Available online at www.sciencedirect.com
SCIENCE DIRECT®
ELSEVIER
Discrete Applied Mathematics 128 (2003) 395–419
DISCRETE APPLIED MATHEMATICS
www.elsevier.com/locate/dam

A lower bound for the job insertion problem

Tamás Kis[a,*], Alain Hertz[b]

[a] *Computer and Automation Research Institute, Hungarian Academy of Sciences, Kende utca 13–17, H-1111 Budapest, Hungary*

[b] *Département de Mathématiques et de Génie Industriel, Ecole Polytechnique, CP 6079, succ. Centre-ville, Montreal (QC), Canada H3C 3A7*

Received 18 April 2000; received in revised form 17 October 2001; accepted 1 April 2002

Abstract

This note deals with the job insertion problem in job-shop scheduling: Given a feasible schedule of n jobs and a new job which is not scheduled, the problem is to find a feasible insertion of the new job into the schedule which minimises the makespan. Since the problem is NP-hard, a relaxation method is proposed to compute a strong lower bound. Conditions under which the relaxation provides us with the makespan of the optimal insertion are derived. After the analysis of the polytope of feasible insertions, a polynomial time procedure is proposed to solve the relaxed problem. Our results are based on the theory of perfect graphs and elements of polyhedral theory.

Keywords: Job-shop scheduling; Perfect graphs; Polyhedral methods

1. The job insertion problem

The job-shop scheduling problem is one of the most basic models in scheduling theory [8]. An instance of the problem is specified by a set of *m machines* $\{M_1, M_2, \ldots, M_m\}$ and a set of *n jobs* $\{J_1, J_2, \ldots, J_n\}$. Each job J_j is a sequence of n_j *operations*, $J_j = (o_{j,1}, o_{j,2}, \ldots, o_{j,n_j})$. Each operation $o_{j,k}$ has to be processed on a pre-specified machine $\mu_{j,k}$ uninterruptedly for $d_{j,k}$ time units. We assume that distinct

* Corresponding author. Address: Computer and Automation Research Institute Hungarian Academy of Sciences, H-1518 Budapest, P.O.Box 63.

E-mail addresses: kistamas@sztaki.hu (T. Kis), alain.hertz@gerad.ca (A. Hertz).

0166-218X/03/$ - see front matter
PII: S0166-218X(02)00502-4

Table 1
An example for a specification of 3+1 jobs

Jobs	Operation sequence
J_1	$(M_1,2) \rightarrow (M_2,1) \rightarrow (M_3,1)$
J_2	$(M_3,2) \rightarrow (M_2,1) \rightarrow (M_1,1)$
J_3	$(M_3,2) \rightarrow (M_2,1) \rightarrow (M_1,2)$
J_4	$(M_1,1) \rightarrow (M_2,2) \rightarrow (M_3,1)$

operations of the same job require different machines. A *feasible schedule* σ specifies the starting time of each operation while respecting the following constraints:

- $\sigma(o_{j,k}) + d_{j,k} \leqslant \sigma(o_{j,k+1})$ for all J_j and $1 \leqslant k \leqslant n_j - 1$.
- for any two operations $o_{j,k}$ and $o_{j',k'}$ with $\mu_{j,k} = \mu_{j',k'}$ either $\sigma(o_{j,k}) + d_{j,k} \leqslant \sigma(o_{j',k'})$ or $\sigma(o_{j',k'}) + d_{j',k'} \leqslant \sigma(o_{j,k})$.

The *makespan* $C(\sigma)$ of some feasible schedule σ is the completion time of the job last finished, i.e., $C(\sigma) = \max\{\sigma(o_{j,n_j}) + d_{j,n_j} \mid 1 \leqslant j \leqslant n\}$. The job-shop scheduling problem aims at finding a feasible schedule that minimises the makespan.

In this paper we will study the *job insertion problem of job-shop scheduling*: We are given $n+1$ jobs, $J_1,\ldots,J_n, J_{n+1}$, m machines, and a feasible schedule σ of the first n jobs. A *feasible insertion* of J_{n+1} into σ inserts simultaneously all operations of J_{n+1} into the sequences of operations on the machines required, such that the result is a feasible schedule. The *makespan of a feasible insertion* is the makespan of the resulting schedule. The insertion problem consists in finding a feasible insertion of J_{n+1} into σ with minimum makespan.

Example. We use the following problem to illustrate concepts and algorithms throughout the paper. Table 1 specifies $3+1$ jobs. A job is a sequence of operations, where each operation is a pair consisting of a machine identifier and a processing time. A schedule of the first three jobs is depicted in Fig. 1(a), whereas Fig. 1(b) shows a possible insertion of J_4 into the schedule.

It can be shown that the job insertion problem is binary NP-hard even if the new job is to be inserted into the schedule of only two other jobs. This fact has implicitly been proven by Sotskov in [19]. A different proof is provided by Kis [14].

1.1. Motivations and related work

The job insertion problem is a special case of the following general insertion problem. Given a set of n jobs, a feasible partial schedule σ that determines the processing order of some operations of these jobs, and a subset S of operations that are not scheduled yet. By *inserting* S we mean that all operations in S are inserted simultaneously into the sequences of operations on the machines required. The general insertion problem consists of inserting S into σ such that the resulting partial schedule is feasible and

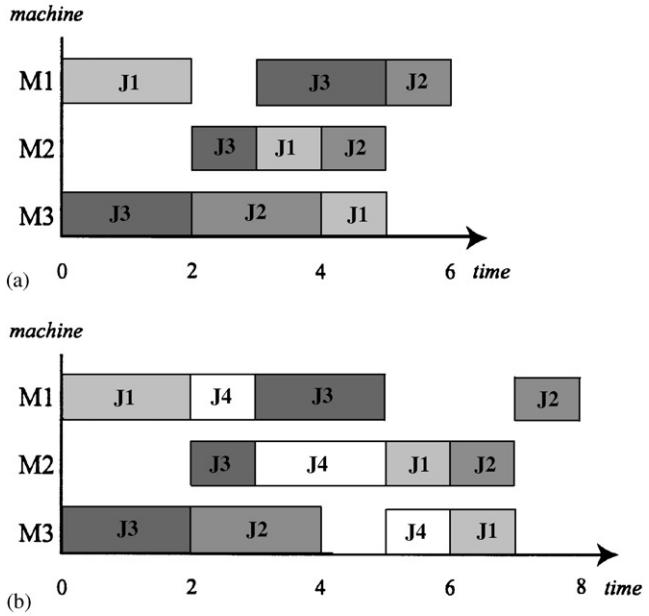

Fig. 1. (a) A schedule of three jobs J_1, J_2, and J_3. (b) A possible insertion of J_4 into the schedule.

some objective function is minimised. In the job insertion problem S is the operation set of a job. Another special case arises when S contains one operation only. A third special case is when S consists of all operations requiring the same machine.

A lot of research has been focused on constructing and improving a schedule by inserting and reinserting, respectively, a *single operation* at a time. For a review and comparison, we refer the reader to Jain and Meeran [13] and to Vaessens et al. [22]. Most of the work on single operation insertion aim at providing easily verifiable sufficient conditions for feasible insertion and to restrict the search space in which an optimal insertion is sought. We mention that the problem of inserting a single operation into a sequence of operations is extensively discussed by e.g., Dell'Amico and Trubian [11], Werner and Winkler [23], Nowicki and Smutnicki [17], Dauzère-Péres and Paulli [9] and Mastrolilli and Gambardella [16]. Moreover, when the operation needs several resources simultaneously for processing, the operation insertion problem becomes more difficult and the approaches become more complicated, see e.g., Dauzère-Péres et al. [10], Brucker and Neyer [6], Kis [14], or, in a more general context than job shop scheduling, Artigues and Roubellat [4].

The job insertion problem plays an important role in the local search algorithm of Werner and Winkler [23] for solving the job shop scheduling problem. In that paper the neighbours of a schedule are obtained by removing all operations of a job having at least one operation on a critical (i.e., longest) path π and inserting them back such that π is eliminated. The authors propose a simple heuristic algorithm to solve the job insertion problem in the context of their local search algorithm.

Various insertion heuristics are discussed by Sotskov et al. [20] to be used in constructive algorithms for the job shop problem with setup times. One of the approaches proposed consists of inserting the jobs one-by-one into a growing schedule. The operations of a job are inserted either sequentially (proceeding e.g., from the first operation of the job to the last one) or in parallel. In the parallel case, only k consecutive operations are inserted at a time, and their best insertion is found by exhaustive search.

The famous Shifting Bottleneck Procedure of Adams et al. [1] repeatedly reschedules (or in other words (*re*)-*inserts*) all operations on some machine with the aim of improving the schedule. A detailed analysis of the problem can be found in Balas et al. [5].

As a summary, the exact or heuristic solution of the job insertion problem can be used to build schedules from scratch, to insert new jobs over time into a schedule, for rescheduling, or it can be a subroutine in new algorithms for the job shop scheduling problem.

1.2. The principal results obtained

The main goal of this paper is to gain more insight into the job insertion problem, or, more generally, to analyse the problem of inserting a set of operations into a schedule in parallel. We provide equivalent characterisations of the set of feasible insertions. It turns out that the set of feasible insertions are the integral solutions of an inequality system whose size is polynomial in the input, or equivalently, the maximum stable sets of a comparability graph associated with the job insertion problem. We use our characterisation in a MILP formulation that models a relaxation of the insertion problem. The formulation is based on the analysis of longest paths in the schedule obtained by inserting J_{n+1} into σ. Necessary and sufficient conditions when the lower bound delivered by the MILP matches the makespan of the optimal insertion are derived, too.

We devise a polynomial time algorithm to solve the MILP. To this end, we define a new transformation of comparability graphs and analyse its properties. This transformation allows us to solve the MILP to optimality in polynomial time.

We have conducted a computational study to evaluate the power of our lower bound. Our results show that the bound is rather strong and can be computed in a short computation time.

1.3. The structure of the paper

Section 2 defines the notation used throughout the paper. In Section 3 we introduce a MILP formulation of the relaxed insertion problem. Section 4 is devoted to the analysis of the polytope of feasible insertions. In Section 5 we devise a polynomial time algorithm to solve the MILP. In Section 6 we discuss our computational results. We conclude the paper in Section 7.

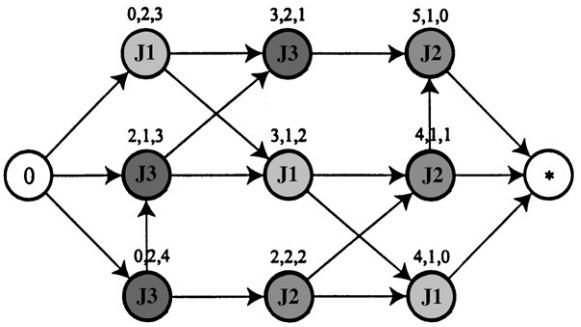

Fig. 2. The directed graph associated with the schedule in Fig. 1(a).

2. Notation

Let σ be a feasible schedule of jobs $J_1, J_2, \ldots, J_n$. We will index the operations *in the schedule* according to their position. Namely, $u_{i,l}$ is the lth operation in the sequence of operations on machine M_i. The processing time of $u_{i,l}$ will be denoted by $p_{i,l}$. The entire sequence is $(u_{i,1}, u_{i,2}, \ldots, u_{i,m_i})$, where m_i is the number of operations on M_i.

We associate a directed graph G_σ with σ. The set of nodes comprises all operations and two extra nodes: 0 and $*$. There is a directed edge from $u_{i,l}$ to $u_{i',l'}$ if and only if

- $i = i'$ and $l' = l + 1$, or
- $i \neq i', u_{i,l}$ and $u_{i',l'}$ are operations of the same job, and $u_{i,l}$ is the immediate predecessor of $u_{i',l'}$ in the sequence of operations of the job.

Furthermore, there is a directed edge from 0 to the first operation on each machine and there is a directed edge from the last operation on each machine to $*$. The directed graph associated with our example schedule is depicted in Fig. 2.

A *directed path* $\pi = (\pi_1, \ldots, \pi_l)$ is a sequence of distinct nodes such that there is a directed edge from π_f to π_{f+1}, $(1 \leqslant f < l)$. The *length of a directed edge* (a, b) is the processing time of a if a is an operation or 0 if $a = 0$. The *length of some path* π is the sum of edge lengths along the path. The *head* $h_{i,l}$ of operation $u_{i,l}$ is the length of the longest path from 0 to $u_{i,l}$ (such a path clearly exists). The tail $t_{i,l}$ of $u_{i,l}$ is the length of the longest path from $u_{i,l}$ to $*$ minus $p_{i,l}$. A *directed cycle* $C = (c_1, \ldots, c_l, c_{l+1})$ is a sequence of nodes such that $(c_1, \ldots, c_l)$ is a directed path, $c_1 = c_{l+1}$ and there is a directed edge from c_l to c_1. Note that G_σ contains no directed cycles, or shortly it is *acyclic*, for σ is feasible.

Let J_{n+1} be the job to be inserted into σ. For the sake of simpler notation, assume that J_{n+1} is a sequence of m operations: $(v_1, v_2, \ldots, v_m)$, where v_i requires M_i for p_i time units. On machine M_i, v_i can be inserted in one of the $m_i + 1$ positions, i.e., before the first operation, or after any of the operations on M_i. These positions are called the

insertion points on M_i, and will be identified with binary variables $x_{i,l}$ $(0 \leqslant l \leqslant m_i)$. An *insertion* is a 0/1 valuation of these variables, such that for each i, precisely one of the variables $x_{i,l}$ is set to 1, the others are set to 0. Let $G_{\sigma,x}$ denote the graph after inserting J_{n+1} in the positions given by x. An insertion x is *feasible*, if $G_{\sigma,x}$ is acyclic. Let I_f denote the set of feasible insertions and let $C(\sigma,x)$ denote the length of the longest path in $G_{\sigma,x}$. The optimal insertion problem can be restated as $\min\{C(\sigma,x) \mid x \in I_f\}$. An *optimal insertion* x' is a feasible insertion minimising $C(\sigma,x)$. The optimal makespan is denoted by $OPT(\sigma,J_{n+1})$.

The head of v_i when inserted in the lth position on M_i is at least $\tilde{h}_{i,l} := h_{i,l} + p_{i,l}$, whereas the tail is at least $\tilde{t}_{i,l} := t_{i,l+1} + p_{i,l+1}$, noting that for each $1 \leqslant i \leqslant m$: $\tilde{h}_{i,0} = \tilde{t}_{i,m_i} = 0$.

For each pair of machines M_i and $M_{i'}$ with $1 \leqslant i < i' \leqslant m$, we define the set $P_{i,i'}$ as follows. An ordered pair of operations $(u_{i,l}, u_{i',l'})$ pertains to $P_{i,i'}$, if and only if there exists a directed path from $u_{i',l'}$ to $u_{i,l}$ in G_σ.

R^D denotes the D-dimensional real space, and $R^D_+ \subset R^D$ consists of all non-negative vectors of R^D. The convex hull of a finite set $V \subset R^D$ is denoted by $Conv(V)$ (cf. [18]).

3. Lower bound by integer programming

In this section we describe a mixed integer-linear program (ILP) that is a *relaxation* of the insertion problem. We will derive conditions under which the optimum value of ILP matches $OPT(\sigma,J_{n+1})$. The relaxation is based on the structure of longest paths in the schedule obtained by inserting J_{n+1} into σ. We start with a brief discussion of this latter topic.

3.1. The structure of longest paths after insertion

Let σ' be a feasible schedule obtained by inserting J_{n+1} into σ. Since σ' is feasible, $G_{\sigma'}$ is acyclic. Let π be a longest path in $G_{\sigma'}$. Then precisely one of the following conditions holds:

(a) π does not contain any operation of J_{n+1},
(b) $\pi = \pi_b \cdot \pi_{J_{n+1}} \cdot \pi_a$, where π_b and π_a do not contain any operation of J_{n+1} (possibly one or both of them are empty), whereas $\pi_{J_{n+1}}$ is a non-empty subsequence of J_{n+1},
(c) $\pi = \pi_b \cdot \pi^1_{J_{n+1}} \cdot \pi_c \cdot \pi^2_{J_{n+1}} \cdot \pi_a$, where π_c is not empty and does not contain any operation of J_{n+1}, π_a and π_b may contain operations of J_{n+1}, and $\pi^1_{J_{n+1}}$ and $\pi^2_{J_{n+1}}$ are non-empty subsequences of J_{n+1}.

We call an insertion of J_{n+1} in σ *fragmented*, if all longest paths in $G_{\sigma'}$ are of type (c). Otherwise, the insertion is *non-fragmented*.

3.2. The MILP formulation

We associate a 0/1 variable with each insertion point in the schedule. Thus we have the binary decision variables $x_{i,l}$, where $1 \leqslant i \leqslant m$ and $0 \leqslant l \leqslant m_i$ (cf. Section 2). A feasible insertion selects for each v_i of J_{n+1} one particular position on M_i. Hence, the equalities

$$\sum_{l=0}^{m_i} x_{i,l} = 1, \quad 1 \leqslant i \leqslant m \tag{1}$$

are respected by all feasible insertions. Furthermore, feasible insertions do not create cycles. Consequently, if $u_{i,k}$ and $u_{i',k'}$ is a pair of operations such that $i < i'$, and there is a directed path from $u_{i',k'}$ to $u_{i,k}$ in G_σ, then no feasible insertion assigns a position to v_i after $u_{i,k}$ on M_i, and a position to $v_{i'}$ before $u_{i',k'}$ on $M_{i'}$, simultaneously. Hence, the inequalities

$$\sum_{l=0}^{k'-1} x_{i',l} + \sum_{l=k}^{m_i} x_{i,l} \leqslant 1, \quad \text{for all pairs } (u_{i,k}, u_{i',k'}) \in P_{i,i'} \tag{2}$$

are valid for all feasible insertions. Note that some of these constraints may be implied by others. That is, the constraint that corresponds to the pair $(u_{i,k}, u_{i',k'}) \in P_{i,i'}$ is implied, iff there exists $(u_{i,l}, u_{i',l'}) \in P_{i,i'}$ with $l' \geqslant k'$ and $l \leqslant k$, and one of these inequalities is strict. Implied constraints can be safely omitted.

Example. The inequality system that defines the set of feasible insertions to our problem is the following:

$$\begin{aligned}
&x_{1,0} + x_{1,1} + x_{1,2} + x_{1,3} = 1,\\
&x_{2,0} + x_{2,1} + x_{2,2} + x_{2,3} = 1,\\
&x_{3,0} + x_{3,1} + x_{3,2} + x_{3,3} = 1,
\end{aligned} \tag{3}$$

$$\begin{aligned}
&x_{2,0} + x_{1,2} + x_{1,3} \leqslant 1,\\
&x_{2,0} + x_{2,1} + x_{2,2} + x_{1,3} \leqslant 1,\\
&x_{3,0} + x_{2,1} + x_{2,2} + x_{2,3} \leqslant 1,
\end{aligned} \tag{4}$$

$$\begin{aligned}
&x_{3,0} + x_{3,1} + x_{2,3} \leqslant 1,\\
&x_{3,0} + x_{1,2} + x_{1,3} \leqslant 1,\\
&x_{3,0} + x_{3,1} + x_{1,3} \leqslant 1.
\end{aligned} \tag{5}$$

The next lemma shows that inequalities (1) and (2) are sufficient to characterise all feasible insertions.

Lemma 1. *A binary vector x satisfies (1) and (2), if and only if it defines a feasible insertion.*

Proof. We know than any feasible insertion satisfies (1) and (2), by construction.

We have to demonstrate the other direction. Suppose that a binary vector x satisfies all equalities (1) and inequalities (2). Due to (1), on each machine precisely one position is selected. We claim that no cycle is created after inserting J_{n+1} in the positions given by x. Assume the contrary and suppose there is a cycle in $G_{\sigma,x}$. Clearly, any cycle C contains at least two operations of J_{n+1}. Let $v_{\min}(C)$ denote the operation of J_{n+1} on C, that has the smallest index. Similarly, let $v_{\max}(C)$ denote the operation of J_{n+1} on C, that has the largest index. If there are several cycles, then we select one that maximises the index of $v_{\min}(C)$. Let C' be such a cycle. Note that C' is of the form $(v_i,\dots,v_{i'},\dots,v_i)$, where $v_i = v_{\min}(C')$ and $v_{i'} = v_{\max}(C')$.

We claim that there is no operation of J_{n+1} on C' between $v_{i'}$ and v_i. If there were, then let v_j be such an operation. By the choice of i and i', $i < j < i'$ holds. Then we know that $C' = (v_i,\dots,v_{i'},\dots,v_j,\dots,v_i)$ and hence $\bar{C} = (v_j,\dots,v_{i'-1},v_{i'},\dots,v_j)$ is a cycle in $G_{\sigma,x}$, with $v_{\min}(\bar{C}) = v_j$. Since $j > i$, this contradicts the choice of C'.

As a consequence, we can state that there exists a path in $G_{\sigma,x}$ emanating from the operation that follows $v_{i'}$ on $M_{i'}$, to the operation that precedes v_i on M_i, that contains no operation of J_{n+1}. Hence, one of the constraints (2) is violated, a contradiction. □

Now we develop inequalities to estimate $C(\sigma,x)$ from below. On the one hand, if v_i is inserted in the kth position on M_i, then there is a path $(u_{i,1},\dots,u_{i,k},v_i,u_{i,k+1},\dots,u_{i,m_i})$ in $G_{\sigma,x}$. Let $C_{\max}$ denote the estimated makespan. Then we have the inequalities:

$$\sum_{l=1}^{m_i} \tilde{h}_{i,l}x_{i,l} + p_i + \sum_{l=0}^{m_i-1} \tilde{t}_{i,l}x_{i,l} \leqslant C_{\max}, \quad 1 \leqslant i \leqslant m. \tag{6'}$$

On the other hand, there are new paths between machines as well. Namely, if v_i is inserted in the kth position on M_i, and $v_{i'}$ is inserted in the kth position on $M_{i'}$, $i < i'$, then $(u_{i,1},\dots,u_{i,k},v_i,v_{i+1},\dots,v_{i'-1},v_{i'},u_{i',k'+1},\dots,u_{i',m_{i'}})$ is a path in $G_{\sigma,x}$. Hence, we have

$$\sum_{l=1}^{m_i} \tilde{h}_{i,l}x_{i,l} + \sum_{l=i}^{i'} p_l + \sum_{l=0}^{m_{i'}-1} \tilde{t}_{i',l}x_{i',l} \leqslant C_{\max}, \quad 1 \leqslant i < i' \leqslant m. \tag{6''}$$

Furthermore, $\max\{\sum_{i=1}^{m} p_i, C(\sigma)\} \leqslant C_{\max}$ clearly holds.

The system of inequalities consisting of (6′), (6″) and this very last inequality will be denoted by

$$A\begin{pmatrix} x \\ C_{\max} \end{pmatrix} \geqslant b. \tag{6}$$

The goal is to minimise $C_{\max}$, hence the entire problem is

$$\min C_{\max}$$

$$K_1 \cdot x = 1,$$

$$K_2 \cdot x \leqslant 1,$$

$$A\begin{pmatrix} x \\ C_{\max} \end{pmatrix} \geqslant b,$$

$$x \in \{0,1\}^D,$$

$$C_{\max} \in R,$$

where K_1 is the matrix of equalities (1), K_2 is the matrix of inequalities (2), and D is the total number of insertion points on all machines. We will call this integer-linear program *ILP*.

Some properties of the estimation are summarised next:

Lemma 2. *Let* $(x^*, C^*_{\max})$ *be an optimal solution to ILP. Then*

(i) $C^*_{\max} \leqslant OPT(\sigma, J_{n+1})$.
(ii) $C^*_{\max} = OPT(\sigma, J_{n+1})$ *implies that all optimal insertions are non-fragmented.*
(iii) *If* $C(\sigma, x^*) = C^*_{\max}$ *then* $C^*_{\max} = OPT(\sigma, J_{n+1})$.

Proof.

(i) Let x' be any optimal insertion. We claim that $(x', OPT(\sigma, J_{n+1}))$ is a feasible solution to ILP. Since x' does not create a cycle, it satisfies (1) and (2). By the construction of inequalities (6″), $(x', OPT(\sigma, J_{n+1}))$ has to satisfy all of them.

(ii) Suppose $C^*_{\max} = OPT(\sigma, J_{n+1})$, and let x' be any optimal insertion. Then $\alpha \cdot x' + OPT(\sigma, J_{n+1}) = \beta$ holds for an inequality in (6). We claim that this equality induces a non-fragmented longest path in $G_{\sigma,x'}$. We distinguish between three cases:
- $OPT(\sigma, J_{n+1}) = \max\{\sum_{i=1}^{m} p_i, C(\sigma)\}$ holds. Then either the operations of J_{n+1} constitute a non-fragmented longest path in $G_{\sigma,x'}$, or there is a longest path of length $C(\sigma)$ in $G_{\sigma,x'}$, which is free of J_{n+1}.
- $-\sum_{l=1}^{m_i} \tilde{h}_{i,l} x'_{i,l} - \sum_{l=0}^{m_i} \tilde{t}_{i,l} x'_{i,l} + OPT(\sigma, J_{n+1}) = p_i$ holds for some $1 \leqslant i \leqslant m$ (cf. (6′)). Since x' is an insertion, there exists a unique k with $x'_{i,k} = 1$. Then the equality can be rewritten as

$$OPT(\sigma, J_{n+1}) = \sum_{l=1}^{m_i} \tilde{h}_{i,l} x'_{i,l} + p_i + \sum_{l=0}^{m_i} \tilde{t}_{i,l} x'_{i,l} = \tilde{h}_{i,k} + p_i + \tilde{t}_{i,k}.$$

  Due to the equality, no operation of J_{n+1} occurs either on the longest path from 0 to $u_{i,k}$, or on the longest path from $u_{i,k+1}$ to $*$ in $G_{\sigma,x'}$. Hence, there is a longest path in $G_{\sigma,x'}$ containing only one operation of J_{n+1}.
- One of the inequalities (6″) holds with equality. An argument similar to the previous one shows that there is a longest path in $G_{\sigma,x'}$ which contains some operations of J_{n+1} consecutively.

(iii) The assumptions imply that G_{σ,x^*} contains a non-fragmented longest path whose length is $C^*_{\max}$. Since x^* is a feasible insertion, $OPT(\sigma, J_{n+1}) \leqslant C^*_{\max}$ holds. Combining this with part (i) the statement follows. □

In the rest of the paper we show how to compute $C^*_{\max}$ in polynomial time. Furthermore, we will obtain an implicit description of all optimal solutions to ILP in the form of the vertices of a polytope.

4. The polytope of feasible insertions

We define the *polytope of feasible insertions* as follows:

$$P(I_f) := Conv(I_f),$$

where I_f is the set of feasible insertions (cf. Section 2). Let $P := \{x \in R^D_+ \mid K_1 \cdot x = 1 \ \& \ K_2 \cdot x \leqslant 1\}$. Let $INT(P) \subset P$ denote the set of integral, i.e. 0/1, vectors of P. We clearly have $INT(P) = I_f$, by Lemma 1. Consequently, $P(I_f) \subseteq P$ must hold. Moreover, the following statement implies that the two polytopes coincide:

Theorem 3. *P is integral, that is, $P = Conv(INT(P))$.*

Proof sketch. The idea of the proof is to show that P is a face of a larger polytope P_c, which is integral. Then clearly, P is integral. It will turn out that P_c can be chosen to be the fractional node packing polytope of a comparability graph, which is known to be integral. We defer the detailed proof until we obtain some more results on the system of inequalities (1) and (2).

4.1. The insertion graph

We define the *insertion graph* $\hat{G}_\sigma$ with respect to J_{n+1} as follows: Let the nodes be identified with the variables $x_{i,l}$. For the edges, each equality or inequality in (1) and (2) induces a clique in the graph. Namely, on each machine, the insertion points are mutually connected (cf. equalities (1)). Furthermore, for each inequality in (2), the insertion points identified with the variables of the inequality are mutually connected.

Example. The insertion graph with respect to J_4 is depicted in Fig. 3(b).

Some useful properties of the insertion graph are summarised next.

Lemma 4. *Let σ be a schedule of n jobs, let J_{n+1} be the job to be inserted, and let $\hat{G}_\sigma$ be the insertion graph with respect to J_{n+1}. Then*

(i) *There are at most m independent nodes in $\hat{G}_\sigma$.*
(ii) *There is no edge adjacent to any two nodes $x_{i,0}$ and $x_{i',0}$.*
(iii) *The stability number $\alpha(\hat{G}_\sigma)$ of $\hat{G}_\sigma$ is m, the number of machines.*

Proof. Part (i) follows from the fact that the equalities in (1) determine m disjoint cliques covering all nodes.

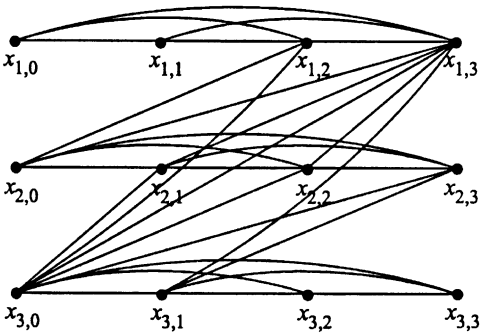

Fig. 3. The insertion graph associated with the example problem.

Part (ii) is due to the structure of inequalities in (2). Namely, there is no inequality containing both $x_{i,0}$ and $x_{i',0}$. Hence, there is no edge adjacent to any two nodes identified with $x_{i,0}$ and $x_{i',0}$.

Part (iii) is an immediate consequence of parts (i) and (ii). □

Our next statement puts the problem into the context of perfect graphs. Recall that an undirected graph is a *comparability graph* if its edges can be transitively oriented. Namely, whenever (a,b) and (b,c) are directed edges after the orientation, then (a,c) is also a directed edge of the oriented graph. For relevant definitions and statements on comparability graphs see [12].

We will orient the edges of $\hat{G}_\sigma$ according to the following *orientation rules*. Let $\{x_{i,k}, x_{i',k'}\}$ be any edge of $\hat{G}_\sigma$. Without loss of generality we may assume that $i \leqslant i'$. Then there are two rules:

- If $i = i'$ orient the edge from $x_{i,k}$ to $x_{i,k'}$, when $k < k'$.
- if $i < i'$, orient the edge from $x_{i',k'}$ to $x_{i,k}$.

Lemma 5. *$\hat{G}_\sigma$ is a comparability graph.*

Proof. Orient the edges of $\hat{G}_\sigma$ according to the orientation rules. We claim that whenever (a,b) and (b,c) are directed edges of the oriented $\hat{G}_\sigma$, then there also exists a directed edge (a,c) in the oriented graph. Let $a = x_{i'',k''}$, $b = x_{i',k'}$ and $c = x_{i,k}$. By the choice of a, b and c and the orientation rules it follows that $i'' \geqslant i' \geqslant i$. We distinguish between four cases:

- a, b, c are all insertion points on the same machine, i.e. $i = i' = i''$. By the orientation rules $k'' < k' < k$ holds. Since $i = i''$, there is an edge in $\hat{G}_\sigma$ adjacent to $x_{i,k''}$ and $x_{i,k}$, which is oriented from the former one to the latter one by the orientation rules.
- $i'' > i' = i$. Then there is an inequality in (2) involving $x_{i,k}$, $x_{i',k'}$ and $x_{i'',k''}$. Consequently, there is an edge adjacent to $x_{i,k}$ and $x_{i'',k''}$ in $\hat{G}_\sigma$. This edge is oriented from $x_{i'',k''}$ to $x_{i,k}$ by the orientation rule.

- $i'' = i' > i$. This case is similar to the previous one.
- $i'' > i' > i$. We have to show that there is an edge adjacent to $x_{i'',k''}$ and $x_{i,k}$ in $\hat{G}_\sigma$, the rest follows from the orientation rules. Since $\{x_{i'',k''}, x_{i',k'}\}$ is an edge of $\hat{G}_\sigma$, there exists a directed path $\pi_{i'',i'}$ in G_σ from $u_{i'',k''+1}$ to $u_{i',k'}$. A similar argument shows that there exists a directed path $\pi_{i',i}$ in G_σ from $u_{i',k'+1}$ to $u_{i,k}$. Hence, $(\pi_{i'',i'}, \pi_{i',i})$ is a directed path in G_σ from $u_{i'',k''+1}$ to $u_{i,k}$. The construction of $\hat{G}_\sigma$ implies that there is an edge adjacent to $x_{i'',k''}$ and $x_{i,k}$, as claimed. □

4.2. Cliques on 3 or more machines

Observe that insertion graphs may have cliques which contain insertion points on 3 or more machines. For instance, in the insertion graph in Fig. 3, nodes $x_{3,0}, x_{2,1}, x_{2,2}, x_{1,3}$ induce a clique, but the inequality

$$x_{3,0} + x_{2,1} + x_{2,2} + x_{1,3} \leqslant 1 \tag{7}$$

is not included in the system defining all feasible insertions. In fact, this inequality can be derived from constraints (3), (4), and (5) as follows: Since $x_{2,0} + x_{2,1} + x_{2,2} = 1 - x_{2,3}$ and $x_{2,1} + x_{2,2} + x_{2,3} = 1 - x_{2,0}$ we can substitute these into (4) and (5), respectively, and taking the sum we obtain

$$x_{3,0} + (1 - x_{2,3}) + (1 - x_{2,0}) + x_{1,3} \leqslant 2.$$

Using (3), we deduce (7). Note that including (7) into the system, some inequalities may become redundant.

Lemma 6. *Let C be a maximal clique of $\hat{G}_\sigma$.*

(a) *If C has at least two insertion points $x_{i,l}$ and $x_{i,r}$, $l < r$, on some machine M_i, then all insertion points $x_{i,k}$ with $l < k < r$ pertain to C.*
(b) *Suppose C has insertion points on two or more machines. Let M_i and $M_{i'}$ be the machines with smallest, respectively highest indices, such that C has some insertion points on them. Then there exists $l \in \{0, \ldots, m_i\}$, such that C contains $x_{i,l}, \ldots, x_{i,m_i}$, and there exists $r' \in \{0, \ldots, m_{i'}\}$, such that C contains $x_{i',0}, \ldots, x_{i',r'}$.*

Proof.

(a) If C is induced by equalities (1) or inequalities (2), then the statement trivially holds. Let C be any other maximal clique. Suppose C has two insertion points l and $r, l < r$, on M_i. We show that all insertion points $x_{i,k}$ with $l < k < r$ pertain to C. Let $x_{i',k'}$ be any other insertion point in C. Since all insertion points on the same machine are mutually connected, we may assume that $i \neq i'$. We distinguish between two cases:
   - $i < i'$. Since C is a clique, there is an edge in $\hat{G}_\sigma$ adjacent to $x_{i',k'}$ and $x_{i,l}$. Then there also exists an inequality in (2) which contains $x_{i',k'}$ and $x_{i,l}$ and hence $x_{i,k}$ as well, for $l < k$. Hence, $x_{i,k}$ and $x_{i',k'}$ are adjacent.
   - $i > i'$. Similar to the previous case.

(b) We prove the statement for M_i only, the other case being similar. In fact, it is enough to show that x_{i,m_i} pertains to C, the rest follows from part (a). Since C contains some insertion points on M_i, let $x_{i,l}$ be the one with smallest (second) index. If $l=m_i$ then there is nothing to prove. Otherwise, let $x_{i',l'}$ be any insertion point in C. We may assume that $i<i'$, for all insertion points on the same machine are mutually connected. Since both $x_{i,l}$ and $x_{i',l'}$ belong to C, there is an inequality in (2) containing both of them. But this inequality has to contain x_{i,m_i} as well, for $m_i > l$. Consequently, there is an edge adjacent to x_{i,m_i} and $x_{i',l'}$. □

Corollary 7. *The maximal cliques on* 1 *or* 2 *machines are precisely those induced by equalities* (1) *and inequalities* (2).

Corollary 8. *Let* χ^C *be the characteristic vector of a maximal clique* C *on three or more machines. Then* $\chi^C \cdot x \leqslant 1$ *is a valid inequality for* P.

Proof. Suppose C has insertion points on machines $M_{i_1}, M_{i_2}, \ldots, M_{i_q}, 1 \leqslant i_1 < i_2 < \cdots < i_q \leqslant m$. We construct a sequence of inequalities $\chi^j \cdot x \leqslant 1, 1 \leqslant j \leqslant q-1$, such that all of them are valid for P, and $\chi^{q-1} = \chi^C$.

Let l_h and $r_h, 1 \leqslant h \leqslant q$, be the smallest and highest indices, respectively, such that x_{i_h,l_h} and x_{i_h,r_h} pertain to C. Note that $r_1 = m_{i_1}$, $l_q = 0$, and $\chi^C \cdot x = \sum_{h=1}^{q} \sum_{k=l_h}^{r_h} x_{i_h,k}$, by Lemma 6.

Since C is a clique, the pairs $(x_{i_j,l_j}, x_{i_{j+1},r_{j+1}})$ are adjacent in $\hat{G}_\sigma$. Hence, there are inequalities $\alpha_j \cdot x \leqslant 1, 1 \leqslant j \leqslant q-1$ in (2), of the form $\alpha_j \cdot x = \sum_{k=0}^{r_{j+1}} x_{i_{j+1},k} + \sum_{k=l_j}^{m_{i_j}} x_{i_j,k} \leqslant 1$.

We define $\chi^j \cdot x$ as follows: $\chi^j \cdot x := \sum_{h=q-j+1}^{q} \sum_{k=l_h}^{r_h} x_{i_h,k} + \sum_{k=l_{q-j}}^{m_{i_{q-j}}} x_{i_{q-j},k}$. One can verify that $\chi^1 \cdot x = \alpha_{q-1} \cdot x$ and $\chi^{q-1} \cdot x = \chi^C \cdot x$ hold. If we manage to show that all inequalities $\chi^j \cdot x \leqslant 1, 1 \leqslant j \leqslant q-1$, are valid for P, then we are done.

Since $\chi^1 \cdot x = \alpha_{q-1} \cdot x$, the statement is true for $j=1$. We proceed with induction on j. Suppose the statement is verified until $j \geqslant 1$. We show that $\chi^{j+1} \cdot x \leqslant 1$ is valid for P, by combining $\chi^j \cdot x \leqslant 1$ with $\alpha_{q-j-1} \cdot x \leqslant 1$.

For the sake of simpler notation, let $i' = i_{q-j}$, $l' = l_{q-j}$, and $r' = r_{q-j}$. Since $\sum_{k=l'}^{m_{i'}} x_{i',k} + \sum_{k=0}^{l'-1} x_{i',k} = 1$, we know that

$$\chi^j \cdot x = \left(\chi^j \cdot x - \sum_{k=l'}^{m_{i'}} x_{i',k} \right) + \left(1 - \sum_{k=0}^{l'-1} x_{i',k} \right). \tag{8}$$

On the other hand, $\sum_{k=0}^{r'} x_{i',k} = 1 - \sum_{k=r'+1}^{m_{i'}} x_{i',k}$, hence

$$\alpha_{q-j-1} \cdot x = \left(1 - \sum_{k=r'+1}^{m_{i'}} x_{i',k} \right) + \sum_{k=l_{q-j-1}}^{m_{i_{q-j-1}}} x_{i_{q-j-1},k}. \tag{9}$$

By the induction hypothesis we know that $\chi^j \cdot x \leqslant 1$ is valid for P. Hence, if we combine (8) and (9) we obtain a valid inequality for P:

$$2 \geqslant \chi^j \cdot x + \alpha_{q-j-1} \cdot x = \left(\chi^j \cdot x - \sum_{k=l'}^{m_{i'}} x_{i',k}\right) + \left(1 - \sum_{k=0}^{l'-1} x_{i',k}\right) + \left(1 - \sum_{k=r'+1}^{m_{i'}} x_{i',k}\right) + \sum_{k=l_{q-j-1}}^{m_{i_{q-j-1}}} x_{i_{q-j-1},k}.$$

Since $1 - \sum_{k=0}^{l'-1} x_{i',k} - \sum_{k=r'+1}^{m_{i'}} x_{i',k} = \sum_{k=l'}^{r'} x_{i',k}$, we obtain

$$2 \geqslant \left(\chi^j \cdot x - \sum_{k=l'}^{m_{i'}} x_{i',k}\right) + \sum_{k=l'}^{r'} x_{i',k} + 1 + \sum_{k=l_{q-j-1}}^{m_{i_{q-j-1}}} x_{i_{q-j-1},k} = 1 + \chi^{j+1} \cdot x,$$

where the last equality follows from the definition of χ^{j+1}. Hence, $\chi^{j+1} \cdot x \leqslant 1$ is valid for P. □

Let K_3 be the matrix of cliques on 3 or more machines. Then we define the polytope P_c as follows.

$$P_c := \{x \in R_+^D \mid K_1 \cdot x \leqslant 1 \,\&\, K_2 \cdot x \leqslant 1 \,\&\, K_3 \cdot x \leqslant 1\}.$$

Now we are ready to complete the proof of Theorem 3.

Proof of Theorem 3. First, we claim that P_c is integral. Since the matrices K_1, K_2, and K_3 contain all maximal cliques of $\hat{G}_\sigma$, P_c is the fractional node packing polytope of this graph. Since comparability graphs are perfect and the fractional node packing polytope of a perfect graph is integral [7], we deduce that P_c is integral.

Second, we claim that P is a face of P_c. To see this, we define the polytope $P_{\leqslant} := \{x \in R_+^D \mid K_1 \cdot x = 1 \,\&\, K_2 \cdot x \leqslant 1 \,\&\, K_3 \cdot x \leqslant 1\}$. Notice that $P_{\leqslant}$ is a face of P_c. Clearly, $P_{\leqslant} \subseteq P$ holds. On the other hand, we have just seen that $K_3 \cdot x \leqslant 1$ is valid on P, hence $P = P_{\leqslant}$. Thus, P is a face of P_c as well.

To finish the proof we note that any face of P_c is integral, since P_c is integral. Hence, P is integral. □

We conclude this section by an immediate consequence of Theorem 3.

Corollary 9. *$P(I_f) = P$, and there is a one-to-one correspondence between the vertices of P and the stable sets of size m in $\hat{G}_\sigma$.*

Due to this corollary, there exists a concise description of all feasible insertions in the form of the system of equalities and inequalities defining P. On the other hand, feasible insertions can equivalently be characterised as maximum size stable sets of the graph $\hat{G}_\sigma$.

5. Solving ILP in polynomial time

In this section we show that ILP can be solved in polynomial time. First, we design a polynomial time procedure to decide whether the optimal $C^*_{\max}$ to ILP is not greater than a given upper bound. Then we apply bisection search to find the optimum. But remember, the optimal $C^*_{\max}$ is a lower bound on $OPT(\sigma, J_{n+1})$, so we do not claim that the insertion problem can be solved in polynomial time.

We start with deriving new clique constraints from a given upper bound $UB > 0$. Recall that the head of v_i would be at least $\tilde{h}_{i,l}$, if it was inserted in the lth position on M_i. Similarly, the tail of $v_{i'}$, would be at least $\tilde{t}_{i',l'}$ if it was inserted in the l'th position on $M_{i'}$ (cf. Section 2). Now, if

$$i < i' \quad \text{and} \quad \tilde{h}_{i,l} + \sum_{k=i}^{i'} p_k + \tilde{t}_{i',l'} > UB, \tag{10}$$

then in any feasible insertion in which $C^*_{\max} \leqslant UB$, v_i and $v_{i'}$ cannot be inserted simultaneously in the insertion points l and l' on M_i and $M_{i'}$, respectively.

Since $\tilde{h}_{i,l} < \tilde{h}_{i,l+1}$ and $\tilde{t}_{i',l'-1} > \tilde{t}_{i',l'}$, we have the following new clique constraints:

$$\sum_{k=0}^{l'} x_{i',k} + \sum_{k=l}^{m_i} x_{i,k} \leqslant 1, \tag{11}$$

for each (i, l, i', l') which satisfies (10). Of course, it is enough to keep those that are not implied.

A variable $x_{i,l}$ has to be set to 0, i.e.,

$$x_{i,l} = 0, \tag{12}$$

$$\text{if } \tilde{h}_{i,l} + p_i + \tilde{t}_{i,l} > UB, \quad \text{or } \tilde{h}_{i,l} + \sum_{k=i}^{m} p_k > UB, \quad \text{or}$$

$$\sum_{k=1}^{i} p_k + \tilde{t}_{i,l} > UB. \tag{13}$$

Example. We compute the new constraints for our standard example with respect to $UB = 7$. To facilitate the computation, consider Fig. 4(a), which shows the $\tilde{h}_{i,l}$ and $\tilde{t}_{i,l}$ values for each insertion point $x_{i,l}$. We obtain the following equalities and inequalities:

$$\begin{aligned}
x_{1,1} + x_{1,2} + x_{1,3} + x_{2,0} + x_{2,1} + x_{2,2} &\leqslant 1, \quad \text{for } 3+3+2 > 7,\\
x_{1,2} &= 0, \quad \text{for } 5+4 > 7,\\
x_{1,3} &= 0, \quad \text{for } 6+4 > 7,\\
x_{3,0} + x_{3,1} + x_{2,1} + x_{2,2} + x_{2,3} &\leqslant 1, \quad \text{for } 3+3+3 > 7,\\
x_{2,1} &= 0, \quad \text{for } 3+2+3 > 7,\\
x_{2,2} &= 0, \quad \text{for } 4+2+2 > 7,\\
x_{2,3} &= 0, \quad \text{for } 5+3 > 7,
\end{aligned}$$

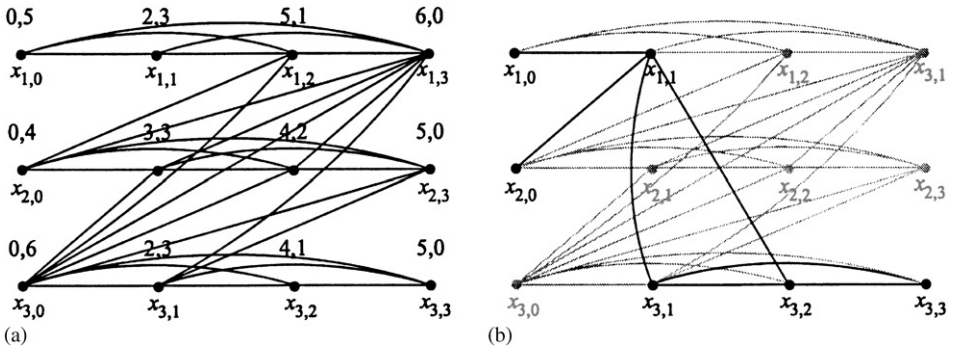

Fig. 4. (a) The $\tilde{h}_{i,l}, \tilde{t}_{i,l}$ pairs for each insertion point $x_{i,l}$. (b) The transformed insertion graph with respect to $UB = 7$.

$$x_{3,0} + x_{3,1} + x_{3,2} + x_{2,2} + x_{2,3} \leqslant 1, \quad \text{for } 4+3+1 > 7,$$
$$x_{3,0} + x_{3,1} + x_{3,2} + x_{1,1} + x_{1,2} + x_{1,3} \leqslant 1, \quad \text{for } 3+4+1 > 7,$$
$$x_{3,0} = 0, \quad \text{for } 4+6 > 7.$$

Let $I_f(UB) \subseteq I_f$ be the set of feasible insertions which satisfy (11) and (12).

Let

$$P(UB) := \{x \in R_+^D \mid K_1 \cdot x = 1,\ K_2 \cdot x \leqslant 1,\ K_{\leqslant}(UB) \cdot x \leqslant 1,\ K_0(UB) \cdot x = 0\},$$

where $K_{\leqslant}(UB)$ is the matrix of new inequalities (11), and $K_0(UB)$ is the matrix of (12).

Let $INT(P(UB))$ denote the set of integral, i.e., 0/1, vectors in $P(UB)$. We claim that $I_f(UB) = INT(P(UB))$. On the one hand, we already know that $I_f = INT(P)$ (cf. Section 4). On the other hand, the inequalities (11) and (12) cut off the same vectors from I_f and $INT(P)$, hence the claim follows. Our goal is to show:

Theorem 10. *$P(UB)$ is integral, and for any $UB \geqslant \max\{C(\sigma), \sum_{i=1}^{m} p_i\} : P(UB) \neq \emptyset$ if and only if $C^*_{\max} \leqslant UB$, where $C^*_{\max}$ is the optimum value of ILP.*

We defer the proof until we obtain some more results on the insertion graph. The next lemma provides us with more insight into the structure of $\hat{G}_\sigma$:

Lemma 11. *Suppose we orient the edges of $\hat{G}_\sigma$ according to the orientation rules (cf. Section 4.1). Let $(a,b) = (x_{i',k'}, x_{i,k})$ be any directed edge of the oriented graph. Then $\tilde{h}_{i',k'} < \tilde{h}_{i,k}$ and $\tilde{t}_{i',k'} > \tilde{t}_{i,k}$ always hold.*

Proof. First, note that the construction of $\hat{G}_\sigma$ implies that there exists a directed path from $u_{i',k'}$ to $u_{i,k}$ in G_σ. Hence, we have $\tilde{h}_{i',k'} = h_{i',k'} + p_{i',k'} \leqslant h_{i,k} < h_{i,k} + p_{i,k} = \tilde{h}_{i,k}$ and $\tilde{t}_{i',k'} = t_{i',k'+1} + p_{i',k'+1} > t_{i',k'+1} \geqslant t_{i,k+1} + p_{i,k+1} = \tilde{t}_{i,k}$. □

We construct a new graph $\hat{G}_\sigma(UB)$ out of $\hat{G}_\sigma$ as follows:

- Add new cliques equivalent to the inequalities (11) to $\hat{G}_\sigma$.
- Delete each insertion point identified with a variable $x_{i,l}$ which is set to 0.

The transformed insertion graph with respect to $UB = 7$ is depicted in Fig. 4(b).

Lemma 12. *$\hat{G}_\sigma(UB)$ is a comparability graph.*

Proof. We orient the edges of $\hat{G}_\sigma(UB)$ according to the orientation rules. Let (a,b) and (b,c) be directed edges of the oriented graph. We claim that there exists a directed edge (a,c) in the oriented $\hat{G}_\sigma(UB)$ as well.

If (a,b) and (b,c) are both directed edges of the oriented $\hat{G}_\sigma$, then the claim is proved by Lemma 5.

So, we have to verify only the cases when at least one of (a,b) and (b,c) is a new edge. Let $a = x_{i'',k''}$, $b = x_{i',k}$, and $c = x_{i,k}$. By the choice of a, b, c, and the rules of the orientation, we have $i'' \geqslant i' \geqslant i$. Furthermore, when $i = i'$ or $i' = i''$ then the three points pertain to the same clique, hence the third edge clearly exists, and it is oriented from a to c by the orientation rules. Thus, we may assume $i'' > i' > i$. We distinguish between three cases:

- $\{a,b\}$ is a new edge, but $\{b,c\}$ is an edge in $\hat{G}_\sigma$. First, $\tilde{h}_{i',k'} < \tilde{h}_{i,k}$ holds, by Lemma 11. Since $\{a,b\}$ is a new edge, $\tilde{h}_{i',k'} + \sum_{l=i'}^{i''} p_l + \tilde{t}_{i'',k''} > UB$. Consequently,
$$UB < \tilde{h}_{i,k} + \sum_{l=i'}^{i''} p_l + \tilde{t}_{i'',k''} < \tilde{h}_{i,k} + \sum_{l=i}^{i''} p_l + \tilde{t}_{i'',k''}$$
holds as well. Hence, $\{a,c\}$ is an edge of $\hat{G}_\sigma(UB)$, too. By the orientation rules it is oriented from a to c in the oriented $\hat{G}_\sigma(UB)$.
- $\{b,c\}$ is a new edge, but $\{a,b\}$ is an edge of $\hat{G}_\sigma$. Similar to the previous case.
- Both $\{a,b\}$ and $\{b,c\}$ are new edges. Since the two edges are new in $\hat{G}_\sigma(UB)$ and b is not deleted from $\hat{G}_\sigma(UB)$, the following inequalities hold:
$$\tilde{h}_{i,k} + \sum_{l=i}^{i'} p_l + \tilde{t}_{i',k'} > UB,$$
$$\tilde{h}_{i',k'} + \sum_{l=i'}^{i''} p_l + \tilde{t}_{i'',k''} > UB,$$
$$\tilde{h}_{i',k'} + p_{i'} + \tilde{t}_{i',k'} \leqslant UB.$$

Consequently,
$$\tilde{h}_{i,k} + \sum_{l=i}^{i''} p_l + \tilde{t}_{i'',k''} = \left(\tilde{h}_{i,k} + \sum_{l=i}^{i'} p_l + \tilde{t}_{i',k'}\right) + \left(\tilde{h}_{i',k'} + \sum_{l=i'}^{i''} p_l + \tilde{t}_{i'',k''}\right) - (\tilde{h}_{i',k'} + p_{i'} + \tilde{t}_{i',k'}) > UB.$$

Hence, $\{a,c\}$ is an edge of $\hat{G}_\sigma(UB)$. By the orientation rules, it is oriented from a to c in the oriented $\hat{G}_\sigma(UB)$.

All cases are verified, the statement is proved. □

Proof of Theorem 10. To show that $Conv(INT(P(UB))) = P(UB)$ we can apply the proof technique of Theorem 3. We must use the fractional node packing polytope of $\hat{G}_\sigma(UB)$, of course. The details are omitted.

We turn to the second part of the theorem. Suppose $P(UB) \neq \emptyset$. Let $x \in P(UB)$ be any vertex of the polytope. We claim that (x, UB) is a feasible solution to ILP. Clearly, x is integral and satisfies (1) and (2). It remains to be shown that (x, UB) satisfies (6) as well. Assume the contrary, and suppose a $\alpha \cdot x + UB < \beta$ holds for some inequality in (6). Clearly, $UB \geqslant \max\{C(\sigma), \sum_{i=1}^{m} p_i\}$ holds by assumption. There are two cases to verify:

- Some inequality in (6′) is violated, i.e., $-\sum_{l=0}^{m_i}(\tilde{h}_{i,l} + \tilde{t}_{i,l})x_{i,l} + UB < p_i$ holds for some $1 \leqslant i \leqslant m$. Since x is an insertion, there exists a unique $k \in \{0,\ldots,m_i\}$, such that $x_{i,k} = 1$. Consequently,

$$p_i > -\sum_{l=0}^{m_i}(\tilde{h}_{i,l} + \tilde{t}_{i,l})x_{i,l} + UB = -(\tilde{h}_{i,k} + \tilde{t}_{i,k}) + UB.$$

  But $x \in P(UB)$ implies $x_{i,k} = 0$, a contradiction.
- Some inequality in (6″) is hurt, i.e., $-\sum_{l=1}^{m_i} \tilde{h}_{i,l}x_{i,l} - \sum_{l=0}^{m_{i'}-1} \tilde{t}_{i',l}x_{i',l} + UB < \sum_{l=i}^{i'} p_l$ for some $1 \leqslant i < i' \leqslant m$. Since x is an insertion, there exists a unique $k \in \{0,\ldots,m_i\}$ such that $x_{i,k} = 1$, and there exists a unique $k' \in \{0,\ldots,m_{i'}\}$ such that $x_{i',k'} = 1$. Consequently,

$$\sum_{l=i}^{i'} p_l > -\sum_{l=1}^{m_i} \tilde{h}_{i,l}x_{i,l} - \sum_{l=0}^{m_{i'}-1} \tilde{t}_{i',l}x_{i',l} + UB = -(\tilde{h}_{i,k} + \tilde{t}_{i',k'}) + UB.$$

  Since $x \in P(UB)$, either $x_{i,k} = 0$ or $x_{i',k'} = 0$ has to hold, a contradiction.

The claim is verified. Since (x, UB) is feasible to ILP, $C^*_{\max} \leqslant UB$ trivially holds.

Conversely, suppose $C^*_{\max} \leqslant UB$. We have to show that $P(UB) \neq \emptyset$. Let $(x^*, C^*_{\max})$ be an optimal solution to ILP. We claim that $x^* \in P(UB)$. It is enough to verify that $K_{\leqslant}(UB) \cdot x^* \leqslant 1$ and $K_0(UB) \cdot x^* = 0$. This is true by construction. □

Corollary 13. *$P(UB) = P(I_f(UB))$ and there is a one-to-one correspondence between the vertices of $P(UB)$ and the stable sets of size m of $\hat{G}_\sigma(UB)$, for any $UB \geqslant 0$.*

In particular, $P(UB) = \emptyset$ if and only if $\alpha(\hat{G}_\sigma(UB)) < m$.

Corollary 14. *$P(C^*_{\max}) \neq \emptyset$ and there is a one-to-one correspondence between the vertices of $P(C^*_{\max})$ and the optimal solutions of ILP.*

Since $I_f(U_1) \subseteq I_f(U_2)$ whenever $U_1 \leqslant U_2$, the optimum value $C^*_{\max}$ of ILP is the smallest UB such that $P(UB) \neq \emptyset$. We can find this value by means of binary search

as follows:

(1) $L = C(\sigma)$, $U = L + \sum_{k=1}^{m} p_k$.
(2) While $L < U$ repeat the following:
  2.1. $UB = (L + U)/2$
  2.2. If $P(UB) \neq \emptyset$, then $U = UB$, otherwise $L = UB + 1$.
(3) Output U.

When the algorithm stops, $L = U = C^*_{\max}$. In order to have polynomial running time, $P(UB) \neq \emptyset$ must be tested in polynomial time. In the sequel we describe a combinatorial approach to do this task. Namely, the remark after Corollary 13 implies that $P(UB) \neq \emptyset$ if and only if $\alpha(\hat{G}_\sigma(UB)) = m$. Moreover, recall that $\hat{G}_\sigma(UB)$ is a comparability graph for all $UB \geqslant 0$, by Lemma 12. Consequently, its stability number can be computed by minimum flow computation (see e.g. [12, p. 134]). The minimum flow problem can be solved in polynomial time (in the size of the network) (see e.g. [2, p. 202]). On the other hand, the graph $\hat{G}_\sigma(UB)$ can be constructed in polynomial time by simply testing the conditions (10) and (13), whose number is polynomially bounded in the size of $\hat{G}_\sigma$. Consequently, we can solve our decision problem in polynomial time.

By a simple trick we can avoid the minimum flow computation. That is, by following the construction in [12], we extend $\hat{G}_\sigma(UB)$ to a transportation network by adding two new vertices s and t and edges (s, a) and (b, t) for each source a and sink b of $\hat{G}_\sigma(UB)$. Assigning a lower capacity 1 to each vertex of $\hat{G}_\sigma(UB)$, we initialise a particular compatible integer-valued flow. Namely, the initial flow is the sum of m unit flows covering the insertion points on each machine. Now, this flow is clearly compatible and is of minimum value if and only if there exists no flow augmenting path. Such a path can be found in linear time, if it exists. Since the value of the minimum flow is exactly the stability number of the comparability graph ([12, p. 134]), we can conclude that $\alpha(\hat{G}_\sigma(UB)) = m$ if and only if there exists no flow augmenting path in the transportation network associated with $\hat{G}_\sigma(UB)$ with respect to the particular initial flow described above.

The second method is based on linear programming. Notice that the matrices $K(UB)_{\leqslant}$ and $K_0(UB)$ can be constructed in polynomial time in the same way as $\hat{G}_\sigma(UB)$. Once the matrices are known, a linear programming package can find a vertex of $P(UB)$ or it states that the polytope is empty. The drawback of this method is that the practical running time of the implementation heavily relies on the linear programming package used.

6. Computational results

In this section we investigate the computational aspects of our lower bound. We aim at two objectives. On the one hand, we would like to demonstrate experimentally that our lower bound is rather strong. To this end, we have computed the lower bounds for more than 1100 insertion problem instances and compared them to the optimal solutions. On the other hand, we would like to show that our combinatorial method for computing the lower bound is very fast, i.e., it is suitable for practical computations.

Table 2
The compositions of the datasets

	Dataset		
	m5	m10	m15
Contents	la01-la15	la16-la35 orb01-orb10 swv01-swv05	abz7-abz9 la36-la40 swv06-swv10

We have obtained the insertion problem instances from job shop problem instances as follows. A job shop problem instance with n jobs gives rise to a series of n insertion problem instances $(\sigma_0, J_1), \ldots, (\sigma_k, J_{k+1}), \ldots, (\sigma_{n-1}, J_n)$, where σ_0 is the empty schedule and σ_{k+1} is obtained from σ_k by inserting J_{k+1} into σ_k in an optimal way, $k=0, \ldots, n-1$. We denote by $OPT(\sigma_k, J_{k+1})$ and $LB(\sigma_k, J_{k+1})$ the makespan of an optimal insertion of J_{k+1} into σ_k and the lower bound computed by our method for the same problem instance, respectively.

To generate the series of insertion problem instances, we used the "abz7-abz9" instances of Adams et al. [1], the "la01-la40" instances of Lawrence [15], the "orb01-orb10" instances of Applegate and Cook [3], and the "swv01-swv10" instances of Storer et al. [21].

When presenting our results, we will distinguish between problem instances with 5, 10 and 15 machines. In fact, in all problem instances the number of machines is equal to the number of operations of the job to be inserted, since each job has exactly one operation on each machine. Clearly, the number of operations to be inserted influences the quality of the lower bound and the computation time as well. Hence, we have 3 datasets, "m5", "m10" and "m15", whose composition is depicted in Table 2.

In Section 6.1 we evaluate the quality of the lower bound obtained by the proposed method while in Section 6.2 we report on the computation times.

6.1. The quality of the lower bound

We computed for each insertion problem instance a lower bound by the method of Section 5 and the makespan of an optimal feasible insertion by an exact method. Various exact methods are discussed in [14], but the description and evaluation of such methods are out of the scope of this paper.

First note that when J_{k+1} is inserted into σ_k, the relative error between $LB(\sigma_k, J_{k+1})$ and $OPT(\sigma_k, J_{k+1})$ is

$$\text{relative error} = (OPT(\sigma_k, J_{k+1}) - \text{LB}(\sigma_k, J_{k+1}))/OPT(\sigma_k, J_{k+1}).$$

Fig. 5 depicts the mean relative error against the job being inserted for each of the 3 datasets. Below we explain how the mean value is computed. Consider, say, the dataset "m5". The 15 job shop problems define 15 series of insertion problem instances. We computed the relative error when J_{k+1} is inserted into σ_k in each of the 15 series and

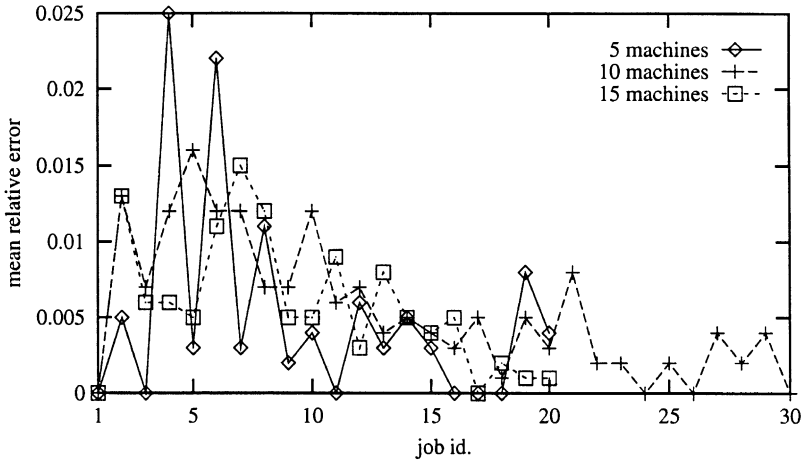

Fig. 5. The mean relative error between the optimal makespan and the ILP lower bound.

we took the mean of these numbers. We did the same with the other two datasets. The result is 3 series of mean relative errors corresponding to the 3 datasets.

Observe that the mean relative error tends to 0 as the problem size, i.e., the number of scheduled jobs, increases for each of the 3 datasets. This is what we expected, since clearly, $OPT(\sigma_k,J_{k+1})/OPT(\sigma_{k-1},J_K)$ decreases as k increases and $OPT(\sigma_{k-1},J_k) \leqslant LB(\sigma_k,\ J_{k+1}) \leqslant OPT(\sigma_k,J_{k+1})$ hold.

In "m10" after inserting 20 jobs the mean relative error increases. The reason is that the sample reduces to 5 insertion problem instances obtained from la31-la35, the other job shop problem instances in the dataset having at most 20 jobs.

Fig. 6 summarises the maximum relative error in the 3 datasets. The results depicted were obtained the same way as those in the previous figure except that we took the maximum instead of the mean of the sample. The development of the maximum relative error is similar to that of the mean, i.e., it tends to 0.

Fig. 7 shows the relative error from a different angle. We can state that in the great majority of cases the relative error is below 1%.

6.2. Computation time

A very important issue is the computation time needed to calculate the lower bounds. The running time of our procedure is primarily determined by the method for testing whether $P(UB) \neq \emptyset$ (cf. Section 5). We have described 2 methods, a combinatorial one and another based on linear programming. It turned out that the combinatorial method is much more efficient than linear programming. Interestingly, when we used the linear programming package CPLEX, most of the time was spent on creating the instance in CPLEX, testing whether a feasible solution exists took much less time. Since the results with linear programming are discouraging, we present the computation times for the combinatorial method only.

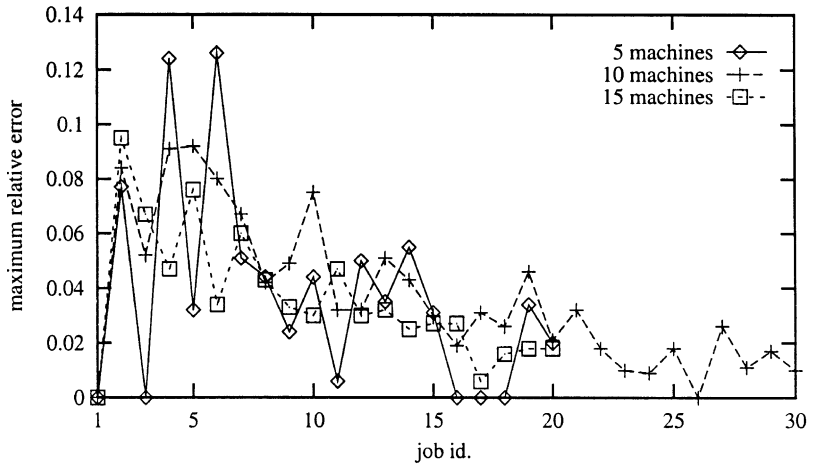

Fig. 6. The maximum relative error between the optimal makespan and the ILP lower bound.

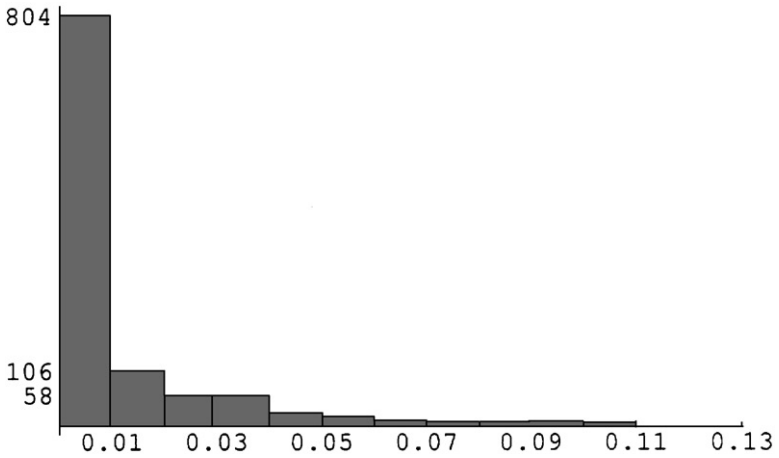

Fig. 7. Cumulative results. The heights of the bars indicate the number of instances with relative error between the cutoff points.

The computing environment was a PC with a 400 MHz CPU under the Linux operating system. In this processing environment the smallest measurable time unit is 0.01 s. Our algorithm was coded in C and was compiled by the "egcs-2.91.66" compiler.

For all instances in the "m5" dataset, the time needed to compute the lower bound was never more than 0.015 s, so we do not provide a detailed diagram.

Figs. 8 and 9 depict the computation times on the "m10" and "m15" datasets, respectively. The two ends of the errorbar over job k indicate the minimum and the

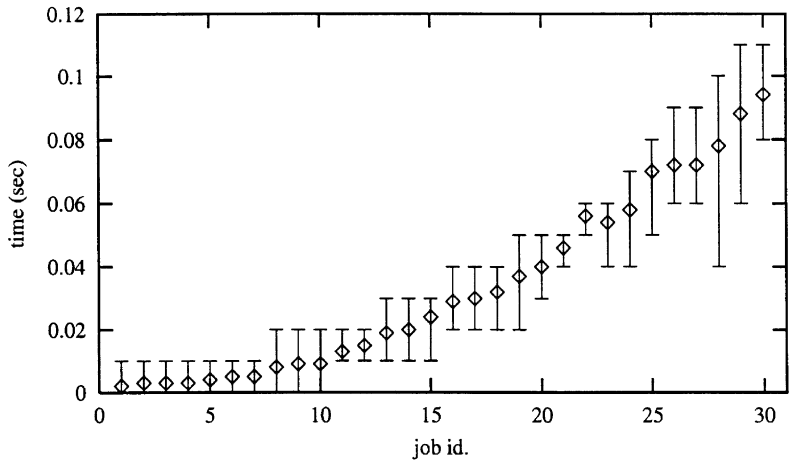

Fig. 8. CPU times of inserting jobs on 10 machines.

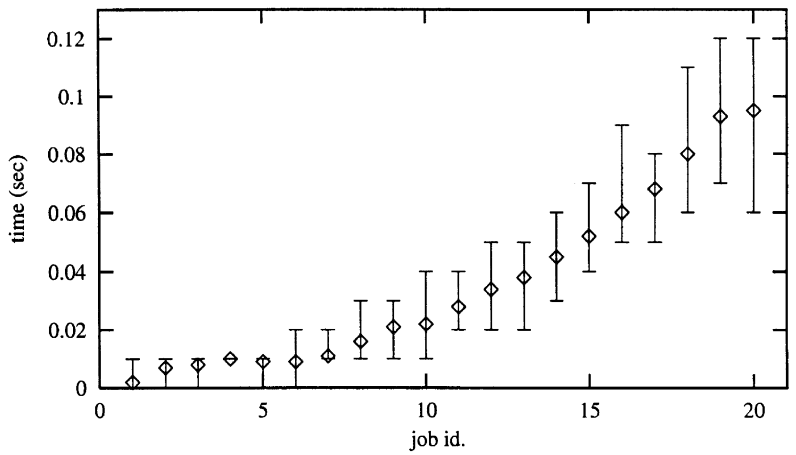

Fig. 9. CPU times of inserting jobs on 15 machines.

maximum time needed to compute the lower bound upon solving the kth instance of the insertion problem series defined by the dataset. Moreover, the "diamonds" on the errorbars indicate the mean computation times.

We can observe that in both datasets the mean computation time follows a quadratic curve. Moreover, in the "m15" dataset it increases more rapidly than in the "m10" dataset, which is quite expected. However, even in this case the computing time is by no means prohibitive for practical computations.

7. Conclusions and future work

In this paper we have studied the job insertion problem. We have given various characterisations of feasible insertions and developed a lower bounding method. To our best knowledge, no lower bound was known to this problem. Our computational results show that the bound is strong and can be computed efficiently in polynomial time.

There are various directions to continue this work, some of them have partially been explored in [14]. We have developed a branch-and-bound algorithm to solve the job insertion problem to optimality in which the lower bound in each search tree node is computed by the method proposed in this paper. The computational results are quite promising and we plan to report on an extensive computational study in a forthcoming paper.

The job insertion problem can be defined for more general scheduling problems as well, e.g., when the operations of the jobs must be processed on sets of machines instead of single machines. In this case the characterisation of the set of feasible insertions of a sequence of operations into a schedule is a major open problem. Note that a lot of work has been done on inserting a single operation that requires a set of machines, see e.g., [10,6,4,14]. We believe that these results can be extended to a theory of the simultaneous insertion of a sequence of operations where each operation in the sequence may require several machines.

Acknowledgements

The authors thank Prof. Dominique de Werra and the two anonymous referees for their constructive comments. The research reported in this paper was partly supported by the Swiss Commission for Technology and Innovation (CTI) grant no. 4173 while both authors were affiliated with the Department of Mathematics of the Swiss Federal Institute of Technology at Lausanne.

References

[1] J. Adams, E. Balas, D. Zawack, The shifting bottleneck procedure for job shop scheduling, Management Sci. 34 (3) (1988) 391–401.
[2] R.K. Ahuja, T.L. Magnanti, J.B. Orlin, Network Flows, Prentice-Hall, Englewood Cliffs, NJ, 1993.
[3] D. Applegate, W. Cook, A computational study of the job-shop scheduling instance, ORSA J. Comput. 3 (1991) 149–156.
[4] C. Artigues, F. Roubellat, A polynomial activity insertion algorithm in a multi-resource schedule with cumulative constraints and multiple modes, European J. Oper. Res. 127 (2000) 297–316.
[5] E. Balas, J.K. Lenstra, A. Vazacopoulos, The one-machine problem with delayed precedence constraints and its use in job shop scheduling, Management Sci. 41 (1) (1995) 94–109.
[6] P. Brucker, J. Neyer, Tabu-search for the multi-mode job-shop problem, OR Spektrum 20 (1998) 21–28.
[7] V. Chvátal, On certain polytopes associated with graphs, J. Combinat. Theory B 18 (1975) 138–154.
[8] E.G. Coffman Jr., Computer and Job-Shop Scheduling Theory, Wiley, New York, 1976.

[9] S. Dauzère-Péres, J. Paulli, An integrated approach for modelling and solving the general multiprocessor job-shop scheduling problem using tabu search, Ann. Oper. Res. 70 (1997) 281–336.
[10] S. Dauzère-Péres, W. Roux, J.B. Lasserre, Multi-resource shop scheduling with resource flexibility, European J. Oper. Res. 107 (1998) 289–305.
[11] M. Dell' Amico, M. Trubian, Applying tabu search to the job-shop scheduling problem, Ann. Oper. Res. 41 (1993) 231–252.
[12] M.C. Golumbic, Algorithmic Graph Theory and Perfect Graphs, Academic Press, New York, 1980.
[13] A.S. Jain, S. Meeran, A state-of-the-art review of job-shop scheduling techniques., Technical Report, Department of Applied Physics, Electronic and Mechanical Engineering, University of Dundee, Dundee, Scotland, 1998.
[14] T. Kis, Insertion techniques for job shop scheduling, Ph.D. Thesis, Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, 2001.
[15] S. Lawrence, Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques, Supplement, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 1984.
[16] M. Mastrolilli, L.M. Gambardella, Effective neighbourhood function for the flexible job shop problem, J. Schedul. 3 (2000) 3–20.
[17] E. Nowicki, C. Smutnicki, A fast taboo search algorithm for the job-shop problem, Management Sci. 42 (6) (1996) 797–813.
[18] A. Schrijver, Theory of Linear and Integer Programming, Wiley, New York, 1986.
[19] Y.N. Sotskov, The complexity of shop-scheduling problems with two or three jobs, European J. Oper. Res. 53 (1991) 326–336.
[20] Y.N. Sotskov, T. Tautenhahn, F. Werner, On the application of insertion techniques for job shop problems with setup times, RAIRO Rech. Opér. 33 (2) (1999) 209–245.
[21] R.H. Storer, S.D. Wu, R. Vaccari, New search spaces for sequencing instances with application to job shop scheduling, Management Sci. 38 (1992) 1495–1509.
[22] R.J.M. Vaessens, E.H.L. Aarts, J.K. Lenstra, Job shop scheduling by local search, INFORMS J. Comput. 8 (1996) 302–317.
[23] F. Werner, A. Winkler, Insertion techniques for the heuristic solution of the job shop problem, Discrete Appl. Math. 58 (1995) 191–211.