Discrete Applied Mathematics 157 (2009) 3643–3655

Contents lists available at ScienceDirect

Discrete Applied Mathematics

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

A new neighborhood and tabu search for the Blocking Job Shop

Heinz Gröflin[a], Andreas Klinkert[b,*]

[a] Department of Informatics, University of Fribourg, Switzerland

[b] Institute of Data Analysis and Process Design, Zurich University of Applied Sciences, Switzerland

ARTICLE INFO

Article history:
Received 29 November 2007
Received in revised form 9 February 2009
Accepted 19 February 2009
Available online 1 April 2009

Keywords:
Job shop scheduling
Blocking
Setup
Disjunctive graph
Tabu search

ABSTRACT

The Blocking Job Shop is a version of the job shop scheduling problem with no intermediate buffers, where a job has to wait on a machine until being processed on the next machine. We study a generalization of this problem which takes into account transfer operations between machines and sequence-dependent setup times. After formulating the problem in a generalized disjunctive graph, we develop a neighborhood for local search. In contrast to the classical job shop, there is no easy mechanism for generating feasible neighbor solutions. We establish two structural properties of the underlying disjunctive graph, the concept of closures and a key result on short cycles, which enable us to construct feasible neighbors by exchanging critical arcs together with some other arcs. Based on this neighborhood, we devise a tabu search algorithm and report on extensive computational experience, showing that our solutions improve most of the benchmark results found in the literature.

1. Introduction

The classical job shop has established itself as a standard problem in scheduling and a substantial body of knowledge has accumulated, benefiting invaluably the scheduling field. However, scheduling problems in practice often cannot be modeled as job shop problems, due to features such as generalized precedence constraints, storage-space and waiting-time constraints, material-handling constraints, sequence-dependent setups, parallel use of resources and processing alternatives (see for instance, Jain and Meeran [14], Pinedo and Chao [26], and Hall and Sriskandarajah [12]).

Several extensions of the classical job shop problem have been proposed in the literature, including the so-called job shop with blocking constraints or *Blocking Job Shop* which arises for instance in manufacturing environments where there are no intermediate buffers between machines. In the Blocking Job Shop, a number of jobs have to be processed, each job involving a sequence of processing steps (operations) to be performed on some machines. An operation occupies a single dedicated machine for some given duration; preemption of operations is not allowed and a machine can process at most one operation at a time. In contrast to the classical job shop, the absence of buffers implies *blocking constraints* in the sense that a job has to stay on a machine (and is blocking it) until the next machine becomes available and the job can be transferred.

The problem considered here is a generalization of the Blocking Job Shop (BJS) and shall be called the Generalized Blocking Job Shop (GBJS). It includes the following additional features: (i) it takes into account *transfer times* for moving a job from one machine to the next machine, and (ii) it allows for *sequence-dependent setup times* between consecutive operations on a machine. The need for taking into account transfer times arises from applications where transferring a job between two machines requires some specific parallel handling on both machines, involving a *hand-over step* on the first machine and a *take-over step* on the second machine, which have to be synchronized and are assumed to have the same duration.

* Corresponding address: Institute of Data Analysis and Process Design (IDP), Zurich University of Applied Sciences (ZHAW), Rosenstrasse 3, P.O. Box, CH-8401 Winterthur, Switzerland. Tel.: +41 58 934 78 02; fax: +41 58 935 78 02.
E-mail address: andreas.klinkert@zhaw.ch (A. Klinkert).

0166-218X/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2009.02.020

In the GBJS, an operation of a job can therefore be viewed as a sequence of four consecutive steps (see also the example in Section 2): (i) a take-over step where the job is taken over from the machine that performed the job's previous operation (or where the job is loaded onto the machine if the operation is the job's first operation); (ii) a processing step where some work is performed on the job; (iii) a (possibly non-existent) waiting period during which the job waits on the machine – and blocks it – until being transferred; (iv) a hand-over step where the job is handed over to the machine performing the job's next operation (or where the job is unloaded, if the operation is the job's last operation). On each machine, a setup occurs before the first operation, between consecutive operations and after the last operation on that machine, and the durations of these setups are sequence-dependent. The GBJS problem consists in finding a feasible schedule of all operations that minimizes the makespan, i.e. the time by which all jobs are completed. Note that BJS is a special case of GBJS, where all transfer times are zero and there are no setups between operations.

Problems of GBJS type arise in practice in complex manufacturing systems as well as in logistics. Indeed, the motivation for the present work stems from the following application (see Klinkert [15]). Among the various types of storage technologies available are so-called automated high-density warehouses. These systems handle and store a large number of pallets. They comprise several floors, subdivided into corridors (where the pallets are stored), and perpendicular to them, cross-aisles linking the corridors to elevators. Pallet movements are automated and computer-controlled, and are executed by transfer carriages integrated in the corridors and cross-aisles, and elevators. As an example, the simplest sequence of operations for a pallet to be retrieved from its storage location to a picking area is the following: the corridor carriage moves to the location where the pallet is stored, loads the pallet and moves to an adjacent cross-aisle. If the cross-aisle carriage is there, the pallet is transferred from the corridor carriage to the cross-aisle carriage (synchronized hand-over/take-over step), otherwise the pallet stays on the corridor carriage (and is blocking it) until the cross-aisle carriage arrives and the pallet can be transferred. The cross-aisle carriage then moves to an elevator and (possibly after having waited for the elevator's arrival) hands the pallet over to the elevator which takes the pallet and brings it to the picking area where it is unloaded. Such a pallet retrieval corresponds to a job in the GBJS model. Its three consecutive operations, where the pallet is handled by the corridor carriage, cross-aisle carriage and elevator, respectively, feature steps (i)–(iv) detailed above. Observe also that sequence-dependent setups are essential in this application since the setup time between two consecutive operations represents the travel time of the carriage's idle move from the hand-over location of the first operation to the take-over location of the next operation.

Job shop models with blocking constraints (BJS) have been discussed by several authors. Hall and Sriskandarajah [12] give a survey on machine scheduling problems with blocking and no-wait constraints. Candar [5] describes several applications of machine scheduling with blocking and no-wait in process and reviews the computational complexity of a variety of related problems.

Mascis and Pacciarelli [17,18] study several types of job shop problems including the ideal (classical) job shop, the Blocking Job Shop (with and without "swaps") and the no-wait job shop, and formulate these problems by means of alternative graphs; a generalization of the disjunctive graph concept similar to the one proposed by Klinkert [15]. They develop three specialized dispatching heuristics for these job shop problems and present numerical results for a large number of benchmark instances. They also solve eighteen of the smaller (10×10) instances to optimality by means of a branch-and-bound method. Pacciarelli [24] uses alternative graphs to describe a complex scheduling problem in a steel-making plant and devises a dispatching heuristic to solve this problem.

Meloni, Pacciarelli and Pranzo [21] present a rollout metaheuristic for the ideal, blocking and no-wait job shop which is based on an alternative graph formulation. This rollout method basically corresponds to a constructive procedure which iteratively extends a partial schedule, represented by a partial selection of alternative arcs, to a complete schedule. At each extension step, all candidate arcs are evaluated according to a scoring function, and the arc with the best score is added to the partial selection. The basic idea for computing the scoring function is a look-ahead strategy based on dispatching procedures (called subheuristics). Any candidate arc is tentatively added to the current partial selection and this new selection is extended to a complete schedule using one or more subheuristics. The score of the candidate arc then corresponds to the objective value of the best complete schedule found. The authors present promising numerical results for the eighteen 10×10 benchmark instances solved to optimality by Mascis and Pacciarelli [18], showing substantial improvements of the results found by the dispatching methods of [18].

Mati et al. [19] investigate scheduling of flexible manufacturing systems, study a multi-resource job shop problem with blocking constraints, where every operation simultaneously requires several resources by a certain quantity, and propose a tabu search based on a geometric approach. The extended abstract of Mati et al. [20] describes a tabu search algorithm for BJS which applies classical permutation moves (i.e. exchange of adjacent operations on a machine) until no further permutation on the critical path leads to a feasible solution. A deadlock recovery move is then employed to emerge from the deadlock. It corresponds to a job rescheduling algorithm that uses an extension of the geometric approach described in [19]. The authors tested their algorithm on the 10×10 benchmark instances provided by Mascis and Pacciarelli [18] and indicated favorable results yielding an optimality gap "in the range [3%, 9%]", without giving detailed test data.

Brizuela et al. [3] develop a genetic algorithm for solving no-wait and Blocking Job Shop problems and show numerical results for four selected benchmark problems.

Klinkert [15] and Gröflin and Klinkert [10] introduce a generalized disjunctive graph framework for modeling various types of scheduling problems and develop a local search approach for the generalized Blocking Job Shop problem with application in automated warehouses (see the previous paragraph).

More recently, some extensions of the BJS have been addressed, e.g. by Brucker and Kampmeyer [4] on cyclic scheduling in the BJS and by Heitmann [13] on job shop scheduling with limited capacity buffers, and more applications related to the job (or flow) shop with blocking have been reported in process industries and logistics. We mention here e.g. scheduling in manufacturing of concrete blocks by Grabowski and Pempera [9], in steel-making by Pacciarelli and Pranzo [25], in chemical batch production by Romero et al. [27], in container handling at a port by Chen et al. [6] and in railway networks by D'Ariano et al. [7].

Most of the above authors emphasize that scheduling problems with blocking constraints (BJS) appear more difficult to solve than the classical job shop (JS). To illustrate the differences, two structural properties of the BJS shall be mentioned here, both being related to feasibility issues. First, in contrast to JS, a feasible partial schedule cannot always be extended to a feasible complete schedule. In fact, Mascis and Pacciarelli [17,18] established that deciding whether this is possible is NP-complete (this was also shown independently by Klinkert [15]). As a consequence, any heuristic that incrementally builds up a solution (e.g. based on priority rules) risks the chance of running into infeasibility. Second, it is not straightforward to construct feasible neighbor solutions in a local search approach as moves based on simple swaps of adjacent operations typically yield infeasible schedules (while in the JS case, it is well known that swapping critical adjacent operations leads to feasible neighbors).

A main part of the present paper is therefore dedicated to the construction of a new neighborhood for the GBJS providing always feasible neighbors. Based on this neighborhood, a tabu search algorithm is developed that consistently provides feasible solutions of competitive quality.

The paper is structured as follows. In the next section, we give a disjunctive programming formulation of the GBJS problem and show how it can be represented as an optimization problem in an associated disjunctive graph. A detailed example illustrates the formal problem description. In Section 3, we establish the theoretical tools and properties needed for developing the neighborhood and illustrate the concepts in the example. We then devise a tabu search in Section 4 and report on extensive computational experience in Section 5, before concluding with some remarks. The Appendix provides a proof on connectivity.

2. Problem formulation

Let M be the set of machines, $\mathcal{J}$ the set of jobs and I the set of operations. σ and τ are dummy start and finish operations of duration zero, having to be performed before, respectively after all operations. A job $J \in \mathcal{J}$ is identified with its set of operations, i.e. $J \subseteq I$ and $\mathcal{J} \subseteq 2^I$ is a partition of I. For ease of notation, the operations of a job J are denoted by simply indexing J with a subscript: the sequence of operations of job J is $J_1, J_2, \ldots, J_{|J|}$. For any job $J \in \mathcal{J}$, the set of ordered *pairs* of consecutive operations J_r and J_{r+1} is denoted by $A_J = \{(J_r, J_{r+1}) : r = 1, \ldots, |J| - 1\}$. For any machine $m \in M$, I_m is the set of operations using m.

In the GBJS, an operation $i \in I$ involves four consecutive steps: a take-over step with duration d_i^t, a processing step with duration $d_i > 0$, a waiting phase of unknown duration, and a hand-over step with duration d_i^h. For any operation $i \in I$, let $t(i)$ denote its take-over step and $h(i)$ its hand-over step.

For any $m \in M$ and $i, j \in I_m$, $i \neq j$, d_{ij}^s denotes the setup time when operation j immediately follows operation i on machine m. Also, for any $m \in M$ and $i \in I_m$, $d_{\sigma i}^s$ denotes the setup time before operation i if i is the first operation on machine m, and $d_{i\tau}^s$ is the setup time after operation i if i is the last operation on m. We assume that these setup times satisfy the triangle inequality (see the remark at the end of the section), i.e. $d_{ij}^s + d_{jk}^s \geq d_{ik}^s$ for any distinct $i, j, k \in I_m$, as well as for $i = \sigma$ and $j, k \in I_m, j \neq k$, respectively $i, j \in I_m, i \neq j$ and $k = \tau$.

Let $x_{t(i)}$ and $x_{h(i)}$ denote the starting time of the take-over step, respectively hand-over step of operation $i \in I$, and x_σ,x_τ the starting time of the dummy operations. The GBJS can now be formulated as the following disjunctive programming problem:

$$\text{Minimize } x_\tau \qquad \text{subject to:} \tag{1}$$

$$\begin{aligned} & x_{t(j)} - x_{h(i)} \geq d_i^h + d_{ij}^s \\ \text{OR } & x_{t(i)} - x_{h(j)} \geq d_j^h + d_{ji}^s \qquad \text{for all } \{i, j\} \subseteq I_m, m \in M \end{aligned} \tag{2}$$

$$\begin{aligned} & x_{t(j)} - x_{h(i)} \geq 0, \\ & x_{h(i)} - x_{t(j)} \geq 0 \qquad \text{for all } (i, j) \in A_J, J \in \mathcal{J} \end{aligned} \tag{3}$$

$$x_{h(i)} - x_{t(i)} \geq d_i^t + d_i \qquad \text{for all } i \in I \tag{4}$$

$$\begin{aligned} & x_{t(i)} - x_\sigma \geq d_{\sigma i}^s, \\ & x_\tau - x_{h(i)} \geq d_i^h + d_{i\tau}^s \qquad \text{for all } i \in I \end{aligned} \tag{5}$$

$$x_{t(i)}, x_{h(i)}, x_\sigma, x_\tau \geq 0 \qquad \text{for all } i \in I. \tag{6}$$

The objective function minimizes the makespan, given by the starting time of τ. The *disjunctive constraints* (2) ensure that no two operations i and j on machine m overlap in time, and that m is *blocked* during the execution of i and j: the hand-over step of operation i must precede the take-over step of operation j or vice versa. The *synchronization constraints* (3) enforce the action that a job is handed over properly from operation i to operation j following i: the starting times $x_{h(i)}$ and $x_{t(j)}$ of the hand-over step of i and the take-over step of j must be equal (equality is expressed by two inequalities). The *processing constraints* (4) ensure that for any operation i, the hand-over step of i takes place after the completion of the take-over and processing step of i. The *setup constraints* (5) take into account initial and final setup times.

As is the case for the classical job shop (Roy and Sussman [28]), GBJS can be formulated as an optimization problem in a (generalized) *disjunctive graph* $G = \langle V, A, E, \mathcal{E}, c\rangle$. In this notation, G can be viewed as an ordinary directed graph with node set V and arc set $A \cup E$, where A and E are the *conjunctive* and *disjunctive* (directed) *arcs*, respectively, and $c \in \mathfrak{R}^{A\cup E}$ defines the arc weights. In addition, the family $\mathcal{E} \subseteq 2^E$ of *disjunctive sets* is given which represents the disjunctive structure of G. Any disjunctive set $D \in \mathcal{E}$ contains exactly two disjunctive arcs, i.e. $D = \{e, \bar{e}\}$ where $\bar{e}$ is called the *mate* of e (and vice versa), and any disjunctive arc $e \in E$ is assumed to be in exactly one disjunctive set. Note that generalized disjunctive graphs differ from classical disjunctive graphs in that a disjunctive set may contain two *arbitrary* arcs, while in the classical case it contains two reverse arcs $e = (v, w)$ and $\bar{e} = (w, v)$ for some $v, w \in V$.

Similarly to classical disjunctive graphs, a subset of disjunctive arcs $S \subseteq E$ is called a *selection* in G. The *subgraph* $G(S)$ associated to S is the (ordinary) directed graph obtained from G by adding the disjunctive arcs of S to the conjunctive arcs A, i.e. $G(S) = (V, A \cup S, c)$. (For ease of notation, we write c instead of the restriction of c to $A \cup S$.) A selection S is *complete* in G if it intersects each disjunctive set in $\mathcal{E}$, and *positive acyclic* in G if $G(S)$ contains no cycle of positive weight. A selection is *feasible* in G if it is complete and positive acyclic in G.

The disjunctive graph associated to a GBJS instance is now defined as follows (see the example below). The node set V contains two nodes for each operation i, a *take-over node* t_i and a *hand-over node* h_i, so that $V = V^I \cup \{\sigma, \tau\}$ where $V^I = \{t_i, h_i : i \in I\}$. The set A of *conjunctive arcs* is given by $A = A^0 \cup A^1 \cup A^{\sigma,\tau}$, where A^0 is the set of *synchronization arcs*, linking for each job J consecutive operations $(i, j) \in A_J$ by a pair of opposite arcs (with weight zero), i.e. $A^0 = \{(h_i, t_j), (t_j, h_i) : (i, j) \in A_J, J \in \mathcal{J}\}$, A^1 is the set of *processing arcs*, joining the take-over node to the hand-over node of each operation, i.e. $A^1 = \{(t_i, h_i) : i \in I\}$, and $A^{\sigma,\tau}$ joins the dummy nodes σ and τ to all operations, i.e. $A^{\sigma,\tau} = \{(\sigma, t_i) : i \in I\} \cup \{(h_i, \tau) : i \in I\}$. The set E of *disjunctive arcs* contains for any two operations $i, j \in I_m$ on a machine m two arcs joining the hand-over node of i to the take-over node of j and vice versa, i.e. $E = \{(h_i, t_j), (h_j, t_i) : i, j \in I_m, i \neq j, m \in M\}$, a *disjunctive set* D consists of a pair $\{(h_i, t_j), (h_j, t_i)\}$ for some distinct $i, j \in I_m$ and the *family* $\mathcal{E} \subseteq 2^E$ *of disjunctive sets* is $\mathcal{E} = \{\{(h_i, t_j), (h_j, t_i)\} : i, j \in I_m, i \neq j, m \in M\}$. Finally, the arcs $e \in A \cup E$ have weight c_e defined as follows: $c_e = 0$ for $e \in A^0$, $c_e = d^t_i + d_i$ for $e = (t_i, h_i) \in A^1$, $c_e = d^s_{\sigma i}$ for $e = (\sigma, t_i) \in A^{\sigma,\tau}$, $c_e = d^h_i + d^s_{i\tau}$ for $e = (h_i, \tau) \in A^{\sigma,\tau}$ and $c_e = d^h_i + d^s_{ij}$ for $e = (h_i, t_j) \in E$.

It is easy to show that GBJS is equivalent to the following problem in G: "Among all feasible selections in G, find a selection S that minimizes the length of a longest path from σ to τ in $G(S)$."

Observe that any disjunctive set $D = \{(h_i, t_j), (h_j, t_i)\}$ is a positive cyclic selection, since (h_i, t_j), (h_j, t_i) together with arcs $(t_i, h_i), (t_j, h_j) \in A^1$ form a positive cycle in $G(D)$. Therefore, in any feasible selection, exactly one arc of each disjunctive set must be chosen. Note also that the disjunctive graph for the GBJS differs from a classical job shop graph in several features. Most importantly, each operation is represented by two nodes, a take-over and a hand-over node, and the constraint that no two operations on a machine may overlap, leads to disjunctive sets consisting of a pair of arcs with different extremities.

Before illustrating the GBJS in an example, two remarks on the formulations and on sequence-dependent setup times are in order.

First, formulation (1)–(6) as well as the disjunctive graph could be given in a more compact form. Clearly, (3) could be used to eliminate variables in (1)–(6). In parallel, in the disjunctive graph, any cycle formed by a pair of synchronization arcs (of length 0) can be contracted to a single node and the arc of any last operation of a job can be contracted to a single node (updating appropriately the weights of the arcs leaving it). This transformation leads to the more compact formulation via alternative graphs proposed by Mascis and Pacciarelli [17]. We believe that both the given graph formulation and the compact version are useful. The first is easy to derive and interpret (see also the example below). As in the classical job shop, any disjunctive pair of arcs occurs between two operations on the same machine, and making a selection amounts to deciding on how to sequence operations on machines. Synchronization arcs express constraints of synchronicity between events, here between the moments a job is handed over by a machine and taken over by its next machine. On the other hand, the compact formulation is better suited for implementation, since the contractions mentioned above make for more efficient longest path computation. The data structure implemented in our tabu search is such that the needed procedures (longest path and so-called closure computations) are performed in (subgraphs of) the compact formulation.

Second, sequence-dependent setup times should satisfy the triangle inequality for the disjunctive programming formulation to be valid. Otherwise, arcs from $e \in E \cup A^{\sigma,\tau}$ between non-consecutive operations may become active when computing longest paths in $G(S)$, yielding a wrong makespan since setups take place only between consecutive operations on a machine. However, the disjunctive graph model and the tabu search developed later can easily be adapted to handle *arbitrary* setup times, simply by ignoring all arcs from $E \cup A^{\sigma,\tau}$ that link non-consecutive operations, when computing longest paths and critical arcs in $G(S)$.

Example. We consider a GBJS instance with 3 machines $M = \{m_1, m_2, m_3\}$, 3 jobs $\mathcal{J} = \{J, K, L\}$ and 7 operations $I = \{1, 2, \ldots, 7\}$. The jobs are identified with their set of operations $J = \{1, 2\}$, $K = \{3, 4\}$ and $L = \{5, 6, 7\}$, where the sequence of operations for job L, for instance, is $L_1 = 5$, $L_2 = 6$ and $L_3 = 7$ and the set of pairs of consecutive operations

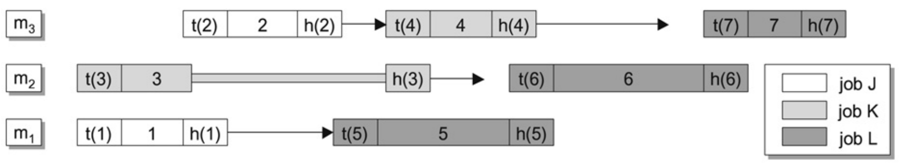

Fig. 1. Gantt diagram for the GBJS example.

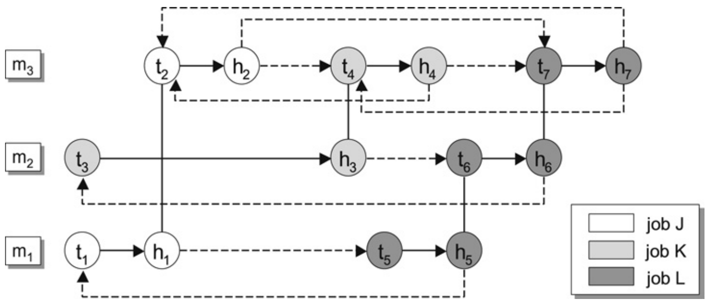

Fig. 2. Disjunctive graph for the GBJS example.

is $A_L = \{(5, 6), (6, 7)\}$. The operations processed on each machine are $I_{m_1} = \{1, 5\}$, $I_{m_2} = \{3, 6\}$ and $I_{m_3} = \{2, 4, 7\}$, i.e. job L for instance is first processed on m_1, then on m_2 and finally on m_3.

The numerical data are as follows: Durations d_i of processing steps $i \in I$ are $(d_1, \ldots, d_7) = (42, 47, 47, 42, 89, 102, 36)$, durations of take-over and hand-over steps are $d_i^t = d_i^h = 29$ for $i \in I$, and setup times to be taken into account are $d_{15}^s = 72$, $d_{36}^s = 54$, $d_{24}^s = 30$, $d_{47}^s = 113$, $d_{42}^s = 59$, $d_{27}^s = 19$ and $d_{51}^s = 25$.

Fig. 1 shows the Gantt diagram of a (semi-active) schedule for this GBJS instance. Each operation involves four consecutive steps: considering for instance job $K = \{3, 4\}$, its first operation 3 comprises (i) an initial loading step of duration d_3^t where the job is loaded onto machine m_2, (ii) a processing step of duration d_3, (iii) a waiting phase (drawn as a thin bar) where the job is blocking machine m_2 until it can be transferred to machine m_3, and (iv) a hand-over step of duration d_3^h where the job is transferred from m_2 to m_3. Operation 4 comprises a take-over step, a processing step, an empty waiting phase and a final unloading step. Note that the hand-over step of operation 3 is synchronized with the take-over step of operation 4. The duration of the (sequence-dependent) setup times between consecutive operations on a machine is indicated as arrows. For instance, the setup between operations 4 and 7 on machine m_3 ends before the processing step of operation 6 is completed on machine m_2; hence the waiting phase of operation 6 is empty.

The disjunctive graph $G = (V, A, E, \mathcal{E}, c)$ associated to this GBJS instance is shown in Fig. 2. For clarity, numerical data and dummy nodes σ, τ have been omitted, pairs of reverse (zero-weighted) synchronization arcs A^0 are drawn as *undirected* edges, and disjunctive arcs E are dashed. G is defined as follows: its node set is $V = \{t_i, h_i : i = 1, 2, \ldots, 7\} \cup \{\sigma, \tau\}$, and its set of conjunctive arcs is $A = A^0 \cup A^1 \cup A^{\sigma,\tau}$ where the zero-weighted synchronization arcs are given by $A^0 = \{(h_1, t_2), (t_2, h_1), (h_3, t_4), (t_4, h_3), (h_5, t_6), (t_6, h_5), (h_6, t_7), (t_7, h_6)\}$, the positive-weighted processing arcs by $A^1 = \{(t_i, h_i) : i = 1, 2, \ldots, 7\}$ and the σ, τ arcs by $A^{\sigma,\tau} = \{(\sigma, t_i), (h_i, \tau) : i = 1, 2, \ldots, 7\}$. The family of disjunctive sets is $\mathcal{E} = \{\{(h_1, t_5), (h_5, t_1)\}, \{(h_3, t_6), (h_6, t_3)\}, \{(h_2, t_4), (h_4, t_2)\}, \{(h_4, t_7), (h_7, t_4)\}, \{(h_2, t_7), (h_7, t_2)\}\}$ and the set of disjunctive arcs is $E = \bigcup_{D \in \mathcal{E}} D$. The schedule shown in Fig. 1 corresponds to the feasible selection $S = \{(h_1, t_5), (h_3, t_6), (h_2, t_4), (h_4, t_7), (h_2, t_7)\}$.

3. A neighborhood for local search

We develop a local search approach for the GBJS where in a generic step, a feasible selection S is given and feasible neighbor selections are generated by exchanging some disjunctive arc $e \in S$ with its mate $\bar{e}$, usually together with exchanges of some other arcs in order to recover feasibility. We shall need the concept of *closure* of a selection as well as a key result on *short cycles* in the disjunctive graph, in order to construct a neighbor selection and prove its feasibility.

Let $S \subseteq E$ be an arbitrary selection in G. The arcs *implied* by S are all disjunctive arcs that must be in *any* feasible selection containing S. Given $e \in E$, clearly, if $G(S \cup \bar{e})$ contains a positive cycle passing through $\bar{e}$, then e is implied by S. Thus, we define

$$\varphi(S) = S \cup \{e \in E : G(S \cup \bar{e}) \text{ contains a positive cycle through } \bar{e}\}.$$

Definition 1. Let $S \subseteq E$ be a selection in G. (i) S is said to be closed in G if $\varphi(S) = S$. (ii) The closure $\Phi(S)$ of S in G is the smallest closed selection in G containing S.

The closure $\Phi(S)$ can easily be computed recursively as follows. The rth iteration of $\varphi(S)$ is given by $\varphi^r(S) = S$ for $r = 0$ and $\varphi^r(S) = \varphi(\varphi^{r-1}(S))$ for $r \geq 1$. Obviously, there exists $r \leq |E|$ such that $\varphi^r(S) = \varphi^{r+1}(S)$. Then $\Phi(S) = \varphi^r(S)$.

We point out two properties which shall be used later. (i) If $S \subseteq S'$, then $\Phi(S) \subseteq \Phi(S')$. (ii) If S is a feasible selection, S is closed, i.e. $S = \varphi(S) = \Phi(S)$. Indeed, for any $e \notin S$, $\overline{e} \in S$ since S is complete, and $G(S \cup \overline{e}) = G(S)$ contains no positive cycle since S is positive acyclic. Therefore $\varphi(S) - S = \emptyset$.

Besides closures, the second ingredient is the existence of "short" positive cycles in G. We shall need the following notations. The extremities of an arc $e = (v, w)$ are denoted by tail $(e) = v$ and head $(e) = w$. For any job $J \in \mathcal{J}$, $V_J = \{t_i, h_i : i \in J\}$ is the node set associated to J, E_J^- and E_J^+ are the sets of disjunctive arcs entering and leaving J, respectively, i.e. $E_J^- = \{e \in E : \text{head}(e) = t_i \text{ for some } i \in J\}$ and $E_J^+ = \{e \in E : \text{tail}(e) = h_i \text{ for some } i \in J\}$, and $E_J = E_J^- \cup E_J^+$ is the set of disjunctive arcs incident with J. Further, for any selection S and node set $W \subseteq V$ in $G(S)$, $\delta(W)$ is the set of arcs with exactly one extremity in W and $\gamma(W)$ is the set of arcs with both extremities in W.

Given a selection $S \subseteq E$, assume now that $G(S)$ contains a positive cycle (with arc set) Z. Observe that for any job J, $Z \cap \delta(V_J) \subseteq E_J$ (i.e. Z enters and leaves J only through disjunctive arcs) since $\delta(V_J) \subseteq E \cup A^{\sigma,\tau}$ and arcs in $A^{\sigma,\tau}$ cannot be contained in any cycle. Furthermore, $|Z \cap E_J^-| = |Z \cap E_J^+|$ since Z enters and leaves J the same number of times. Let $\Delta(Z, J)$ be this number, which can be seen as the number of times Z *visits* J. The following result asserts the existence of "short" positive cycles in $G(S)$.

Theorem 2. *For any positive cycle Z in $G(S)$, there exists a "short" positive cycle Z' with $Z' \cap E \subseteq Z \cap E$ and visiting each job at most once.*

Proof. Let Z be a positive cycle in $G(S)$. If $\Delta(Z, J) \leq 1$ for all $J \in \mathcal{J}$, we are done. Otherwise, let $J \in \mathcal{J}$ with $\Delta(Z, J) = |Z \cap E_J^-| > 1$. We will construct a cycle Z' with $Z' \cap E \subseteq Z \cap E$ and visiting J only once.

For any $e \in E_J^-$, define the rank $r(e)$ to be the order of the operation of J to which e is incident, i.e. $r(e) = s$ if head$(e) = J_s$, $s \in \{1, \ldots, |J|\}$. Obviously, all $e \in Z \cap E_J^-$ have distinct ranks. Among them, let g be the arc of highest rank. Starting from head$(g) \in V_J$, traverse Z until it enters the first time V_J through arc, say $e \in Z \cap E_J^-$. The path traversed from head(g) to head(e) can be written as the concatenation (P, f, Q, e) where subpath $P \subseteq \gamma(V_J)$ starts in head(g), $f \in Z \cap E_J^+$, and $Q \subseteq \gamma(V - V_J)$ ends in tail(e). Now, since $r(g) > r(e)$, there is a path R in $\gamma(V_J)$ from head(e) to head(g), with the first arc (t_i, h_i) if $t_i =$ head(e). Form the following cycle Z'. Starting from head(e), traverse R until encountering a node of P, then continue on P, f, Q and e. Since arc (t_i, h_i) in Z' has positive weight, Z' has positive length. Also, since Z' enters V_J (through e), respectively leaves V_J (through f) exactly once, $\Delta(Z, J) = 1$. Finally, by the construction of Z', $Z' \cap E \subseteq Z \cap E$, so that $\Delta(Z', J') \leq \Delta(Z, J')$ for all $J' \in \mathcal{J}$, proving the theorem. ■

Given now a feasible selection S in G, a neighbor is generated by selecting some arc $e \in S$, a job J "incident" with e, and rescheduling job J (or a part of that job) in such a way that arc e is exchanged with $\overline{e}$, possibly together with exchanges of other arcs incident with J. Note that there are two jobs incident with $e = (h_i, t_j)$, denoted as $J^{e,T}$ and $J^{e,H}$, where $J^{e,T}$ is incident with the tail of e, i.e. $i \in J^{e,T}$, and $J^{e,H}$ with the head of e, i.e. $j \in J^{e,H}$.

Specifically, given $e \in S$ and $J \in \{J^{e,T}, J^{e,H}\}$, let $R = S - E_J$ be the set of selected disjunctive arcs not incident with J, and for any $U \subseteq E$, denote by $E(U) = \{e \in E : \{e, \overline{e}\} \cap U \neq \emptyset\}$ the set of all disjunctive arcs contained in disjunctive sets intersected by U. The following result whose proof relies on Theorem 2, shows how to construct a neighbor $S_{e,J}$ with $e \notin S_{e,J}$ and $\overline{e} \in S_{e,J}$.

Theorem 3. *Construct $Q = \Phi(R \cup \overline{e})$. Then $S_{e,J} = Q \cup (S - E(Q))$ is a feasible neighbor selection of S.*

Proof. Suppose $J = J^{e,T}$, i.e. $e \in E_J^+$ and $\overline{e} \in E_J^-$ (the proof for $J = J^{e,H}$ is analogous). We first show that $Q = \Phi(R \cup \overline{e})$ is positive acyclic in G. Consider the (complete) selection $R \cup E_J^-$ which corresponds to the solution where J is scheduled after all other operations. Obviously, $R \cup E_J^-$ is positive acyclic. Indeed, suppose there is a positive cycle Z in $G(R \cup E_J^-)$. Since Z is not contained in $G(R)$, $Z \cap E_J^- \neq \emptyset$, hence $Z \cap E_J^+ \neq \emptyset$, contradicting $(R \cup E_J^-) \cap E_J^+ = \emptyset$. Thus $R \cup E_J^-$ is feasible and therefore closed. From $R \cup \overline{e} \subseteq R \cup E_J^-$, it follows that $\Phi(R \cup \overline{e}) \subseteq \Phi(R \cup E_J^-) = R \cup E_J^-$, so that $\Phi(R \cup \overline{e})$ is positive acyclic.

Let $\widehat{S} = S - E(Q)$. We now show that $Q \cup \widehat{S}$ is feasible. It is complete since $\widehat{S}$ intersects all disjunctive sets not intersected by Q. Suppose that $G(Q \cup \widehat{S})$ contains a positive cycle Z. By Theorem 2, there exists a "short" positive cycle Z' visiting job J at most once, i.e. $\Delta(Z', J) \leq 1$. Since Q is positive acyclic, $Z' \cap \widehat{S} \neq \emptyset$, and since $\widehat{S} \subseteq E_J$, Z' visits J, hence $\Delta(Z', J) = 1$, i.e. $|Z' \cap E_J| = 2$. Let $Z' \cap E_J = \{f, g\}$, i.e. $G(R \cup \{f, g\})$ contains Z'. Since $R \cup \widehat{S} \subseteq S$ is positive acyclic, $|\widehat{S} \cap \{f, g\}| \leq 1$, and since Q is positive acyclic and $R \subseteq Q$, $|Q \cap \{f, g\}| \leq 1$. Thus, since $\{f, g\} \subseteq Q \cup \widehat{S}$, $f \in \widehat{S}$ and $g \in Q$. But then, since $G(Q \cup f)$ contains the positive cycle Z' through f, $\overline{f} \in \Phi(Q) = Q$, hence $\{f, \overline{f}\} \subseteq E(Q)$, contradicting $f \in \widehat{S} = S - E(Q)$. ■

Neighbor $S_{e,J}$ of S is obtained by imposing the condition that some $e \in S$ be replaced by its mate $\overline{e}$. If $e = (h_i, t_j)$, $\overline{e} = (h_j, t_i)$ and operation i, positioned in S before operation j on a processor, is positioned after j in $S_{e,J}$. Disjunctive arcs besides e might also be changed since all arcs of $\Phi(R \cup \overline{e})$ are implied by imposing $\overline{e}$. Suppose $J = J^{e,T}$ so that $i \in J$ and $\Phi(R \cup \overline{e}) - R \subseteq E_J^-$. Then i is moved to the "right", as well as possibly other operations of J. If $J = J^{e,H}$, $j \in J$ is moved to the "left", possibly with other operations of J.

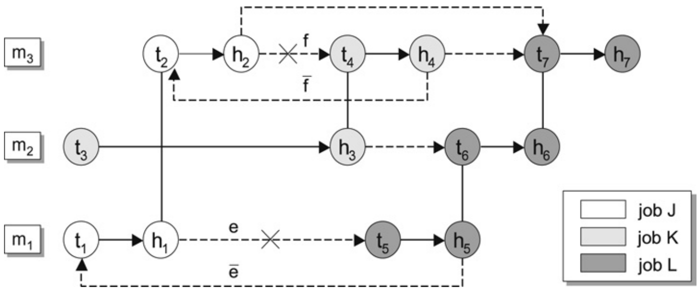

Fig. 3. Graph $G(S_{e,J})$ for neighbor selection $S_{e,J}$.

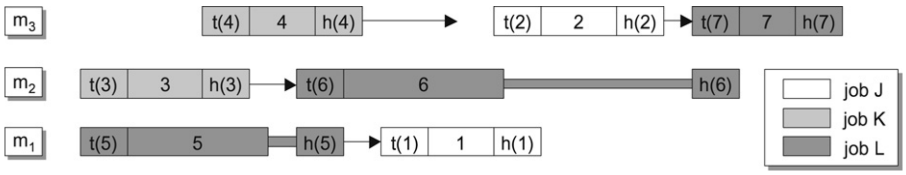

Fig. 4. Gantt diagram for neighbor selection $S_{e,J}$.

Observe that $S_{e,J}$ is a neighbor *closest* to S in the sense that $S_{e,J} - S \subset S' - S$ for any feasible selection S' containing $R \cup \bar{e}$ and distinct from $S_{e,J}$. Hence, the number of exchanged arcs in $S_{e,J}$ is *minimal* among all neighbors obtained from S by exchanging e with $\bar{e}$ and allowing some rescheduling of job J.

Theorem 3 defines a non-empty neighborhood $\mathcal{N}(S)$ of feasible neighbors defined for any feasible selection S:

$$\mathcal{N}(S) = \{S_{e,J} : e \in S, J \in \{J^{e,T}, J^{e,H}\}\}, \text{ where} \tag{7}$$

$$S_{e,J} = Q \cup (S - E(Q)) \tag{8}$$

$$Q = \Phi(\bar{e} \cup (S - E_J)) \tag{9}$$

$$E(Q) = \{e \in E : \{e, \bar{e}\} \cap Q \neq \emptyset\}. \tag{10}$$

In the Appendix, we show that this neighborhood is connected.

Example (*Continued*). Given the feasible selection $S = \{(h_1, t_5), (h_3, t_6), (h_2, t_4), (h_4, t_7), (h_2, t_7)\}$ corresponding to the schedule shown in Fig. 1, we shall now construct a feasible neighbor selection of S, based on Theorem 3. First, a critical disjunctive arc $e \in S$ is chosen to be exchanged with its mate $\bar{e}$. Let $e = (h_1, t_5)$. Replacing e by $\bar{e} = (h_5, t_1)$ corresponds to a swap of operations 1 and 5. Observe that simply swapping these two operations leads to an *infeasible* schedule since the resulting selection $S - e \cup \bar{e}$ yields a positive cycle (with a node sequence) $C = (t_1, h_1, t_2, h_2, t_4, h_3, t_6, h_5)$ in $G(S)$.

In order to obtain a *feasible* neighbor selection, one of the two jobs $J^{e,T}$ and $J^{e,H}$ incident to e is chosen to be rescheduled. Let $J = J^{e,T}$ be the chosen job. First, all arcs incident to J are removed from S, yielding $R = S - E_J = \{(h_3, t_6), (h_4, t_7)\}$, and the mate $\bar{e}$ is added, yielding $R \cup \bar{e} = \{(h_3, t_6), (h_4, t_7), (h_5, t_1)\}$. Then the *closure* $Q = \Phi(R \cup \bar{e})$ is constructed in order to determine additional disjunctive arcs implied by $R \cup \bar{e}$. Since the disjunctive arc $f = (h_2, t_4)$ together with $R \cup \bar{e}$ forms a positive cycle in $G(R \cup \{\bar{e}, f\})$ (corresponding to the cycle C mentioned above), $\bar{f} = (h_4, t_2)$ is contained in the closure. No other arcs forming a positive cycle with $R \cup \{\bar{e}, \bar{f}\}$ can be found and the closure is therefore given by $Q = \Phi(R \cup \bar{e}) = \{(h_3, t_6), (h_4, t_7), (h_5, t_1), (h_4, t_2)\}$. Finally, the *neighbor selection* $S_{e,J} = Q \cup (S - E(Q))$ is obtained. By definition, $E(Q) = \{e \in E : \{e, \bar{e}\} \cap Q \neq \emptyset\}$, hence $E(Q) = \{(h_3, t_6), (h_6, t_3), (h_4, t_7), (h_7, t_4), (h_5, t_1), (h_1, t_5), (h_4, t_2), (h_2, t_4)\}$ and $S - E(Q) = \{(h_2, t_7)\}$. Thus, $S_{e,J}$ is given by $S_{e,J} = \{(h_3, t_6), (h_4, t_7), (h_5, t_1), (h_4, t_2), (h_2, t_7)\}$. Fig. 3 shows the conjunctive graph $G(S_{e,J})$ associated to the neighbor selection $S_{e,J}$, and the corresponding (semi-active) schedule is drawn in Fig. 4. Observe that operations 1 and 5 are swapped together with operations 2 and 4, corresponding to the exchange of $e, \bar{e}$ and $f, \bar{f}$.

Note that a different feasible neighbor selection $S_{e,J}$ can be constructed by choosing job $J = J^{e,H}$ to be rescheduled instead of $J^{e,T}$. Then $R = \{(h_2, t_4)\}$, $R \cup \bar{e} = \{(h_2, t_4), (h_5, t_1)\}$, $Q = \Phi(R \cup \bar{e}) = \{(h_2, t_4), (h_5, t_1), (h_6, t_3), (h_7, t_4)\}$, $S - E(Q) = (h_2, t_7)$ and $S_{e,J} = Q \cup (h_2, t_7)$. The corresponding schedule contains three swaps, namely operations 1 and 5, 3 and 6, as well as 4 and 7.

4. Tabu search algorithm

General features of our tabu search for GBJS are similar to the approach proposed by Nowicki and Smutnicki [23] for the classical job shop, although the moves are different and more involved than those in the job shop case, as detailed in the previous section. For a general introduction to tabu search, see for instance Glover and Laguna [8].

Our tabu search algorithm is based on the *neighborhood* $\mathcal{N}$ defined in (7)–(10), but moves are restricted to neighbors associated to critical arcs. More precisely, the following restricted neighborhood $\mathcal{N}^c(S) = \{S_{e,J} : e \in \Lambda(S), J \in \{J^{e,T}, J^{e,H}\}\}$ has been implemented, where $\Lambda(S)$ denotes the set of all critical arcs in a feasible selection S, and arc $e \in S$ is said to be critical if it is on a longest path from σ to τ in $G(S)$. $\mathcal{N}^c$ has been proven to be computationally adequate, based on extensive numerical experience, but we do not know whether $\mathcal{N}^c(S)$ is opt-connected (see the Appendix for the definitions of connectivity and opt-connectivity). Numerical tests have also shown that it is sufficient to consider only selections $S_{e,J} \in \mathcal{N}^c(S)$ for which $J = J^{e,H}$, i.e. J is incident with the head of e.

At any iteration of the tabu search, a set (or list) of candidate solutions $\mathcal{N}'(S) \subseteq \mathcal{N}^c(S)$ is evaluated. The *candidate list strategy* is as follows: Determine a longest path P from σ to τ in $G(S)$ and let $\Lambda_P(S)$ be the set of all disjunctive arcs on P. Then the candidate set is given by $\mathcal{N}'(S) = \{S_{e,J} : e \in \Lambda_P(S), J \in \{J^{e,T}, J^{e,H}\}\}$.

A *tabu list* of fixed length *maxt* is maintained to keep track of the last moves executed. After the execution of a move defined by (e, J) with $J \in \{J^{e,T}, J^{e,H}\}$, i.e. after a move to neighbor $S^{e,J}$, arc $\bar{e}$ is appended to the tabu list and the first list entry is dropped. The presence of arc $\bar{e}$ in the tabu list means that both moves associated to $\bar{e}$, i.e. $(\bar{e}, J^{\bar{e},T})$ and $(\bar{e}, J^{\bar{e},H})$, are tabu. The idea behind is that a move (e, J) with $e = (t_i, h_j)$ swaps the two operations i and j, and swapping them back soon after by a move $(\bar{e}, J)$ should be avoided.

Evaluation of the candidate set comprises the following steps: For all candidates $S_{e,J} \subseteq \mathcal{N}'(S)$, calculate the makespan $\lambda(S_{e,J})$. If e is contained in the tabu list and $\lambda(S_{e,J})$ does not improve the best makespan found so far ("improved-best aspiration criterion"), remove candidate $S_{e,J}$ from the candidate set. Take the best of the remaining candidates as the next selection. If all candidates have been removed (i.e. all candidates are tabu and not globally improving), take the candidate corresponding to the oldest tabu move as the next selection.

In addition to this short term memory tabu search, an *intensification strategy* based on longer term memory is used. A list of bounded length *maxl* containing historically found *elite solutions* is kept, and new solutions encountered during the search are appended to the list if they are better than any previously encountered. If the tabu search runs for a specified number of iterations (*maxiter*) without improving the best solution (or if a cycle is detected), the current search path is terminated and the search is resumed with the last elite solution in the list. An elite solution is stored together with its associated tabu list and the moves already taken from the solution. When resuming search from an elite solution, its associated tabu list is restored and the moves previously taken are removed from the candidate set. An elite solution is dropped from the list if its candidate set is empty, i.e. all search paths starting from this solution have been explored.

Finally, a *cycle detection procedure* (similar to Nowicki and Smutnicki [23]) is used that keeps track of the sequence of makespans encountered during the iterations and scans this sequence for cycles (i.e. repeated subsequences of values).

5. Computational results

Extensive computational tests on a set of 81 benchmark instances have been performed for both the classical Blocking Job Shop (BJS) and GBJS. The instances for BJS are obtained from the following standard job shop benchmark problems: *abz5-9* proposed by Adams et al. [1], *ft06/10/20* by Fisher and Thompson (in Muth and Thompson [22]), *la01-40* by Lawrence [16], *orb01-06/08-10* by Applegate and Cook [2], *swv01-20* by Storer et al. [29], and *yn1-4* by Yamada and Nakano [30]. Since GBJS has not yet been considered in the literature, we created a new collection of benchmark instances for this problem, based on the instances for BJS. We added to each instance randomly generated transfer times and sequence-dependent setup times, uniformly distributed in the intervals [1, 20] and [0, 50], respectively.

The tabu search has been implemented in C++ and tested under Microsoft Windows XP on an Intel Pentium IV - 2.8 GHz platform with 512 MB of physical memory.

Starting solutions for BJS are obtained by a randomized dispatching heuristic of Brizuela et al. [3]. For GBJS, this heuristic almost always runs into infeasibility; therefore random permutation schedules are used as starting solutions.

The tabu search parameters have been set identical for all numerical experiments: $maxt = 8$, $maxl = 300$ and $1000 \leq maxiter \leq 2500$, where *maxiter* decreases with increasing length of the elite solution list. For each instance, five independent runs have been executed with different starting solutions, and the time limit for each run was 1800 s.

The numerical results for BJS have been compared to the results of Mascis and Pacciarelli [17,18], Meloni, Pacciarelli and Pranzo [21] and Brizuela et al. [3] which, to the best of our knowledge, are the only contributions containing explicit computational results for BJS. Mascis and Pacciarelli (MP) presented results for 59 BJS problems (a subset of our 81 benchmark instances). They applied three specialized dispatching heuristics that often – but not always – find a feasible solution. Using all three heuristics, they found solutions for 53 of the 59 instances. For eighteen of the smaller (10×10) instances, they also computed the optimal solution with a branch-and-bound algorithm. Meloni, Pacciarelli and Pranzo (MPP) tested their rollout metaheuristic on the eighteen 10×10 instances provided by MP. They found feasible solutions for seventeen of the eighteen instances, showing substantial improvements of the heuristic results given by MP. Unfortunately, MPP do not mention how their algorithm performs on larger instances and whether the method is able to construct feasible solutions for larger instances.

Brizuela et al. [3] proposed a genetic algorithm for BJS. They offer however very sparse computational results on only 4 instances (on which our tabu search performs better). For this reason, only a comparison with the results of MP and MPP is reported.

Table 1
Summary of tabu search results for BJS and GBJS.

Instances		BJS						GBJS				
Size	*#inst*	*#bench*	*%bench*	*iter*	300 s	600 s	1200 s	*%init*	*iter*	300 s	600 s	1200 s
6 × 6	1	1	1.59	14755	0.00	0.00	0.00	29.14	3724	0.00	0.00	0.00
10 × 5	5	5	40.30	53914	0.55	0.07	0.00	82.41	19352	0.00	0.00	0.00
15 × 5	5	5	42.30	99344	2.38	1.40	0.69	86.06	55924	0.00	0.00	0.00
20 × 5	6	6	44.88	59571	2.95	1.31	0.40	79.52	91325	0.35	0.00	0.00
10 × 10	17	16(17)	2.11	63154	2.93	1.55	0.35	130.61	74716	0.28	0.03	0.00
15 × 10	5	5	80.10	24636	5.67	3.08	1.67	168.72	90085	4.17	1.06	0.30
20 × 10	10	4(5)	64.76	12967	4.95	2.92	1.12	112.46	51461	5.59	2.66	0.90
30 × 10	5	1(5)	42.31	4416	5.69	4.17	1.77	142.23	23814	10.67	7.47	3.35
50 × 10	10	0	–	1302	5.05	2.62	0.63	79.01	8019	3.66	2.41	1.06
15 × 15	5	5	71.58	9768	6.92	4.24	1.44	224.67	49448	5.03	2.12	0.73
20 × 15	8	1(3)	59.52	4951	7.05	4.17	1.85	116.08	27286	6.97	4.23	1.79
20 × 20	4	0	–	1877	10.65	6.58	1.69	174.84	17494	12.49	7.25	2.75

Table 1 summarizes our numerical results for BJS and GBJS. Each line corresponds to a group of test instances of the same dimension $n \times m$, where n and m are the numbers of jobs and machines, respectively. Column '*#inst*' shows the number of instances in the group. The values in each line are averaged over the 5 independent tabu runs for each instance and over all instances of the group.

The results for BJS are compared to the best solutions found by MP or MPP: column '*#bench*' shows the number of instances in the group that MP could solve, and in parenthesis (if different) the number of instances they tried to solve. Column '*%bench*' gives the mean relative gap between the best solution found by MP or MPP ('bench') and our best tabu solution ('best'), i.e. '$(bench - best)/best * 100$'. Column '*iter*' indicates the average number of iterations of the tabu search. Columns '300 s', '600 s' and '1200 s' show intermediate tabu results after 300, 600 and 1200 s, respectively: the value in column '300 s' for instance corresponds to the mean relative gap between the best tabu solution found after 300 s and the final tabu solution, i.e. '$(300s - tabu)/tabu * 100$'.

The results for GBJS are presented similarly. Since there are no benchmark results available for comparison, the tabu results are compared to the makespan of the initial solution: column '*%init*' shows the mean relative gap between the initial solution ('init') and the tabu solution, i.e. '$(init - tabu)/tabu * 100$'.

In summary, our solutions for BJS are about 40%–80% better than MP's solutions (see '*%bench*'), and for the 10×10 instances, they are of similar quality as MPP's solutions. Furthermore, the table shows that most of the improvement in the tabu search is already achieved before the set time limit of 1800 s (see '300 s', '600 s', '1200 s'). For GBJS, the mean improvement of the initial solution is around 80%–220%.

Tables 2 and 3 give detailed results for the BJS and GBJS instances, respectively. Columns '*init*' and '*final*' show for each instance the mean initial and final makespan of our tabu search, averaged over the five independent runs (with different starting solutions). Column '*best*' gives the best makespan found among these five tabu solutions. For BJS, column '*bench*' shows the best makespan found by MP or MPP (results obtained by MPP are flagged with an asterisk, and 'ns' indicates that the instance could not be solved by MP), and column '*opt*' gives the optimal solutions provided by MP for the eighteen 10×10 instances.

Comparing our tabu solutions for these 10×10 instances to the optimal solutions provided by MP, the mean relative gap from the optimum, given by '$(best - opt)/opt * 100$', is 6.48% for our solutions, while this gap is 5.99% and 70.79% for MPP's and MP's solutions, respectively.

6. Concluding remarks

Solving job shop problems with blocking constraints is a challenge since approaches developed for the classical job shop do not seem to easily extend to this problem. Constructive heuristics risk the chance of running into infeasibility. Local search procedures based on simple swaps of operations produce infeasible solutions. Also, bottleneck approaches appear problematic since sequencing a single machine while fixing operation sequences on other machines typically yields infeasible schedules.

We developed a neighborhood for the Generalized Blocking Job Shop that always generates feasible neighbors, based on the exchange of critical arcs, closures and job reinsertion. In order to prove feasibility of these neighbors, we established a key theorem on short cycles in the disjunctive graph. We integrated the neighborhood in a tabu search method and presented numerical results that compare quite favorably with benchmark results, when available. In addition, the constructive heuristics establishing these benchmarks fail sometimes, although infrequently, to terminate with a feasible solution while our method always maintains feasibility.

The structural results of this paper are also useful in other contexts. We used for instance in Gröflin and Klinkert [11] the short cycle property in deriving polyhedral descriptions of the feasible job and block insertions for various extensions of the job shop model. Furthermore, in view of the general nature of the tools used, the approach presented for designing a neighborhood for local search is applicable to other complex scheduling problems.

Table 2
Detailed tabu search results for BJS.

Instances		Tabu			MP, MPP*	
Inst	Size	*init*	*final*	*best*	*bench*	*opt*
abz5	10 × 10	2393	1639	1626	*1595	1468
abz6	10 × 10	2109	1287	1241	*1222	1145
abz7	20 × 15	1924	1273	1224	ns	–
abz8	20 × 15	2034	1263	1226	ns	–
abz9	20 × 15	1827	1290	1166	1860	–
ft06	6 × 6	92	63	63	64	–
ft10	10 × 10	1821	1136	1103	*1144	1068
ft20	20 × 5	2239	1527	1495	2047	–
la01	10 × 5	1272	836	832	1137	–
la02	10 × 5	1142	824	793	1055	–
la03	10 × 5	1152	765	747	1088	–
la04	10 × 5	1233	774	769	971	–
la05	10 × 5	1009	711	698	1116	–
la06	15 × 5	1815	1191	1180	1493	–
la07	15 × 5	1582	1119	1091	1737	–
la08	15 × 5	1758	1161	1125	1569	–
la09	15 × 5	1814	1269	1223	1943	–
la10	15 × 5	1910	1222	1203	1533	–
la11	20 × 5	2378	1615	1584	2189	–
la12	20 × 5	2052	1433	1391	2122	–
la13	20 × 5	2180	1576	1548	2296	–
la14	20 × 5	2340	1648	1620	2423	–
la15	20 × 5	2303	1667	1650	2371	–
la16	10 × 10	1797	1175	1142	*1109	1060
la17	10 × 10	1662	1040	1026	*982	929
la18	10 × 10	1683	1112	1078	*1114	1025
la19	10 × 10	1797	1124	1093	*1115	1043
la20	10 × 10	1960	1184	1154	*1118	1060
la21	15 × 10	2645	1617	1545	2701	–
la22	15 × 10	2456	1525	1458	2566	–
la23	15 × 10	2524	1645	1611	3044	–
la24	15 × 10	2592	1623	1571	3350	–
la25	15 × 10	2505	1541	1499	2211	–
la26	20 × 10	3294	2182	2162	4024	–
la27	20 × 10	3449	2258	2175	ns	–
la28	20 × 10	3369	2186	2071	3070	–
la29	20 × 10	3200	2161	2124	3792	–
la30	20 × 10	3494	2199	2171	3173	–
la31	30 × 10	4845	3266	3167	ns	–
la32	30 × 10	5139	3549	3418	ns	–
la33	30 × 10	4617	3225	3131	ns	–
la34	30 × 10	4809	3354	3205	4561	–
la35	30 × 10	4833	3445	3311	ns	–
la36	15 × 15	3178	1974	1932	3369	–
la37	15 × 15	3314	2133	2053	3009	–
la38	15 × 15	3123	1939	1875	3187	–
la39	15 × 15	3099	1987	1950	3787	–
la40	15 × 15	3273	1982	1936	3345	–
orb01	10 × 10	2015	1300	1268	*1287	1175
orb02	10 × 10	1836	1128	1092	*1110	1041
orb03	10 × 10	1996	1226	1203	*1188	1160
orb04	10 × 10	2489	1237	1203	*1223	1146
orb05	10 × 10	1688	1102	1083	*1083	995
orb06	10 × 10	2275	1302	1265	*1278	1199
orb08	10 × 10	1671	1063	1028	1496	995
orb09	10 × 10	2182	1138	1127	*1046	1039
orb10	10 × 10	1921	1214	1181	*1153	1146
swv01	20 × 10	3439	2050	2001	–	–
swv02	20 × 10	3283	2159	2069	–	–
swv03	20 × 10	3358	2099	2029	–	–
swv04	20 × 10	3606	2183	2156	–	–
swv05	20 × 10	3359	2177	2108	–	–
swv06	20 × 15	4557	2762	2612	–	–
swv07	20 × 15	4332	2643	2610	–	–
swv08	20 × 15	4502	2995	2878	–	–
swv09	20 × 15	4210	2743	2669	–	–
swv10	20 × 15	4348	2881	2675	–	–
swv11	50 × 10	8120	5425	5215	–	–
swv12	50 × 10	7977	5778	5623	–	–
swv13	50 × 10	8193	5414	5256	–	–
swv14	50 × 10	7965	5335	5231	–	–
swv15	50 × 10	7749	5384	5061	–	–
swv16	50 × 10	8055	5614	5376	–	–
swv17	50 × 10	7860	5677	5368	–	–
swv18	50 × 10	7823	5685	5544	–	–
swv19	50 × 10	7679	6041	5830	–	–
swv20	50 × 10	7683	5782	5545	–	–
yn1	20 × 20	2609	1855	1699	–	–
yn2	20 × 20	2514	1800	1697	–	–
yn3	20 × 20	2640	1813	1688	–	–
yn4	20 × 20	2852	1938	1777	–	–

The present paper is a substantial expansion of the unpublished report "The Synchronized Job Shop Problem: Local Search in Generalized Disjunctive Graphs", presented at the Integer Programming Conference in honor of Egon Balas, Pittsburgh, 2002. In particular, the tabu approach presented here replaces the descent method used initially.

Acknowledgements

We thank the anonymous referees for their helpful comments and constructive suggestions.

Appendix

In a local search approach, a neighborhood $\mathcal{N}$ is defined which associates to a solution S a subset of neighbor solutions $\mathcal{N}(S)$. A transition from S to a neighbor $S' \in \mathcal{N}(S)$ is a move. $\mathcal{N}$ is said to be *connected* if, given two arbitrary solutions S and T, it is possible to proceed from S to T through a sequence of moves in $\mathcal{N}$, i.e. there exists a sequence $S^0, \ldots, S^l$ such that $S^0 := S, S^k \in \mathcal{N}(S^{k-1}), 1 \le k \le l$, and $S^l = T$. A neighborhood $\mathcal{N}$ is *opt-connected* if, given an arbitrary solution S, it is possible, starting from S, to reach some optimal solution through a sequence of moves in $\mathcal{N}$, i.e. there is a sequence $S^0, \ldots, S^l$ such that $S^0 := S, S^k \in \mathcal{N}(S^{k-1}), 1 \le k \le l$, and S^l is optimal.

In what follows, we show that the neighborhood $\mathcal{N}$ defined in Section 3 is connected. Whether $\mathcal{N}^c$ is opt-connected or not, remains however an open question.

Theorem 4. *The neighborhood $\mathcal{N}$ defined by (7)–(10) is connected, and for any pair of feasible selections S, T, at most $|E|$ moves are necessary to proceed from S to T.*

The proof is based on two lemmas. Given two feasible selections S and T, the first lemma shows how to proceed from S to a feasible selection S' that is closer to T than S by rescheduling some job. The second lemma ensures that the transition from S to S' can be done with moves in $\mathcal{N}$.

Table 3
Detailed tabu search results for GBJS.

Instances		Tabu			Instances		Tabu		
Inst	Size	*init*	*final*	*best*	Inst	Size	*init*	*final*	*best*
abz5	10 × 10	7480	2562	2462	la34	30 × 10	15304	6618	6242
abz6	10 × 10	5963	2117	2075	la35	30 × 10	15424	6312	5990
abz7	20 × 15	9387	3669	3393	la36	15 × 15	11569	3696	3458
abz8	20 × 15	9312	3713	3431	la37	15 × 15	12457	3808	3533
abz9	20 × 15	9331	3606	3229	la38	15 × 15	11544	3571	3392
ft06	6 × 6	740	573	565	la39	15 × 15	11458	3587	3400
ft10	10 × 10	4172	2380	2290	la40	15 × 15	11743	3496	3307
ft20	20 × 5	4506	2989	2906	orb01	10 × 10	3640	2239	2170
la01	10 × 5	3075	1570	1494	orb02	10 × 10	5374	2096	2030
la02	10 × 5	2812	1579	1531	orb03	10 × 10	3479	2301	2135
la03	10 × 5	2530	1497	1467	orb04	10 × 10	5455	2222	2154
la04	10 × 5	2602	1470	1439	orb05	10 × 10	5071	2125	2021
la05	10 × 5	2680	1399	1342	orb06	10 × 10	4323	2310	2245
la06	15 × 5	4288	2259	2232	orb08	10 × 10	2953	1995	1925
la07	15 × 5	3873	2274	2156	orb09	10 × 10	4930	2046	2004
la08	15 × 5	4292	2201	2040	orb10	10 × 10	5254	2303	2233
la09	15 × 5	4652	2341	2293	swv01	20 × 10	6330	3825	3520
la10	15 × 5	4122	2349	2271	swv02	20 × 10	6369	3886	3772
la11	20 × 5	5599	3121	3081	swv03	20 × 10	6625	3891	3783
la12	20 × 5	5095	2873	2793	swv04	20 × 10	6383	3986	3906
la13	20 × 5	5581	2858	2756	swv05	20 × 10	6570	4030	3830
la14	20 × 5	5630	2935	2813	swv06	20 × 15	9152	4693	4550
la15	20 × 5	5415	2981	2866	swv07	20 × 15	9143	4815	4469
la16	10 × 10	5240	2066	1994	swv08	20 × 15	9607	4928	4559
la17	10 × 10	5085	1960	1938	swv09	20 × 15	9276	4995	4704
la18	10 × 10	5479	2068	1963	swv10	20 × 15	9269	4800	4647
la19	10 × 10	5396	1966	1786	swv11	50 × 10	15471	10244	9888
la20	10 × 10	5415	2119	2041	swv12	50 × 10	15506	10094	9660
la21	15 × 10	8019	2987	2792	swv13	50 × 10	15765	10504	10160
la22	15 × 10	7376	2875	2771	swv14	50 × 10	15131	10199	9547
la23	15 × 10	8069	2979	2921	swv15	50 × 10	15095	9964	9748
la24	15 × 10	8252	3019	2979	swv16	50 × 10	25169	12196	10985
la25	15 × 10	7909	2890	2754	swv17	50 × 10	24526	12319	12041
la26	20 × 10	10263	4013	3716	swv18	50 × 10	24283	12330	11763
la27	20 × 10	10828	4110	4057	swv19	50 × 10	26054	11905	11080
la28	20 × 10	10697	4017	3869	swv20	50 × 10	24906	11741	11189
la29	20 × 10	10263	4029	3957	yn1	20 × 20	13225	4858	4485
la30	20 × 10	10682	4126	4076	yn2	20 × 20	13556	5340	4440
la31	30 × 10	14976	6506	6172	yn3	20 × 20	13639	4777	4613
la32	30 × 10	16615	6404	6117	yn4	20 × 20	13346	4761	4239
la33	30 × 10	15667	6423	6116					

We shall need the following notations and definitions. For any two disjoint subsets $\mathcal{J}^a$, $\mathcal{J}^b$of job set $\mathcal{J}$, $\gamma(\mathcal{J}^a)$ denotes the set of all arcs with one extremity in a job K and the other in a job L, $K, L \in \mathcal{J}^a$, and $\delta(\mathcal{J}^a, \mathcal{J}^b)$ the set of all arcs with tails in a job $K \in \mathcal{J}^a$ and heads in a job $L \in \mathcal{J}^b$.

Definition 5. Given a partition of $\mathcal{J}$ into two sets $\mathcal{J}^a$, $\mathcal{J}^b$, a feasible selection S schedules $\mathcal{J}^a$ before $\mathcal{J}^b$ if $S \cap \delta(\mathcal{J}^b, \mathcal{J}^a) = \emptyset$.

Definition 6. Given a subset $\mathcal{J}^b$ of $\mathcal{J}$ and a feasible selection T, a feasible selection S agrees with T on $\mathcal{J}^b$ if $S \cap \gamma(\mathcal{J}^b) = T \cap \gamma(\mathcal{J}^b)$.

Now let the set $\mathcal{J}$ of all jobs be partitioned into two sets $\mathcal{J}^a$ and $\mathcal{J}^b$ and let J be an arbitrary job of $\mathcal{J}^a$. The sets of disjunctive arcs E_J^-, E_J^+ and E_J entering J, leaving J and incident with J can be partitioned according to whether they are arcs incident to jobs of $\mathcal{J}^a$ or $\mathcal{J}^b$: $E_J^- = E_{aJ}^- \cup E_{bJ}^-$, $E_J^+ = E_{aJ}^+ \cup E_{bJ}^+$ and $E_J = E_{aJ} \cup E_{bJ}$, where $E_{aJ}^- = E_J^- \cap \delta(\mathcal{J}^a - J, J)$, $E_{aJ}^+ = E_J^+ \cap \delta(J, \mathcal{J}^a - J)$, $E_{bJ}^- = E_J^- \cap \delta(\mathcal{J}^b, J)$, $E_{bJ}^+ = E_J^+ \cap \delta(J, \mathcal{J}^b)$ and $E_{aJ} = E_{aJ}^- \cup E_{aJ}^+$, $E_{bJ} = E_{bJ}^- \cup E_{bJ}^+$.

Lemma 7. *Let S and T be feasible selections such that S schedules $\mathcal{J}^a$ before $\mathcal{J}^b$ and agrees with T on $\mathcal{J}^b$. Then S' defined by*

$$S' - E_J = S - E_J \tag{11}$$

$$S' \cap E_{aJ} = E_{aJ}^- \tag{12}$$

$$S' \cap E_{bJ} = T \cap E_{bJ} \tag{13}$$

is a feasible selection, and, letting $\mathcal{J}'^a = \mathcal{J}^a - J$ and $\mathcal{J}'^b = \mathcal{J}^b \cup J$, S' schedules $\mathcal{J}'^a$ before $\mathcal{J}'^b$ and agrees with T on $\mathcal{J}'^b$.

Proof. Observe that by (11), changing S to S' corresponds to a rescheduling of job J. We show that S' is feasible. First, S' is complete, as it is easy to verify $|S'| = |E|/2$. S' is also positive acyclic. Suppose the contrary, i.e. $G(S')$ contains a positive cycle, and hence a short positive cycle Z visiting J exactly once. (Note that Z must visit J since by (11), $G(S'-E_J) = G(S-E_J)$ is positive acyclic.) Let $Z \cap S' \cap E_j = \{f, g\}$ with $f \in E_J^- \cap S'$ and $g \in E_J^+ \cap S'$. Z consists of the concatenation (f, P, g, Q), where P is a path of conjunctive arcs in $\gamma(J)$, and Q a path in $\gamma(V-J)$. By (12), $S' \cap E_{aJ}^+ = \emptyset$, and therefore $g \in E_{bJ}^+$ and the starting node head(g) of Q is in $\bigcup_{J' \in \mathcal{J}^b} J'$. By (11), Q is a path in $G(S-E_J)$. Since S schedules $\mathcal{J}^a$ before $\mathcal{J}^b$, i.e. $S \cap \delta(\mathcal{J}^b, \mathcal{J}^a) = \emptyset$, not only the starting node head(g), but all nodes of Q are in $\bigcup_{J' \in \mathcal{J}^b} J'$ and therefore, interpreting Q as its arc set, $Q \cap S' = Q \cap S \subseteq \gamma(\mathcal{J}^b) \cap S$. But then, since S agrees with T on $\mathcal{J}^b$, i.e. $S \cap \gamma(\mathcal{J}^b) = T \cap \gamma(\mathcal{J}^b)$, Q is a path in $G(T - E_J)$. Since $S' \cap E_{bJ} = T \cap E_{bJ}$ and both f and $g \in E_{bJ}$, Z itself is contained in $G(T)$, a contradiction to T being a feasible selection.

Finally, since S schedules $\mathcal{J}^b$ before $\mathcal{J}^a$ and (12) holds, and S agrees with T on $\mathcal{J}^b$ and (13) holds, S' schedules $\mathcal{J}'^a$ before $\mathcal{J}'^b$ and agrees with T on $\mathcal{J}'^b$, where $\mathcal{J}'^a = \mathcal{J}^a - J$ and $\mathcal{J}'^b = \mathcal{J}^b \cup J$. ■

Lemma 8. *Let S and S' be two feasible selections differing only in the way job J is scheduled, i.e. $S - E_J = S' - E_J$ and define the neighborhood $\mathcal{N}_J(S) = \{S_{e,J} : e \in E_J\} \subseteq \mathcal{N}(S)$. There exists a sequence of selections S^0, S^1, S^l such that $S^0 := S$, $S^k \in \mathcal{N}_J(S^{k-1})$, $1 \le k \le l$, and $S^l = S'$. Moreover $l \le |E_J|/2$.*

Proof. It is enough to show that if $S \neq S'$, taking any $e \in S - S'$ and exchanging e with $\bar{e}$ yield $S_{e,J}$ with $|S_{e,J} \cap S'| > |S \cap S'|$. Since $S - E_J = S' - E_J$ and $\bar{e} \in S - S'$, $(S - E_J) \cup \bar{e} \subseteq S'$, and since S' is feasible and hence closed, $Q = \Phi((S - E_J) \cup \bar{e}) \subseteq S'$. Now, $S = (S \cap E(Q)) \cup (S - E(Q))$ and $S_{e,J} = (S_{e,J} \cap E(Q)) \cup (S - E(Q)) = Q \cup (S - E(Q))$ are bipartitions of S and $S_{e,J}$ and $|S| = |S_{e,J}|$, hence $|S \cap E(Q)| = |Q|$. Then $|S \cap S'| - |S_{e,J} \cap S'| = |S \cap E(Q) \cap S'| - |Q|$, and since $e \in (S \cap E(Q)) - S'$, $|S \cap E(Q) \cap S'| < |S \cap E(Q)| = |Q|$, hence $|S \cap S'| - |S_{e,J} \cap S'| < 0$. ■

Proof of Theorem 4. Start with selection S and job set partitions $\mathcal{J}^a = \mathcal{J}$ and $\mathcal{J}^b = \emptyset$, and perform repeatedly the following steps while $\mathcal{J}^a \neq \emptyset$. Choose $J \in \mathcal{J}^a$, construct S' as specified in Lemma 7, with at most $|E_J|/2$ moves (Lemma 8), and set $S := S'$, $\mathcal{J}'^a := \mathcal{J}^a - J$ and $\mathcal{J}'^b := \mathcal{J}^b \cup J$. At termination, $S' = T$. The total number of moves is bounded by $\sum_{J \in \mathcal{J}} |E_J|/2 = |E|$, each $e \in E$ appearing exactly in two E_J's. ■

References

[1] J. Adams, E. Balas, D. Zawack, The shifting bottleneck procedure for job shop scheduling, Management Science 34 (3) (1988) 391–401.
[2] D. Applegate, W. Cook, A computational study of the job shop scheduling problem, ORSA Journal on Computing 3 (2) (1991) 149–156.
[3] C.A. Brizuela, Y. Zhao, N. Sannomiya, No-wait and blocking job-shops: Challenging problems for GA's, IEEE 0-7803-77-2/ 01 (2001) 2349–2354.
[4] P. Brucker, T. Kampmeyer, Cyclic job shop scheduling problems with blocking, Annals of Operations Research 159 (2008) 161–181.
[5] O. Candar, Machine scheduling problems with blocking and no-wait in process, Working Paper [April-99], Department of Industrial Engineering, Bilkent University, Ankara, Turkey, 1999.
[6] L. Chen, N. Bostel, P. Dejax, J. Cai, L. Xi, A tabu search algorithm for the integrated scheduling problem of container handling systems in a maritime terminal, European Journal of Operational Research 181 (2007) 40–58.
[7] A. D'Ariano, D. Pacciarelli, M. Pranzo, A branch and bound algorithm for scheduling trains in a railway network, European Journal of Operational Research 183 (2007) 643–657.
[8] F. Glover, M. Laguna, Tabu Search, Kluwer, Boston, 1997.
[9] J. Grabowski, J. Pempera, Sequencing of jobs in some production system, European Journal of Operational Research 125 (2000) 535–550.
[10] H. Gröflin, A. Klinkert, Local search in job shop scheduling with synchronization and blocking constraints, Internal working paper [04–06], Department of Informatics, University of Fribourg, Switzerland, 2004.
[11] H. Gröflin, A. Klinkert, Feasible insertions in job shop scheduling short cycles and stable sets, European Journal of Operational Research 177 (2007) 763–785.
[12] N.G. Hall, C. Sriskandarajah, A survey of machine scheduling problems with blocking and no-wait in process, Operations Research 44 (3) (1996) 510–525.
[13] S. Heitmann, Job-shop scheduling with limited buffer capacities, Doctoral Thesis, University of Osnabrück, Germany, 2007.
[14] A.S. Jain, S. Meeran, Deterministic job-shop scheduling: Past present and future, European Journal of Operational Research 113 (1999) 390–434.
[15] A. Klinkert, Optimization in design and control of automated high-density warehouses, Doctoral Thesis No. 1353, University of Fribourg, Switzerland, 2001.
[16] S. Lawrence, Supplement to Resource Constrained Project Scheduling: An Experimental Investigation of Heuristic Scheduling Techniques, GSIA, Carnegie Mellon University, Pittsburgh, PA, 1984.
[17] A. Mascis, D. Pacciarelli, Machine scheduling via alternative graphs, Research Report, RT-DIA-46-2000, Italy, 2000.
[18] A. Mascis, D. Pacciarelli, Job-shop scheduling with blocking and no-wait constraints, European Journal of Operational Research 143 (2002) 498–517.
[19] Y. Mati, N. Rezg, X. Xie, Geometric approach and taboo search for scheduling flexible manufacturing systems, IEEE Transactions on Robotics and Automation 17 (6) (2001) 805–818.
[20] Y. Mati, N. Rezg, X. Xie, Scheduling problem of job-shop with blocking: A taboo search approach, Extended Abstracts, MIC 2001-4th Metaheuristics International Conference, Portugal, 2001, pp. 643–648.
[21] C. Meloni, D. Pacciarelli, M. Pranzo, A rollout metaheuristic for job shop scheduling problems, Annals of Operations Research 131 (2004) 215–235.
[22] J.F. Muth, G.L. Thompson (Eds.), Industrial Scheduling, Kluwer, Dordrecht, 1963.
[23] E. Nowicki, C. Smutnicki, A fast taboo search algorithm for the job shop problem, Management Science 42 (6) (1996) 797–812.
[24] D. Pacciarelli, Alternative graph formulation for solving complex factory-scheduling problems, International Journal of Production Research 40 (15) (2002) 3641–3653.
[25] D. Pacciarelli, M. Pranzo, Production scheduling in a steelmaking-continuous casting plant, Computers and Chemical Engineering 28 (2004) 2823–2835.
[26] M. Pinedo, X. Chao, Operations Scheduling with Applications in Manufacturing and Services, Irwin/McGraw-Hill, Boston, 1999.

[27] J. Romero, L. Puigjaner, T. Holczinger, F. Friedler, Scheduling intermediate storage multipurpose batch plants using the s-graph, AIChE Journal 50 (2004) 403–417.
[28] B. Roy, B. Sussman, Les problèmes d'ordonnancement avec contraintes disjonctives, Note DS No. 9 bsi, SEMA, Paris, France, 1964.
[29] R.H. Storer, S.D. Wu, R. Vaccari, New search spaces for sequencing problems with application to job shop scheduling, Management Science 38 (10) (1992) 1495–1509.
[30] T. Yamada, R. Nakano, A genetic algorithm applicable to large-scale job-shop problems, in: Proceedings of the 2nd Parallel Problem Solving from Nature, PPSN '92, 1992, pp. 281–290.