Linear programming based algorithms for preemptive and non-preemptive RCPSP

Jean Damay, Alain A. Quilliot, Eric Sanlaville

Linear Programming based algorithms for preemptive and non-preemptive RCPSP

Jean Damay, Alain Quilliot and Eric Sanlaville
{jean.damay, alain.quilliot, eric.sanlaville}@isima.fr
Laboratoire LIMOS-CNRS UMR 6158, Université de Clermont II
Complexe Scientifique des Cézeaux
63173 Aubière Cedex, France

Abstract: In this paper, the RCPSP (Resource Constrained Project Scheduling Problem) is solved using a linear programming model. Each activity may or may not be preemptive. Each variable is associated to a subset of independent activities (antichains). The properties of the model are first investigated. In particular, conditions are given that allow a solution of the linear program to be a feasible schedule. From these properties, an algorithm based on neighbourhood search is derived. One neighbour solution is obtained through one Simplex pivoting, if this pivoting preserves feasibility. Methods to get out of local minima are provided. The solving methods are tested on the PSPLIB instances in a preemptive setting and prove efficient. They are used when preemption is forbidden with less success, and this difference is discussed.

Keywords: Scheduling, Resource Constrained Project, Preemption, Linear Programming, Column Generation.

1 Introduction

We consider in this paper the Resource Constrained Project Scheduling Problem (RCPSP). A project is composed of activities and subject to precedence and resource constraints, and the objective is minimizing the makespan. A precedence constraint consists in completing some activity before another one can start. The resource constraints specify that each activity requires constant amounts of renewable resources while being processed, these resources having limited capacities.

The approach developed in this paper involves variables associated to subsets of activities that can be simultaneously processed during the schedule. A time-indexed linear formulation of the non-preemptive version of the RCPSP involving these feasible subsets (here called valid antichains) has been introduced by Mingozzi et al. [7]: precedence and non-preemption constraints can be simply formulated in spite of a very large number of variables (depending on the length of the discretized time interval $[0; T]$, where T is an upper bound of the makespan, and on the number of valid antichains of the project). In such approaches, the resource constraints are taken into account in the computation of these antichains, which allows to eliminate these constraints from the linear formulation.

By evading the non-preemption constraints and partially the precedence constraints, Mingozzi et al. also formulate an LP-relaxation of the RCPSP, which was slightly modified by Brucker and Knust [3] and then Baptiste and Demassey [1] to give the best known lower bounds of the problem. They both implement a destructive approach on T and use constraint propagation techniques cooperating with the resolution of this linear relaxation through column generation.

Concerning the preemptive version of the RCPSP (an activity can be interrupted and later resumed without penalty), it is assumed in papers that look

for optimality [5] [11], that all processing times are integer and that preemption only occurs at integer times. This may increase the optimal makespan, as in the following classical example: mono-resource project of capacity 2 with three unit-length and unit-resource-requiring unrelated activities. The optimal preemptive makespan is 1.5, whereas the makespan with integer preemptions is 2. In this paper, it is assumed that for a subset of activities, preemption is allowed and occurs at arbitrary rational times.

In section 2, the model based on Linear Programming and valid antichains is defined. In section 3, algorithms that explore the set of (preemptive or non-preemptive) feasible solutions are presented. The next section provides numerical results on classical instances of the literature, and the paper ends with some conclusions and perspectives.

2 The valid antichain based model

2.1 Notations and definitions

Let $\mathcal{R}$ be the set of renewable resources, with a time-independent capacity R_k for each resource $k \in \mathcal{R}$. Let $V = \{1, 2, \ldots, n\}$ be the set of n activities. Each activity i has a processing time p_i and requires an amount of r_{ik} units of resource $k \in \mathcal{R}$. Denote by E the transitive closure of the precedence relation between activities: $(i, j) \in E$ if and only if activity i must be completed before the beginning of activity j. The graph $G = (V, E)$ is called *precedence graph*.

Let I be a set of activities, such that $i \in I$ if and only if preemption is not allowed for activity i. Of course $I = \emptyset$ (resp. $I = V$) for the preemptive (resp. non-preemptive) version of the problem. An I-schedule is defined by, for each activity $i \in V$, a list of time intervals on which i is executed, such that

- at any time, the total resource requirements are not greater than the corresponding capacities
- precedence constraints between activities are respected
- if $i \in I$, the number of associated time intervals is equal to 1

The I-RCPSP consists in finding an I-schedule that minimizes the makespan C_{max} (the maximum ending value of all time intervals). Without loss of generality, it is assumed that for any $t < C_{max}$, at least one activity is executed at t. In Figure 1, the precedence graph (except for transitivity arcs) of an instance with 7 activities and one resource is presented. Suppose $I = \{1, 2, 5, 6, 7\}$. A Gantt chart is also given, representing a solution for the instance. E.g. for activity 4, the time intervals are $\{[0; 1], [3, 4]\}$. This solution is an I-schedule (and, in fact, optimal among preemptive solutions - when $I = \emptyset$).

The core of our approach is the notion of subsets of independent activities, hereafter called *antichains*. A subset $a \subseteq V$ is an antichain if and only if $\forall i, j \in a, (i, j) \notin E$ (antichain property in the graph G). It is a *valid* antichain if $\forall k \in \mathcal{R}, \sum_{i \in a} r_{ik} \leq R_k$ (resource property). Denote by $\mathcal{A}$ the set of all valid antichains of the project.

Let us define on $\mathcal{A}$ a precedence relation $\prec$: for two antichains $a, b \in \mathcal{A}$, $a \prec b$ if and only if there exist two activities i, j such that $i \in a, j \in b,$ and $(i, j) \in E$.

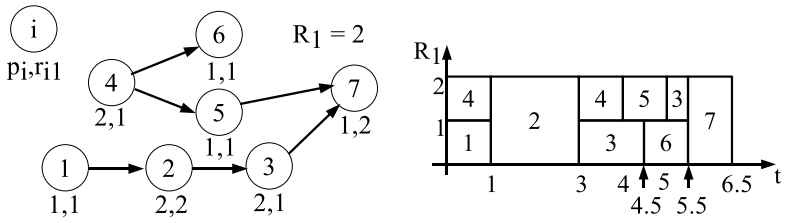


Figure 1: precedence graph and Gantt chart of a $\{1, 2, 5, 6, 7\}$-schedule

For each activity $i \in V$ we denote by $\mathcal{A}_i$ the set of valid antichains containing i. Let $\sigma = (a_1, \ldots, a_L)$ be a sequence of valid antichains, and for all $i \in V$, let $\sigma_i = \mathcal{A}_i \cap \sigma$ be the set of antichains of σ containing i. σ is said *ordered* if the order of this sequence respects the partial order given by $\prec$, that is to say $\forall l, l' \in \{1, \ldots, L\}, l < l' \Rightarrow a_{l'} \not\prec a_l$. σ is said *I-consecutive* if and only if for each activity $i \in I$, the antichains of σ_i are consecutive in σ. σ is said *I-candidate* if it is ordered and I-consecutive. Note now that for the preemptive version of the problem, a sequence is $\emptyset$-candidate if and only if it is ordered.

The I-schedule of Figure 1 can be seen as the sequence of valid antichains $(\{1,4\},\{2\},\{3,4\},\{3,5\},\{5,6\},\{3,6\},\{7\})$, valued by durations $(1, 2, 1, 0.5, 0.5, 0.5, 1)$. The equivalence between schedules and sequences is stated below.

Proposition 2.1 *Let σ be a sequence of valid antichains, and $\boldsymbol{z}$ be a positive vector indexed on σ. The couple $(\sigma, \boldsymbol{z})$ represents an I-schedule if and only if σ is I-candidate and $\boldsymbol{z}$ verifies $\forall i \in V, \sum_{a \in \sigma_i} z_a = p_i$. Its makespan is $\sum_{a \in \sigma} z_a$.*

Proof. ($\Leftarrow$) We can easily transform σ into a list of intervals for each activity, weighted by the value of $\mathbf{z}$ for the associated antichains, and that respects the two first properties of an I-schedule. As the sequence is by definition I-consecutive, the initial non-preemption property for the activities of I is respected.
($\Rightarrow$) If we define an *event* as the beginning or the end of a time interval, an I-schedule can be transformed into a sequence of valid antichains by generating a new antichain at each event. An antichain is valued by the time spent till the next event. This sequence is ordered and for each $i \in I$, the antichains containing i are consecutive, so we can define an I-candidate valid antichain sequence. By construction and because of the lack of activity-empty time intervals, the time spent between 0 and the latest event is the sum of the values of $\mathbf{z}$ for all generated antichains. ■

By a slight abuse of notation, a couple $(\sigma, \mathbf{z})$ where σ is I-candidate and $\forall i \in V, \sum_{a \in \sigma_i} z_a \geq p_i$ represents also an I-schedule.

Let f be a family of valid antichains, and for all $i \in V$, let $f_i = \mathcal{A}_i \cap f$ be the set of antichains of f containing i. In the following, $\prec_f$ is the restriction of $\prec$ to the family f. By extension, a family f of valid antichains is said *I-candidate* if and only if we can build with exactly all the antichains of f an I-candidate

sequence.

f is said *complete* if and only if $\forall i \in V, f_i \neq \emptyset$.

2.2 Antichain formulation

Mingozzi et al. presented in [7] the linear model $\mathcal{P}$ for the RCPSP:

$$(\mathcal{P}) : \min Z = \mathbf{1} \cdot \mathbf{z}, A\ \mathbf{z} \geq \mathbf{p},\ \mathbf{z} \in \mathbf{R}_+^{|\mathcal{A}|}.$$

The *incidence* matrix $A(n, |\mathcal{A}|)$ is defined as follows: $A_{il} = 1$ if activity i belongs to antichain a_l, 0 otherwise. $\mathbf{p}$ is the vector of all processing times $\mathbf{p} = (p_i)_{i \in V}$. This formulation relaxes the non-preemption constraints, and partially the precedence constraints (any precedence constraint becomes a disjunction).

We say that an antichain a_l is *active* for a solution $\mathbf{z}$ of $\mathcal{P}$ if and only if $z_l > 0$. Let us denote by $ACT(\mathbf{z})$ the set of active antichains for $\mathbf{z}$.

We now define $\mathcal{S}_I$ as the set of solutions $\mathbf{z}$ of $\mathcal{P}$ for which $ACT(\mathbf{z})$ is I-candidate.

Theorem 2.2 *Solving the I-RCPSP consists in finding an element of $\mathcal{S}_I$ that minimizes Z.*

Proof. Let $\mathbf{z}^*$ be an element of $\mathcal{S}_I$ minimizing Z. $ACT(\mathbf{z}^*)$ is an I-candidate family, therefore it is associated to an I-candidate sequence, say σ^*, so that $(\sigma^*, \overline{\mathbf{z}})$ (where $\overline{\mathbf{z}}$ is the restriction of $\mathbf{z}^*$ to $ACT(\mathbf{z}^*)$, re-indexed on σ^*) represents an I-schedule of makespan $\sum_{a \in \sigma^*} \overline{z}_a = \sum_{a \in \mathcal{A}} z_a^*$. Suppose there exists an I-schedule of makespan m' strictly lower. Thanks to Proposition 2.1, we obtain a couple $(\sigma', \mathbf{z}')$ such that $\sum_{a \in \sigma'} z'_a = m'$. Thus, we can obtain a vector $\mathbf{z}'^+$ indexed on $\{1, \ldots, \mathcal{A}\}$, whose components are $z_a'^+ = z'_a$ if $a \in \sigma'$, 0 else. This vector belongs by definition to $\mathcal{S}_I$ and has an objective value of m', a contradiction with the hypothesis on $\mathbf{z}^*$. ■

Let us now state the following:

Proposition 2.3 *If a valid antichain family f is I-candidate, then a (not necessarily complete) subfamily ϕ of f is also I-candidate.*

Proof. The valid antichain sequence σ associated to f is ordered and I-consecutive. If we remove the antichains of $f \backslash \phi$ from σ, the new sequence σ' (associated to ϕ) is obviously still ordered, and for each activity $i \in I$, σ'_i is still consecutive in σ'. ■

Remark The set of basis solutions (in the Simplex sense) for $\mathcal{P}$ in $\mathcal{S}_I$ is dominant.
Indeed, let $\mathbf{z}$ be an element of $\mathcal{S}_I$, and $\mathcal{P}_\mathbf{z}$ the restriction of $\mathcal{P}$ to the columns indexed by $ACT(\mathbf{z})$. $\mathcal{P}_\mathbf{z}$ admits at least one solution ($\mathbf{z}$ itself), hence admits one optimal basis solution, which is also a basis solution of $\mathcal{P}$, with a non-increasing value of Z. Proposition 2.3 ensures that the family associated to this basis solution is still I-candidate.
In the following, we shall only consider the basis solutions of $\mathcal{S}_I$. Hence, an element of $\mathcal{S}_I$ is characterized by exactly n variables of $\mathcal{P}$ (corresponding to its active antichains and some slack variables).

The proof of the following connexity theorem is written in appendix. It gives us further information on the set $\mathcal{S}_I$ of solutions of the I-RCPSP.

Theorem 2.4 (Connexity Theorem) *$\mathcal{S}_I$ is connected in the sense that two elements of $\mathcal{S}_I$ are neighbours if and only if one can be obtained from the other by exactly one Simplex pivoting.*

It follows that one may go from one solution of $\mathcal{S}_I$ to any other by a sequence of Simplex pivotings.

2.3 Characterization of I-candidate antichain families

We define now the elementary relation R_I between two directed pairs of antichains of f:

Definition 1 (relation R_I) *Let a, b, a', b' be four valid antichains of f. $(a,b)R_I(a',b')$ if and only if one of the following conditions is met:*

i) $(a,b) = (a',b')$

ii) $a = a'$ *and* $\exists i \in I$ *such that* $i \notin a$ *and* $i \in b, i \in b'$

iii) $b = b'$ *and* $\exists i \in I$ *such that* $i \notin b$ *and* $i \in a, i \in a'$.

Then we denote by R_I^* the transitive closure of relation R_I. As R_I is reflexive and symmetrical, R_I^* is an equivalence relation. We also set, for two antichains $a, b \in f$, $C^f_{(a,b)}$ as the equivalence class (a directed pair subset) of (a,b) for the R_I^* relation.

Let $(a',b') \in \mathcal{C}^f_{(a,b)}$. Informally, this means that, if a precedes b in a sequence σ associated to the I-candidate family f, then I-consecutiveness entails that a' precedes b' in σ (see Sequence Lemma below).

Let us consider now the same instance as in Figure 1, except that $I = V$. On Figure 2, antichains of the family determined by the schedule given in Figure 1, are represented as the nodes of two antichain graphs. The equivalence classes of the oriented pairs $(\{3,5\},\{5,6\})$ and $(\{1,4\},\{2\})$ are the arcs of the first and the second graph, respectively. Clearly the whole family is not V-candidate: it suffices to consider either of these two equivalence classes.

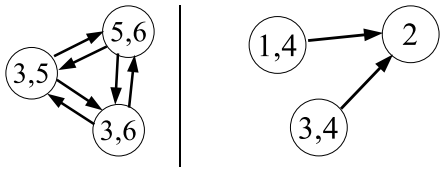


Figure 2: two equivalence classes for two subfamilies of valid antichains

Let us remark that $C^f_{(a,b)} = \{(a',b'), (b',a') \in C^f_{(b,a)}\}$.
Denote by $\prec_f^I = \bigcup_{u \in \prec_f} \mathcal{C}_u^f$ the set of all propagated arcs of $\prec_f$ through R_I^*. Let us give two preliminary lemmas before giving the main theorem of this section.

Lemma 2.5 (Sequence Lemma) *Let f be an I-candidate family of valid antichains and σ be an associated I-candidate sequence. Let a, b, a' and b' be four elements of f such that $(a', b') \in \mathcal{C}^f_{(a,b)}$ and a precedes b in σ. Then a' precedes b' in σ.*

Proof. Suppose first $(a, b)R_I(a', b')$, and let us consider $a = a'$. Then $\exists i \in I, i \in b \cap b' \backslash a$. Suppose b' precedes a in σ, then the antichains of σ_i are not consecutive in σ. The same holds if $b = b'$. In the general case ($(a, b)R^*_I(a', b')$), a simple induction provides the result. ■

Lemma 2.6 (Triangle Lemma) *If a, b, a', b', c in an antichain family f are such that $(a', b') \in \mathcal{C}^f_{(a,b)}$, $(a', c) \notin \mathcal{C}^f_{(a,b)}$ and $(c, b') \notin \mathcal{C}^f_{(a,b)}$, then $(c, a') \in \mathcal{C}^f_{(c,a)}$ and $(b', c) \in \mathcal{C}^f_{(b,c)}$.*

Proof. Suppose first $(a, b)R_I(a', b')$, and let us consider $a = a'$ (immediately $(c, a)R_I(c, a')$ and $(a, c) \notin \mathcal{C}^f_{(a,b)}$). Then $\exists i \in I, i \in b \cap b' \backslash a$. If $i \notin c$, then by definition $(b, c)R_I(b', c)$. If $i \in c$, then $(a', c)R_I(a, b)$, a contradiction with the hypotheses. So $(b, c)R_I(b', c)$, and $(c, b) \notin \mathcal{C}^f_{(a,b)}$. The same reasoning holds for $b = b'$. In the general case ($(a, b)R^*_I(a', b')$), an induction provides the result. ■

Theorem 2.7 (Characterization theorem) *A valid antichain family f is I-candidate if and only if both the following conditions are met:*

- C_1) *for all $a, b \in f$, $a \neq b$, such that neither (a, b) nor (b, a) belongs to $\prec^I_f$, we have $(b, a) \notin \mathcal{C}^f_{(a,b)}$*
- C_2) *the antichain subgraph $(f, \prec^I_f)$ does not contain any oriented cycle.*

Proof. This proof is inspired from [2], [9] involving ordered interval hypergraphs.
($\Rightarrow$): Let σ be an I-candidate sequence associated to f: C_1 comes directly from the Sequence lemma with $(a', b') = (b, a)$ in σ. To prove C_2, suppose there exists an oriented cycle in $(f, \prec^I_f)$. Then it is impossible to find a total order of all antichains of f which includes $\prec^I_f$, so impossible to find a sequence that is ordered and that respects I-consecutiveness.
($\Leftarrow$): Let us suppose that both conditions C_1 and C_2 are satisfied. In case $\prec^I_f$ is a total order, the result is obvious. Otherwise, we may choose a and b in f such that neither (a, b) nor (b, a) is in $\prec^I_f$. We shall prove that it is possible to insert either (a, b) or (b, a) into $\prec^I_f$ without losing condition C_2.

Suppose $\prec^I_f \cup \mathcal{C}^f_{(a,b)}$ contains some oriented cycle Γ_{ab} and $\prec^I_f \cup \mathcal{C}^f_{(b,a)}$ contains some oriented cycle Γ_{ba}. We may choose Γ_{ab} and Γ_{ba} in such a way that their lengths are minimal. Clearly Γ_{ab} must contain at least one oriented pair (a', b') of $\mathcal{C}^f_{(a,b)}$. It also must be of cardinality larger than 2 thanks to condition C_1. Let α and β be respectively the predecessor of a' and the successor of b' in Γ_{ab} ($\alpha \neq b'$ and $\beta \neq a'$). Because of the minimality of Γ_{ab}, $(\alpha, b') \notin \mathcal{C}^f_{(a,b)}$ and because its cardinality cannot be 2, $(a', \alpha) \notin \mathcal{C}^f_{(a,b)}$. The Triangle Lemma with $c = \alpha$ may be applied to provide $(\alpha, a)\ R^*_I\ (\alpha, a')$. The same reasoning applied

on β leads to the relation: $(b,\beta)\ R_I^*\ (b',\beta)$. So (a',b') can be replaced by (a,b) in Γ_{ab}. Suppose another arc $(a'',b'') \in \Gamma_{ab} \cap \mathcal{C}^f_{(a,b)}$. Then replacing it again by (a,b) implies another shorter oriented cycle, and a contradiction.

Γ_{ab} can be written $\Gamma_{ab} = (a,b) \cup \Gamma'_{ab}$, where Γ'_{ab} is a path from b to a made only with arcs of $\prec_f^I$. By proceeding the same way with Γ_{ba}, $\Gamma_{ba} = (b,a) \cup \Gamma'_{ba}$ where Γ'_{ba} is a path from a to b made only with arcs of $\prec_f^I$. But the concatenation of Γ'_{ab} and Γ'_{ba} makes a cycle in $(f, \prec_f^I)$, a contradiction . For sake of simplicity, notation $\prec_f^I$ is kept, even after adding such arcs of an equivalence class. We reiterate on the next oriented pair (a,b) until $\prec_f^I$ is a total order. ■

In the preemptive version of the problem, $I = \emptyset$, and $\forall a,b \in f, \mathcal{C}^f_{(a,b)} = \{(a,b)\}$. Thus, C_1 is always true, and C_2 becomes: *the antichain subgraph $(f, \prec_f)$ does not contain any oriented cycle.*

2.4 Incremental test of feasibility for Simplex pivotings

We now present an incremental test to check as simply as possible if, starting from some element of $\mathcal{S}_I$ (basis of $\mathcal{P}$), the next basis of $\mathcal{P}$ obtained by Simplex pivoting is still I-candidate.

Let f be a basis of $\mathcal{S}_I$, a_o the leaving variable, a_e the entering variable for some Simplex pivoting. Note first that thanks to Proposition 2.3, the removal of the antichain a_o of the I-candidate active antichain family f does not withdraw the I-candidate property. The following corollary may be applied on family $f \backslash a_o$ and a_e to test the feasibility of the next basis.

Corollary 2.8 (Incrementality corollary) *Let f be an I-candidate valid antichain family and $a_e \notin f$ be a valid antichain. Then the family $f' = f \cup a_e$ is I-candidate if and only if*

C'_1) *for all $a \in f, (a_e, a) \notin \mathcal{C}^{f'}_{(a,a_e)}$*

C'_2) *there is no oriented cycle containing a_e in the antichain subgraph $(f', \prec_{f'}^I)$.*

Proof. The part (*Only if*) is trivial thanks to the Characterization Theorem for f'. For the other part, condition C_1 for f and C'_1 ensures condition C_1 for f'. For condition C_2 of f', suppose there is an oriented cycle Γ in $(f', \prec_{f'}^I)$. Assuming Γ is of minimal length, it clearly includes an arc of a class of an arc containing a_e. As in the proof of the Characterization Theorem, we demonstrate that there exists an oriented cycle containing this arc itself, so containing vertex a_e. ■

For the preemptive version, C'_1) is always true, and C'_2) becomes: *there is no oriented cycle containing a_e in the antichain subgraph $(f', \prec'_f)$.*

3 Building of feasible schedules

3.1 A descent algorithm based on Simplex pivoting

We have already established that solving the problem is equivalent to minimizing Z on $\mathcal{S}_I$, that $\mathcal{S}_I$ is connected, and that an incremental test of the I-candidate property of an antichain family can be implemented. We can now present a descent algorithm based on the Simplex. Indeed, if we have an initial solution $\mathbf{z}_0$ of $\mathcal{S}_I$, the idea of this algorithm is to check if the switch of variables proposed by the Simplex pivoting will lead to a still I-candidate associated antichain family. If the incremental test proves negative, the switch does not occur, and we try another one proposed by the Simplex. Otherwise, the classical Simplex pivoting is computed. The process is iterated until no candidate variable verifies the test: this situation corresponds to a **local minimum** of the problem.

As the number of valid antichains is very large and a few of them are interesting for our problem, we set up a column generation module. Each dual variable π_i is associated to activity $i \in V$ and an entering column a_e has to verify $\sum_{i \in a_e} \pi_i > 1$ due to our formulation of $\mathcal{P}$. Hence, the subproblem is a multidimensional knapsack-stable problem: each activity i has a unit profit π_i and a multidimensional weight $r_{ik}, k \in \mathcal{R}$. The disjunctions (i, j) are defined by some resource constraint violations ($\exists k \in \mathcal{R}, r_{ik} + r_{jk} > R_k$) and the precedence relations between activities ($(i, j) \in E$ or $(j, i) \in E$), or can be strengthened through constraint propagation techniques (see [1], [4], [12]), at least when a better upper bound of the makespan is found. A preprocessing filtering step is added on this subproblem, ensuring that the generated column a_e does not lead to an oriented cycle in the graph induced by the current associated family (without computing and removing a_o) plus a_e. This preprocessing is removed to check if a local minimum is indeed reached.

The aim of the following subsections is to show how to get out of these local minima.

3.2 Reconstruction algorithm

We build here from the family f (of cardinality L) associated to a solution of the relaxed problem $\mathcal{P}$, an I-candidate sequence σ, allowing some antichain modifications. In [8], the authors generate their schedules by applying list algorithms with priority lists derived from the order of the jobs in the solution of a Lagrangian relaxation. The following method is inspired from Moukrim and Quilliot [10] (concerning the identical parallel machines scheduling problem).

The idea is to build step by step an ordered sequence of antichains. Let us consider an ordered subsequence $s = (a_1, \ldots, a_\lambda), \lambda < L$ of antichains of f, and an antichain $\alpha \in f \backslash s$. We define a_l as the first antichain of s such that $\alpha \prec a_l$. If a_l does not exist, then the sequence resulting from the concatenation of α at

the end of s is ordered. Otherwise, let us define the six following sub-antichains:

$$\begin{aligned}
MIN(a_l) &= \{i \in a_l \text{ such that } \exists j \in \alpha, (i,j) \in E\} \\
MIN(\alpha) &= \{i \in \alpha \text{ such that } \exists j \in a_l, (i,j) \in E\} \\
MAX(a_l) &= \{i \in a_l \text{ such that } \exists j \in \alpha, (j,i) \in E\} \\
MAX(\alpha) &= \{i \in \alpha \text{ such that } \exists j \in \{a_l, \ldots, a_q\}, (j,i) \in E\} \\
EQ(a_l) &= \{i \in a_l \backslash MIN(a_l) \backslash MAX(a_l)\} \\
EQ(\alpha) &= \{i \in \alpha \backslash MIN(\alpha) \backslash MAX(\alpha)\}.
\end{aligned}$$

Proposition 3.1 *Given an ordered sequence $s = (a_1, \ldots, a_l, \ldots, a_\lambda)$ and an antichain $\alpha \notin s$, if we state:*

$$a'_l = MIN(a_l) \cup MIN(\alpha) \cup EQ(a_l) \cup EQ(\alpha)$$

$$\alpha' = MAX(a_l) \cup MAX(\alpha) \cup EQ(a_l) \cup EQ(\alpha)$$

then

i) a'_l and α' are antichains

ii) the sequence $(a_1, \ldots, a'_l, \ldots, a_\lambda)$ is still ordered

iii) $\alpha' \not\prec a'_l$.

Proof. It is only a matter of applying the definitions, keeping in mind that s is ordered and that a couple of activities of an antichain cannot be linked by a precedence relation of E, which is transitive. ■

Note first that a'_l and α' are not necessarily valid, even if a_l and α are. This fact is discussed in section 3.4. Thanks to Proposition 3.1, for a given $\alpha \in f$ located just after the subsequence s, we proceed iteratively until the modified s, concatenated with the final α' is an ordered sequence. By induction on α (from the second to the last antichain of f in its initial arbitrary order), we provide from the initial family f an ordered sequence σ (of cardinality L) of non-valid antichains. This reconstruction algorithm has a complexity of $\mathcal{O}(n^2L^3)$ (with $L \leq n$), since the computing of our six sub-antichains for a_l and α (in position l' in f) is $\mathcal{O}((l'-l)n^2)$.

In Table 1, we give an example of reconstruction algorithm applied to the instance shown in Figure 1, $I = V$, and on the represented family f, solution of $\mathcal{P}$:

- Line 1: the family of active antichains in an arbitrary order (makespan 6.5)
- Line 2: the sequence after reconstruction (removing already present antichains)
- Line 3: the sequence after treatment of non-valid antichains (see 3.4)
- Line 4: the sequence after treatment of non-consecutiveness (see 3.4)
- Line 5: the sequence after loading in $\mathcal{P}$ (makespan 7: optimal non-preemptive makespan).

Table 1: A complete reconstruction for the instance of Figure 1, $I = V$

1	5,6	1,4	3,4	2	3,6	7	3,5		
2	1,4	2,5,6	3,5,6	3,6	3,5	7			
3	1,4	2	5,6	3,5	3,6	5,6	3,6	3,5	7
4	1,4	2	5	3,5	3,6	6	6	7	
5	1,4	2	3,5	3,6	7				

3.3 Perturbation algorithm

During the descent algorithm described in section 3.1, one may observe that some valid and Simplex candidate antichain a_e is not allowed to enter the basis f instead of an antichain a_o because of an oriented cycle in $(f \cup \{a_e\} \backslash \{a_o\}, \prec_f)$ containing a_e. The idea of this algorithm is to force the entering of a_e. Indeed, a_e may be present in some "good" solution, but some antichains of f forbid a_e to enter the basis through one Simplex pivoting.

First of all, we have to get the corresponding I-candidate sequence σ from the associated valid antichains f of the current basis solution. To do that, we only have to compute a topological order in the antichain subgraph $(f, \prec_f)$.

The algorithm consists in building a sequence σ' given by the following proposition (for any a_l, notations of section 3.2 are used):

Proposition 3.2 *Given an ordered sequence $\sigma = (a_1, \ldots, a_l, \ldots, a_L)$ and an antichain $\alpha \notin \sigma$. For any $a_l, l \leq L$, we define:*

$$a'_l = MIN(a_l) \cup MIN(\alpha) \cup EQ(a_l) \cup EQ(\alpha)$$

$$\alpha'_l = MAX(a_l) \cup MAX(\alpha).$$

Then a'_l and α'_l are antichains and the sequence $\sigma' = (a'_1, \ldots, a'_L, \alpha, \alpha'_1, \ldots, \alpha'_L)$ is ordered.

Proof. As for Proposition 3.1, it is a matter of applying the definitions, first to prove that $(a'_1, \ldots, a'_L)$ (resp. $(\alpha'_1, \ldots, \alpha'_L)$) is ordered, so that for any $l, m \in \{1, \ldots, L\}$ with $l < m$, $a'_m \not\prec a'_l$ (resp. $\alpha'_m \not\prec \alpha'_l$), and then that the insertion of α preserves this property. This is left to the reader. ■

We build this way an ordered sequence σ' (of cardinality $2L+1$) of non-valid antichains, containing the desired entering column $\alpha = a_e$. Note that a'_l and α'_l depend only on the original sequence σ and of α, hence the order of building is not important. This perturbation algorithm has a complexity $\mathcal{O}(n^2L^2)$.

In Table 2, an example of perturbation algorithm is given, for an element z of $\mathcal{S}_I$ (makespan 8), applied on instance of Figure 1, with $I = \emptyset$:

- Line 1: the ordered sequence of active antichains of z (makespan 8), and the rejected antichain
- Line 2: the sequence after perturbation (removing already present antichains)
- Line 3: the sequence after treatment of non-valid antichains (see 3.4)
- Line 4: the sequence after loading in $\mathcal{P}$ (makespan 7).

Table 2: A complete perturbation for the instance of Figure 1

1	4	1,5	2	3,6	7			3,4
2	4	1,4	2,4	3,4	3	3,5	3,6	7
3	4	1,4	2	3,4	3	3,5	3,6	7
4	1,4	2	3,4	3,5	3,6	7		

3.4 Dealing with non-valid antichains and non-I-consecutive sequences

In Propositions 3.1 and 3.2, the created antichains a'_l, α' or α'_l may not be valid because the resource requirements are not necessarily lower than or equal to the corresponding capacities. One may create in this case two valid antichains: for example, one would define $a'_l = MIN(a_l) \cup EQ(a_l)$ and $a''_l = MIN(\alpha) \cup EQ(\alpha)$ to be inserted instead of a_l. Another idea considers these antichains as representatives of a sub-family of valid antichains. Indeed, we can easily compute all maximal valid antichains included in a non-valid antichain. The advantage is that we can choose only a subset of them, without violating the ordered property of the whole created sequence called σ here. This choice is conditionned by the fact that we have to ensure the I-consecutiveness of this sequence.

Note that for the preemptive version, σ is already $\emptyset$-consecutive. For $I \neq \emptyset$, consider some non-I-consecutive ordered sequence σ. For each activity $i \in V$, we associate a list of index intervals of the antichains of σ_i in σ. The idea of this step is to enlarge these intervals for all activities i in I as much as possible, in order to finally have only one interval, and ensure the I-consecutiveness of the sequence, hence its I-candidate property.

For an activity $i \in I$ and an index interval of σ_i, this enlargement is processed by trying to add i to the left and right neighbour antichains of the interval, with respect to the resource constraints and the order property of σ. Some intervals of an activity $i \in I$ may be connected through this enlargement. If there is only one interval at the end of this process, then the antichains of i are consecutive. Otherwise, we select one of these intervals (heuristically the one with maximum cardinality) and simply remove i from the others.

3.5 Global method

The global method described in Algorithm 1 computes first an initial (non-preemptive) solution (line 2). In our implementation, this is done by a list algorithm with random priority. It also involves the descent (3.1), reconstruction (3.2) and perturbation (3.3) algorithms. Note that all I-candidate families corresponding to the reconstructed or perturbated sequences are also obviously complete, thus they can be reloaded in basis of $\mathcal{P}$. Note also that the column chosen to perturb the solution is the first Simplex candidate one which was rejected by the incremental test.

In this method, K_i, K_r and K_p are given parameters representing the number of initial solutions, the number of reconstructions for each initial solution

Algorithm 1 Global method

Require: K_i, K_r, K_p

1: $nbInit \leftarrow 1$
2: compute an initial randomized I-candidate family
3: $nbReconst \leftarrow 0$
4: $nbPerturb \leftarrow 0$
5: perform a descent with incremental tests to stay within $\mathcal{S}_I$
6: **if** $nbPerturb < K_p$ **then**
7: perturb the local minimum into another I-candidate family
8: $nbPerturb \leftarrow nbPerturb + 1$
9: goto step 5
10: **end if**
11: **if** $nbReconst < K_r$ **then**
12: solve $\mathcal{P}$ from the current solution
13: reconstruct from this solution an I-candidate family
14: $nbReconst \leftarrow nbReconst + 1$
15: goto step 4
16: **end if**
17: **if** $nbInit < K_i$ **then**
18: $nbInit \leftarrow nbInit + 1$
19: goto step 2
20: **end if**
21: give the best solution found so far

and the number of perturbations for each reconstruction, respectively.

This approach is of neighbourhood search kind (e.g. Taboo). The reconstruction algorithm (after the solving of $\mathcal{P}$) may be considered as a diversification stage, whereas perturbation is similar to the search of a better solution in the neighbourhood to escape a local minimum, except that a perturbation is equivalent to several moves in a classical neighbourhood search.

Finally note that each solving of $\mathcal{P}$ provides a lower bound of the optimal makespan, and that the algorithm is stopped as soon as the value of the best solution is equal to this lower bound.

4 Computational experiments

The tests of the algorithms presented in this paper have been performed on a Duron 1GHz through a C++ Code compiled by g++. Our algorithms have been tested on the 480 instances of the standard PSPLIB benchmarks [6] with $n = 30$ or 60. For each instance, K_i initial feasible solutions are computed: $INIT$ is the minimum makespan of them. OPT is the makespan of the optimal solution for each instance, if we have it. Note that our algorithm does not use any information from these hypothetical previously known optimal values. The following values are obtained for different values of K_i, K_r, K_p (see section 3.5):

- SOL makespan of the best solution obtained by the global algorithm (within time limit of 300 seconds)
- LB lower bound obtained by solving $\mathcal{P}$. This value is computed several

times during the algorithm (before each reconstruction), and the constraint propagation mentioned in section 3.1 can improve the lower bound obtained. The best one is stored in LB.

All gaps are given as percentages in average, with the maximum value between parentheses, and the number of times the two values are equal is provided too. Note that a descent process is stopped if a local minimum is found or if 10 000 antichains have been successively rejected by the incremental test.

We define two sets of hard instances called *hard*1 and *hard*2. Let us first recall one of the three parameters of *ProGen*, the standard project generator for the $PSPLIB$, called RS (Resource Strength). The capacity of resource k is defined as $R_k = R_k^{min} + round(RS.(R_k^{max} - R_k^{min}))$, where R_k^{min} is the maximum resource requirement of an activity, and R_k^{max} is the maximum requirement of activities simultaneously executed in an earliest start schedule. RS takes the following values $\{0.2, 0.5, 0.7, 1\}$. In case $RS = 1$, the earliest start schedule is feasible.

- An instance belongs to *hard*1 if $INIT \neq LB$. It excludes the instances with $RS = 1$.
- An instance belongs to *hard*2 if $RS \in \{0.2, 0.5\}$, for which it is well known that the gap between the best-known upper and lower bounds of the literature (for $n \geq 60$ and V-RCPSP) is maximum.

Finally, let us define the *reconstruction gap* as the average gap between the makespan of the reconstructed solution and of the value of the solution of $\mathcal{P}$ from which it was computed.

4.1 The preemptive case

4.1.1 A Branch and Bound algorithm for the preemptive version

A Branch and Bound algorithm in case $I = \emptyset$ was implemented to find the optimal preemptive solutions of our instances, or at least improve our best known preemptive lower and upper bounds. Remember that optimal values are not available from the PSPLIB itself. A node of the search tree consists in a set of forbidden disjunctions, that is to say the associated antichains of the solution of $\mathcal{P}$ for this node must not contain these couples of activities. If the graph induced by these associated antichains does not contain any oriented cycle, then this solution belongs to $\mathcal{S}_\emptyset$ (leaf node). Otherwise, we consider a set of disjunctions producing an oriented cycle of minimal length (chordless): for example if we detect the antichain oriented cycle $(\{1, 2, 7, 8\}, \{3, 4, 9\}, \{5, 6\})$ due to the following precedence relations between activities: $\{(2, 3), (4, 5), (6, 1)\} \in E$, then the corresponding forbidden disjunctions added to the three descendant nodes are respectively $\{1, 2\}$, $\{3, 4\}$ and $\{5, 6\}$. The set of forbidden disjunctions may be enlarged by graph techniques.

4.1.2 Numerical results

A complete study of our algorithm is performed for the 480 instances of 30 activities ($PSPLIB30$), for which the optimal makespan is always obtained by the Branch & Bound algorithm presented in section 4.1.1. These optimal

Table 3: numerical results for $PSPLIB30$ in preemptive case

K_i	K_r	K_p	inst.(#)	$INIT/LB$	#eq.	SOL/LB	#eq.	SOL/OPT	#eq.	CPU (s.)
1	0	0	all(480)	18.15(97.12)	139	4.69(52.63)	245	2.37(34.88)	291	0.5(5)
30	0	0	all(480)	11.33(48.78)	142	2.44(28.57)	272	0.23(6.11)	418	3.0(24)
1	20	0	all(480)	17.91(92.11)	139	2.32(31.75)	289	0.29(13.51)	415	2.4(26)
1	0	20	all(480)	18.15(97.12)	139	2.83(26.98)	267	0.59(16.22)	385	1.5(13)
1	20	5	all(480)	17.90(92.11)	139	2.15(26.98)	293	0.15(5.41)	442	9.9(300)
1	20	5	$hard1$(341)	25.19(92.11)	0	3.03(26.98)	154	0.21(5.41)	303	13.9(300)
1	20	5	$hard2$(240)	29.13(92.11)	6	4.05(26.98)	76	0.30(5.41)	202	17.4(300)

makespans are available on *http://www.isima.fr/damay*.

In Table 3, each line presents our results for one set of given parameters. As the best results are obtained when reconstruction and perturbation are combined, the last two lines detail the results for hard instances. The reconstruction gap is 42.1(96.6). Note that for these values of K_i, K_r, K_p, the gap between OPT and LB is 2.00(21.16) (294 equalities), and for information the number of Simplex pivotings, degeneracy pivotings, generated columns and rejected columns by the incremental test are, respectively 17(157), 7(85), 273(2 623) and 12 400(315 000).

The results of the algorithm with these final values of K_i, K_r, K_p on the instances of 60 activities ($PSPLIB60$), for which we do not have the optimal makespan, is given in Table 4. We can expect that our algorithm also produces feasible solutions of very good quality, since the gap between OPT and LB is much larger than the gap between SOL and OPT for $n = 30$.

The number of Simplex pivotings, degeneracy pivotings, generated columns and rejected columns are respectively 141(1326), 60(390), 931(15 342) and 59 500(428 000). The reconstruction gap is 56.22(110.94).

4.2 The non-preemptive case

Table 5 reports the results of our algorithms on $PSPLIB30$ in the non-preemptive case ($I = V$). The optimal values of these instances are available on the web page of PSPLIB, cf [6]. The perturbation algorithm is not well adapted to this case, so its results are not presented here. As a matter of fact, we observe experimentally that most of the rejected columns for this test are not allowed to enter the basis because of the non-consecutivity of the antichains containing the same activity, and not because of an oriented cycle in the antichain graph.

Only with reconstructions, the gap between OPT and LB is 3.72(30.16)

Table 4: numerical results for $PSPLIB60$ in preemptive case

K_i	K_r	K_p	inst.(#)	$INIT/LB$	#eq.	SOL/LB	#eq.	CPU (s.)
1	20	5	all(480)	19.45(104.62)	141	2.38(21.09)	334	103.4(300)
1	20	5	$hard1$(339)	27.54(104.62)	0	3.37(21.09)	193	146.2(300)
1	20	5	$hard2$(240)	33.85(104.62)	3	4.68(21.09)	100	193.5(300)

Table 5: numerical results for $PSPLIB30$ in non-preemptive case

K_i	K_r	inst.(#)	$INIT/LB$	#eq.	SOL/LB	#eq.	SOL/OPT	#eq.	CPU (s.)
1	0	all(480)	17.99(92.31)	139	15.21(92.11)	144	10.74(69.77)	158	1.0(16)
30	0	all(480)	11.19(48.78)	142	6.77(36.54)	207	2.75(22.06)	267	24.3(300)
1	30	all(480)	17.87(92.31)	139	5.76(31.75)	221	1.90(17.07)	291	67.47(300)
1	30	$hard1$(341)	25.16(92.31)	0	8.11(31.75)	82	2.67(17.07)	152	95.0(300)
1	30	$hard2$(240)	29.04(92.31)	6	10.22(31.75)	32	3.37(17.07)	77	93.6(300)

(243 equalities), and also larger than SOL/OPT. The reconstruction gap is 80.5(230.0). The number of Simplex pivotings, degeneracy pivotings, generated columns and rejected columns are, respectively 16(58), 6(33), 230(552) and 25 800(160 000).

These gaps can be explained by the very high number of columns rejected by the non-preemptive test. As a result, the current solution is very often a local minimum. The reconstruction algorithm, used for diversification, improves the result but not sufficiently to obtain a really small gap. An adaptation of the perturbation algorithm might be useful. However, as mentioned above, it is not easy to design.

5 Concluding remarks

This paper presents a characterization of the solution set for the preemptive and non-preemptive RCPSP, based on a linear programming model. An exploration algorithm of this solution space has been developed, in order to visit as many local minima as possible. The experimental results are very good in the preemptive case. The gap is much larger for non-preemptive activities. This is due to the difficulty for a given improving column to pass the feasibility test. Other combinatorial problems may be formulated as linear programs with additional conditions on the column generation. Our method should behave well for these, provided the feasibility test is not too demanding, as in the preemptive case.

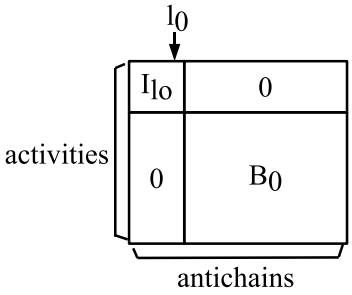
Appendix: Proof of the connexity theorem

Proof. This proof is written for the case when the constraints of $\mathcal{P}$ are equalities. The proof for inequalities is based on the same scheme, but it is simpler to identify basis solutions in the first case, especially without slack variables.
The proof is to show that there exists a sequence of basis solutions of $\mathcal{S}_I$, corresponding to a sequence of Simplex pivotings, leading from any solution of $\mathcal{S}_I$ to the solution corresponding to the family of singleton antichains, which evidently belongs to $\mathcal{S}_I$.
Suppose there exists a basis solution of $\mathcal{P}$ in $\mathcal{S}_I$ that does not verify this property. Then we choose such a solution $\mathbf{z}$ and the I-candidate sequence σ associated to its antichain family f in basis, such that:

- the number l_0 of singleton antichains in the beginning of σ is maximal;
- the sum $\sum_{a_l \in f} |a_l|$ is minimal, for l_0 fixed.

Suppose $l_0 \geq 1$ and $\exists l > l_0, \exists i_0 \leq l_0, i_0 \in a_l$. Let λ be the highest index of antichains in σ such that $i_0 \in a_\lambda$. Replacing antichain a_λ by $a_\lambda \backslash \{i_0\}$ in the family f, we obtain a family denoted by f_λ in which the n associated incidence vectors are still linearly independent, since the new column is a linear combination of the old one with an other ($\{i_0\}$). f_λ also provides a new sequence still trivially I-candidate (when $i_0 \in I$ or not). Moreover, the sum $\sum_{a_l \in f_\lambda} |a_l|$ is lower, so a contradiction on the minimality of the sum for f.

Hence, the basis incidence matrix corresponding to the family f has the following shape (with $l_0 \geq 0$):



Let ϵ be a strictly positive real number. We define the vector $\mathbf{z}^\epsilon$ by $\forall a_l \in f, \mathbf{z}^\epsilon_l = \mathbf{z}_l + \epsilon$, so that the time durations associated to the antichains of f are all non null, that is to say they are all active. This vector $\mathbf{z}^\epsilon$ is a solution of the linear problem in which the constraints are:

$$\sum_{a_l \in \mathcal{A}_i} z_l = p_i + |f_i|.\epsilon \tag{1}$$

Let i_1 be the smallest index of the activities such that $i_1 \in a_{l_0+1}$ $(i_1 > l_0)$. Let λ' be the highest index of the antichains in f such that $i_1 \in a_{\lambda'}$. Let us define the valid antichains a_μ and $a_{\mu'}$ such that $a_\mu = \{i_1\}$ and $a_{\mu'} = a_{\lambda'} \backslash i_1$, and the family $f' = f \cup \{f_\mu, f_{\mu'}\}$.

Let us denote by $\mathcal{P}^\epsilon$ the linear problem defined by the constraints 1 and the objective function max z_μ.

The solving of $\mathcal{P}^\epsilon$ by the Simplex, in the variable space reduced to those associated to f', provides a sequence S_ϵ of basis solutions linked by the Simplex pivoting neighbourhood operator, going from $\mathbf{z}^\epsilon$ to an optimal solution $\mathbf{z}^{\epsilon *}$ such that $z^{\epsilon *}_\mu > 0$, so for which a_μ is inside the basis. Furthermore, the insertions of a_μ between a_{l_0} and a_{l_0+1} and of $a_{\mu'}$ between $a_{\lambda'}$ and $a_{\lambda'+1}$ give trivially a sequence of antichains still I-candidate. Thanks to Proposition 2.3, all solutions of S_ϵ correspond to I-candidate families.

The number of basis solutions taking their antichains in f' is finite ($\frac{(n+1)(n+2)}{2}$), hence we can find a sequence $\epsilon_k, k \in \mathbf{N}$, tending toward 0 such that the sequences S_{ϵ_k} are identical. We deduce that there exists a sequence of basis solutions in $\mathcal{S}_I$ corresponding to a sequence of Simplex pivotings, going from $\mathbf{z}$ to a solution $\mathbf{z}^*$ such that a_μ is inside the basis. It implies a contradiction with the maximality of l_0.

■

References

[1] P. Baptiste and S. Demassey. Tight LP bounds for Resource Constrained Project Scheduling. *OR Spectrum*, 26:251–262, 2004.

[2] F. Bendali and A. Quilliot. Representation of ordered families of intervals, and applications. *RAIRO, Recherche Opérationnelle/Operations Research*, 31(1):73–101, 1997.

[3] P. Brucker and S. Knust. A linear programming and constraint propagation-based lower bound for the RCPSP. *European Journal of Operational Research*, 127:355–362, 2000.

[4] S. Demassey, C. Artigues, and P. Michelon. Constraint propagation based cutting planes : an application to the resource-constrained project scheduling problem. *INFORMS Journal on Computing*, 17(1):52–65, 2005.

[5] E. Demeulemeester and W. Herroelen. An efficient optimal solution procedure for the preemptive resource-constrained project scheduling problem. *European Journal of Operational Research*, 90:334–348, 1996.

[6] R. Kolisch and A. Sprecher. PSPLIB - A project scheduling library. *European Journal of Operational Research*, pages 205–216, 1996.

[7] A. Mingozzi, V. Maniezzo, S. Ricciardelli, and L. Bianco. An exact algorithm for the resource-constrained project scheduling based on a new mathematical formulation. *Management Science*, 44:714–729, 1998.

[8] R. H. Möhring, A. S. Schulz, F. Stork, and M. Uetz. Solving project scheduling problems by minimum cut computations. *INFORMS Management Science*, 49(3):330–350, 2003.

[9] A. Moukrim and A. Quilliot. A relation between multiprocessor scheduling and linear programming. *Order*, 14:269–278, 1998.

[10] A. Moukrim and A. Quilliot. Optimal preemptive scheduling on a fixed number of identical machines. *Operations Research Letters*, 33:143–150, 2005.

[11] C. Le Pape and Ph. Baptiste. Resource constraints for preemptive job-shop scheduling. *Constraints*, 3(4):263–287, 1998.

[12] A. Schoo, O. Thiele, P. Brucker, and S. Knust. A branch and bound algorithm for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 107:272–288, 1998.