J Comb Optim (2011) 22:121–144
DOI 10.1007/s10878-009-9278-x

The flexible blocking job shop with transfer and set-up times

Heinz Gröflin · Dinh Nguyen Pham · Reinhard Bürgy

Published online: 12 November 2009

Abstract The Flexible Blocking Job Shop (FBJS) considered here is a job shop scheduling problem characterized by the availability of alternative machines for each operation and the absence of buffers. The latter implies that a job, after completing an operation, has to remain on the machine until its next operation starts. Additional features are sequence-dependent transfer and set-up times, the first for passing a job from a machine to the next, the second for change-over on a machine from an operation to the next. The objective is to assign machines and schedule the operations in order to minimize the makespan. We give a problem formulation in a disjunctive graph and develop a heuristic local search approach. A feasible neighborhood is constructed, where typically a critical operation is moved (keeping or changing its machine) together with some other operations whose moves are "implied". For this purpose, we develop the theoretical framework of job insertion with local flexibility, based on earlier work of Gröflin and Klinkert on insertion. A tabu search that consistently generates feasible neighbor solutions is then proposed and tested on a larger test set. Numerical results support the validity of our approach and establish first benchmarks for the FBJS.

Keywords Job shop scheduling · Flexible machines · Blocking · Setup · Disjunctive graph · Tabu search

H. Gröflin (✉) · R. Bürgy
Department of Informatics, University of Fribourg, Boulevard de Pérolles 90, 1700 Fribourg, Switzerland
e-mail: heinz.groeflin@unifr.ch

D.N. Pham
FortisBC Inc, Kelown, BC V1Y 7V7, Canada

1 Introduction

Job shop scheduling problems in practice often display features that are not addressed in the "classical" Job Shop problem. Among such features are limited buffer capacity between machines or even its absence, forcing a job to wait on a machine until the machine for the job's next operation becomes free. This latter case is commonly referred to as the Blocking Job Shop. Another feature is the option to choose the machine for each operation from an (operation-dependent) set of machines. This flexibility results in scheduling problems where not only a starting time, but also an assigned machine has to be determined for each operation.

In this paper, we study a job shop problem possessing both features, which we call the Flexible Blocking Job Shop or FBJS. Additional features of the FBJS will be to allow for transfer steps passing a job from a machine to the next, and sequence-dependent set-up times between two consecutive operations on a machine. Both features are not included in (what one defines as) a classical Blocking Job Shop, but turn out to be useful in practice. Indeed, a job, after completion of an operation on a machine has to be handed over to a transport processor or to the machine working on the job's next operation. Such steps have to be taken into account if transfer times are not negligible. The option of set-up times is also valuable, for instance when processors are mobile and have to execute an "idle" move between two consecutive operations.

A frequent objective in scheduling problems is to minimize the makespan, i.e. the time by which all operations have been completed. Another, somewhat more general objective is to minimize maximum lateness, i.e. the maximum delay (if any) by which a job finishes after its due date. (Note that with all due dates equal to zero, the second objective becomes the first.) We shall present a solution approach for the FBJS with the more common makespan minimization objective. Maximum lateness would require only minimal adaptations.

Literature related to the FBJS is mainly dedicated to the non-flexible Blocking Job Shop. Indeed, we are not aware of previous literature on the blocking job shop with flexibility. Blocking constraints together with flexibility have been addressed in the simpler flow shop version of the job shop (Thornton and Hunsucker 2004).

The non-flexible Blocking Job Shop (BJS) has found increasing attention over the last years. In Mascis and Pacciarelli (2000) and Mascis and Pacciarelli (2002) several scheduling problems, among them the BJS, are formulated with the help of alternative graphs and solved with dispatching heuristics. Klinkert (2001) studies the scheduling of pallet moves in automated high-density warehouses, proposes a generalized disjunctive graph framework similar to the alternative graphs and devises a local search heuristic with feasible neighborhood for a BJS with transfer and set-up times. Meloni et al. (2004) develop a "rollout" metaheuristic which they apply among other problems also to the BJS. Brizuela et al. (2001) propose a genetic algorithm for the BJS. Brucker and Kampmeyer (2008) present tabu search algorithms for cyclic scheduling in the BJS. Gröflin and Klinkert (2007) study a general insertion problem with, among others, an application to job insertion in the BJS. They also present in Gröflin and Klinkert (2009) a tabu search for the BJS with transfer and set-up times. Job shop scheduling problems with limited buffer capacity (extending in a sense the BJS where buffer capacity is zero) are studied in Brucker et al. (2006) and Heitmann

(2007). Various ways in which buffers occur are analyzed and tabu search approaches for the flow shop and job shop problems are proposed for specific buffer configurations, including the BJS.

The paper is organized as follows. The next section describes formally the FBJS and gives a problem formulation in a disjunctive graph which forms the basis for the subsequent development. Additionally, it provides a mixed-integer linear programming formulation. Section 3 is devoted to the design of a feasible neighborhood of a given solution. A structural framework—the so-called job insertion with local flexibility—is introduced first, building on previous work of the authors. Based on this framework, feasible neighbors of a given solution are generated by moving an operation of a job, together with implied moves of other operations of the job. Section 4 is devoted to a tabu search which makes uses of the neighborhood derived in the previous section. Computational results are presented in Sect. 5.

Graphs are needed for the formulation of the FBJS and the derivation of the neighborhood. They will be directed and the following standard notation will be used. An arc $e = (v, w)$ has a *tail* (node v), and a *head* (node w), denoted by $t(e)$ and $h(e)$ respectively. Also, given a graph $G = (V, E)$, for any $N \subseteq V$, $\gamma(N) = \{e \in E : t(e)$ and $h(e) \in N\}$, $\delta^-(N) = \{e \in E : t(e) \notin N$ and $h(e) \in N\}$, $\delta^+(N) = \{e \in E : t(e) \in N$ and $h(e) \notin N\}$ and $\delta(N) = \delta^-(N) \cup \delta^+(N)$. These sets are defined in G, we abstain however from a heavier notation, e.g. $\delta^+_G(N)$ for $\delta^+(N)$. It will be clear from the context which underlying graph is meant. Finally, in a graph $G = (V, E, d)$ with arc valuation $d \in R^E$, a path or a cycle in G will be called *positive* if its length is positive.

2 The flexible blocking job shop

2.1 Formulation, notation and data

The FBJS can be described as follows. Let I be a set of operations $i \in I$ and $\mathcal{J} \subseteq 2^I$ a set of jobs such that $\mathcal{J}$ forms a partition of I, i.e. a job $J \in \mathcal{J}$ is a set of operations $\{i : i \in J\}$ and any operation $i \in I$ is in exactly one job $J \in \mathcal{J}$. We assume that the set of operations of a job is ordered in a sequence and denote sometimes $\{i : i \in J\}$ as the ordered set $\{J_1, J_2, \ldots, J_{|J|}\}$, J_r denoting the r-th operation of job J. Two operations i, j of job J are consecutive if $i = J_r$ and $j = J_{r+1}$ for some r, $1 \leq r < |J|$. Furthermore, let M be a set of machines. Each operation $i \in I$ needs one machine, say m, for its execution. This machine m is not fixed, and there is flexibility in choosing it from some subset of alternative machines $M_i \subseteq M$.

An important process feature arises from the assumption that there are no buffers between machines. Then a job J, after having its operation J_r executed on machine m, might have to wait on m, thus *blocking* m, until the machine for its next operation J_{r+1} becomes available.

Specifically, consider operation $j = J_r$ of job J and assume that j is processed on $m \in M_j$, its job predecessor $i = J_{r-1}$ is processed on $p \in M_i$ and its job successor $k = J_{r+1}$ is processed on $q \in M_k$. Operation j can be viewed as consisting of the following successive steps: (i) a take-over step where, after completion of i, the job is taken over from machine p to machine m; (ii) a processing step, (iii) a possible waiting time of the job on m; (iv) a hand-over step where, after completion of j, the job is

handed over from machine m to machine q for its next operation k. The durations of these steps—except (iii)—are input data. Let $d(i, p; j, m)$, $d(j, m)$ and $d(j, m; k, q)$ denote the duration of the take-over step, the processing step and the hand-over step respectively. Due to blocking, the take-over step of operation j must occur simultaneously with the hand-over step of its predecessor operation i. The starting times of the two steps, as well as their duration are therefore the same.

Two remarks on the discussed steps are in order. First, if j is the first operation of a job, the take-over step might be called more appropriately a loading step of duration $d(ld; j, m)$, and similarly, if j is the last operation of a job, the hand-over step might be called an unloading step of duration $d(j, m; uld)$. Second, since no assumptions are made for the subsets M_i, $i \in I$, if $M_i \cap M_j \neq \emptyset$ for two consecutive operations i and j of some job J, J might stay on a machine $m \in M_i \cap M_j$ for both operations. Duration $d(i, m; j, m)$ might be zero, essentially combining both operations into a single operation, or positive if some change-over time, e.g. if a tool change on m is required.

We also allow for set-up times between two consecutive operations on a machine. If two operations i and j with $i \in J$, $j \in J'$ and $J \neq J'$, are executed on a machine m, and j immediately follows i on m, then there is a set-up time of duration $d(i, m; j, m)$ occurring between the hand-over step of i and the take-over step of j. Initial and final set-up times for the first and last operation of a job might also be present, of respective duration $d(\sigma; j, m)$ and $d(j, m; \tau)$ (the choice of the symbols σ and τ will become clear in the sequel).

A few standard assumptions are made. All durations are non-negative, processing times $d(j, m)$ are positive and set-up times satisfy the triangle inequality, i.e. $d(i, m; j, m) + d(j, m; k, m) \geq d(i, m; k, m)$ for any operations i, j and k from three distinct jobs on a common machine m; similarly, $d(\sigma; j, m) + d(j, m; k, m) \geq d(\sigma; k, m)$ and $d(i, m; j, m) + d(j, m; \tau) \geq d(i, m; \tau)$. Note that hand-over/take–over times as well as set-up times can have value 0, allowing also for the case where so-called swapping is permitted. (Swapping occurs when two jobs swap machines.)

The FBJS can be stated as follows. A schedule of the jobs is a specification for each operation of its chosen machine and its starting time, fulfilling the process constraints described above as well as the standard constraints that a machine can be occupied by at most one operation at a time and no preemption is allowed. The objective is to schedule the jobs so that the makespan is minimal.

2.2 A disjunctive graph formulation

Main features of the disjunctive graph will be the following. Each operation, for each of its possible machines, is represented by a chain of three arcs for the take-over, processing and hand-over steps. In a job, two consecutive operations i and j are linked by a transfer arc and arcs synchronizing the hand-over of i with the take-over of j. Finally, a pair of disjunctive arcs is present between any two operations on a same machine, joining the last node of one operation to the first node of the other operation.

Specifically, the disjunctive graph $G = (V, A, E, \mathcal{E}, d)$ for the FBJS is constructed as follows. To each operation $i \in I$ and machine $m \in M_i$ is associated a chain with

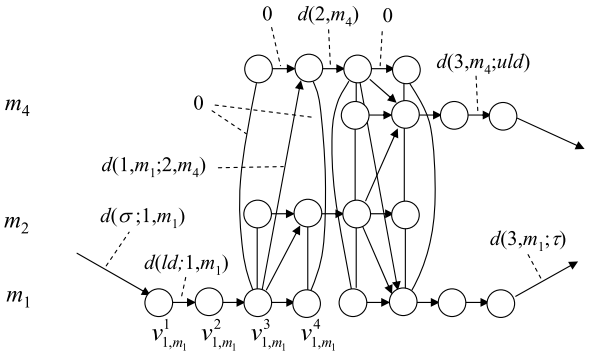

Fig. 1 A job in the disjunctive graph

node set $V_{im} = \{v^1_{im}, v^2_{im}, v^3_{im}, v^4_{im}\}$ and arcs (v^1_{im}, v^2_{im}), (v^2_{im}, v^3_{im}), (v^3_{im}, v^4_{im}). Node set V of G consists of the union of the V_{im} over all $i \in I$ and $m \in M_i$, together with two additional nodes σ and τ, representing fictive start and end operations of duration 0 to occur before, respectively after, all operations of I. Denote by I^{first} and I^{last} the subsets of all operations that are first and last operations of jobs, respectively.

The set A of conjunctive arcs of G consists of the following arcs, indicated with their weights:

1. For each $i \in I$ and $m \in M_i$, three arcs (v^1_{im}, v^2_{im}), (v^2_{im}, v^3_{im}), (v^3_{im}, v^4_{im}), with respective weights: $d(ld; i, m)$ if $i \in I^{first}$ and 0 if $i \in I - I^{first}$; $d(i, m)$; $d(i, m; uld)$ if $i \in I^{last}$ and 0 if $i \in I - I^{last}$. The three arcs are referred to as *take-over arc, processing arc and hand-over arc*.
2. For any two consecutive operations $i = J_r$ and $j = J_{r+1}$ of a job J and $m \in M_i$, $m' \in M_j$, two pairs of *synchronization arcs* $(v^3_{im}, v^1_{jm'})$, $(v^1_{jm'}, v^3_{im})$ and $(v^4_{im}, v^2_{jm'})$, $(v^2_{jm'}, v^4_{im})$ of weight 0 joining the (starts and the ends of the) hand-over step of i and take-over step of j, and a *transfer arc* $(v^3_{im}, v^2_{jm'})$ of weight $d(i, m; j, m')$. Note that $m = m'$ is possible.
3. For each $i \in I^{first}$ and $m \in M_i$, an *initial set-up arc* (σ, v^1_{im}) of weight $d(\sigma; i, m)$ and for each $i \in I^{last}$ and $m \in M_i$, a *final set-up arc* (v^4_{im}, τ) of weight $d(i, m; \tau)$.

The set E of disjunctive arcs is given as follows: for any two operations i and j with $i \in J$, $j \in J'$, $J \neq J'$ and $M_i \cap M_j \neq \emptyset$, and any $m \in M_i \cap M_j$, there are two disjunctive arcs (v^4_{im}, v^1_{jm}), (v^4_{jm}, v^1_{im}) with respective weights $d(i, m; j, m)$, $d(j, m; i, m)$. The family $\mathcal{E}$ is the collection of all *pairs* of disjunctive arcs $((v^4_{im}, v^1_{jm}), (v^4_{jm}, v^1_{im}))$, $i \in J$, $j \in J'$, $J \neq J'$ and $m \in M_i \cap M_j$. An element of $\mathcal{E}$, i.e. a pair of disjunctive arcs, will sometimes be denoted by D, and the two elements of D by e and $\overline{e}$. Arc $\overline{e}$ will be said to be the *mate* of e and vice-versa.

We illustrate how a job is represented in the disjunctive graph and introduce a small problem with four jobs that will be used in the sequel.

Figure 1 shows all conjunctive arcs incident to a job. The job has three operations, say $i = 1, 2, 3$, and corresponding sets of alternative machines $M_1 = \{m_1\}$, $M_2 = \{m_2, m_4\}$ and $M_3 = \{m_1, m_4\}$. A pair of synchronization arcs is represented by an (undirected) edge. Selected arc weights are also indicated.

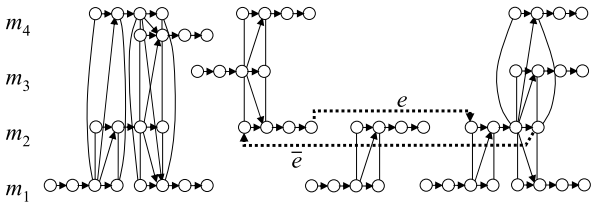

Fig. 2 Example with four jobs

An example with four jobs is sketched in Fig. 2, the job at the left being the job described above. The figure depicts the associated disjunctive graph in which, however, nodes σ and τ as well as all disjunctive arcs, except for one pair, denoted by e and $\overline{e}$, have been omitted for clarity. Also, specific numerical data will not be needed for the subsequent use of the example, it will only be assumed that transfer times are positive.

As in the classical job shop problem and its associated disjunctive graph, the FBJS can be formulated as a combinatorial problem in disjunctive graph G.

Definition 1 A mode is a tuple $\mu = (\mu_1, \mu_2, \ldots, \mu_{|I|}) \in \mathbf{M} = M_1 \times M_2 \times \cdots \times M_{|I|}$, assigning to each operation $i \in I$ a machine $\mu_i \in M_i$.

A mode μ selects a node-induced subgraph G^μ in G defined as follows. Consider node subset $V^\mu = \bigcup_{i \in I} V_{i\mu_i} \cup \{\sigma, \tau\}$ and let $A^\mu = A \cap \gamma(V^\mu)$, $E^\mu = E \cap \gamma(V^\mu)$ and $\mathcal{E}^\mu = \{D \in \mathcal{E} : D \subseteq E^\mu\}$. The resulting graph $G^\mu = (V^\mu, A^\mu, E^\mu, \mathcal{E}^\mu, d)$ is the disjunctive graph associated to mode μ. (We take the liberty of denoting the restriction of d to $A^\mu \cup E^\mu$ again by d.)

Definition 2 For any mode $\mu \in \mathbf{M}$ and set $S \subseteq E^\mu$, (μ, S) is called a selection. Selection (μ, S) is said to be positive cyclic if the graph $(V^\mu, A^\mu \cup S, d)$ contains a positive cycle, and positive acyclic otherwise. (μ, S) is said to be complete if $S \cap \{e, \overline{e}\} \neq \emptyset$ for all $\{e, \overline{e}\} \in \mathcal{E}^\mu$, and to be feasible if it is positive acyclic and complete.

The FBJS can now be formulated as the following problem: "Among all feasible selections (μ', S'), $\mu' \in \mathbf{M}$, $S' \subseteq E^{\mu'}$, find a selection (μ, S) minimizing the length of a longest path from σ to τ in subgraph $(V^\mu, A^\mu \cup S, d)$".

2.3 A mixed-integer linear programming formulation

An alternative problem statement for the FBJS is given by the following mixed-integer linear program. For each operation i, let x_i^1 and x_i^2 be the starting time and ending time of its take-over step, and x_i^3 and x_i^4 the starting time and ending time of its hand-over step. Also, for each $i \in I$ and $m \in M_i$, let y_{im} be a binary variable with value 1 if machine m is chosen for i and value 0 otherwise. Define by P^1 the set of all ordered pairs (i, j) of operations which are consecutive in a job (i preceding j) and by P^2 the set of all unordered pairs (i, j) of operations i and j belonging to distinct jobs and having potentially a common machine, i.e. with $M_i \cap M_j \neq \emptyset$. The FBJS can be formulated as follows.

Minimize x_τ

subject to

$$x_i^1 \geq d(\sigma; i, m) - H(1 - y_{im}) \quad \forall i \in I^{first}, m \in M_i$$

$$x_i^2 - x_i^1 \geq d(ld; i, m) - H(1 - y_{im}) \quad \forall i \in I^{first}, m \in M_i$$

$$x_i^2 - x_i^1 \geq 0 \quad \forall i \in I - I^{first}$$

$$x_i^3 - x_i^2 \geq d(i, m) - H(1 - y_{im}) \quad \forall i \in I, m \in M_i$$

$$x_i^4 - x_i^3 \geq d(i, m; uld) - H(1 - y_{im}) \quad \forall i \in I^{last}, m \in M_i$$

$$x_i^4 - x_i^3 \geq 0 \quad \forall i \in I - I^{last}$$

$$x_\tau - x_i^4 \geq d(i, m; \tau) - H(1 - y_{im}) \quad \forall i \in I^{last}, m \in M_i$$

$$x_i^3 = x_j^1 \quad \text{and} \quad x_i^4 = x_j^2 \quad \forall (i, j) \in P^1$$

$$x_j^2 - x_i^3 \geq d(i, m; j, m') - H(2 - y_{im} - y_{jm'}) \quad \forall (i, j) \in P^1, m \in M_i, m' \in M_j$$

$$x_j^1 - x_i^4 \geq d(i, m; j, m) - H(3 - y_{im} - y_{jm} - z_{ij}) \quad \forall (i, j) \in P^2, m \in M_i \cap M_j$$

$$x_i^1 - x_j^4 \geq d(j, m; i, m) - H(2 - y_{im} - y_{jm} + z_{ij}) \quad \forall (i, j) \in P^2, m \in M_i \cap M_j$$

$$\sum_{m \in M_i} y_{im} = 1 \quad \forall i \in I$$

$$y_{im} \in \{0, 1\} \quad \forall i \in I, m \in M_i; z_{ij} \in \{0, 1\} \; \forall (i, j) \in P^2$$

Herein, H denotes a large number. We examine briefly its role, the interpretation of the constraints becoming then easy. Looking at the first set of constraints, the term $-H(1 - y_{im})$ ensures that for each $i \in I^{first}$, the constraint $x_i^1 \geq d(\sigma; i, m)$ holds precisely when machine m is chosen for i (i.e. $y_{im} = 1$), while the inequalities for the machines not chosen (for which $y_{im} = 0$) are inactivated, i.e. hold trivially. The action of $-H(1 - y_{im})$ in the other sets of constraints is similar. The term $-H(2 - y_{im} - y_{jm'})$ activates $x_j^2 - x_i^3 \geq d(i, m; j, m')$ precisely when machine m is chosen for i and m' for j. Finally $-H(3 - y_{im} - y_{jm} - z_{ij})$ activates the disjunctive precedence constraint $x_j^1 - x_i^4 \geq d(i, m; j, m)$ exactly when machine m is chosen for both i and j, and i is before j (i.e. $z_{ij} = 1$). Its "mate" constraint $x_i^1 - x_j^4 \geq d(j, m; i, m)$ is activated if m is chosen for i and j, and i is after j (i.e. $z_{ij} = 0$). (Note that the interpretation of the variables z_{ij} (i before or after j) is only valid if i and j are on a common machine.)

We conclude this section with a remark. Clearly, the above formulation could be more compact since it includes constraints $x_i^3 = x_j^1$ and $x_i^4 = x_j^2$ for all $(i, j) \in P^1$. We refrained from variable elimination for easy readability of the formulation and its relation to the disjunctive graph.

3 A feasible neighborhood

In view of the state of the art in the simpler non-flexible Blocking Job Shop (BJS), solving the FBJS with exact methods appears only feasible for small problem sizes. For this reason, the solution proposed in this paper is of heuristic nature, namely a tabu search using a neighborhood concept based on the following idea.

Given a feasible solution, feasible neighbor solutions will be generated by re-scheduling an operation either by moving it within the sequence on the machine it was assigned to, or moving it to another machine and inserting it in the sequence of operations on that machine. However, as is the case in the BJS, moves of other operations might be necessary if feasibility is to be maintained. We first provide a structural framework for deriving a feasible neighborhood, based on previous work on the Blocking Job Shop (Gröflin and Klinkert 2009) and on a theoretical framework for insertion in a variety of scheduling problems (Gröflin and Klinkert 2007). The development of the next section can be seen as an application of this insertion theory, specializing it (to job insertion in the blocking job shop), but also extending it by allowing local flexibility. Definitions, e.g. of the insertion graph, and results are adapted accordingly.

3.1 Job insertion with local flexibility

Given a feasible selection (μ, S) in the disjunctive graph $G = (V, A, E, \mathcal{E}, d)$, let $i \in I$ be an arbitrary operation and let J be the job to which i belongs, i.e. $i \in J$. We will consider the problem of (extracting and) reinserting job J, allowing operation i to choose any machine $m \in M_i$ and keeping the machine assignments of all other operations of J. The disjunctive graph $G' = (V', A', E', \mathcal{E}', d)$ for this scheduling problem is obtained as follows. Delete from V node sets V_{jm} for all $j \neq i$ and $m \in M_j - \mu_j$, obtaining V', then add to the set of conjunctive arcs all arcs of S not incident to job J, obtaining A', and delete all disjunctive edges not incident to J, obtaining E'. Formally,

$$V' = \{\sigma, \tau\} \cup \left(\bigcup_{j \in I - i} V_{j\mu_j} \right) \cup \left(\bigcup_{m \in M_i} V_{im} \right),$$

and, defining the subset of V' associated to job J and the part of S not incident to job J by

$$V'_J = \left(\bigcup_{j \in J - i} V_{j\mu_j} \right) \cup \left(\bigcup_{m \in M_i} V_{im} \right),$$

$$R = S - \delta(V'_J),$$

the sets of conjunctive arcs, disjunctive arcs and disjunctive pairs are

$$A' = (A \cap \gamma(V')) \cup R,$$

$$E' = (E \cap \gamma(V')) \cap \delta(V'_J) \quad \text{and}$$

$$\mathcal{E}' = \{(e, \overline{e}) : e \text{ and } \overline{e} \in E'\}.$$

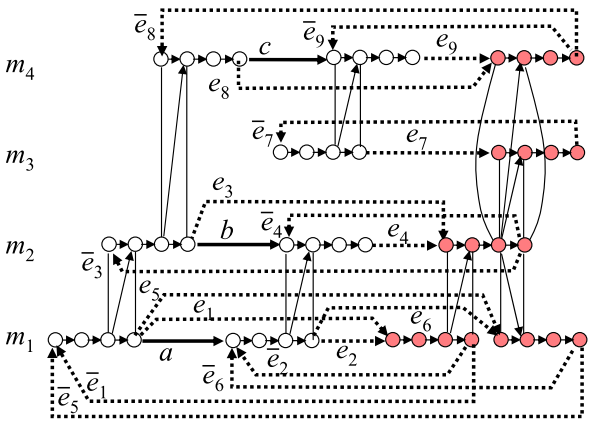

Fig. 3 Job insertion with local flexibility: insertion graph G'

G' will be called the *insertion graph of job* J with *local flexibility at* i. We illustrate its construction in the example. Identifying in Fig. 2 the four jobs from left to right by J^1 to J^4, we choose i to be the third operation of job J^4, i.e. $i = J_3^4$ and $J = J^4$. We assume the following selection (μ, S). Mode μ assigns respectively machines m_1, m_2 and m_4 to the operations J_1^1, J_2^1 and J_3^1 of job J^1; m_3 and m_4 to J_1^2 and J_2^2 ; m_1 and m_2 to J_1^3 and J_2^3, and m_1, m_2 and m_4 to J_1^4, J_2^4 and J_3^4. S is the selection of disjunctive arcs corresponding to the permutation schedule with job order J^1, J^2, J^3, J^4. Figure 3 shows graph G'. The node set V'_J of job J is the set of shaded nodes. The disjunctive arcs (set E') are drawn in dotted lines. The three arcs in bold lines represent $R = S - \delta(V'_J)$. Nodes σ and τ and arcs incident to them have been omitted.

As previously in G, we can define in G' selections which we call now *insertions* as follows. For given $m \in M_i$, let $G^m = (V^m, A^m, E^m, \mathcal{E}^m, d)$ be the subgraph obtained from G' by deleting node sets $V_{im'}$, $m' \in M_i - m$, and denote by $V_J^m = V'_J \cap V^m$ the node set associated to J.

Definition 3 For any $m \in M_i$ and $T \subseteq E^m$, (m, T) is called an insertion in G'. (m, T) is positive cyclic if the graph $(V^m, A^m \cup T, d)$ contains a positive cycle, and positive acyclic otherwise. (m, T) is complete if $T \cap \{e, \overline{e}\} \neq \emptyset$ for all $\{e, \overline{e}\} \in \mathcal{E}^m$, and feasible if it is positive acyclic and complete.

Clearly, for any $m \in M_i$ and $T \subseteq E^m$, (m, T) is a positive acyclic (complete, feasible) insertion if and only if $(\mu', T \cup R)$ is a positive acyclic (complete, feasible) selection in G where μ' is defined by $\mu'_i = m$ and $\mu'_j = \mu_j$ for all $j \in I - i$.

In particular, for $T^S = S \cap \delta(V'_J)$, (μ_i, T^S) is the feasible insertion corresponding to the given selection (μ, S). In order to construct a neighbor selection of (μ, S), we will construct a neighbor insertion of (μ_i, T^S) in G'.

For this purpose, we need to establish a key property of G', or more accurately, of its subgraphs G^m, $m \in M_i$, and introduce the concept of closure. The following partitions of E', respectively E^m, will be needed in the sequel. The first (bi)partition

splits the disjunctive arcs into arcs entering and leaving J,

$$E^- = E' \cap \delta^-(V'_J), \qquad E^+ = E' \cap \delta^+(V'_J).$$

The second partition of E' into $|J| + |M_i| - 1$ sets is induced by the operations of J,

$$E^j = E' \cap \delta(V_{j\mu_j}), \quad j \in J - i, \qquad E^{im} = E' \cap \delta(V_{im}), \quad m \in M_i.$$

For given $m \in M_i$, we show now that graph $(V^m, A^m \cup E^m, d)$, has the so-called short cycle property (Gröflin and Klinkert 2007). Let Z be (the arc set of) a positive cycle in $(V^m,\ A^m \cup E^m,\ d)$. Since (V^m, A^m, d) does not contain a positive cycle, $Z \cap E^m \neq \emptyset$. Since $E^m \subseteq \delta(V_J^m)$ and $\delta(V_J^m) - E^m$ is made up only of arcs of type (σ, v) or (v, τ) which cannot appear in any cycle, $Z \cap E^m = Z \cap \delta(V_J^m)$. Finally, $|Z \cap \delta(V_J^m)| = 2|Z \cap \delta^+(V_J^m)| = 2|Z \cap \delta^-(V_J^m)| = 2k$ for some $k \geq 1$. k can be seen as the *number of times Z visits V_J^m*. The following property holds.

Lemma 1 *For any positive cycle Z in $(V^m, A^m \cup E^m, d)$ visiting V_J^m a number $k \geq 1$ of times, there exists a positive cycle Z' with $Z' \cap E^m \subseteq Z \cap E^m$ visiting V_J^m exactly once, i.e. $Z' \cap E^m = \{e, f\}$ for some $e \in Z \cap \delta^-(V_J^m)$, $f \in Z \cap \delta^+(V_J^m)$.*

Proof Suppose $|Z \cap \delta^-(V_J^m)| = k > 1$. For any $e \in \delta^-(V_J^m)$, $e \in E^j$ for some $j \in J - i$ or $e \in E^{im}$, i.e. e is incident to some operation $j \in J$. If j is the s-th operation J_s of job J, define the *rank* of e as $r(e) = s$. Obviously, distinct arcs $e, e' \in Z \cap \delta^-(V_J^m)$ have distinct ranks. Choose from all $e \in Z \cap \delta^-(V_J^m)$ the arc, say e' of highest rank. Starting out from node $h(e') \in V_J^m$, traverse partially Z, leaving V_J^m the first time through arc $f \in \delta^+(V_J^m)$, then entering V_J^m the first time through arc $e \in Z \cap \delta^-(V_J^m)$ and stopping in node $h(e)$. By definition of e', $r(e) < r(e')$. Therefore there exists a path P_J in $(V_J^m, \gamma(V_J^m), d)$ from $h(e)$ to $h(e')$, passing through the processing arc of operation $J_{r(e)}$, hence P_J is a positive path. The traversed subpath in Z from $h(e')$ to $h(e)$ together with P_J is a closed walk of positive length entering V_J^m through e and leaving V_J^m through f, hence it contains a positive cycle Z' as stipulated. □

We define now the concept of closure in G^m, using for any $T \subseteq E^m$, the short notation $G^m(T)$ for graph $(V^m, A^m \cup T, d)$. Given an insertion (m, T), let

$$\varphi(T) = T \cup \{e \in E^m - T : G^m(T \cup \overline{e}) \text{ contains a positive cycle } Z \text{ with } Z \ni \overline{e}\},$$

and call T *closed* if $\varphi(T) = T$. It is easy to see that if T and T' are closed, $T \cap T'$ is closed, so that the following closure operator Φ is well-defined.

Definition 4 For any $T \subseteq E^m$, the closure $\Phi(T)$ of T in G^m is the smallest closed set containing T.

Obviously, $\Phi(T)$ can be computed by recursively applying φ, for instance by defining $\varphi^r(T) = T$ for $r = 0$, and applying $\varphi^r(T) = \varphi(\varphi^{r-1}(T))$ for $r > 1$. For some $r \leq |E^m|$, $\varphi^{r+1}(T) = \varphi^r(T)$, so that $\Phi(T) = \varphi^r(T)$. Note that $\varphi(T) = T \Leftrightarrow \Phi(T) = T$. Also, for any $T, T' \subseteq E^m$, $T \subseteq T' \Rightarrow \Phi(T) \subseteq \Phi(T')$.

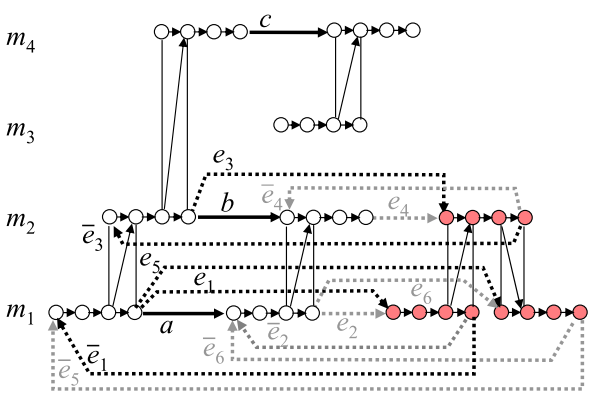

Fig. 4 Deriving $\Phi(e_5) = \{e_5, e_3, e_1\}$ in G^{m_1}

The closure $\Phi(T)$ adds arcs to T that are "implied" by T, i.e. that must be chosen by any feasible selection containing T, as expressed by the following proposition.

Proposition 1 *If (m, T') is feasible and $T \subseteq T'$, then $\Phi(T) \subseteq T'$ and $(m, \Phi(T))$ is positive acyclic.*

Proof Since (m, T') is complete, $e \in E^m - T'$ implies $\overline{e} \in T'$, and since (m, T') is positive acyclic, $G^m(T' \cup \overline{e}) = G^m(T')$ does not contain a positive cycle. Hence $\varphi(T') = T' = \Phi(T')$, i.e. T' is closed. From $\Phi(T) \subseteq \Phi(T') = T'$ follows that $(m, \Phi(T))$ is positive acyclic. □

Continuing the example, we briefly illustrate how the closure $\Phi(e_5)$ is derived in G^m for $m = m_1$, i.e. machine m_1 is chosen for i, the third operation of J. Figure 4 depicts G^{m_1}. One easily identifies a positive cycle entering J in e_5 and leaving J in $\overline{e}_3$, hence e_5 implies e_3, hence $e_3 \in \varphi^1(e_5)$. In fact, no further arc is implied by e_5, i.e. $\varphi^1(e_5) = \{e_5, e_3\}$. Also, there is a positive cycle entering J in e_3 and leaving J in $\overline{e}_1$, hence $e_1 \in \varphi^2(e_5)$. One can check that no other arc is implied by e_1, i.e. $\varphi^2(e_5) = \{e_5, e_3, e_1\}$, and that e_1 does not imply any further arc, hence $\Phi(e_5) = \{e_5, e_3, e_1\}$.

Besides closures, we will also need the notion of span in the sequel:

Definition 5 For any $T \subseteq E^m$, the span of T in G^m is the set

$$[T] = \{e \in E^m : \{e, \overline{e}\} \cap T \neq \emptyset\}.$$

3.2 Operation moves

Given a feasible selection (μ, S), neighbor selections can be generated with the following scheme. An operation $i \in I$, a machine $m \in M_i$ and a disjunctive arc $f \notin S$ with $t(f) \in V_{im}$ are chosen, and a corresponding feasible neighbor selection (μ', S_f) is constructed, which places i on machine m and enforces $f \in S_f$, i.e. places i before operation k such that $h(f) \in V_{km}$. Disjunctive arcs other than f might also be enforced, in order to maintain feasibility. They will be determined by performing two

successive closures in the insertion graph of the job J to which i belongs, hereby re-inserting J in a neighbor position. Specifically, (μ', S_f) is derived as follows. Mode μ' is given by

$$\mu'_j = \mu_j \quad \text{for all } j \in I - i, \mu'_i = m. \tag{1}$$

S_f is derived as follows. Let J be the job to which i belongs, G' be the insertion graph of J with local flexibility in i and $G^m = (V^m, A^m, E^m, \mathcal{E}^m, d)$ be the subgraph of G' corresponding to the machine choice $\mu'_i = m$ for operation i. Then

$$S_f = T_f \cup R \tag{2}$$

where, as before, $R = S - \delta(V'_J)$ is the part of S not incident to J, and (m, T_f) is a feasible neighbor of the insertion (μ_i, T^S) corresponding to (μ, S), to be constructed as follows.

Let $T^m = S \cap E^m$ and $E^{im-} = E^{im} \cap E^-$. Apply successively the following three steps. (i) Take f and all arcs implied, i.e. form the closure $\Phi(f)$ in G^m. (ii) Place before i all other operations on machine m that have not already been sequenced with respect to i in step (i), forming the closure $\Phi(E^{im-} - [\Phi(f)])$ in G^m. (iii) Keep T^m on the remaining part. Specifically:

$$T_f = P \cup Q \cup (T^m - [P \cup Q]) \quad \text{where} \tag{3}$$

$$P = \Phi(f), \qquad Q = \Phi(E^{im-} - [P]) \tag{4}$$

Theorem 1 *Insertion (m, T_f) given by (3) and (4) is a feasible neighbor insertion of (μ_i, T^S).*

Proof We show that (m, T_f) is a feasible insertion by proving that (i) its "components" (m, P), (m, Q) and $(m, T^m - [P \cup Q])$ are positive acyclic, (ii) (m, T_f) is complete and (iii) (m, T_f) is positive acyclic.

(i) Clearly, (m, E^{m+}) and (m, E^{m-}) are feasible insertions where job J is placed before, respectively after, all other jobs. By Proposition 1, $f \in E^{m+}$ and $E^{im-} - [P] \subseteq E^{m-}$ imply that $P = \Phi(f) \subseteq E^{m+}$ and $Q = \Phi(E^{im-} - [P]) \subseteq E^{m-}$, hence (m, P) and (m, Q) are positive acyclic. Also $(m, T^m - [P \cup Q])$ is obviously positive acyclic.

(ii) To establish completeness of (m, T_f), we verify that $T_f \cap \{e, \overline{e}\} \neq \emptyset$ for all pairs $\{e, \overline{e}\} \subseteq E^m$. By construction of P and Q, $(P \cup Q) \cap \{e, \overline{e}\} \neq \emptyset$ for all $\{e, \overline{e}\} \subseteq E^{im}$. Also, for any $\{e, \overline{e}\} \subseteq E^m - E^{im}$ for which $(P \cup Q) \cap \{e, \overline{e}\} = \emptyset$, completeness of (μ_i, T^S) implies that $(T^m - [P \cup Q])) \cap \{e, \overline{e}\} \neq \emptyset$.

(iii) Since $E^{m+} \cap E^{m-} = \emptyset$, P and Q are disjoint. We show that also their spans are disjoint:

$$[P] \cap [Q] = \emptyset. \tag{5}$$

It is enough to prove $[P] \cap Q = \emptyset$. Assuming the contrary, there exists some set N such that $E^{im-} - [P] \subseteq N \subset Q$, $[P] \cap N = \emptyset$ and $[P] \cap \varphi(N) \neq \emptyset$. Let $e \in (\varphi(N) - N)) \cap [P]$. Note that $e \in \varphi(N) \subseteq Q \subseteq E^{m-}$, $e \in [P]$ and $P \subseteq E^{m+}$ imply $\overline{e} \in P \subseteq E^{m+}$. By definition of $\varphi(N)$, there exists a positive cycle Z in $G^m(N \cup \overline{e})$

going through $\overline{e}$. By Lemma 1, there is a short positive cycle Z' in $G^m(N \cup \overline{e})$ with $\{g, h\} = Z' \cap E^m \subseteq Z \cap E^m$ for some $g \in E^{m-}$ and $h \in E^{m+}$. Since (m, Q) is positive acyclic (see (i)) and $N \subset Q$, any positive cycle in $G^m(N \cup \overline{e})$ and hence Z', must contain $\overline{e}$. Therefore $\overline{e} = h$, and $g \in N$ with $g \neq e$ since $e \notin N$, and $g \notin P$ since $g \in N$ and $N \cap [P] = \emptyset$.

Now, P is closed so that $\overline{e} \in P$, $g \notin P$, $g \neq e$ and Z' is a positive cycle in $G^m(\overline{e} \cup g)$ imply that $\overline{g} \in \varphi(P) = P$. But then $g \in N \cap [P]$, a contradiction to $N \cap [P] = \emptyset$.

We show now that (m, T_f) is positive acyclic. Suppose the contrary. By Lemma 1, there exists a short positive cycle Z' with $\{g, h\} = Z' \cap E^m$ in $G^m(T_f) = G^m(P \cup Q \cup (T^m - [P \cup Q]))$. Suppose $g \in P$. If $h = \overline{g}$, $h \subseteq [P]$. If $h \neq \overline{g}$, since P is closed and Z' is a positive cycle in $G^m(g \cup h)$, $\overline{h} \in P$, and hence $h \subseteq [P]$. Similarly, if $g \in Q$, $h \subseteq [Q]$. Hence $\{g, h\}$ is contained in $[P]$ or $[Q]$ or $T^m - [P \cup Q]$. By (5), these three cases are mutually exclusive, hence $\{g, h\} \subseteq T_f$ implies $\{g, h\} \subseteq P$ or $\{g, h\} \subseteq Q$ or $\{g, h\} \subseteq T^m - [P \cup Q]$, a contradiction to (m, P), (m, Q) and $(m, T^m - [P \cup Q])$ being positive acyclic. □

Corollary 1 *If $m = \mu_i$, i.e. operation i remains on its current machine and $\overline{f} \in S$, then (3) is equivalent to*

$$T_f = P \cup (T^m - [P]). \tag{6}$$

Proof Let T_f be defined by (3) and $\widehat{T}_f = P \cup (T^m - [P])$. First, $(m, \widehat{T}_f)$ is a feasible insertion. This is easily shown using the arguments of the previous proof and noting that completeness of $(m, \widehat{T}_f)$ results from the fact that now, with $m = \mu_i$, (m, T^m) is feasible and hence complete. Next, we show that

$$Q = \Phi(E^{im-} - [P]) \subseteq T^m - [P]. \tag{7}$$

Let $i_1, \ldots, i_l$ be the operations on machine m, ordered in that sequence by T^m, i.e. $T^m \cap E^{im}$ consists of all disjunctive arcs g with $t(g) \in V_{i_r m}$, $h(g) \in V_{i_s m}$, $1 \leq r < s \leq l$. Since $\overline{f} \in T^m$ with $h(\overline{f}) \in V_{im}$, $i = i_q$ for some q with $1 < q \leq l$, and for its mate f, $t(f) \in V_{i_q m}$ and $h(f) \in V_{i_p m}$ for some p with $1 \leq p < q \leq l$. Obviously, any $g \in E^{im+}$ with $t(g) \in V_{i_q m}$ and $h(g) \in V_{i_s m}$ for $s \geq p$, $s \neq q$, is contained in P, since $i = i_q$ being placed ahead of i_p implies that i also precedes any operation i_s with $s \geq p$. Therefore $g \in E^{im+} - P$ implies $t(g) \in V_{i_q m}$ and $h(g) \in V_{i_s m}$ for some s with $1 \leq s < p$, hence $g \in E^{im-} - [P]$ implies $t(g) \in V_{i_s m}$ and $h(g) \in V_{i_q m}$ for some s with $1 \leq s < p < q$. But then $g \in T^m \cap E^{im}$, proving

$$E^{im-} - [P] \subseteq T^m - [P]. \tag{8}$$

Since (m, T_m) is feasible, by (8) and Proposition 1, $Q = \Phi(E^{im-} - [\Phi(f)]) \subseteq T^m$ and by (5), $Q \subseteq T^m - [P]$, proving (7).

From (7) follows $P \cup Q \subseteq P \cup (T^m - [P])$. Also $T^m - [P \cup Q] \subseteq T^m - [P]$, hence $T_f = P \cup Q \cup (T^m - [P \cup Q]) \subseteq P \cup (T^m - [P]) = \widehat{T}_f$. But $T_f \subseteq \widehat{T}_f$ implies $T_f = \widehat{T}_f$ since both (m, T_f) and $(m, \widehat{T}_f)$ are feasible insertions with same mode m and therefore $|T_f| = |\widehat{T}_f|$. □

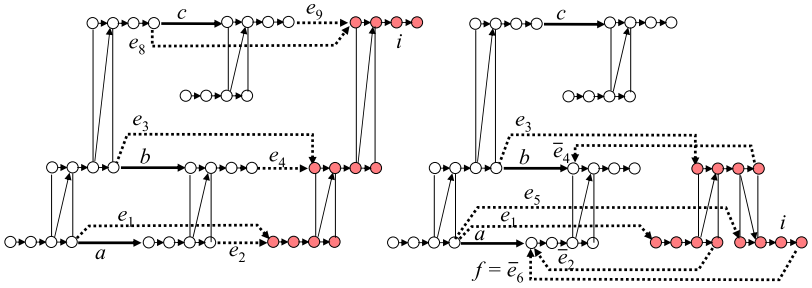

Fig. 5 Current and neighbor selections

By Corollary 1, the neighbor (m, T_f) of insertion (m, T^m) if operation i remains on its current machine and $\overline{f} \in S$, is obtained more simply by taking f together with the arcs it implies, forming $P = \Phi(f)$, and keeping T^m on the remaining part. We remark that we used neighbors of this type in previous work on the simpler non-flexible Blocking Job Shop (Gröflin and Klinkert 2009).

We illustrate in the example the generation of a neighbor selection according to (3)–(4). The current selection (μ, S) is depicted in Fig. 5, left, i.e. $S = \{a, b, c, e_1, e_2, e_3, e_4, e_8, e_9\}$ consists of the bold and dotted arcs. The part of S not incident to job J is $R = \{a, b, c\}$. We choose the operation i to be moved as the third operation of the job J at the right, so that the insertion graph is the graph previously given in Fig. 3. In the current selection, operation i is on machine m_4. The insertion (m, T^m) corresponding to (μ, S) is defined by $m = m_4$ and $T^{m_4} = \{e_1, e_2, e_3, e_4, e_8, e_9\}$.

We move now operation i to machine $m = m_1$ and choose $f = \overline{e}_6$. According to (3)–(4), we determine $P = \Phi(\overline{e}_6) = \{\overline{e}_6, \overline{e}_4, \overline{e}_2\}$, $E^{im_1-} - [\Phi(\overline{e}_6)] = \{e_5\}$ and $Q = \Phi(E^{im_1-} - [\Phi(\overline{e}_6)]) = \{e_5, e_3, e_1\}$, so that $P \cup Q = \{\overline{e}_6, \overline{e}_4, \overline{e}_2, e_5, e_3, e_1\}$. Also, $T^{m_1} = T^{m_4} \cap E^{m_1} = \{e_1, e_2, e_3, e_4\}$, hence $T^{m_1} - [P \cup Q] = \emptyset$. Therefore, $T_f = P \cup Q$ and the neighbor selection is $S_f = T_f \cup R = \{\overline{e}_6, \overline{e}_4, \overline{e}_2, e_5, e_3, e_1, a, b, c\}$. It is depicted at the right.

Two remarks are in order before concluding this section. First, in the move of operation i on a machine $m \in M_i$, we introduce in S_f some disjunctive arc $f \notin S$ with $t(f) \in V_{im}$, $h(f) \in V_{jm}$, placing i *before* some operation j on m. Obviously, a similar construction of a neighbor selection S_g introducing now a disjunctive arc $g \notin S$ with $h(g) \in V_{im}$, $t(g) \in V_{jm}$, can be derived in a straightforward manner, placing i *after* operation j on m. Formulas (3)–(4) become

$$T_g = P \cup Q \cup (T^m - [P \cup Q]) \quad \text{where} \tag{9}$$

$$P = \Phi(g), \qquad Q = \Phi(E^{im+} - [P]) \tag{10}$$

Second, consider the case where an operation i is moved to a machine $m \in M_i - \mu_i$ on which there is no operation in the current selection. Since there is no f with $t(f) \in V_{im}$ or g with $h(g) \in V_{im}$ to select, this case is not accounted for by (3)–(4) or (9)–(10). The corresponding neighbor selection $(\mu', S_\emptyset)$ obtained from such a move is obviously given by (1) and $S_\emptyset = T^m \cup R$. In fact, (3)–(4) (or (9)–(10)) still apply by simply setting $f = \emptyset$ (or $g = \emptyset$), justifying the notation $S_\emptyset$. Indeed, in this case

$E^{im-} = E^{im+} = \emptyset$ and $\Phi(f) = \Phi(g) = \emptyset$, so that (3)–(4) or (9)–(10) yield $S_\emptyset = T^m \cup R$.

4 A tabu search

Tabu search is a well-known heuristic approach in combinatorial optimization (see for instance Glover and Laguna 1997). In particular, various job shop scheduling problems have been tackled with this method by numerous authors, e.g. by Nowicki and Smutnicki (1996) for the classical job shop. We present now a tabu search for the FBJS based on the feasible neighborhood developed in the previous section, and using some general features of Nowicki and Smutnicki (1996), in particular Back Jump Tracking with the maintenance of a list of elite solutions. These features have proved useful in the (non-flexible) Blocking Job Shop (Gröflin and Klinkert 2009).

First, we reduce the number of potential neighbors by considering only moves of *critical* operations. Given a feasible selection (μ, S) in the disjunctive graph $G = (V, A, E, \mathcal{E}, d)$, let L be the arc set of an (arbitrary) longest path from σ to τ in subgraph $(V^\mu, A^\mu \cup S, d)$. The set of critical operations is defined by

$$I^c = \{i \in I : \{h(e), t(e)\} \cap V_{i\mu_i} \neq \emptyset \text{ for some } e \in S \cap L\}. \tag{11}$$

For each operation $i \in I^c$, neighbors (μ', S_f) according to (1), (2), (3) and (4) are generated by proceeding for each $m \in M_i$ as follows.

(i) If $m = \mu_i$ and $h(e) \in V_{i\mu_i}$ for a (unique) $e \in S \cap L$, i.e. i stays on its machine and is preceded on it by a critical operation, namely operation $j(i)$ such that $t(e) \in V_{j(i)m}$, then f is the mate of e. The move reverses the precedence between i and its immediate predecessor $j(i)$ on m. Note that, due to the closures taken, other operations of job J to which i belongs might be moved as well, and while i is moved ahead of $j(i)$, it might not be its immediate predecessor. We also point out that if there is no $e \in S \cap L$ with $h(e) \in V_{i\mu_i}$, i.e. i is the first critical operation of a "block" on machine μ_i, then no move of i staying on μ_i is generated.

(ii) If $m \neq \mu_i$, i.e. operation i changes its machine, the choice of f is less immediate. In principle, i could be inserted anywhere in the sequence of operations on m and one could generate as many neighbors as there are insertion positions. However, in order to limit the size of the neighborhood, only one f is chosen with the following idea in mind. In a neighbor selection that is close to the current selection, i is likely to be scheduled on the new machine m at a time not too far from its time on the current machine μ_i. We proceed therefore as follows. In selection (μ, S), for any operation k, let x_k denote the starting time of k on its machine μ_k, i.e. x_k is the length of a longest path from σ to v^1_{k,μ_k} in subgraph $(V^\mu, A^\mu \cup S, d)$. Let $j(i)$ be an operation on m such that $x_{j(i)} \geq x_i$ and either $j(i)$ is first in the sequence on m or its predecessor, say k, has starting time $x_k < x_i$. Then f with $h(f) \in V_{j(i),m}$, $t(f) \in V_{im}$ is chosen, placing i before $j(i)$. If all operations k on m have starting times $x_k < x_i$, then set $f = \emptyset$ in (3)–(4). $S_\emptyset$ thus obtained will place i last in the sequence on m. If there are no operations on m, take again $S_\emptyset$ (as discussed at the end of Sect. 3).

The neighborhood just described will be referred to as $\mathcal{N}^1_c$. Since for each operation $i \in I^c$, $|M_i|$ neighbors (or $|M_i| - 1$ neighbors if i is the first critical operation

of a "block") are generated, the size of $\mathcal{N}_c^1$ is relatively large and increases with the degree of flexibility. For this reason, we also examined a second neighborhood $\mathcal{N}_c^2 \subseteq \mathcal{N}_c^1$ of smaller size (approximately $|I^c|$) independent of the degree of flexibility, obtained as follows. In selection (μ, S), let the current workload of a machine m be the sum of the sojourn times of the operations on m (counted for each operation from start of take-over step to end of hand-over-step), and corresponding set-up times. For each operation $i \in I^c$, a single neighbor is generated by first choosing among all machines in M_i, the machine, say m, with smallest current workload and constructing neighbor (μ', S_f) as above.

Two tabu search versions with neighborhood $\mathcal{N}_c^1$ and $\mathcal{N}_c^2$ respectively have been implemented. A tabu list storing entries of the last *maxt* iterations is maintained. Specifically, a tabu list $L_{(\mu,S)}$ is associated to the current selection (μ, S), of maximal length *maxt*. The list is empty for the initial selection. In an iteration, i.e. after execution of a move from selection (μ, S) to a neighbor (μ', S_f), the list $L_{(\mu',S_f)}$ is obtained from $L_{(\mu,S)}$ by dropping the oldest entry if $|L_{(\mu,S)}| = maxt$, and entering in first position arc e chosen as follows. (i) If operation $i \in I^c$ was moved while staying on machine $m = \mu_i$, then $e \in S \cap L$ with $h(e) \in V_{i\mu_i}$ is chosen. (ii) If $i \in I^c$ was moved to machine $m \neq \mu_i$, then $e \in S \cap L$ with $h(e) \in V_{i\mu_i}$, or, if there is no such e, $e \in S \cap L$ with $t(e) \in V_{i\mu_i}$ is entered. The tabu list is used to define tabu neighbors: a neighbor (μ', S_f) of (μ, S) is *tabu* if $S_f \cap L_{(\mu,S)} \neq \emptyset$.

The choice of the move to be executed is based on the evaluation of the so-called candidate set. This set consists initially of the whole neighborhood. The (minimal) makespan for each neighbor selection is computed. Then any neighbor which is tabu and whose makespan does not improve the best makespan found so far (improved-best aspiration criterion), is removed from the candidate set. If this set becomes empty (i.e. all neighbors are tabu and non-improving), the neighbor corresponding to the oldest entry in the tabu list is added to it. The move to be executed is then to the neighbor with best makespan in the candidate set.

A long term memory is also used in the tabu search by maintaining a list of bounded length *maxl* of so-called elite solutions, which is initially empty. A new solution is appended to the list if it is better than any previously encountered solution. If the tabu search runs for a given number *maxiter* of iterations without improving the best makespan or if a cycle is detected, the current search path is terminated and search is resumed from the last elite solution in the list. For this reason, an elite solution is stored together with its associated tabu list and the moves already taken from the solution. Thus, when resuming the search from the solution, its tabu list is restored and previously executed moves are removed from the candidate set. An elite solution is deleted from the list if its candidate set is empty. Finally, a procedure is used to detect cycling that keeps track of the sequence of makespans encountered during the search and scans for repeated subsequences.

5 Computational results

The tabu search has been implemented in C++ and run on a PC with 3.0 GHz processor and 2 GB memory. Since the FBJS does not appear in the literature, benchmark

instances and results were not available. We created therefore a total of 320 instances starting from data used by Gröflin and Klinkert (2009) in the Blocking Job Shop (BJS).

We started with the 40 instances la01 to la40. For each lapq, two groups of 4 instances were generated, the first group without transfer and set-up times and referred to as FBJS^0, the second group with transfer and set-up times, referred to as FBJS^{ts}. Within a group, the 4 instances are generated by introducing increasing flexibility: in the first instance, 1 machine is available for each operation (i.e. the non-flexible instance), while in the 2nd, 3rd and 4th instance, 2, 3 and 5 machines respectively, are available for each operation. This is achieved by adding randomly and successively 1, 1 and 2 machines, creating for each operation four machine sets of size 1, 2, 3 and 5 that are nested. Such a choice will allow to evaluate the impact of increasing flexibility.

Transfer and set-up times were generated randomly as described in Gröflin and Klinkert (2009) for the BJS. Note however that in the FBJS, it is possible that two operations that are consecutive in a job are on a same machine. In this case, both corresponding transfer and set-up times are set to zero.

Computations were performed as follows. Tabu search parameters were identical for all runs and set as: $maxt = 14$, $maxl = 300$ and $maxiter = 2500$. Both versions of the tabu search with neighborhood $\mathcal{N}_c^1$ and $\mathcal{N}_c^2$ were executed.

For each instance, five runs were performed with different starting solutions. The latter were randomly generated permutation schedules, using the choice of machines (i.e. the mode) of the corresponding non-flexible instance. Computation time of a run was limited to 1800 seconds.

Detailed computational results are provided in Table 1. Its first line splits the results into the two groups FBJS^0 and FBJS^{ts} of problems without, respectively with transfer and set-up times. The second line refers to the degree of flexibility, i.e. the number ($flex = 2, 3$ or 5) of machines per operation, and the third line to the neighborhood ($\mathcal{N}_c^1$ or $\mathcal{N}_c^2$) used. The rest of the table is subdivided horizontally according to the size of the problem, e.g. the first block reports on instances 10×5 with 10 jobs and 5 machines. In each line lapq, the mean makespan (averaged over the five runs) is reported for varying degree of flexibility, neighborhood and absence or presence of transfer and set-up times. We discuss now these results, assessing solution quality, convergence behavior of the tabu search and impact of increasing flexibility on the makespan.

As is often the case in complex scheduling problems, a definitive assessment of attained solution quality by comparing it with the mathematically proven optimum is not possible in the current state of the art. Also, as already mentioned, no benchmarks for the FBJS are available in the literature. For these reasons, we resorted to compare performance of our tabu search when applied on the (non-flexible) BJS. Benchmarks are available in this case. We used the results of Gröflin and Klinkert (2009) which provide the most comprehensive benchmarks for the BJS without and with set-up and transfer times.

Since the neighbor construction (3) yields in the non flexible case the same neighbor as used for the benchmark results (see Corollary 1), one would expect that our tabu search for FBJS performs at best as well or somewhat worse due to computational overhead. Surprisingly, our method not only matched, but improved nearly all

Table 1 Detailed tabu search results for various degrees of flexibility

	FBJS^0						FBJS^{ts}					
flex	2		3		5		2		3		5	
inst	$\mathcal{N}_c^1$	$\mathcal{N}_c^2$	$\mathcal{N}_c^1$	$\mathcal{N}_c^2$	$\mathcal{N}_c^1$	$\mathcal{N}_c^2$	$\mathcal{N}_c^1$	$\mathcal{N}_c^2$	$\mathcal{N}_c^1$	$\mathcal{N}_c^2$	$\mathcal{N}_c^1$	$\mathcal{N}_c^2$
10 × 5												
la01	666	689	617	634	–	–	1018	1057	860	914	–	–
la02	613	637	576	591	–	–	956	1010	808	842	–	–
la03	563	580	515	536	–	–	936	960	744	785	–	–
la04	588	612	538	554	–	–	914	940	758	778	–	–
la05	553	559	549	550	–	–	846	879	751	780	–	–
15 × 5												
la06	920	956	861	892	–	–	1387	1469	1154	1231	–	–
la07	864	900	805	837	–	–	1375	1458	1115	1196	–	–
la08	897	925	827	859	–	–	1432	1500	1167	1205	–	–
la09	1014	1048	919	953	–	–	1565	1641	1219	1280	–	–
la10	945	987	874	912	–	–	1542	1596	1193	1264	–	–
20 × 5												
la11	1268	1300	1147	1188	–	–	2014	2063	1600	1648	–	–
la12	1108	1133	1020	1045	–	–	1825	1892	1462	1506	–	–
la13	1228	1262	1119	1163	–	–	1920	2014	1529	1586	–	–
la14	1248	1295	1155	1178	–	–	1979	2009	1549	1614	–	–
la15	1286	1309	1185	1225	–	–	1965	2021	1592	1701	–	–
10 × 10												
la16	818	860	724	796	717	749	1313	1344	1165	1225	967	1043
la17	693	766	646	701	646	651	1213	1256	1081	1122	853	968
la18	776	850	697	778	663	733	1281	1331	1163	1200	957	1058
la19	830	898	705	829	648	782	1308	1353	1194	1224	990	1110
la20	834	895	756	824	756	764	1386	1425	1215	1270	985	1110
15 × 10												
la21	1174	1230	1066	1112	1034	1059	1899	1933	1601	1705	1289	1427
la22	1058	1102	989	1020	947	970	1762	1836	1516	1637	1235	1361
la23	1169	1211	1098	1144	1052	1086	1887	1922	1609	1701	1307	1407
la24	1103	1154	1013	1064	1005	1032	1865	1896	1582	1693	1251	1376
la25	1072	1147	982	1050	931	979	1823	1866	1512	1610	1219	1342

Table 1 (*Continued*)

	FBJS^0						FBJS^{ts}					
flex	2		3		5		2		3		5	
inst	$\mathcal{N}_c^1$	$\mathcal{N}_c^2$	$\mathcal{N}_c^1$	$\mathcal{N}_c^2$	$\mathcal{N}_c^1$	$\mathcal{N}_c^2$	$\mathcal{N}_c^1$	$\mathcal{N}_c^2$	$\mathcal{N}_c^1$	$\mathcal{N}_c^2$	$\mathcal{N}_c^1$	$\mathcal{N}_c^2$
20 × 10												
la26	1523	1572	1402	1438	1294	1355	2421	2474	2067	2146	1675	1854
la27	1561	1636	1433	1495	1353	1378	2543	2608	2178	2262	1742	1887
la28	1552	1630	1420	1476	1343	1371	2513	2573	2142	2268	1725	1890
la29	1476	1529	1337	1337	1263	1276	2396	2461	2060	2159	1671	1827
la30	1534	1577	1397	1443	1318	1341	2539	2578	2158	2269	1724	1913
30 × 10												
la31	2303	2332	2089	2122	1981	1959	3707	3736	3118	3206	2424	2656
la32	2533	2508	2265	2325	2147	2142	3946	3988	3335	3468	2634	2860
la33	2233	2236	2046	2140	1935	1957	3675	3793	3139	3285	2418	2633
la34	2327	2324	2091	2105	2022	1996	3761	3800	3187	3257	2432	2704
la35	2331	2376	2136	2136	2019	2032	3717	3852	3134	3296	2492	2754
15 × 15												
la36	1346	1435	1240	1342	1158	1248	2090	2178	1864	1965	1646	1791
la37	1486	1579	1351	1474	1264	1409	2230	2305	2002	2113	1791	1922
la38	1269	1365	1144	1247	1061	1175	2061	2110	1811	1909	1592	1726
la39	1327	1424	1202	1339	1144	1279	2055	2200	1815	1927	1590	1729
la40	1361	1450	1288	1352	1156	1293	2112	2165	1845	1927	1616	1762

(all but one out of 80) makespans to an extent of a few percent to over ten percent. Detailed comparisons can be found in Table 2. The columns with heading "bench" and "new" refer respectively to the results reported in Gröflin and Klinkert (2009) and achieved with our tabu search. (Note that $\mathcal{N}_c^1 = \mathcal{N}_c^2$ if there is no flexibility, so that there is no distinction between the two neighborhoods to be made here.) Altogether, our tabu search for the FBJS is competitive even for the BJS, and its logic suggests that its performance should be comparable in flexible instances.

It is of interest to examine the evolution of attained solution quality during computation. Table 3 gives an overview for a subset of the benchmark instances, namely all instances of FBJS^{ts} with two machines per operation (*flex* = 2), and both neighborhoods $\mathcal{N}_c^1$ and $\mathcal{N}_c^2$. For each run of each instance, the best makespan ω at the beginning (initial solution), and after 100, 200, 300, 600 and 1200 seconds of computation has been recorded and its (relative) gap from the final solution $(\omega - \omega_{final})/\omega_{final}$ computed. Table 3 shows in columns 4 to 9 these gaps in an aggregated way, reporting mean gaps (over runs and instances of same size). Additionally, average number of iterations and running time are given in columns 2 and 3. The following observations can be made.

Table 2 Detailed tabu search results without flexibility

	FBJS^0		FBJS^{ts}			FBJS^0		FBJS^{ts}	
	bench	new	bench	new		bench	new	bench	new
10 × 5					*15 × 10*				
la01	836	820	1570	1487	*la21*	1617	1576	2987	2805
la02	824	817	1579	1518	*la22*	1525	1467	2875	2575
la03	765	740	1497	1439	*la23*	1645	1570	2979	2705
la04	774	764	1470	1398	*la24*	1623	1546	3019	2738
la05	711	666	1399	1320	*la25*	1541	1523	2890	2681
15 × 5					*20 × 10*				
la06	1191	1180	2259	2117	*la26*	2182	2125	4013	3620
la07	1119	1084	2274	2070	*la27*	2258	2201	4110	3760
la08	1161	1162	2201	2109	*la28*	2186	2167	4017	3661
la09	1269	1258	2341	2205	*la29*	2161	1990	4029	3645
la10	1222	1208	2349	2239	*la30*	2199	2097	4126	3742
20 × 5					*30 × 10*				
la11	1615	1591	3121	2872	*la31*	3266	3137	6506	5550
la12	1433	1398	2873	2664	*la32*	3549	3316	6404	5863
la13	1576	1541	2858	2710	*la33*	3225	3061	6423	5566
la14	1648	1638	2935	2830	*la34*	3354	3146	6618	5595
la15	1667	1630	2981	2822	*la35*	3445	3171	6312	5555
10 × 10					*15 × 15*				
la16	1175	1143	2066	1913	*la36*	1974	1919	3696	3274
la17	1040	977	1960	1786	*la37*	2133	2029	3808	3386
la18	1112	1098	2068	1880	*la38*	1939	1828	3571	3119
la19	1124	1102	1966	1844	*la39*	1987	1882	3587	3158
la20	1184	1162	2119	1958	*la40*	1982	1925	3496	3173

Initial solutions are often far away from the obtained final solutions: their makespan is about three times as large for small problem sizes, and 4 to 5 times as large for larger sizes. Second, most of the improvements are made in the first 100 (300) seconds depending on problem size, and no or only marginal improvements are reached after 1200 seconds even for larger sizes. Similar results have been achieved for instances with higher flexibility, suggesting good improvement performance and convergence of the tabu search.

We also examined the extent to which increasing flexibility decreases makespan. For each instance, the mean makespan ω (averaged over the five runs) is compared to the mean makespan $\omega^{nonflex}$ of the corresponding instance with no flexibility and the

Table 3 Time analysis for FBJS^{ts} tabu search with flexibility 2

size	iter	time	init	100 s	200 s	300 s	600 s	1200 s
$\mathcal{N}_c^1$								
10 × 5	*683679*	1774	1.95	0.03	0.01	0.01	0.00	0.00
15 × 5	*326775*	1801	1.88	0.03	0.02	0.02	0.01	0.00
20 × 5	*194100*	1801	1.83	0.03	0.02	0.01	0.01	0.00
10 × 10	*215991*	1801	3.04	0.03	0.02	0.01	0.01	0.00
15 × 10	*97719*	1801	3.31	0.05	0.03	0.03	0.01	0.01
20 × 10	*55007*	1801	3.20	0.07	0.05	0.03	0.02	0.01
30 × 10	*25999*	1801	3.12	0.08	0.06	0.05	0.02	0.01
15 × 15	*51883*	1801	4.51	0.08	0.05	0.03	0.02	0.01
$\mathcal{N}_c^2$								
10 × 5	*363552*	552	1.84	0.02	0.01	0.01	0.00	0.00
15 × 5	*566514*	1801	1.74	0.02	0.01	0.01	0.00	0.00
20 × 5	*334144*	1801	1.75	0.03	0.02	0.01	0.01	0.00
10 × 10	*379275*	1801	2.91	0.04	0.02	0.02	0.00	0.00
15 × 10	*173712*	1801	3.21	0.04	0.03	0.02	0.01	0.00
20 × 10	*100534*	1801	3.11	0.06	0.04	0.03	0.02	0.01
30 × 10	*45378*	1801	3.04	0.08	0.05	0.04	0.02	0.00
15 × 15	*92562*	1801	4.30	0.07	0.04	0.03	0.02	0.01

relative variation $(\omega - \omega^{nonflex})/\omega^{nonflex}$ is computed. These relative variations are reported in Table 4 in an aggregated way (averaged over instances of same size) for both neighborhoods $\mathcal{N}_c^1$ and $\mathcal{N}_c^2$, and for FBJS^0 and FBJS^{ts} (without/with transfer and set-up times). The following observations can be made.

First, flexibility offers more potential for makespan reduction when transfer and set-up times are present. For instance, for larger problems (10×10 to 15×15 in Table 4) and 5 machines per operation, the makespan is reduced by approximately 37% in the absence of transfer and set-up times and by about 53% otherwise. This can be attributed to the opportunity for two consecutive operations in a job to be performed on the same machine if their machine sets intersect, in which case transfer and set-up times are saved. This effect is quite visible when schedules are displayed in Gantt charts (see example below).

Second, when the number of machines per operation increases from 2 to 3 to 5, the makespan is reduced on average by 25%, 31% and 37% in the absence of transfer- and set-up times, and by 32%, 43% and 53% otherwise. While these figures are only estimates, they give an interesting indication on the benefit of flexibility and also of the diminishing return of adding flexibility.

We conclude the discussion of computational results by observing that the tabu search version with neighborhood $\mathcal{N}_c^1$ appears to be better than the neighborhood $\mathcal{N}_c^2$ with respect to solution quality. As is apparent in Table 1, $\mathcal{N}_c^1$ is systematically better for problems FBJS^{ts}. However, $\mathcal{N}_c^2$ yields in six out of 105 instances of FBJS^0

Table 4 Impact of flexibility

	FBJS^0						FBJS^{ts}					
flex	*2*		*3*		*5*		*2*		*3*		*5*	
size	$\mathcal{N}_c^1$	$\mathcal{N}_c^2$	$\mathcal{N}_c^1$	$\mathcal{N}_c^2$	$\mathcal{N}_c^1$	$\mathcal{N}_c^2$	$\mathcal{N}_c^1$	$\mathcal{N}_c^2$	$\mathcal{N}_c^1$	$\mathcal{N}_c^2$	$\mathcal{N}_c^1$	$\mathcal{N}_c^2$
10 × 5	−22%	−19%	−26%	−25%	–	–	−35%	−32%	−45%	−43%	–	–
15 × 5	−21%	−18%	−27%	−24%	–	–	−32%	−29%	−46%	−42%	–	–
20 × 5	−21%	−19%	−28%	−26%	–	–	−30%	−28%	−44%	−42%	–	–
10 × 10	−28%	−22%	−36%	−28%	−37%	−33%	−31%	−28%	−38%	−36%	−49%	−44%
15 × 10	−27%	−24%	−33%	−30%	−35%	−33%	−32%	−30%	−42%	−38%	−53%	−49%
20 × 10	−28%	−25%	−34%	−32%	−38%	−36%	−33%	−31%	−42%	−40%	−54%	−49%
30 × 10	−26%	−26%	−33%	−32%	−36%	−36%	−33%	−32%	−43%	−41%	−56%	−52%
15 × 15	−29%	−24%	−35%	−30%	−40%	−33%	−35%	−32%	−42%	−39%	−49%	−45%
Average	−25%	−22%	−31%	−28%	−37%	−35%	−32%	−30%	−43%	−40%	−53%	−48%

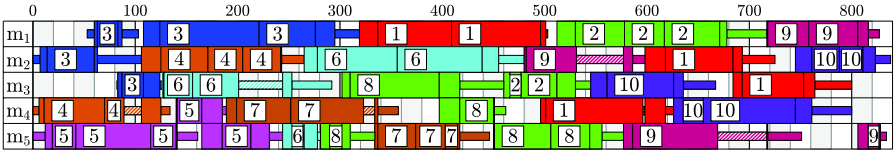

Fig. 6 Gantt chart of instance *la*05 with flexibility 2 and transfer and set-up times

better results. $\mathcal{N}_c^2$ appears to be at an advantage when generating a sufficient number of iterations in the available computing time is critical. Indeed, since $\mathcal{N}_c^2 \subseteq \mathcal{N}_c^1$, more iterations are performed with $\mathcal{N}_c^2$ than with $\mathcal{N}_c^1$ for a given instance and computation time. Also, generating a neighbor is computationally more expensive in problems FBJS^0 than in FBJS^{ts}, since determining closures in FBJS^0 involves detection of *positive* cycles (cycles of length 0, corresponding to so-called swaps, being allowed).

Figure 6 shows a Gantt chart of the best schedule with makespan 837 obtained in instance *la*05 with two machines per operation and transfer and set-up times. The narrow bars represent set-up times (filled) and waiting times (hatched), while thick bars represent take-over, processing and hand-over steps of operations. The numbers refer to the jobs, e.g. the five 3 in the chart identify the five operations of job 3. Note that, taking advantage of machine flexibility, several consecutive operations in jobs are executed on a same machine, thus saving transfer and set-up times.

6 Concluding remarks

It is well-known that adding blocking constraints in the classical Job Shop, turning it into a Blocking Job Shop (BJS), makes the problem substantially more complex. Theoretical results as well as empirical evidence with various solution approaches support this fact. The difficulty stems from feasibility issues. For instance, determining whether a partial positive acyclic selection can be extended to a feasible selection

is an NP-complete decision problem (and is easy in the classical Job Shop). Some heuristic approaches that have been proposed do not always terminate with a feasible solution, others, based on local search, resort to "repair" steps in order to maintain a feasible solution, at the risk of deteriorating solution quality.

The Flexible Blocking Job Shop (FBJS), as a generalization of the BJS allowing for machine choices for the operations, represents therefore a challenge. Up to now, this problem does not seem to have been tackled, at least we are not aware of previous work in the literature. We presented in this paper a tabu search for the FBJS that consistently generates feasible neighbor solutions and leads to substantial solution improvement. The neighborhood is based on moves of critical operations. An operation is moved on its currently assigned machine or to another machine, together with implied moves of other operations.

We conducted numerical experiments on a larger test set which support the validity of the approach. Our tabu search is competitive even in the non-flexible case (BJS), improving most benchmarks available in the literature on the test set, and it establishes first benchmarks for the FBJS. Additionally, it provided interesting information on how adding flexibility in a BJS reduces makespan.

Our approach required some theoretical tools. First, we chose to give a comprehensive formulation of the FBJS in a disjunctive graph that includes flexibility and sequence-dependent transfer times and set-up times. For a given assignment of machines, a so-called mode, the corresponding disjunctive graph is then a node-induced subgraph. Second, we needed to introduce the framework of job insertion with local flexibility as the proper structural setting for describing the moves and proving their feasibility.

Several directions for future research suggest themselves. First, the framework of job insertion with local flexibility could be exploited further. Indeed, based on Gröflin and Klinkert (2007), it is not difficult to characterize all feasible insertions in the insertion graph G' through stable sets in the so-called conflict graph, so that other feasible neighbors than the ones used here are at hand.

Second, reducing computation times is desirable, especially in view of applications in practice. This might be achieved by efficiency improvements in the implementation of the tabu search. Designing heuristics of constructive nature would be another alternative. A first step was taken in Pham (2008) where, besides an early version of the tabu search presented in this paper, also constructive heuristics are proposed that are faster, however at the cost of markedly lower solution quality.

Finally, extending the FBJS to the Flexible Job Shop with Limited Buffer Capacity (FJSLBC) suggests itself. In fact the latter problem can be captured as a FBJS by modeling each buffer capacity unit as a machine. However, as Brucker et al. (2006) have shown, there are benefits in studying specific buffer configurations. Future work on the FJSLBC could combine elements of Brucker et al. (2006) with the approach presented here.

Acknowledgements We are greatly indebted to Andreas Klinkert for letting us benefit from his code for the blocking job shop. It formed the basis in our code development.

References

Brizuela C, Zhao Y, Sannomiya N (2001) No-wait and blocking job-shops: challenging problems for ga's. In: IEEE international conference on systems, man, and cybernetics, vol 4, pp 2349–2354

Brucker P, Kampmeyer T (2008) Cyclic job shop scheduling problems with blocking. Ann Oper Res 159(1):161–181

Brucker P, Heitmann S, Hurink J, Nieberg T (2006) Job-shop scheduling with limited capacity buffers. OR Spectrum 28:151–176

Glover F, Laguna M (1997) Tabu search. Kluwer, Boston

Gröflin H, Klinkert A (2007) Feasible insertions in job shop scheduling, short cycles and stable sets. Eur J Oper Res 177:763–785

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

Heitmann S (2007) Job-shop scheduling with limited buffer capacities. PhD thesis, University Osnabrück, Germany

Klinkert A (2001) Optimization in design and control of automated high-density warehouses. PhD thesis, University of Fribourg, Switzerland

Mascis A, Pacciarelli D (2000) Machine scheduling via alternative graphs. Internal Report RT-DIA-46-2000, Università degli Studi Roma Tre, Dipartimento di Informatica e Automazione, Italy

Mascis A, Pacciarelli D (2002) Job-shop scheduling with blocking and no-wait constraints. Eur J Oper Res 143:498–517

Meloni C, Pacciarelli D, Pranzo M (2004) A rollout metaheuristic for job shop scheduling problems. Ann Oper Res 131:215–235

Nowicki E, Smutnicki C (1996) A fast taboo search algorithm for the job shop problem. Manag Sci 42:797–812

Pham DN (2008) Complex job shop scheduling: Formulations, algorithms and a healthcare application. PhD thesis, University of Fribourg, Switzerland

Thornton HW, Hunsucker JL (2004) A new heuristic for minimal makespan in flow shops with multiple processors and no intermediate storage. Eur J Oper Res 152:96–114