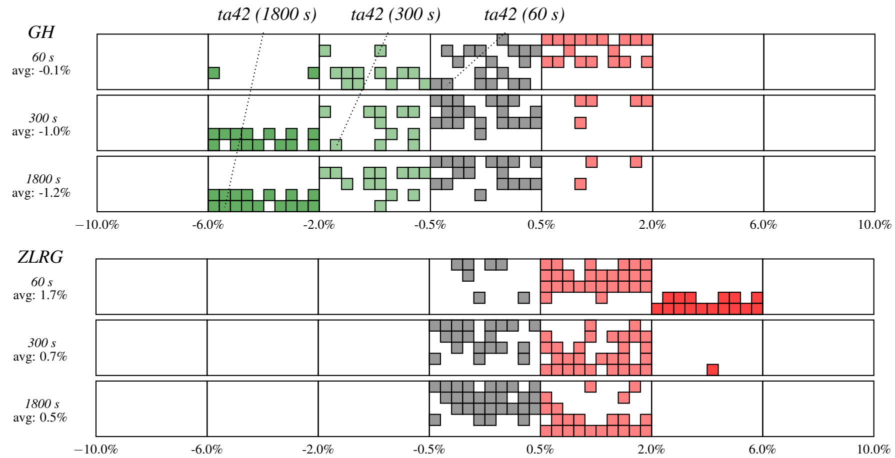

Figure 7: Comparison of the results obtained in the JS with makespan objective with results of Grimes and Hebrard (2015) (above) and Zhang et al. (2008) (below) in the Taillard instances.

González et al. (2010) developed a hybrid solution method for the JS with flow time objective, combining a genetic algorithm and a tabu search method. The tabu search applies a neighborhood in which a critical operation is moved (forward or backward) in its critical block provided that a sufficient condition for feasibility of the move is satisfied. We consider their average results over 20 runs (see González et al., 2010, Table 4, p. 41) and abbreviate these results by *GVSV*. The authors used a Intel Xeon 2.66 GHz processor, and the computation time per run varies between 98 and 895 seconds.

Figure 8 illustrates the relative gaps of JILS to GVSV in instances la01 to la20 ordered in the small multiples according to the number in the instance name. It can be observed that JILS gives slightly higher values than GVSV after 60 seconds, but after 1800 seconds, the performances of JILS and GVSV are similar. Indeed, the relative gap of JILS to GVSV is then only 0.2% averaged over all instances.

A considerable amount of work was dedicated to study the JS with total weighted tardiness objective, which is a generalization of the JS with total flow time objective. We compare JILS with the following two approaches.

As mentioned in Section 1, Mati et al. (2011) recently developed a tabu search method that is applicable to the JS with general regular objective. We consider their results for the JS with total weighted tardiness objective (see Mati et al., 2011, results with $f = 1.3$ in Table 3, p. 38) and abbreviate these results by *MDL*. The authors used a 2.6 GHz processor. As they have a computation time of only 18 seconds per run, we compare the JILS results to their best results over 10 runs.

González et al. (2012a) recently addressed the JSS problem with weighted tardiness objective, and they also provide benchmark results for the JS (without setup times). They propose a hybrid heuristic combining tabu search and genetic algorithms. Neighbors are built in the tabu search by reversing a single critical arc provided that some feasibility condition is satisfied. We consider their average results over 10 runs (see González et al., 2012a, column GTN with $f = 1.3$, Table 2, p. 2108 and Table 3, p. 2110) and abbreviate these results by *GGVV*. They used an Intel Core 2 Duo processor with 2.66 GHz, and the computation time of a run varies between 18 and 1931 seconds.

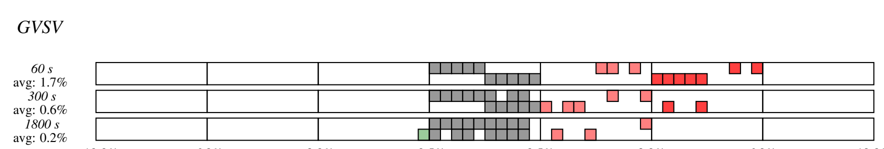

Figure 8: Comparison of the results obtained in the JS with total flow time objective with results of González et al. (2010).

Figure 9 (upper part) illustrates the relative gaps of JILS to MDL in the instances la17 to la20 and orb01 to orb10 using this order in the small multiples, e.g. instance la17 is illustrated on the top left position. It can be observed that the results of JILS are quite close to those of MDL after 300 and 1800 seconds, concluding that both methods perform quite similarly.

Figure 9 (lower part) illustrates the relative gaps of JILS to GGVV for the instances la01 to la40 and orb01 to orb10, ordered in the small multiples according to their size as given in Table 6. It can be observed that JILS quickly finds results of a similar quality than GGVV in the small instances, and on average finds similar results in all instances after 1800 seconds. Indeed, the average relative gap is then 0.0%. Interestingly, the relative gaps are quite high for some instances. For example, it is -7.5% for instance la34 and 7.3% for la29 (after 1800 seconds). It might be concluded that there is some further improvement potential for both approaches in these instances.

As there are no benchmarks results available for the total squared flow time and total tardiness objectives for our benchmark instances, we resorted to compare the performance of JILS with results obtained via a mixed integer programming (MIP) model that we derived in a straightforward manner from the disjunctive programming formulation of the CJS-R problem (see Section 2.1). As preliminary tests revealed that the simple MIP approach only finds good solutions in small instances, we decided to use the smallest instances la01 to la10 for these experiments. We ran the MIP using the solver Gurobi 6.5 with a time limit of 7200 seconds for each instance, and Table 1 reports the obtained results.

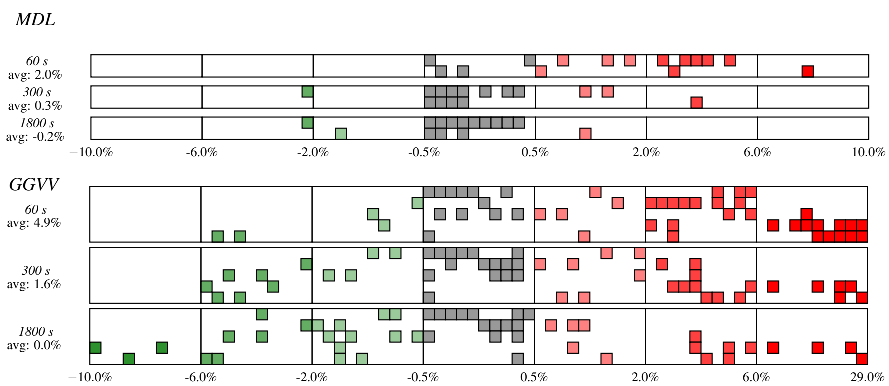

Figure 9: Comparison of the results obtained in the JS with total weighted tardiness objective with results of Mati et al. (2011) (above) and González et al. (2012a) (below).

Table 1: Comparison of the MIP results to the JILS results for JS instances with total squared flow time objective (left, in units of 1000) and total tardiness objective (right). Columns avg-1800 provide the results obtained with JILS after 1800 seconds, columns LB; UB give the MIP lower and upper bounds ("opt" if they are the same), and columns diff present the relative difference of the results avg-1800 and UB. The instances are grouped according to their size $n \times m$, where n is the number of jobs and m the number of machines.

objective	total squared flow time			total tardiness		
results	avg-1800	LB ; UB	diff	avg-1800	LB ; UB	diff
10×5						
la01	2599	2599 (opt)	0.0%	1194	1194 (opt)	0.0%
la02	2296	2110 ; 2339	-1.8%	1065	1065 (opt)	0.0%
la03	1956	1956 (opt)	0.0%	1076	1076 (opt)	0.0%
la04	2060	1916 ; 2060	0.0%	1096	1096 (opt)	0.0%
la05	1862	1862 (opt)	0.0%	1164	1164 (opt)	0.0%
15×5						
la06	5953	2193 ; 6396	-6.9%	3491	916 ; 3643	-4.2%
la07	5252	2022 ; 5651	-7.1%	3282	713 ; 3333	-1.5%
la08	5188	1969 ; 6135	-15.4%	3056	735 ; 3872	-21.1%
la09	6553	2364 ; 7340	-10.7%	3567	786 ; 3984	-10.5%
la10	6348	2150 ; 6916	-8.2%	3655	818 ; 4097	-10.8%

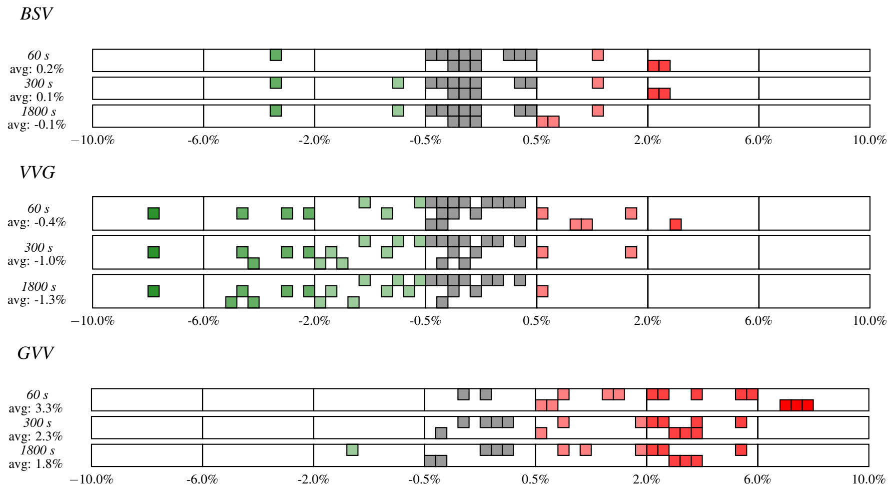

Figure 10: Comparison of the results obtained in the JSS with makespan objective to the results of Balas et al. (2008); Vela et al. (2010); González et al. (2012b).

For the instances of size 10×5 the following can be observed. The MIP approach shows that JILS always found an optimal solution for the total tardiness objective. For the total squared tardiness objective, the MIP approach could prove optimality of three (out of five) results, and JILS obtained results of similar quality than the MIP. In the instances of size 15×5 the optimality gaps of the MIP approach are large, and on average, JILS gives 9.7% and 9.6% lower results than the MIP approach for the total squared tardiness objective and the total tardiness objective, respectively.

In summary, JILS is competitive in the JS with all considered objectives.

5.3.3 JSS with makespan objective

The JSS has attracted the attention of many researchers. As in the JS, most articles are discussing the makespan objective.

A prominent contribution addressing the JSS with makespan objective is (Balas et al., 2008), in which a shifting bottleneck procedure is proposed. The single machine scheduling subproblems are treated as traveling salesman problems with time windows and are solved with dynamic programming. We consider their results obtained by the multi-run SB-RGLS10 version (see Balas et al., 2008, Table 6, p. 2108 and Table 3, p. 260) and abbreviate these results by *BSV*. They used a UltraSPARC-II processor with 360 MHz, and the computation time of a run varies between 264 and 3033 seconds.

Vela et al. (2010) also addressed the JSS problem with makespan objective. They propose a hybrid heuristic combining a local search and a genetic algorithm. Neighbors are built in the tabu search by reversing the orders of operations in a critical block. They consider only moves where feasibility is ensured by some sufficient condition. We consider their average results over 30 runs (see Vela et al., 2010, Table 5, p. 161 and Table 6, p. 162) and abbreviate these results by *VVG*. They used an Intel Pentium IV processor with 1.7 GHz, and the computation time of a run varies between 2 and 166 seconds.

González et al. (2012b) tackled the JSS problem with makespan objective. They propose a similar hybrid heuristic as Vela et al. (2010) based on a genetic algorithm with a local search component. We consider their average results (over 30 runs) obtained with Lamarckian evolution (see González et al., 2012b, Table 1, p. 157)) and abbreviate these results by *GVV*. They used an Intel Core 2 Duo processor with 2.6 GHz, and the computation time of a run varies between 35 and 180 seconds.

Figure 10 (upper part) illustrates the relative gaps of JILS to BSV for the instances t2-ps01 to t2-ps15 using this order in the small multiples. Observing that the average gaps are very small, it can be concluded that the results of JILS are of a similar quality than those of BSV.

Figure 10 (middle part) illustrates the relative gaps of JILS to VVG for the instances t2-ps01 to t2-ps15 and t2-laXYsdst, $XY \in \{21, 24, 25, 27, 29, 38, 40\}$, ordered in the small multiples according to their size as in Table 7. It can be observed that the results of JILS are slightly better than VVG, even after a short run time.

Figure 10 (lower part) illustrates the relative gaps of JILS to GVV for the instances t2-ps11 to t2-ps15, t2-laXYsdst, $XY \in \{21, 24, 25, 27, 29, 38, 40\}$, t2-abz$Z$sdst, $Z \in \{7, 8, 9\}$, ordered in the small multiples according to their size as in Table 7. It can be seen that GVV provides better results than JILS, especially when compared to the JILS results obtained after 60 and 300 seconds. Nevertheless, the average gap of the JILS results after 1800 seconds is only about 1.8%.

In summary, JILS has a quite good performance in the JSS with makespan objective.

5.3.4 JSS with other objectives

Only few papers address other objectives than the makespan in the JSS.

González et al. (2012a) tackle the JSS with total weighted tardiness objective. The approach was already described in Section 5.3.2. We consider their average results over 10 runs (González et al., 2012a, column GTN with $f = 1.6$ in Table 1, p. 2106) and abbreviate these results by *GGVV2*. They used an Intel Core 2 Duo processor with 2.66 GHz and the computation time of a run varies between 45 and 810 seconds.

Figure 11 illustrates the relative gaps of JILS to GGVV2 for the instances t2-ps01 to t2-ps15, t2-laXYsdst, $XY \in \{21, 24, 25, 27, 29, 38, 40\}$, and t2-abz$Z$sdst, $Z \in \{7, 8, 9\}$, ordered in the small multiples according to their size as in Table 7. It can be seen that GGVV2 provides substantially lower values when compared to the average results of JILS after 60 and 300 seconds. Also, when compared to the results obtained after 1800 seconds, GGVV2 gives 2.9% lower results on average. However, as illustrated in the last row, the best results of JILS (over the five runs) meet the performance of GGVV2. We also recognized in preliminary tests that JILS can be made more robust for the JSS with total weighted tardiness if a more specialized neighborhood is used. A further investigation is left for future work.

As there are no benchmarks results available for the total flow time, total squared flow time, and total tardiness objectives, we compare the performance of JILS with results obtained with the same MIP approach that was applied for the JS (see Section 5.3.2). For this purpose, we used the smallest JSS instances ts-ps01 to ts-ps10 and ran the MIP for each instance using the solver Gurobi 6.5 with a time limit of 7200 seconds. Table 2 reports the obtained results.

The following can be observed for the instances of size 10×5. The average results of JILS are equal or lower than the MIP results in all the instances. Also, in the instances where the MIP approach could prove optimality, JILS always found an optimal solution. In the instances of size 15×5, JILS gives substantially lower results than the MIP approach. Indeed, the relative gap is -8.2%, -15.8%, and -13.0% averaged over all instances of this size for the total flow time, total squared flow time, and total tardiness objective, respectively.

Altogether, we conclude that JILS performs well in the JSS with all considered objectives.

5.3.5 BJS with makespan objective

A considerable amount of work has been dedicated to the BJS with makespan objective. The benchmarks are taken from the following two recently published articles.

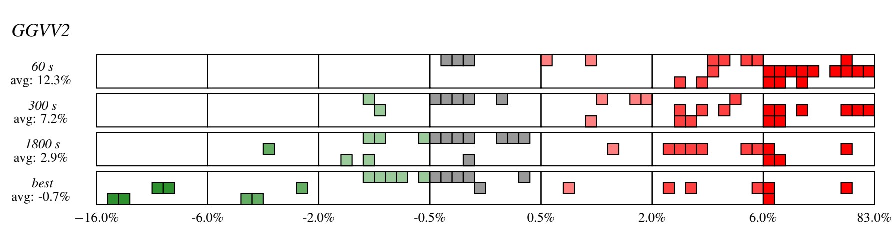

Figure 11: Comparison of the results obtained in the JSS with total weighted tardiness objective to the results of González et al. (2012a).

Table 2: Comparison of the MIP results to the results of JILS for JSS instances with total flow time, total squared flow time (in units of 1000), and total tardiness objective. See Table 1 for a detailed explanation.

objective	total flow time			total squared flow time			total tardiness		
results	avg-1800	LB ; UB	diff	avg-1800	LB ; UB	diff	avg-1800	LB ; UB	diff
10×5									
t2-ps01	6050	6050 (opt)	0.0%	4053	3753 ; 4210	-3.7%	1623	1623 (opt)	0.0%
t2-ps02	5505	5220 ; 5522	-0.3%	3479	2511 ; 3479	0.0%	1395	1395 (opt)	0.0%
t2-ps03	5092	4773 ; 5171	-1.5%	2973	2467 ; 2973	0.0%	1361	1361 (opt)	0.0%
t2-ps04	5478	4793 ; 5549	-1.3%	3375	2449 ; 3467	-2.7%	1583	1020 ; 1620	-2.3%
t2-ps05	5041	5041 (opt)	0.0%	2874	2173 ; 3054	-5.9%	1545	1190 ; 1553	-0.5%
15×5									
t2-ps06	10706	6522 ; 11261	-4.9%	8685	2756 ; 10690	-18.8%	4379	591 ; 5241	-16.4%
t2-ps07	10144	5936 ; 11211	-9.5%	7912	2458 ; 9349	-15.4%	4180	747 ; 4521	-7.5%
t2-ps08	10193	6231 ; 11074	-8.0%	8019	2620 ; 9937	-19.3%	4083	537 ; 4966	-17.8%
t2-ps09	10879	6578 ; 12253	-11.2%	9346	2849 ; 10564	-11.5%	4155	976 ; 4672	-11.1%
t2-ps10	10828	6324 ; 11709	-7.5%	8950	2731 ; 10433	-14.2%	4305	807 ; 4897	-12.1%

Oddi et al. (2012) developed a constrained programming approach for the BJS with makespan objective. They propose an iterative improvement algorithm based on iterative flattening search and perform tests using IBM ILOG CP optimizer. We consider their results obtained with the IBM ILOG CP optimizer (see Oddi et al., 2012, Table 3, p. 7) and abbreviate these results by *ORCS*. They used an AMD Phenom II X4 Quad processor with 3.5 GHz, and executed one run of 1800 seconds for each instance.

Pranzo and Pacciarelli (2015) developed an iterated greedy algorithm for the BJS with makespan objective. Their approach is based on a repetition of a destruction phase that is deleting a part of the solution and a construction phase that starts from a partial solution and applies a greedy approach trying to construct a new feasible solution. We consider their results from (Pranzo and Pacciarelli, 2015, Table 7, p. 21) and abbreviate them by *PP*. They used an Intel Core 2 Duo processor with 2.6 GHz. As they have a computation time of only 60 seconds per run, we compare the average results of JILS with their best results over 10 runs.

Figure 12 illustrates the relative gaps of JILS to ORCS (above) and to PP (below) for the instances la01 to la40, ordered in the small multiples according to this order. It can be observed that JILS is substantially better than ORCS. Indeed, the average gap is -4.5%, -6.0%, and -7.6% after 60, 300, and 1800 seconds, respectively. Especially in the large instances, JILS is much better than ORCS. For example, the average gap is -12.0% in the instances of size 15×15 (i.e., la36 to la40).

Considering the results of PP, it can be seen that JILS gives substantially lower values. Indeed, the average gap is -2.0%, -3.6%, and -5.2% after 60, 300, and 1800 seconds, respectively. JILS is at a particular advantage in large

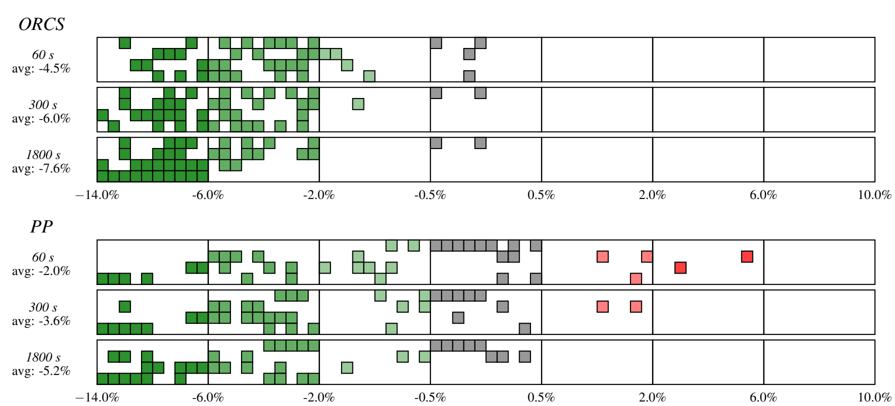

Figure 12: Comparison of the results obtained in the BJS with makespan objective to results of Oddi et al. (2012); Pranzo and Pacciarelli (2015).

instances. For example, the results of JILS are, on average, 13.3% lower than PP in instances of size 30×10 (i.e., la31 to la35).

We conclude that JILS is substantially improving the best results of the literature in the BJS with makespan objective.

Note that we proposed a tabu search for the flexible version of the BJS with makespan objective in a previous article (Gröflin et al., 2011), and a refined version for more general complex job shops with makespan criterion in (Bürgy, 2014). These methods formed the basis for the development of JILS, and the attained results in (Bürgy, 2014) are of a similar quality than the present results.

5.3.6 BJS with other objectives

To our best knowledge, there are no benchmark results available for the BJS with other objectives than the makespan using the standard benchmark instances. Therefore, we compare the performance of JILS with the other objectives to the results obtained with the same MIP approach that was applied for the JS and JSS (see Section 5.3.2). For this purpose, we used the smallest instances la01 to la10 and ran the MIP for each instance using the solver Gurobi 6.5 with a time limit of 7200 seconds. Table 3 reports the obtained results. The following can be observed for the instances of size 10×5. The MIP approach shows that JILS always found an optimal solution for the total flow time, total tardiness, and total weighted tardiness objectives. In addition, the JILS results are of the same quality or slightly better than the MIP results for the total squared flow time objective. Similar as for the JS and JSS, the optimality gaps of the MIP results are large in the instances of size 15×5 for all four objectives, and JILS gives substantially lower results.

In summary, JILS has a very good performance for the BJS with all considered objectives.

Table 3: Comparison of the MIP results to JILS for the BJS instances with total flow time objective, total squared flow time objective (in units of 1000), total tardiness objective, and total weighted tardiness objective. See Table 1 for a detailed explanation.

objective	total flow time			total squared flow time			total tardiness			total weighted tardiness		
results	avg-1800	LB ; UB	diff	avg-1800	LB ; UB	diff	avg-1800	LB ; UB	diff	avg-1800	LB ; UB	diff
10×5												
la01	5152	5152 (opt)	0.0%	3123	2928 ; 3123	0.0%	743	743 (opt)	0.0%	1486	1486 (opt)	0.0%
la02	4937	4937 (opt)	0.0%	2982	2313 ; 2982	0.0%	700	700 (opt)	0.0%	1251	1251 (opt)	0.0%
la03	4605	4605 (opt)	0.0%	2574	2108 ; 2682	-4.0%	736	736 (opt)	0.0%	1461	1461 (opt)	0.0%
la04	4764	4764 (opt)	0.0%	2663	2318 ; 2663	0.0%	716	716 (opt)	0.0%	1236	1236 (opt)	0.0%
la05	4447	4447 (opt)	0.0%	2309	1816 ; 2309	0.0%	796	796 (opt)	0.0%	1244	1244 (opt)	0.0%
15×5												
la06	9715	5705 ; 10721	-9.4%	7769	2356 ; 10296	-24.5%	3075	165 ; 3475	-11.5%	5469	680 ; 7069	-22.6%
la07	9136	5301 ; 10310	-11.4%	6715	1899 ; 8944	-24.9%	2946	402 ; 3764	-21.7%	4727	748 ; 6634	-28.7%
la08	9042	5716 ; 11051	-18.2%	6722	2080 ; 9840	-31.7%	2831	387 ; 3260	-13.1%	4951	872 ; 6049	-18.1%
la09	10237	6001 ; 11169	-8.3%	8690	2276 ; 14300	-39.2%	3127	212 ; 3762	-16.9%	5685	942 ; 7656	-25.7%
la10	9758	5881 ; 10684	-8.7%	7986	2417 ; 10650	-25.0%	2994	291 ; 3953	-24.3%	5785	561 ; 6275	-7.8%

6 Concluding remarks

We developed a neighborhood that can be applied to a wide range of job shop scheduling problems with regular objectives. By casting the neighborhood in a tabu search, we evaluated its performance with an extensive experimental study using three well-known job shop scheduling problems and five regular objectives. The obtained results show that the proposed approach has a good and robust performance. Indeed, it is quite competitive with the state of the art for all considered objectives in the job shop and in the job shop with sequence-dependent setup times, substantially improves the state of the art results in the blocking job shop with makespan objective, and establishes first results in the blocking job shop with the other considered objectives.

The class of regular objectives is certainly valuable in practice. For example, the total (weighted) squared tardiness objective allows to keep the maximum tardiness low while maintaining an acceptable level for the total tardiness, which cannot be accomplished by a linear (weighted) penalization of the tardiness. In further research, it might be interesting to investigate non-regular objectives penalizing earliness and tardiness. These kind of objectives are especially relevant in a just in time production context.

The proposed neighbor generation scheme provides a general tool to derive a large set of feasible neighbors. It might be interesting to consider other types of neighbors than those used in the considered neighborhood. Introducing larger moves of operations within a critical block might be promising. Also, it might be interesting to consider the optimal (re-) insertion of a job instead of searching for a neighbor close to the current solution. This optimal job insertion problem might be used, for example, to construct a good initial solution and in a large neighborhood search approach. Although the optimal job insertion problem is already NP-hard for the classical job shop, the nice characterization of all feasible insertions given in Section 3 might help in the development of an efficient method inserting a job in a near-optimal fashion.

7 Appendix

7.1 Proofs of job insertion properties

Hereafter, we provide proofs of Proposition 2, Theorem 1, and Theorem 2. Although they can easily be deduced from Gröflin and Klinkert (2007), we give them for convenience and completness.

of Proposition 2. (i) Observe that any acyclic insertion T is stable in H by construction of H. If insertion T is also complete hence feasible, then it is of size $|E^J|/2$ as $|T \cap \{e, \overline{e}\}| = 1$ for each pair $\in \mathscr{E}^J$ by Assumption 1 (iv), so $|T| = |\mathscr{E}^J|$, and $\mathscr{E}^J$ is a partition of E^J into pairs, so $|T| = |\mathscr{E}^J| = |E^J|/2$.

(ii) In view of Proposition 1 (i), any stable set T in H picks at most one node of each pair $\{e, \overline{e}\} \in \mathscr{E}^J$, hence $|T| \leq |E^J|/2$. Asking $|T| = |E^J|/2$ implies that exactly one node is picked from each pair $\{e, \overline{e}\} \in \mathscr{E}^J$, hence insertion T is complete. □

of Theorem 1. In view of Proposition 2, we just need to show that any stable set T of size $|E^J|/2$ corresponds to a positive acyclic insertion. Assume the contrary, i.e., insertion T is positive cyclic, and let Z' be a positive cycle in $(V, A^J \cup T, d)$. By the SCP there exists a short positive cycle Z with $Z \cap E^J \subseteq Z' \cap E^J \subseteq T$ and $|Z \cap E^J| = 2$, so $\{e, f\} = Z \cap E^J$ for some $e \in E^{J-}, f \in E^{J+}$. Then $\{e, f\}$ is a positive cyclic insertion, hence $\{e, f\} \in U$ contradicting the stability of T in H. □

of Theorem 2. We show that i) $\Phi(\{g\})$ is stable in H, ii) T_g is stable in H, iii) T_g is complete.

i) By definition of the closure and the bipartition of H, $\Phi(\{g\}) \subseteq E^{J+}$ if $g \in E^{J+}$ and $\Phi(\{g\}) \subseteq E^{J-}$ if $g \in E^{J-}$. E^{J+} and E^{J-} are the partitions of the bipartite conflict graph H by Proposition 1 ii), so $\Phi(\{g\})$ is stable.

ii) T_g is stable in H. Assume the contrary, so there exists some pair $e, f \in T_g$ so that $\{e, f\} \in U$. By i) $\Phi(\{g\})$ is stable in H, and as T^S is a feasible insertion hence it is stable in H by Theorem 1, then $T^S \setminus [\Phi(\{g\})]$ is stable in H. Therefore we can assume that $e \in \Phi(\{g\})$ and $f \in T^S \setminus [\Phi(\{g\})]$. But by definition $e \rightsquigarrow \overline{f}$, so $\overline{f} \in \Phi(\{g\})$, contradicting $f \in T^S \setminus [\Phi(\{g\})]$.

iii) As T^S is complete and by construction of $\Phi(\{g\})$ and $T^S \setminus [\Phi(\{g\})]$, $\{e, \overline{e}\} \cap T_g \neq \emptyset$ for all pairs $\{e, \overline{e}\} \in \mathscr{E}^J$ implying completness of T_g. □

7.2 Detailed results

The detailed results of the JILS are recorded in Table 4 to 10. All values are rounded to the nearest integer.

Table 4: Objective values obtained with JILS for the JS with makespan objective in the Taillard benchmark instances. For each instance, the four columns display the average results (over the five runs) after 60, 300, and 1800 seconds and the best results after 1800 seconds. The instances are grouped according to their size $n \times m$, where n is the number of jobs and m the number of machines.

objective	makespan					makespan			
time	60	300	1800	best		60	300	1800	best
15×15									
ta01	1239	1233	1232	1231	ta26	1671	1661	1659	1654
ta02	1256	1248	1245	1244	ta27	1711	1697	1694	1687
ta03	1225	1224	1223	1221	ta28	1633	1626	1624	1620
ta04	1182	1181	1177	1175	ta29	1643	1637	1633	1629
ta05	1234	1233	1232	1231	ta30	1623	1608	1603	1590
ta06	1246	1244	1242	1240	30×15				
ta07	1228	1228	1228	1228	ta31	1788	1767	1766	1764
ta08	1224	1219	1218	1217	ta32	1861	1832	1829	1821
ta09	1287	1286	1283	1281	ta33	1858	1823	1818	1808
ta10	1262	1249	1245	1244	ta34	1881	1845	1843	1837
20×15					ta35	2007	2007	2007	2007
ta11	1387	1377	1375	1369	ta36	1849	1834	1828	1820
ta12	1381	1377	1375	1373	ta37	1830	1803	1800	1795
ta13	1364	1355	1355	1352	ta38	1715	1693	1688	1685
ta14	1352	1346	1345	1345	ta39	1816	1806	1805	1797
ta15	1366	1357	1353	1343	ta40	1730	1707	1706	1697
ta16	1378	1371	1369	1362	30×20				
ta17	1489	1483	1476	1470	ta41	2110	2068	2060	2053
ta18	1429	1418	1418	1412	ta42	2033	1990	1982	1974
ta19	1358	1349	1345	1344	ta43	1943	1900	1897	1889
ta20	1376	1367	1363	1356	ta44	2061	2029	2017	2006
20×20					ta45	2068	2031	2020	2007
ta21	1673	1664	1660	1657	ta46	2108	2072	2069	2060
ta22	1636	1624	1619	1609	ta47	1983	1952	1942	1932
ta23	1579	1570	1566	1560	ta48	2047	1993	1987	1975
ta24	1671	1663	1661	1658	ta49	2054	2018	2011	2000
ta25	1630	1615	1604	1599	ta50	2021	1977	1966	1947

Table 5: Objective values obtained with JILS for the JS with makespan, total flow time, and total squared flow time objective. Results for the total squared flow time are provided in units of 1000.

objective	makespan				total flow time				total squared flow time			
time	60	300	1800	best	60	300	1800	best	60	300	1800	best
10×5												
la01	666	666	666	666	4834	4832	4832	4832	2602	2599	2599	2599
la02	655	655	655	655	4481	4468	4463	4459	2298	2297	2296	2296
la03	597	597	597	597	4152	4151	4151	4151	1956	1956	1956	1956
la04	590	590	590	590	4259	4259	4259	4259	2060	2060	2060	2060
la05	593	593	593	593	4074	4072	4072	4072	1862	1862	1862	1862
15×5												
la06	926	926	926	926	8725	8649	8629	8626	6038	5967	5953	5941
la07	890	890	890	890	8166	8122	8081	8069	5364	5287	5252	5240
la08	863	863	863	863	8130	7989	7958	7946	5279	5216	5188	5093
la09	951	951	951	951	9215	9069	9050	9043	6669	6571	6553	6520
la10	958	958	958	958	9068	8885	8866	8818	6461	6399	6348	6309
20×5												
la11	1222	1222	1222	1222	14530	14249	14053	13936	13060	12727	12505	12441
la12	1039	1039	1039	1039	12454	12031	11870	11759	9709	9489	9420	9265
la13	1150	1150	1150	1150	13945	13564	13410	13358	11864	11544	11350	11254
la14	1292	1292	1292	1292	15072	14781	14644	14622	13980	13783	13596	13274
la15	1207	1207	1207	1207	15007	14544	14316	14216	13368	13033	12884	12754
10×10												
la16	945	945	945	945	7383	7376	7376	7376	5771	5771	5771	5771
la17	784	784	784	784	6565	6547	6537	6537	4594	4565	4557	4556
la18	848	848	848	848	6993	6971	6970	6970	5153	5132	5127	5117
la19	842	842	842	842	7248	7227	7217	7217	5351	5335	5335	5335
la20	902	902	902	902	7403	7371	7354	7345	5840	5808	5792	5764
orb01	1061	1059	1059	1059	8187	8093	8044	8023	6980	6881	6858	6856
orb02	888	888	888	888	7344	7333	7316	7308	5616	5611	5606	5606
orb03	1005	1005	1005	1005	8202	8085	8056	8032	7072	6898	6865	6865
orb04	1005	1005	1005	1005	7897	7893	7893	7893	6744	6744	6744	6744
orb05	889	889	888	887	6989	6983	6983	6983	5153	5148	5141	5135
orb06	1012	1010	1010	1010	8179	8137	8130	8127	7065	7003	6984	6982
orb07	397	397	397	397	3293	3259	3257	3257	1146	1111	1108	1108
orb08	899	899	899	899	6981	6951	6944	6931	5276	5234	5209	5198
orb09	934	934	934	934	7452	7433	7433	7433	6068	6040	6016	6008
orb10	944	944	944	944	7905	7850	7846	7846	6461	6365	6364	6364
15×10												
la21	1051	1048	1047	1046	13024	12473	12353	12208	11613	11265	11207	11010
la22	932	932	929	927	12457	11919	11825	11720	10291	9955	9798	9711
la23	1032	1032	1032	1032	13063	12822	12692	12650	11606	11329	11210	11112
la24	943	940	938	935	12686	12137	12058	11936	10460	10279	10153	9846
la25	978	977	977	977	12383	11952	11868	11818	10385	10076	9911	9802
20×10												
la26	1218	1218	1218	1218	23842	21428	20271	19614	21671	20942	20477	19838
la27	1240	1237	1235	1235	25710	22158	20232	19939	22908	22498	21674	21517
la28	1216	1216	1216	1216	24534	22262	20425	20000	21573	21106	20696	20237
la29	1173	1170	1166	1164	22521	20641	18657	17992	19357	18533	18006	17731
la30	1355	1355	1355	1355	23328	21020	19486	19249	22579	21839	21014	20784
30×10												
la31	1784	1784	1784	1784	57420	54393	49893	45168	55463	54369	53638	51998
la32	1850	1850	1850	1850	64989	60314	55884	51938	62797	61517	60394	58639
la33	1719	1719	1719	1719	61882	56276	49519	45454	51619	50801	49765	48622
la34	1721	1721	1721	1721	63195	58745	54917	49862	55660	54501	53440	52042
la35	1888	1888	1888	1888	60661	57369	51645	46655	57721	56895	56239	55458
15×15												
la36	1272	1268	1268	1268	18871	17428	16806	16620	20474	19873	19607	19331
la37	1410	1405	1403	1401	19467	18404	17765	17670	22479	21776	21647	21321
la38	1201	1199	1196	1196	18238	16524	16040	15717	17309	16899	16669	16233
la39	1234	1234	1234	1233	18345	16589	16198	15922	18193	17866	17460	16985
la40	1229	1227	1226	1226	18712	16995	16466	16069	18378	17886	17537	17234

Table 6: Objective values obtained with JILS for the JS with total tardiness and total weighted tardiness objective.

objective	total tardiness				total weighted tardiness			
time	60	300	1800	best	60	300	1800	best
10×5								
la01	1194	1194	1194	1194	2299	2299	2299	2299
la02	1076	1071	1065	1065	1762	1762	1762	1762
la03	1076	1076	1076	1076	1951	1951	1951	1951
la04	1096	1096	1096	1096	1917	1917	1917	1917
la05	1165	1164	1164	1164	1878	1878	1878	1878
15×5								
la06	3541	3495	3491	3488	6026	5883	5813	5810
la07	3359	3309	3282	3272	5952	5855	5769	5765
la08	3105	3083	3056	3027	5532	5475	5475	5475
la09	3699	3598	3567	3567	5811	5608	5608	5608
la10	3805	3688	3655	3604	6795	6687	6618	6618
20×5								
la11	7301	7136	7032	6929	12474	12355	12043	11885
la12	5801	5750	5702	5668	11009	10862	10737	10542
la13	6756	6692	6659	6618	11878	11652	11539	11475
la14	7838	7771	7758	7675	13453	13345	13244	13107
la15	7306	7166	7109	7089	13016	12668	12595	12354
10×10								
la16	616	612	612	612	1169	1169	1169	1169
la17	711	694	694	694	927	899	899	899
la18	484	484	484	484	935	929	929	929
la19	486	486	486	486	973	949	948	948
la20	555	546	538	536	831	820	805	805
orb01	1414	1262	1246	1246	2625	2578	2568	2568
orb02	707	693	676	656	1423	1416	1408	1408
orb03	1381	1320	1271	1271	2186	2112	2111	2111
orb04	829	823	823	823	1635	1623	1623	1623
orb05	825	809	806	806	1742	1688	1667	1667
orb06	982	973	959	959	1821	1790	1790	1790
orb07	286	284	284	284	591	590	590	590
orb08	1240	1237	1216	1211	2534	2466	2436	2429
orb09	857	856	844	844	1316	1316	1316	1316
orb10	891	863	863	863	1795	1748	1709	1679
15×10								
la21	2263	2163	2120	2098	3974	3670	3608	3560
la22	2599	2448	2392	2264	4909	4638	4579	4356
la23	2522	2359	2267	2244	4133	4073	3922	3777
la24	2356	2262	2148	2113	3945	3804	3759	3579
la25	2297	2186	2162	2092	3862	3586	3527	3313
20×10								
la26	6082	5766	5540	5417	10756	10607	10074	9803
la27	6023	5927	5658	5480	10158	9725	9570	9226
la28	6088	5888	5566	5302	10572	10208	9810	9606
la29	5412	5132	4867	4637	10577	10377	10041	9556
la30	5672	5509	5243	5150	9864	9594	9268	8981
30×10								
la31	17329	17065	16521	16158	31072	30809	30200	29039
la32	18077	17861	17265	16846	33380	32851	32544	32258
la33	16440	16253	15746	15554	28996	28441	28098	26931
la34	16576	16323	16010	15465	30128	29461	28806	28352
la35	17481	17214	16636	16397	32288	31945	31259	30730
15×15								
la36	1988	1842	1797	1792	3578	3286	3196	2981
la37	2128	1884	1753	1565	3053	2726	2608	2290
la38	1484	1348	1278	1101	2795	2496	2400	2159
la39	1200	1040	971	907	2341	1912	1820	1676
la40	1530	1318	1148	1107	2811	2561	2347	2159

Table 7: Objective values obtained with JILS for the JSS with makespan, total flow time, and total squared flow time objective. Results for the total squared flow time are provided in units of 1000.

objective	makespan				total flow time				total squared flow time			
time	60	300	1800	best	60	300	1800	best	60	300	1800	best
10×5												
t2-ps01	798	798	798	798	6057	6053	6050	6050	4083	4060	4053	4053
t2-ps02	784	784	784	784	5509	5505	5505	5505	3479	3479	3479	3479
t2-ps03	749	749	749	749	5113	5092	5092	5092	2988	2973	2973	2973
t2-ps04	732	730	730	730	5521	5485	5478	5478	3406	3375	3375	3375
t2-ps05	691	691	691	691	5050	5041	5041	5041	2885	2876	2874	2874
15×5												
t2-ps06	1026	1026	1026	1025	11017	10773	10706	10608	9035	8870	8685	8610
t2-ps07	970	970	970	970	10332	10187	10144	10039	8188	7982	7912	7851
t2-ps08	975	967	965	963	10428	10255	10193	10140	8403	8121	8019	7816
t2-ps09	1060	1060	1060	1060	11303	11045	10879	10835	9936	9503	9346	9329
t2-ps10	1018	1018	1018	1018	11040	10924	10828	10785	9311	9027	8950	8800
20×5												
t2-ps11	1506	1504	1489	1460	21185	20506	20180	19666	28742	27633	25810	24650
t2-ps12	1334	1333	1324	1319	19682	19176	18783	18534	22397	22069	21055	20285
t2-ps13	1437	1437	1434	1430	21315	21044	20666	20162	26512	25750	24805	24011
t2-ps14	1489	1489	1480	1469	21738	21378	21015	20595	28982	27816	26097	24860
t2-ps15	1531	1526	1524	1518	21977	21438	21065	20657	29371	27908	26871	26338
15×10												
t2-LA21sdst	1279	1273	1273	1273	15529	15233	15116	15009	17543	17137	16993	16644
t2-LA24sdst	1163	1157	1155	1154	15317	14899	14780	14630	16538	15854	15658	15380
t2-LA25sdst	1198	1192	1188	1184	14964	14693	14590	14441	16079	15713	15485	15234
20×10												
t2-LA27sdst	1826	1812	1791	1767	30471	30005	29445	28705	52949	51839	50598	47885
t2-LA29sdst	1720	1698	1691	1678	28663	28384	27571	27231	46925	46267	44236	42311
15×15												
t2-LA38sdst	1471	1463	1456	1447	19283	18853	18733	18521	25947	25412	25024	24265
t2-LA40sdst	1499	1492	1488	1480	20051	19615	19458	19267	28114	26915	26464	26002
20×15												
t2-ABZ7sdst	1359	1305	1299	1287	23536	23376	22916	22446	30374	29786	29566	28321
t2-ABZ8sdst	1367	1343	1329	1307	23754	23500	23173	22630	31503	31185	30423	28890
t2-ABZ9sdst	1343	1306	1296	1286	23366	23176	22753	22388	30955	30445	30331	29121

Table 8: Objective values obtained with JILS for the JSS with total tardiness and total weighted tardiness objective.

objective	total tardiness				total weighted tardiness			
time	60	300	1800	best	60	300	1800	best
10×5								
t2-ps01	1659	1631	1623	1623	2868	2852	2852	2852
t2-ps02	1395	1395	1395	1395	2312	2301	2301	2301
t2-ps03	1376	1361	1361	1361	2689	2677	2677	2677
t2-ps04	1594	1583	1583	1583	2538	2538	2538	2538
t2-ps05	1546	1545	1545	1545	2636	2602	2570	2570
15×5								
t2-ps06	4531	4441	4379	4361	7727	7493	7359	7359
t2-ps07	4406	4262	4180	4143	7587	7418	7407	7358
t2-ps08	4378	4164	4083	4062	8153	7823	7627	7524
t2-ps09	4391	4215	4155	4147	7619	7411	7314	7303
t2-ps10	4602	4473	4305	4237	8330	7975	7872	7837
20×5								
t2-ps11	12884	12446	11947	11561	22873	22506	22110	21365
t2-ps12	12378	11907	11510	10918	21935	21674	21171	20937
t2-ps13	12766	12565	12224	11904	22068	21684	21364	20929
t2-ps14	13197	12937	12511	12365	24423	23776	23140	22446
t2-ps15	12977	12586	12179	12023	24099	23200	22721	22127
15×10								
t2-LA21sdst	2961	2772	2674	2609	4851	4597	4454	4347
t2-LA24sdst	2911	2661	2490	2300	5457	5114	4920	4492
t2-LA25sdst	3181	2918	2831	2763	5645	4880	4683	4454
20×10								
t2-LA27sdst	12939	12566	12022	11592	23105	22605	21631	20368
t2-LA29sdst	12723	12449	11648	11360	23395	22973	22149	21493
15×15								
t2-LA38sdst	1925	1689	1445	1282	3642	3118	2632	2417
t2-LA40sdst	2421	2100	1828	1637	4651	3844	2912	2137
20×15								
t2-ABZ7sdst	12019	11716	11175	10804	20452	20210	19336	18378
t2-ABZ8sdst	11613	11293	10818	10331	20453	19986	19168	18759
t2-ABZ9sdst	11291	11013	10664	10473	19877	19470	19091	18710

Table 9: Objective values obtained with JILS for the BJS with makespan, total flow time, and total squared flow time objective. Results for the total squared flow time are provided in units of 1000.

objective	makespan				total flow time				total squared flow time			
time	60	300	1800	best	60	300	1800	best	60	300	1800	best
10×5												
la01	793	793	793	793	5174	5152	5152	5152	3123	3123	3123	3123
la02	793	793	793	793	4937	4937	4937	4937	2982	2982	2982	2982
la03	715	715	715	715	4605	4605	4605	4605	2574	2574	2574	2574
la04	743	743	743	743	4781	4773	4764	4764	2663	2663	2663	2663
la05	664	664	664	664	4447	4447	4447	4447	2309	2309	2309	2309
15×5												
la06	1099	1085	1066	1060	10035	9860	9715	9683	8113	7963	7769	7689
la07	1054	1038	1025	1016	9349	9264	9136	9083	7141	6920	6715	6684
la08	1091	1064	1044	1040	9338	9138	9042	9042	7503	7091	6722	6604
la09	1181	1167	1145	1141	10590	10312	10237	10185	9273	8958	8690	8613
la10	1143	1120	1111	1096	9966	9840	9758	9735	8376	8166	7986	7920
20×5												
la11	1497	1478	1461	1446	16778	16258	16073	15787	17973	17811	17146	16444
la12	1295	1273	1254	1239	14242	13897	13634	13464	13808	13292	12843	12098
la13	1461	1428	1399	1361	16083	15791	15379	15243	17090	16486	16189	15912
la14	1512	1498	1473	1464	16869	16569	16295	16180	18376	18247	17796	17521
la15	1533	1512	1492	1477	17072	16772	16575	16315	20066	18935	18517	18264
10×10												
la16	1078	1066	1060	1060	8222	8166	8123	8109	7378	7218	7187	7187
la17	932	930	929	929	7385	7224	7159	7159	5677	5581	5570	5529
la18	1039	1033	1025	1025	7744	7652	7525	7525	6455	6366	6350	6350
la19	1065	1049	1045	1043	8216	8165	8086	8030	7204	7009	6919	6919
la20	1089	1067	1060	1060	8132	8020	7978	7951	7125	7062	7054	7054
15×10												
la21	1486	1467	1425	1396	15312	14914	14665	14483	18985	16899	16601	15967
la22	1340	1328	1309	1293	14411	14117	13946	13479	15612	15110	14697	14490
la23	1481	1451	1426	1417	15682	15494	15229	15108	18426	17706	17268	16809
la24	1410	1390	1378	1366	15038	14698	14406	14209	17441	16124	15640	15107
la25	1413	1380	1330	1311	14466	14244	13937	13797	16543	15615	14918	14396
20×10												
la26	1957	1908	1860	1848	25374	25124	24152	23437	38682	36714	35168	34597
la27	2031	1971	1934	1920	26045	25587	24884	24444	40974	39169	37271	36006
la28	1969	1926	1842	1829	25577	25208	24624	24023	39600	37149	36199	35492
la29	1799	1789	1749	1721	23578	23277	22398	22348	32782	30903	30211	29731
la30	1980	1927	1888	1869	24637	24292	23634	23190	37990	35864	35607	34754
30×10												
la31	2841	2756	2694	2677	50773	49859	49444	48552	106731	104192	99488	96030
la32	3077	2979	2911	2860	55042	54271	53200	52092	126460	121433	116269	110294
la33	2747	2682	2615	2590	49697	49646	49163	47260	102757	98181	94674	91772
la34	2852	2773	2652	2619	51516	51301	50931	50251	111042	106008	101361	96075
la35	2887	2777	2717	2682	52149	51629	51095	50573	109471	103423	99284	97650
15×15												
la36	1731	1719	1670	1639	19768	19588	19406	19059	28934	27923	27511	26621
la37	1871	1835	1804	1761	21588	21175	20790	20537	34209	31780	30889	29660
la38	1680	1635	1607	1593	18992	18854	18381	18199	25288	24826	23972	22757
la39	1749	1723	1659	1634	19415	19216	18889	18602	28135	26616	25605	24938
la40	1749	1699	1645	1609	19527	19300	19006	18810	28666	28068	26502	24132

Table 10: Objective values obtained with JILS for the BJS with total tardiness and total weighted tardiness objective.

objective	total tardiness				total weighted tardiness			
time	60	300	1800	best	60	300	1800	best
10×5								
la01	757	743	743	743	1486	1486	1486	1486
la02	705	700	700	700	1251	1251	1251	1251
la03	736	736	736	736	1461	1461	1461	1461
la04	720	716	716	716	1240	1236	1236	1236
la05	796	796	796	796	1245	1244	1244	1244
15×5								
la06	3349	3192	3075	2994	5723	5569	5469	5339
la07	3088	3034	2946	2929	4920	4809	4727	4727
la08	3040	2878	2831	2813	4991	4956	4951	4949
la09	3379	3239	3127	3127	5800	5704	5685	5685
la10	3144	3060	2994	2984	5996	5950	5785	5514
20×5								
la11	7289	7169	6873	6572	12943	12733	12077	11627
la12	5980	5909	5557	5417	11304	10618	10132	9789
la13	7300	6898	6566	6317	12650	11992	11483	11310
la14	7660	7533	7182	7089	12869	12486	12245	12129
la15	7739	7464	7109	6985	13550	12749	12321	12223
10×10								
la16	63	25	22	20	96	56	44	40
la17	153	114	104	98	274	215	144	132
la18	94	53	11	3	91	71	24	6
la19	179	103	63	43	364	229	169	86
la20	36	0	0	0	42	4	0	0
15×10								
la21	2181	2066	1856	1764	3639	3123	2787	2688
la22	2616	2360	2045	1862	4061	3798	3511	3318
la23	2701	2457	2200	2120	4942	4128	3642	3537
la24	2344	2227	1975	1882	4265	3915	3398	3268
la25	2129	2038	1766	1655	3742	3162	2738	2598
20×10								
la26	8053	7117	6377	6163	13191	11945	11180	10260
la27	8533	7447	6584	6347	13281	12790	11453	10769
la28	7418	6718	6340	6217	12965	12169	10973	9621
la29	6898	5821	5496	5246	12067	11367	9746	9315
la30	6453	5856	5337	5224	10617	9783	8770	7379
30×10								
la31	28484	25177	23913	21857	41804	41747	41090	39088
la32	38955	32441	27292	25136	45338	44723	44251	42510
la33	37945	35852	32246	28101	40062	39894	39164	36088
la34	43523	41059	36426	33874	42488	41882	41381	40125
la35	27341	25163	24507	23735	42060	41296	40418	38181
15×15								
la36	1556	986	717	602	2302	1754	1338	1078
la37	1744	1184	932	822	2555	1978	1664	1377
la38	1451	1069	903	726	1984	1639	1099	726
la39	1467	813	561	405	1383	1075	786	630
la40	3458	1211	983	943	2195	1768	1204	1113

References

Applegate D, Cook W (1991) A computational study of the job-shop scheduling problem. ORSA Journal on Computing 3(2): 149–156

Balas E, Simonetti N, Vazacopoulos A (2008) Job shop scheduling with setup times, deadlines and precedence constraints. Journal of Scheduling 11(4):253–262

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

Brucker P, Knust S (2011) Complex Scheduling, 2nd edn. Springer

Brucker P, Thiele O (1996) A branch & bound method for the general-shop problem with sequence dependent setup-times. OR Spectrum 18(3):145–161

Brucker P, Jurisch B, Sievers B (1994) A branch and bound algorithm for the job-shop scheduling problem. Discrete Applied Mathematics 49(1):107–127

Bülbül K, Kaminsky P (2013) A linear programming-based method for job shop scheduling. Journal of Scheduling 16(2):161–183

Bürgy R (2014) Complex Job Shop Scheduling: A General Model and Method. PhD thesis, Department of Informatics, University of Fribourg

Bürgy R, Gröflin H (2016) The blocking job shop with rail-bound transportation. Journal of Combinatorial Optimization 31(1): 151–181

Eilon S, Hodgson R (1967) Job shops scheduling with due dates. International Journal of Production Research 6(1):1–13

Glover FW, Laguna M (1997) Tabu search. Kluwer

González MA, Vela CR, Sierra M, Varela R (2010) Tabu search and genetic algorithm for scheduling with total flow time minimization. In: COPLAS 2010: ICAPS Workshop on Constraint Satisfaction Techniques for Planning and Scheduling Problems, 33–41

González MÁ, González-Rodríguez I, Vela CR, Varela R (2012a) An efficient hybrid evolutionary algorithm for scheduling with setup times and weighted tardiness minimization. Soft Computing 16(12):2097–2113

González MA, Vela CR, Varela R (2012b) A competent memetic algorithm for complex scheduling. Natural Computing 11(1): 151–160

Graham RL, Lawler EL, Lenstra JK, Rinnooy Kan AHG (1979) Optimization and approximation in deterministic sequencing and scheduling: a survey. Annals of Discrete Mathematics 5:287–326

Grimes D, Hebrard E (2015) Solving variants of the job shop scheduling problem through conflict-directed search. INFORMS Journal on Computing 27(2):268–284

Gröflin H, Klinkert A (2007) Feasible insertions in job shop scheduling, short cycles and stable sets. European Journal of Operational Research 177(2):763–785

Gröflin H, Klinkert A (2009) A new neighborhood and tabu search for the blocking job shop. Discrete Applied Mathematics 157(17):3643–3655

Gröflin H, Pham DN, Bürgy R (2011) The flexible blocking job shop with transfer and set-up times. Journal of Combinatorial Optimization 22(2):121–144

Lawrence S (1984) Supplement to resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques. GSIA, Carnegie Mellon University, Pittsburgh, PA

Mascis A, Pacciarelli D (2002) Job-shop scheduling with blocking and no-wait constraints. European Journal of Operational Research 143(3):498–517

Mati Y, Dauzère-Pérès S, Lahlou C (2011) A general approach for optimizing regular criteria in the job-shop scheduling problem. European Journal of Operational Research 212(1):33–42

Nowicki E, Smutnicki C (1996) A fast taboo search algorithm for the job shop problem. Management Science 42(6):797–813

Nowicki E, Smutnicki C (2005) An advanced tabu search algorithm for the job shop problem. Journal of Scheduling 8(2):145–159

Oddi A, Rasconi R, Cesta A, Smith SF (2012) Iterative improvement algorithms for the blocking job shop. In: Twenty-Second International Conference on Automated Planning and Scheduling

Perregaard M, Clausen J (1998) Parallel branch-and-bound methods for the job-shop scheduling problem. Annals of Operations Research 83(0):137–160

Pinedo ML (2012) Scheduling: Theory, Algorithms, and Systems, 4th edn. Springer

Potts CN, Strusevich VA (2009) Fifty years of scheduling: a survey of milestones. Journal of the Operational Research Society 60:S41–S68

Pranzo M, Pacciarelli D (2015) An iterated greedy metaheuristic for the blocking job shop scheduling problem. Journal of Heuristics

Taillard E (1994) Parallel taboo search techniques for the job shop scheduling problem. ORSA Journal on Computing 6:108

Tufte ER (2001) The Visual Display of Quantitative Information, 2nd edn. Bertrams

Vela C, Varela R, González M (2010) Local search and genetic algorithm for the job shop scheduling problem with sequence dependent setup times. Journal of Heuristics 16:139–165

Zhang CY, Li P, Rao Y, Guan Z (2008) A very fast TS/SA algorithm for the job shop scheduling problem. Computers and Operations Research 35(1):282–294