RAIRO-Oper. Res. 52 (2018) 513–528
https://doi.org/10.1051/ro/2018031

RAIRO Operations Research
www.rairo-ro.org

BRANCH-AND-CUT-AND-PRICE ALGORITHMS FOR THE PREEMPTIVE RCPSP

Pierre Fouilhoux[1,*], A.Ridha Mahjoub[2], Alain Quilliot[3] and Hélène Toussaint[3]

Abstract. In this article, we address the preemptive Resource-Constrained Precedence Scheduling Problem. We propose two mixed integer formulations containing an exponential number of variables and inequalities. An antichain is a set of pairwise incomparable elements with respect to the precedence constraints. In the first formulation, the integer variables are associated with the antichains. For the second, the integer variables are limited to a particular subset of antichains. We propose two Branch-and-Cut-and-Price algorithms for each of these formulations. We introduce some valid inequalities in order to reinforce the second formulation. Finally, we give some computational results on instances of the PSPLIB and compare the formulations.

Mathematics Subject Classification. 90C11, 90C27, 90C08, 90C90.

Received June 30, 2017. Accepted April 27, 2018.

1. Introduction

Let $V = \{1, \ldots, n\}$ be a set of n activities that are associated with processing times $p_i, i \in V$. Let $G = (V, <)$ be a partially ordered set (poset) where $<$ is a precedence relation over the activities of V so that for every pair of activities $i, j \in V$ with $i < j$, activity i must be completed before activity j starts. We consider κ resources such that each resource $k \in \{1, \ldots, \kappa\}$ corresponds to a total availability $R_k \in I\!R$ and let $r_{ik} \leq R_k$ be the constant amount of resource k required by activity $i \in V$ at each time of its execution. We will refer at instance $P = (V, <, p, \kappa, R, r)$ where $p = (p_1, \ldots, p_n)$, $R = (R_1, \ldots, R_\kappa)$ and $r = (r_{ik}, i \in V, k \in \{1, \ldots, \kappa\})$ as a *project*. Given an activity $i \in V$, an *execution sequence* for activity $i \in V$ is a sequence of n_i time intervals $S_i = \left([s_i^1, e_i^1], \ldots, [s_i^{n_i}, e_i^{n_i}]\right)$ where $e_i^j < s_i^{j+1}$, $j \in \{1, \ldots, n_i - 1\}$, such that $\sum_{j=1}^{n_i}(e_i^j - s_i^j) = p_i$. A *schedule* of project P is a set $S = \{S_1, \ldots, S_n\}$ of execution sequences for the activities $i \in \{1, \ldots, n\}$ so that i) $e_i^{n_i} \leq s_j^1$ for every pair $i, j \in V$ with $i < j$ and ii) at any time, the total amount of consumed resource $k \in \{1, \ldots, \kappa\}$ does not exceed R_k. The *makespan* of a S is the total length of the schedule, *i.e.*, $\max_{i \in V}(e_i^{n_i}) - \min_{i \in V}(s_i^1)$. A schedule is

Keywords and phrases: Resource-constrained precedence scheduling problem, Preemptive case, Antichain, Branch-and-Cut-and-Price algorithm.

[1] Sorbonne Universités, Université Pierre et Marie Curie, Laboratoire LIP6, CNRS UMR 7606, 4 place Jussieu 75005 Paris, France.
[2] Université Paris-Dauphine, LAMSADE, CNRS UMR 7243, Place du Maréchal de Lattre de Tassigny, 75775 Paris CEDEX 16, France.
[3] Université Blaise Pascal Clermont II, Laboratoire LIMOS, CNRS UMR 6158, Complexe Scientifique des Cézeaux, 63117 Aubière Cedex, France.

* Corresponding author: pierre.fouilhoux@lip6.fr

called *non-preemptive* if $n_i = 1$ for every $i \in V$ and *preemptive* otherwise. The preemptive *Resource-Constrained Precedence Scheduling Problem* (RCPSP) consists in determining a preemptive schedule of minimum makespan. In this article, we assume that the limited resources are renewable, *i.e.*, their amounts are constant and available at each time of the project (*e.g.* manpower, electric power,...). For instance, such projects may arise in industrial applications like production planning or process scheduling in computer science.

The decision variant of the non-preemptive RCPSP (sometimes named *single-mode RCPSP*) is the problem of determining whether there exists a schedule of makespan smaller than a given deadline T. This problem is NP-hard in the strong sense [6]. Two surveys of this problem and its extensions can be found in [3, 10].

Several integer formulations of the non-preemptive RCPSP have been given in the literature, *e.g.*, [1, 11, 15, 17–19]. Most are based on a discretization of the time horizon. In particular, in [15] Mingozzi et al. introduce a 0–1 linear formulation for the non-preemptive RCPSP using time-indexed variables. They also propose several linear relaxations of their formulation that permit to compute lower bounds. Brucker and Knust [4] consider one of these relaxations and devise a destructive approach based algorithm using column generation and constraint programming. In [9], Hardin et al. describe strong valid inequalities for another time-indexed formulation. Baptiste and Demassey [2] develop a preprocessing step for this latter algorithm using constraint programming techniques and provide the best known lower bounds for the instances of the PSPLIB of Kolisch and Sprecher[12].

Given a poset, an *antichain* is a set of pairwise incomparable elements. This concept has been first introduced by Mingozzi et al. [15]. Using this they propose a formulation containing an exponential number of variables and develop a column generation algorithm for solving it. The linear relaxation of this formulation permits to produce a very good lower bound for the preemptive RCPSP, which is in turn a good lower bound for the non-preemptive RCPSP. In [5], Damay et al. point out a correspondence between the solutions of the RCPSP and particular subsets of antichains, called *feasible subsets*. Then they devise a heuristic column generation based algorithm for the RCPSP. Recently, in [16] Moukrim et al. propose a Branch-and-Bound algorithm for the preemptive RCPCP in which each subproblem is related to a particular subset of antichains and can be solved to optimality using the linear relaxation of [15]. Using this approach, they solve instances up to 30 activities coming from the PSPLIB.

The formulation of [15] was inspired from a similar column generation process proposed by Mehrotra and Trick [14] for the vertex coloring problem. Efficient extensions of the formulation have been discussed in [7, 8, 13]. Moreover in [7], Gualandi and Malucelli succeed to solve very large instances of the vertex coloring problem using a technique combining column generation and constraint programming.

In this article, we propose two mixed integer formulations (MIP) for the preemptive RCPSP. In the first one, the integer variables are associated with the antichains and are then in exponential number. For the second, the integer variables are limited to a particular set of antichains and are in $O(|V|^2)$. For both formulations, we propose a Branch-and-Cut-and-Price (BCP) algorithm. We also propose some valid inequalities in order to reinforce the second formulation. Finally, we present some computational results for instances from the PSPLIB and compare the two formulations.

The paper is organized as follows. In the following section we show that the preemptive RCPSP reduces to find a particular set of antichains. Using this, we introduce two MIPs for this problem in Section 3. In Section 4, we present two Branch-and-Cut-and-Price algorithms which are dedicated to these formulations. In Section 5, we propose some valid inequalities for the second formulation. Finally, in Section 6, we present our computational study. In the rest of this section, we give more definitions and notations.

The graphs we consider are finite, directed and without loops and multiple arcs. We denote a graph by $H = (N, B)$ where N is the *node set* and B the *arc set* of H. An arc $b \in B$ from node u to node v will be denoted as (u, v). A *path* is a sequence of nodes $(v_1, v_2, \ldots, v_k)$ such that B contains the arcs $(v_1, v_2), (v_2, v_3), \ldots, (v_{k-1}, v_k)$. If $(v_1, v_2, \ldots, v_k)$ is a path linking v_1 to v_k and arc $(v_k, v_1) \in E$, then the sequence $(v_1, v_2, \ldots, v_k,)$ is called a *circuit*. If $W \subseteq N$, $B(W)$ denotes the set of all arcs of H with both endnodes in W. The graph $(W, B(W))$ is the subgraph of H *induced* by W. An induced subgraph is called *acyclic* if it does not contain a circuit.

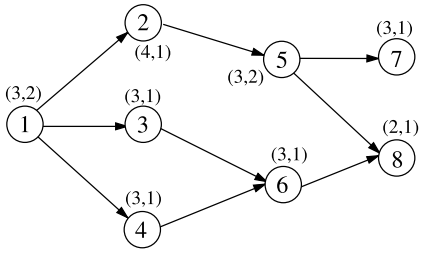

FIGURE 1. An instance P_1 of project with 1 resource and $R_1 = 2$.

2. An antichain model

In this section, we show that the preemptive RCPSP reduces to finding a particular subset of antichains of a poset. Such subsets can be seen as a generalization of the feasible subsets introduced by Mingozzi et al. in [15] and of the candidate families introduced in [5].

Given a project $P = (V, <, p, \kappa, R, r)$, a *R-antichain* a of P is an antichain of the poset $G = (V, <)$ which satisfies the resource constraints, *i.e.*, $\sum_{i \in a} r_{ik} \leq R_k$ for all $k \in \{1, \ldots, \kappa\}$.

Figure 1 illustrates a project P_1 of 8 activities and a single resource, called 1, with $R_1 = 2$. In the figure, project P_1 is fully given by the graph G_1: each activity $i \in V$ is represented by a circle containing its number; the pair (p_i, r_{i1}) of its processing time and its resource requirement; and the arcs represent the precedence relations between the activities. Note that in such a graph, for every two activities $i, j \in V$, we have $i < j$ if and only if there is a path from i to j. Remark that sets $\{3, 4\}$ and $\{2, 6\}$ are R_1-antichains, whereas $\{3, 5\}$ is an antichain which is not an R_1-antichain since $r_{31} + r_{51} = 3 > R_1$.

Let us consider $\mathscr{A}$ as the set of all the R-antichains of P. We define a precedence relation $\prec$ on $\mathscr{A}$ such that for $a, a' \in \mathscr{A}$, $a \prec a'$ if and only if there exists a pair of activities $i \in a$ and $j \in a'$ with $i < j$. For instance, for the project P_1 of Figure 1, remark that $\{3, 4\} \prec \{2, 6\}$. Given a subset of R-antichains $A \subseteq \mathscr{A}$, let us define the *antichain graph* as the directed graph $\Gamma(A) = (A, \mathcal{E}(A))$ where the arc set $\mathcal{E}(A)$ contains the arc (a, a'), $a, a' \in A$, if and only if $a \prec a'$.

Remark that an antichain graph can contain a circuit. For instance, in project P_1 of Figure 1, the R_1-antichains $\{2, 6\}$ and $\{3, 7\}$ induce a circuit. A subset of R-antichains $A \subseteq \mathscr{A}$ is said to be *acyclic* if the graph $\Gamma(A)$ is acylic.

Given a subset $A \subseteq \mathscr{A}$ of R-antichains and an activity i of V, we define A_i as the subset of A containing activity i. Let us consider the vector set $\mathcal{S}(A)$ introduced by Mingozzi et al. in [15]:

$$\mathcal{S}(A) = \left\{ z \in \mathbb{R}_+^{|A|} \text{ s.t. } \sum_{a \in A_i} z_a = p_i \text{ for all } i \in V \right\}.$$

The set $\mathcal{S}(A)$ is the solution set of a set of equalities containing an exponential number of variables: there is one variable z_a associated with every R-antichain a of $\mathscr{A}$. A subset of R-antichains $A \subseteq \mathscr{A}$ is said to be *time-full* if $S(A) \neq \emptyset$. Remark that, if $\mathcal{S}(A) \neq \emptyset$, a component z_a corresponds to an amount of time associated with the R-antichain a of A.

Figure 2a shows a schedule associated with the instance of Figure 1. Remark that activity 4 has an execution sequence of 2 time intervals. Figure 2b presents the same schedule divided into the 9 corresponding R_1-antichains. For instance, the R_1-antichain a_1 (resp. a_2) corresponds to an amount of time $z_{a_1} = 3$ (resp. $z_{a_2} = 2$). The two R_1-antichains containing activity 4 are a_2 and a_4 and their total amount of time is $p_4 = 3$. It can be seen that the set A_1 of all these R_1-antichains is time-full and that graph $\Gamma(A_1) = (A_1, \mathcal{E}(A_1))$ is acyclic.

We then can propose a reduction for the preemptive RCPSP.

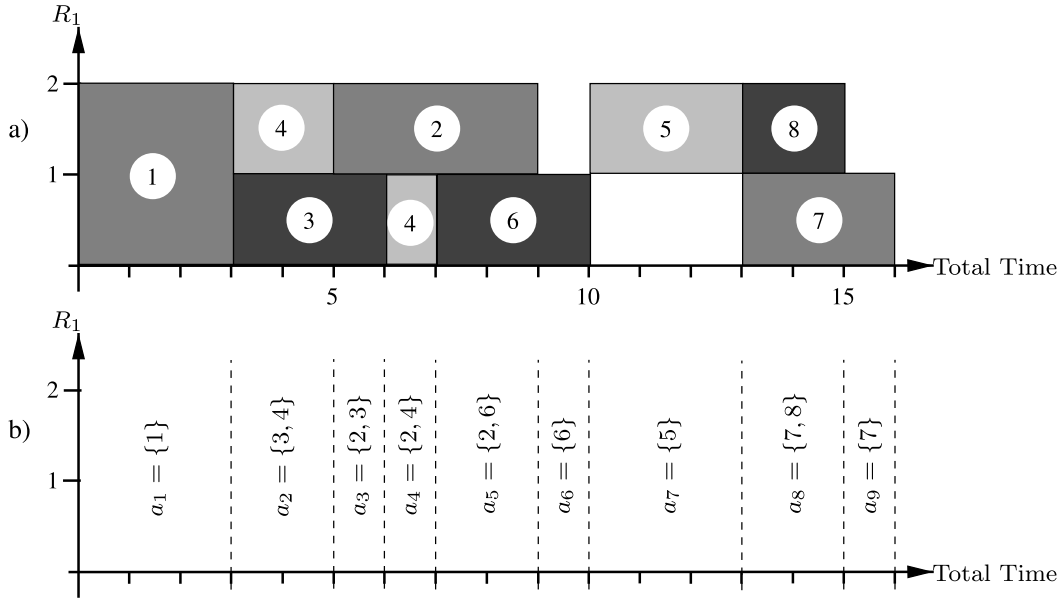

FIGURE 2. Gantt chart corresponding to the instance of Figure 1.

Theorem 2.1. *Given a project P, determining an optimal preemptive schedule of P is equivalent to determining an acyclic time-full subset A of $\mathscr{A}$ such that $\mathcal{S}(A) \neq \emptyset$ and $\min_{z \in \mathcal{S}(A)} \sum_{a \in A} z_a$ is minimum.*

Proof. First, let us consider a schedule $S = \{S_1, \ldots, S_n\}$ where each execution sequence S_i is divided into n_i time intervals, for each activity $i \in V$. Let us consider the set of the events on the schedule, that is to say the dates between the time 0 and the makespan T when a time interval of an execution sequence of an activity starts or ends. Let us denote $(d_1, \ldots, d_\tau)$ the sequence of the τ events of the schedule with $d_1 = 0$ and $d_\tau = T$. Given $t = 1, \ldots, \tau - 1$, let us consider the set a_t of activities that are in execution within the interval $[d_t, d_{t+1}]$. Since the activities of an antichain a_t, $t \in \{1, \ldots, \tau\}$, have been activated simultaneously, then $\sum_{i \in a_t} r_{ik} \leq R_k$ for every $k \in \{1, \ldots, \kappa\}$, and thus a_t is an R-antichain. Moreover, since in schedule S, an activity $j \in V$ begins after the end of every activity i of V with $i < j$, by construction, the subset $A = \cup_{t \in \{1, \ldots, \tau\}} a_t$ is acyclic. Finally, we can see that A is time-full. Indeed, by setting $z_{a_t} = d_{t+1} - d_t$, for every $t \in \{1, \ldots, \tau\}$, z is a solution of $\mathcal{S}(A)$ where $\sum_{a \in A} z_a$ is the makespan of the schedule S.

Conversely, let us consider an acyclic time-full subset A of p R-antichains with $\mathcal{S}(A) \neq \emptyset$. Using a deep-first search on graph $\Gamma(A)$ it can then be obtained a topological order $\sigma = (a_1, \ldots, a_p)$ over the R-antichains of A. Let z^* be an optimal solution of the linear program $\min_{z \in \mathcal{S}(A)} \sum_{a \in A} z_a$. Note that, since A is time-full, this program is feasible. Given $l \in \{1, \ldots, p\}$ and an activity $i \in a_l$, we set $s_i^l = \sum_{\lambda=1}^{l-1} z^*_{a_\lambda}$ and $e_i^l = \sum_{\lambda=1}^{l} z^*_{a_\lambda}$. By construction, for a given $i \in V$, the sequence $S_i = ([s_i^1, e_i^1], \ldots, [s_i^p, e_i^p])$ is an execution sequence of activity i. Indeed, since A is time-full, $\sum_{\{l \mid a_l \in \mathscr{A}_i\}} e_i^l - s_i^l = p_i$. Moreover, since σ is a topological order and A is composed of R-antichain, $S = (S_1, \ldots, S_n)$ is a preemptive schedule. Finally, we can remark that the makespan of S is exactly $\sum_{a \in A} z^*_a$. □

3. MIP FORMULATIONS

In this section, we present two MIP formulations for the preemptive RCPSP based on the reduction introduced in the previous section.

3.1. Antichain formulation

We first introduce a MIP formulation directly inspired from Theorem 2.1. Given a project P, we associate a 0–1 variable x_a and a continuous variable z_a to every R-antichain a of $\mathscr{A}$. Vector x will be an incidence vector of a subset A of $\mathscr{A}$ and z_a an amount of time associated with $a \in A$. Let $P_a = \min\{p_i \mid i \in a\}$ for every $a \in \mathscr{A}$. Let us consider the following mixed linear program $\mathcal{P}_1$.

$$
\begin{aligned}
Min \sum_{a\in\mathscr{A}} z_a & & & \\
\sum_{a\in\mathscr{A}_i} z_a = p_i & & \text{for all } i \in V, & \quad (3.1)\\
z_a \le P_a x_a & & \text{for all } a \in \mathscr{A}, & \quad (3.2)\\
\sum_{a\in C} x_a \le |C| - 1 & & \text{for all circuit } C \text{ in } \Gamma(\mathscr{A}), & \quad (3.3)\\
x_a \in \{0,1\} & & \text{for all } a \in \mathscr{A}, & \\
z_a \ge 0 & & \text{for all } a \in \mathscr{A}. &
\end{aligned}
$$

Given a solution (z, x) of $\mathcal{P}_1$, let us consider the set $A = \{a \in \mathscr{A} \mid x_a = 1\}$. Constraints (3.1), called *time-full inequalities*, ensure that A is time-full and constraints (3.3), called *circuit antichain inequalities*, ensure that A is acyclic. Constraints (3.2) force z_a to be equal to 0 whenever the R-antichain $a \in \mathscr{A}$ is not considered in the solution, *i.e.*, $x_a = 0$. Consequently, A is an acyclic time-full subset of $\mathscr{A}$ so that $\mathscr{A} \neq \emptyset$. Since, on the other hand, every incidence vector of an acyclic time-full subset of $\mathscr{A}$ corresponds to a solution of $\mathcal{P}_1$, then by Theorem 2.1, $\mathcal{P}_1$ is equivalent to the preemptive RCPSP.

3.2. Two-indices formulation

In this section, we will give another reduction for the preemptive RCPSP. Given a project P, let us consider the set $\mathcal{F}$ of R-antichains containing exactly two activities. Given a subset $A \subseteq \mathscr{A}$, we then consider the set

$$f(A) = \{\{i, j\} \in F \mid \exists\, a \in A \text{ with } \{i, j\} \subseteq a\}$$

and the graph $\Gamma^2(A) = (f(A), \mathcal{E}(f(A)))$ which is a subgraph of $\Gamma(\mathscr{A})$. We then can give the following result.

Theorem 3.1. *Let $A \subseteq \mathscr{A}$ be a subset of R-antichains. $\Gamma(A)$ is acyclic if and only if $\Gamma^2(A)$ is acyclic.*

Proof. Let us first consider a set $A \subseteq \mathscr{A}$ such that $\Gamma(A)$ is acyclic and assume that $\Gamma^2(A)$ contains a circuit $(\{i_1, j_1\}, \ldots, \{i_k, j_k\})$ with $k \ge 2$, such that $i_l < j_{l+1}$ for $l = 1, \ldots, k-1$ and $i_k < j_1$. By construction of $f(A)$, there exists $a_k \in A$ such that $\{i_k, j_k\} \subseteq a_k$ for every $k \in \{1, \ldots, k\}$. Then $(a_1, \ldots, a_k)$ is such that $a_l \prec a_{l+1}$ for $l = 1, \ldots, k-1$ and $a_k \prec a_1$. Consequently $(a_1, \ldots, a_k)$ forms a circuit of $\Gamma(A)$, a contradiction.

Let us now suppose that $\Gamma^2(A)$ is acyclic and $\Gamma(A)$ contains a circuit $(a_1, \ldots, a_k)$ with $k \ge 2$. By definition of $\Gamma(A)$, for every $l \in \{1, \ldots, k-1\}$ (resp. $l = k$), there exist two activities $i_l \in a_l$ and $j_{l+1} \in a_{l+1}$ (resp. $i_k \in a_k$ and $j_1 \in a_1$) such that $i_l < j_{l+1}$ (resp. $i_k < j_1$). Let us then consider the sequence $s = (\{i_1, j_1\}, \ldots, \{i_k, j_k\})$. Let us first suppose that there exists $l_0 \in \{1, \ldots, k\}$ such that $i_{l_0} = j_{l_0}$. Since $i_{l_0-1} < j_{l_0} = i_{l_0} < j_{l_0+1}$ (indices are taken modulo k), $i_{l_0-1} < j_{l_0+1}$. Let us then remove $\{i_{l_0}, j_{l_0}\}$ from list s. Using iteratively this argument, we then obtain a subsequence $s' = (\{i'_1, j'_1\}, \ldots, \{i'_{k'}, j'_{k'}\})$ of s such that $i'_l \neq j'_l$ for every $l \in \{1, \ldots, k'\}$. Moreover observe that $k' \ge 2$. Consequently, $\{i'_l, j'_l\} \in f(A)$ for $l = 1, \ldots, k'$, and thus s' is a circuit of $\Gamma^2(f(A))$, which is a contradiction. □

From Theorem 3.1, we can propose a second MIP formulation, called Two-indices formulation, for the preemptive RCPSP. With every element $\{i, j\}$ of $\mathcal{F}$, we associate a 0–1 variable x_{ij}. We also still consider continuous

variables z_a associated with every R-antichain a of $\mathscr{A}$. Let $P_{ij} = \min(p_i, p_j)$ for every $\{i,j\} \in \mathcal{F}$. Let us consider the following MIP formulation $\mathcal{P}_2$.

$$
\begin{aligned}
Min \sum_{a \in \mathscr{A}} z_a, & & & \\
\sum_{a \in \mathscr{A}_i} z_a = p_i & & \text{for all } i \in V, & \quad (3.1)\\
\sum_{a \in \mathscr{A} \mid \{i,j\} \subseteq a} z_a \leq P_{ij} x_{ij} & & \text{for all } \{i,j\} \in \mathcal{F}, & \quad (3.4)\\
\sum_{\{i,j\} \in C} x_{ij} \leq |C| - 1 & & \text{for all circuit } C \text{ of } \Gamma^2(\mathscr{A}), & \quad (3.5)\\
x_{ij} \in \{0,1\} & & \text{for all } \{i,j\} \in \mathcal{F}, & \\
z_a \geq 0 & & \text{for all } a \in \mathscr{A}. &
\end{aligned}
$$

Given a solution (z,x) of $\mathcal{P}_2$, let us consider the set $A = \{a \in \mathscr{A} \mid z(a) > 0\}$. Clearly, time-full inequalities (3.1) ensure that A is time-full. By inequalities (3.4), for every pair $\{i,j\}$ of $\mathcal{F}$ that are contained in an R-antichain of A, we have $x_{ij} = 1$. Note that, for a given pair $\{i,j\} \in \mathscr{F}$, the corresponding inequality (3.4) is valid from the two inequalities (3.1) for i and j. Moreover, constraints (3.5), called *circuit inequalities*, ensure that A is acyclic. From Theorem 3.1, since every acyclic time-full subset A of R-antichains correspond to a solution of $\mathcal{P}_2$, then program $\mathcal{P}_2$ is equivalent to the preemptive RCPSP.

We can remark that since inequalities (3.4) are associated with pairs of activities, they will be easier to handle within a column generation process than inequalities (3.2) of the previous antichain formulation.

4. Branch-and-Cut-and-Price algorithms

In this section we devise two BCP algorithms for the two MIP introduced in the previous section.

4.1. Branching rules and pricing phase for the antichain formulation

In this section, we describe the branching rules and the pricing phase for the antichain formulation. For this, we first present the branching rules of the algorithm.

4.1.1. Branching rules

Classical branching on variables $x_a, a \in \mathscr{A}$, will not be efficient in the framework of this BCP algorithm. Indeed, a better strategy would be to determine a partition of the solution set into two subproblems of similar size. We then use an idea close to the well-known Ryan and Foster rule: either two activities are never activated simultaneously or they are (partially) activated simultaneously. This can be done as follows: for two given activities $i, j \in V$ such that neither $i < j$ nor $j < i$, we construct two subproblems so that:

- Rule 0: either there is no R-antichain containing both activities i and j, *i.e.*, $x_a = 0$ for all $a \in \mathscr{A}$ such that $i, j \in a$,
- Rule 1: there exists at least one R-antichain containing both i and j. This is equivalent to adding the following inequality

$$\sum_{a \in \mathscr{A} \mid \{i,j\} \subseteq a} x_a \geq 1. \quad (4.1)$$

For a given node of the branching tree, we will denote by L_0 (resp. L_1) the set of pairs i, j for which rule 0 (resp. rule 1) has been applied. Given a subset $A \subseteq \mathscr{A}$ of R-antichains, let us then consider the set

$$A^{L_0} = \{a \in A \mid \{i,j\} \not\subseteq a \text{ for all } \{i,j\} \in L_0\}.$$

Therefore each node of the branching tree is characterized by a program $\mathcal{P}_1^{L_0,L_1}$ obtained from $\mathcal{P}_1$ by removing variables x_a, z_a with $a \in \mathscr{A} \setminus \mathscr{A}^{L_0}$ and adding the inequalities (4.1) for every $i, j \in L_1$.

The choice of a pair of activities for these branching rules can be performed by determining a pair i, j so that $\sum_{a \in \mathscr{A} \mid \{i,j\} \subseteq a} x_a$ is close to 0.5.

4.1.2. Pricing problem

At each node of the branching tree induced by the branching rules, we need to solve problem $\mathcal{P}_1^{L_0,L_1}$. This will be performed by a Price-and-Cut algorithm. The pricing phase of this Price-and-Cut algorithm consists in solving $\mathcal{P}_1^{L_0,L_1}$ where only a subset K of circuits of $\Gamma(\mathscr{A}^{L_0})$ is considered.

In order to initialize a pricing algorithm to solve $\mathcal{P}_1^{L_0,L_1}$, we consider a subset $A \subseteq \mathscr{A}$ of R-antichains such that $K \in \Gamma(A^{L0})$. Moreover, we will suppose that A contains all the R-antichains limited to only one activity $\{i\}, i \in V$, this implies that every linear program involved in the pricing phase will always contain a solution. Let us then consider the following linear program $\widetilde{\mathcal{P}}_1^{L_0,L_1}(A)$

$$\begin{aligned}
Min \sum_{a \in A^{L_0}} z_a & & & \\
\sum_{a \in A_i^{L_0}} z_a = p_i & & \text{for all } i \in V, & \quad (3.1)\\
z_a \le P_a x_a & & \text{for all } a \in A^{L_0}, & \quad (3.2)\\
\sum_{a \in C} x_a \le |C| - 1 & & \text{for all circuit } C \in K, & \quad (3.3)\\
\sum_{a \in A^{L_0} \mid \{i,j\} \subseteq a} x_a \ge 1 & & \text{for all } \{i,j\} \in L_1, & \quad (4.1)\\
x_a \le 1 & & \text{for all } a \in A^{L_0}, & \quad (4.2)\\
x_a \ge 0 & & \text{for all } a \in A^{L_0}, & \\
z_a \ge 0 & & \text{for all } a \in A^{L_0}. &
\end{aligned}$$

Let (z^*, x^*) be an optimal solution of $\widetilde{\mathcal{P}}_1^{L_0,L_1}(A)$. Let (z^0, x^0) be the vector such that $z_a^0 = z_a^*$, $x_a^0 = x_a^*$ if $a \in A^{L_0}$ and $z_a^0 = 0$, $x_a^0 = 0$ if $a \in \mathscr{A}^{L_0} \setminus A^{L_0}$. We now can address the pricing subproblem associated with $\widetilde{\mathcal{P}}_1^{L_0,L_1}(A)$.

Theorem 4.1. *Let λ_i^*, $i \in V$ be the dual optimal values of constraints (3.1) of $\widetilde{\mathcal{P}}_1^{L_0,L_1}(A)$. Then (z^0, x^0) is optimal for the linear program $\widetilde{P}_1^{L_0,L_1}(\mathscr{A})$ if $\sum_{i \in a} \lambda_i^* \le 1$ for every $a \in \mathscr{A}^{L_0} \setminus A^{L_0}$.*

Proof. Let us consider the linear problem $\widetilde{\mathcal{P}}_1'^{L_0,L_1}(A)$ obtained from $\widetilde{\mathcal{P}}_1^{L_0,L_1}(A)$ by adding inequalities $x_a \ge 0$ and $x_a \le 1$, for all $a \in \mathscr{A}^{L_0} \setminus A^{L_0}$; and replacing inequalities (4.1) by inequalities

$$\sum_{a \in \mathscr{A}^{L_0} \mid \{i,j\} \subseteq a} x_a \ge 1 \quad \text{for all } \{i,j\} \in L_1. \qquad (4.3)$$

Note that inequalities (4.1) and (4.3) differ by the fact that the sum in (4.1) is restricted to A^{L_0}. Let us then define vector x^1 as $x_a^1 = x_a^*$ if $a \in A^{L_0}$ and $x_a^1 = 1$ if $a \in \mathscr{A}^{L_0} \setminus A^{L_0}$. Remark that (z^0, x^1) is a solution of

$\widetilde{\mathcal{P}}_1^{'L_0,L_1}(A)$. Moreover, if (z^0, x^1) is an optimal solution of $\widetilde{P}_1^{L_0,L_1}(\mathscr{A})$, then (z^0, x^0) will be also optimal since they correspond to the same objective value. We will now show that (z^0, x^1) is an optimal solution of $\widetilde{P}_1^{L_0,L_1}(\mathscr{A})$.

Let $\lambda_i, i \in V$ (resp. $\mu_a, a \in A^{L_0}$; $\sigma_C, C \in K$; $\rho_{i,j}, \{i,j\} \in L_1$; $\nu_a, a \in \mathscr{A}^{L_0}$) be the dual variables of $\widetilde{\mathcal{P}}_1^{'L_0,L_1}(A)$ corresponding to the constraints (3.1) (resp. (3.2), (3.3), (4.3), (4.2)). The constraints of the dual of $\widetilde{\mathcal{P}}_1^{L_0,L_1}(\mathscr{A})$ are

$$\sum_{i \in a} \lambda_i + \mu_a \leq 1 \qquad \text{for all } a \in \mathscr{A}^{L_0} \tag{4.4}$$

$$-P_a \mu_a + \sum_{C \in K \mid a \in C} \sigma_C + \sum_{\{i,j\} \in L_1 \mid \{i,j\} \in a} \rho_{ij} + \nu_a \leq 0 \qquad \text{for all } a \in \mathscr{A}^{L_0} \tag{4.5}$$

with $\lambda_i \in I\!R, \forall i \in V$; $\mu_a \leq 0, \forall a \in \mathscr{A}^{L_0}$, $\sigma_C \leq 0, \forall C \in K$, $\rho_{ij} \geq 0, \forall \{i,j\} \in L_1$; and $\nu_a \leq 0, \forall a \in \mathscr{A}^{L_0}$.

Let $(\mu^*, \sigma^*, \rho^*, \nu^*)$ be an optimal solution of the dual of $\widetilde{\mathcal{P}}_1^{'L_0,L_1}(A)$. Let us consider the vector μ^1 given by $\mu_a^1 = \mu_a^*$ if $a \in A^{L_0}$ and $\mu_a^1 = 0$ if $a \in \mathscr{A}^{L_0} \setminus A^{L_0}$. We will first show that $(\lambda^*, \mu^1, \sigma^*, \rho^*, \nu^*)$ is a solution of the dual of $\widetilde{\mathcal{P}}_1^{L_0,L_1}(\mathscr{A})$. By construction, inequalities (4.4) and (4.5) are satisfied for every $a \in A^{L_0}$. Let us now consider $a \in \mathscr{A}^{L_0} \setminus A^{L_0}$. Since, by hypothesis, $\sum_{i \in a} \lambda_i^* \leq 1$, the inequality (4.4) corresponding to a is then satisfied. Note that the inequality (4.5) of the dual of $\widetilde{\mathcal{P}}_1^{'L_0,L_1}(A)$ corresponding to a is $\sum_{\{i,j\} \in L_1 \mid \{i,j\} \in a} \rho_{ij} + \nu_a \leq 0$. Moreover, since $\mu_a \geq 0$ and since any circuit $C \in K$ does not contain the R-antichain a, then the inequality (4.5) of the dual of $\widetilde{\mathcal{P}}_1^{L_0,L_1}(\mathscr{A})$ is satisfied.
Since the objective values of $\widetilde{P}_1^{L_0,L_1}(\mathscr{A})$ and its dual are the same for (z^0, x^1) and for $(\lambda^*, \mu^1, \sigma^*, \rho^*, \nu^*)$, by weak duality, (z^0, x^1) is an optimal solution of $\widetilde{P}_1^{L_0,L_1}(\mathscr{A})$. □

4.1.3. Pricing phase for $\mathcal{P}_1$

From Theorem 4.1, the pricing problem for formulation $\widetilde{\mathcal{P}}_1^{L_0,L_1}(A)$ is, given the dual optimal value λ^* of inequalities (3.1), to test whether every R-antichain $a \in \mathscr{A}^{L_0} \setminus A^{L_0}$ satisfies $\sum_{i \in a} \lambda_i^* \leq 1$, and, if not, to find an R-antichain $a \in \mathscr{A}^{L_0} \setminus A^{L_0}$ with $\sum_{i \in a} \lambda_i^* > 1$. This pricing problem then reduces to find a set of activities inducing an R-antichain $\mathscr{A}^{L_0} \setminus A^{L_0}$ so that $\sum_{i \in a} \lambda_i^*$ is minimum. This problem is clearly equivalent to the following program where y_i is a 0-1 variable associated with activity i, for every $i \in V$.

$$\begin{aligned}
\text{Max} \sum_{i \in V} \lambda_i y_i & & \\
\sum_{i \in V} r_{ik} y_i \leq R_k & & \text{for all } k \in \{1, \ldots, \kappa\}, \\
y_i + y_j \leq 1 & & \text{for all } i, j \in V \times V \text{ with } i < j, \\
y_i + y_j \leq 1 & & \text{for all } (i,j) \in L_0, \\
\{i \in V \mid y_i = 1\} \notin A^{L_0}, & & \\
y_i \in \{0,1\} & & \text{for all } i \in V.
\end{aligned} \tag{4.6}$$

Given an optimal solution y^* of this program, constraint (4.6) forces the R-antichain $a = \{i \in V \mid y_i^* = 1\}$ to be in $\mathscr{A}^{L_0} \setminus A^{L_0}$. Indeed, the reduced cost of a variable z_a corresponding to an R-antichain $a \in A^{L_0}$ is different from the one of z_a if $a \in \mathscr{A}^{L_0} \setminus A^{L_0}$ due to the dual cost of inequalities (3.2).

Since constraint (4.6) can easily be taken into account during a Branch-and-Bound method, this program can be solved using a MIP solver. Moreover, the linear inequalities of this program arise as valid inequalities in the knapsack and stable set problems. This may explain the fact that MIP solvers efficiently solve this program.

4.2. Branching rules and pricing phase for the two-index formulation

We now present the branching rules and the pricing algorithm for the two-index formulation $\mathcal{P}_2$.

4.2.1. Branching rules

For formulation $\mathcal{P}_2$, the chosen branching rule is to fix a variable use x_{ij} to 0 or 1. In the particular case of $\mathcal{P}_2$, this classical branching rule exactly corresponds to the more complicated rule we used for $\mathcal{P}_1$ in the previous section.

A subproblem of $\mathcal{P}_2$, using the classical branching rule, is then defined by a set L_0 of pairs (i,j) of activities such that $x_{ij}=0$ and a set L_1 such that $x_{ij}=1$. A MIP program $\mathcal{P}_2^{L_0,L_1}$ equivalent to this subproblem is then obtained from $\mathcal{P}_2$ by removing variables x_{ij} with $(i,j)\in L_0$ together with the corresponding inequalities (3.4) as well as every variable z_a such that $\{i,j\}\in a$; and to set to 1 every variable x_{ij} with $(i,j)\in L_1$.

4.2.2. Pricing problem

As for the Antichain formulation, program $\mathcal{P}_2^{L_0,L_1}$ will be solved by a Price-and-Cut algorithm. The pricing phase of this algorithm phase consists in solving $\mathcal{P}_2^{L_0,L_1}$ where only a subset K of circuits of $\Gamma(\mathscr{A})$ is considered. This pricing phase starts with a given R-antichain subset $A\subseteq\mathscr{A}$ such that $K\in\Gamma(A^{L_0})$ and containing an acyclic time-full subset. Let us then consider the following linear problem $\widetilde{\mathcal{P}}_2^{L_0,L_1}(A)$

$$
\begin{aligned}
&Min \sum_{a\in A} z_a, \\
&\sum_{a\in A_i^{L_0}} z_a = p_i && \text{for all } i\in V, && (3.1)\\
&\sum_{a\in A^{L_0}\ |\ \{i,j\}\subseteq a} z_a \le P_{ij}x_{ij} && \text{for all } \{i,j\}\in\mathcal{F}\setminus L_0, && (3.4)\\
&\sum_{\{i,j\}\in C} x_{ij} \le |C|-1 && \text{for all circuit } C\in K, && (3.5)\\
&x_{ij}=1 && \text{for all } \{i,j\}\in L_1,\\
&x_{ij}\in\{0,1\} && \text{for all } \{i,j\}\in\mathcal{F}\setminus L_0,\\
&z_a\ge 0 && \text{for all } a\in A^{L_0}.
\end{aligned}
$$

Let (z^*,x^*) be the optimal solution of $\widetilde{\mathcal{P}}_2^{L_0,L_1}(A)$. Let us consider the vector z^0 obtained as $z^0_a=z^*_a$ if $a\in A^{L_0}$ and $z^0_a=0$ if $a\in\mathscr{A}^{L_0}\setminus A^{L_0}$. Let λ^*_i, $i\in V$ (resp. μ_{ij}, $(i,j)\in\mathcal{F}\setminus L_0$), be the dual optimal values of constraints (3.1) (resp. (3.4)) of $\widetilde{\mathcal{P}}_2^{L_0,L_1}(A)$. It can be easily seen that the reduced cost of variable z_a, $a\in\mathscr{A}^{L_0}$, is

$$\rho_a = 1-\sum_{i\in a}\lambda_i+\sum_{\{i,j\}\subseteq a}\mu_{ij}.$$

Consequently (z^0,x^*) is optimal for the linear program $\widetilde{\mathcal{P}}_2^{L_0,L_1}(\mathscr{A})$ if $\rho_a>0$ for every R-antichain a of $\mathscr{A}^{L_0}$.

4.2.3. Pricing phase for $\mathcal{P}_2$

The pricing problem for this formulation reduces to find an R-antichain with minimum reduced cost. Let y_i be a 0-1 variable associated with activity $i\in V$ and t_{ij} a 0-1 variable associated with every $(i,j)\in\mathcal{F}\setminus L_0$. Then

the pricing subproblem is equivalent to the following program that can be easily solved using a MIP solver.

$$
\begin{aligned}
\text{Max } & \sum_{i\in V}\lambda_i y_i - \sum_{(i,j)\in\mathcal{F}}\mu_{ij}t_{ij} & \\
& \sum_{i\in V} r_{ik}y_i \le R_k && \text{for } k\in\{1,\dots,\kappa\},\\
& y_i + y_j \le 1 && \text{for all } i,j\in V\times V \text{ with } i<j,\\
& y_i + y_j \le 2t_{ij} && \text{for all } (i,j)\in\mathcal{F}\setminus L_0,\\
& y_i + y_j \le 1 && \text{for all } (i,j)\in L_0,\\
& t_{ij} \le y_i && \text{for all } (i,j)\in\mathcal{F}\setminus L_0,\\
& t_{ij} \le y_j && \text{for all } (i,j)\in\mathcal{F}\setminus L_0,\\
& y_i \in\{0,1\} && \text{for all } i\in V,\\
& t_{ij} \in\{0,1\} && \text{for all } (i,j)\in\mathcal{F}\setminus L_0.
\end{aligned}
$$

4.3. Price-and-Cut for the antichain and the two-index formulation

In this section, we precise how the whole Price-and-Cut algorithms are organized.

4.3.1. Circuit inequalities separation algorithm

We first present a separation algorithm for the circuit inequalities (3.3) and (3.5) of the two formulations. Let us consider a directed graph $H=(N,B)$ and a weight $x^*_u\in[0,1]$ associated with every node u of N. For a given set $A\subseteq\mathscr{A}$, graph H can be either $\Gamma(A)$ or $\Gamma^2(A)$. The separation problem for inequalities (3.3) and (3.5) can be then seen as deciding whether x^* satisfies the inequalities $\sum_{u\in C}x^*_u\le|C|-1$ for every circuit of H, and, if not, to find an inequality that is violated by x^*.

This separation problem can be easily performed in $|O(|N|t(N,B))$ where $t(N,B)$ is the complexity of a shortest path arborescence algorithm on graph H. Let $w_{uv}=1-\frac{x^*_u+x^*_v}{2}$ be a weight associated with arc (u,v) for every $(u,v)\in B$. For every node $u\in N$, we compute a shortest path arborescence rooted in u. For every node v such that $(v,u)\in B$, we then consider the circuit C obtained by concatenating the path from u to v in the arborescence plus arc (v,u). Note that if $w(C)<1$, then $\sum_{u\in C}x^*_u>|C|-1$. Moreover if for every node u,v the corresponding circuit C is so that $w(C)\ge1$, then x^* satisfies all the circuit inequalities.

4.3.2. Price-and-Cut algorithm

Given a subproblem created by the branching rules L_0 and L_1, the linear relaxation of the subproblem is then solved using a Price-and-Cut algorithm as follows. The sets A and K are initialized so that A contains a time-full acyclic subset and K is a set of circuits involving R-antichains of A.

(1) Consider linear program $\widetilde{\mathcal{P}}_i^{L_0,L_1}(A)$ $(i=1,2)$
(2) Using iteratively the pricing phase, find a set A' of new columns so that there is no column with nonnegative reduced cost to add in $\widetilde{\mathcal{P}}_i^{L_0,L_1}(\mathcal{A}\cup\mathcal{A}')$
(3) Using iteratively the separation phase, find a set K' of circuits so that there is no violated circuit inequalities to add in $\widetilde{\mathcal{P}}_i^{L_0,L_1}(A\cup A'))$
(4) If $A'=\emptyset$ and $K'=\emptyset$, then STOP
(5) $A\leftarrow A\cup A'$ and $K\leftarrow K\cup K'$
(6) Go to (1)

Since this Price-and-Cut algorithm solves the linear relaxation of the subproblem, the branching tree induced by the branching rules will give an optimal solution of $\mathcal{P}_1$ or $\mathcal{P}_2$.

5. Valid inequalities for the two-index formulation

In formulation $\mathcal{P}_2$, a preemptive schedule of a project corresponds to several solutions of $\mathcal{P}_2$. For instance, given a solution (z, x) of $\mathcal{P}_2$, for a given $\{i, j\} \in \mathcal{F}$, if $\sum_{a \in \mathscr{A} \mid \{i,j\} \subseteq a} z_a = 0$, then x_{ij} can sometimes be indifferently set to 0 or to 1. This implies that the linear relaxation $\widetilde{\mathcal{P}}_2$ of $\mathcal{P}_2$ may produce unnecessary nodes in the branching tree. In particular, an extreme point (z, x) of $\widetilde{\mathcal{P}}_2$ may satisfy one or several inequalities (3.4) with equality, this implies that some components of x may not be integer. However, another solution $(z, \tilde{x})$ corresponding to the same R-antichain subset and the same variable z can be obtained by increasing as much as possible the value of x_{ij}, for every $\{i, j\} \in \mathcal{F}$. Hence either $x_{ij} \leq 1$ or a circuit inequality (3.5) will be tight for $\tilde{x}$. Consequently, the resulting point $\tilde{x}$ will satisfy more (resp. less) circuit inequalities (resp. inequalities (3.4)) with equality than x.

In order to construct a 1-to-1 correspondence between a subset of R-antichains and a solution (x, z) of $\mathcal{P}_2$, we can restrict the solutions to the incidence vectors x of inclusion-wise maximal acyclic subgraphs of $\Gamma^2(\mathscr{A})$. Let us consider the polytope Q_{max} defined as the convex hull of all the incidence vectors of inclusion-wise maximal acyclic subgraphs of $\Gamma^2(\mathscr{A})$.

Given a subset W of $\mathcal{F}$, let us consider the set $\mathcal{AC}(W)$ of all subsets of pairs of $\mathcal{F}$ inducing an acyclic subgraph of $\Gamma^2(W)$. Given an R-antichain $\{i, j\} \in \mathcal{F}$, let $\mathscr{C}_{ij}$ the set of all subsets of pairs of $\mathcal{F}$ inducing a circuit containing the R-antichain $\{i, j\}$. We then consider the set $\mathcal{C}_{ij}$ obtained as the union of the pairs of every subsets of $\mathscr{C}_{ij}$. Note that $\mathcal{C}_{ij}$ contains pair $\{i, j\}$. Consider the inequalities

$$x_{ij} + x(\mathcal{C}_{ij} \setminus W) \geq 1 \quad \text{for all } \{i, j\} \in \mathcal{F} \text{ and for all } W \in \mathcal{AC}(\mathcal{C}_{ij}) \text{ with } \{i, j\} \in W. \tag{5.1}$$

Theorem 5.1. *Inequalities (5.1) are valid for Q_{max}.*

Proof. Let M be an inclusion-wise maximal subset of $\mathcal{AC}$ and let us consider χ^M its incidence vector. Let $\{i, j\} \in \mathcal{F}$ and $W \in \mathcal{AC}(\mathcal{C}_{ij})$ with $\{i, j\} \in W$. Let us remark, that since W induces an acyclic subgraph and contains $\{i, j\}$, then $\mathcal{C}_{ij} \setminus W$ is a non-empty set of nodes. If $\chi^M_{ij} = 1$, the inequality is trivially satisfied by χ^M. Let us now suppose that $\chi^M_{ij} = 0$ and $\chi^M(\mathcal{C}_{ij} \setminus W) = 0$. Hence, M induces an acyclic subgraph and contain no pair of $\mathcal{C}_{ij} \setminus W$. Then $M \cup \{\{i, j\}\}$ still induces an acyclic subgraph of $\Gamma^2(\mathscr{A})$, a contradiction. □

Moreover we have the following property.

Theorem 5.2. *Let (z, x) be a solution of the formulation obtained by adding inequalities (5.1) to formulation $\mathcal{P}_2$. Then x is the incidence vector of an inclusion-wise maximal acyclic subgraph of $\Gamma^2(\mathscr{A})$.*

Proof. Let $M = \{\{i, j\} \in \mathcal{F} \mid x_{ij} = 1\}$. Clearly, M induces an acyclic subgraph of $\Gamma^2(\mathscr{A})$ due to inequalities (3.5). Let us suppose that M is not an inclusion-wise maximal subset of $\mathcal{AC}$, *i.e.*, there exists a pair $\{i, j\} \in \mathcal{F}$ such that $x_{ij} = 0$ and $W = M \cup \{\{i, j\}\}$ still induces an acyclic graph. Let $W' = W \cap \mathcal{C}_{ij}$. Since $x(\mathcal{C}_{ij} \setminus W') = 0$, the inequality (5.1) associated with $\{i, j\}$ and W' is then violated by x, a contradiction. □

Unfortunately, inequalities (5.1) are in exponential number and their associated separation problem appears to be hard to solve. In fact, we do not know its complexity. In this article, we will only consider particular inequalities of this class. Let us consider the following *connected component inequalities*

$$x_{ij} + \sum_{\{i', j'\} \in \pi(i,j)} x_{i'j'} \geq 1 \quad \text{for all } \{i, j\} \in \mathcal{F} \tag{5.2}$$

$$x_{ij} + \sum_{\{i', j'\} \in \sigma(i,j)} x_{i'j'} \geq 1 \quad \text{for all } \{i, j\} \in \mathcal{F}, \tag{5.3}$$

TABLE 1. Linear relaxation values for the three formulations.

	Ropt	Ming. Gap	NBbest	TT
$\widetilde{\mathcal{P}}_1$	56.83	0.16	76	0.2
$\widetilde{\mathcal{P}}_2$	57.02	0.49	136	1.6
$\widetilde{\mathcal{P}}_3$	57.05	0.54	144	2.2

where $\pi(i,j)$ (resp. $\sigma(i,j)$) is the set of pairs $\{i',j'\}$ such that $\{i',j'\}$ belongs to the strongly component of $\mathcal{F}$ containing $\{i,j\}$ and such that $(\{i',j'\},\{i,j\})$ (resp. $(\{i,j\},\{i',j'\})$) is an arc of $\Gamma^2(\mathscr{A})$. Clearly, given $\{i,j\} \in \mathcal{F}$, inequality (5.2) (resp. (5.3)) is a particular case of inequalities (5.1) when $W = \mathcal{C}_{ij} \setminus \pi(i,j)$ (resp. $W = \mathcal{C}_{ij} \setminus \sigma(i,j)$). Note that it can be produced an incidence vector of $\mathcal{AC}(\mathcal{F})$ which satisfies inequalities (5.2) and (5.3)) and which is not the incidence vector of a subset inducing an inclusion-wise maximal acyclic subgraph.

In what follows we present our experimental results. For these experiments, we will also consider a further formulation, denoted by $\mathcal{P}_3$, obtained from $\mathcal{P}_2$ by adding inequalities (5.2) and (5.3). Note that $\mathcal{P}_3$ is also equivalent to the preemptive RCPSP.

6. Experimental results

We can now present some computational results obtained using our BCP algorithms for the three formulation $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$. The algorithm has been implemented in C++ using the SCIP 3.0.1 framework. It was tested on a PC with 2.40GHz, 1 TB Ram for 80 processors, running under linux.

For these experiments, we have focused on the PSPLIB instances of 30 activities. Indeed, these are the only instances for which the preemptive RCPSP have been solved to optimality in the literature. In [16], an optimal solution has been produced within a few seconds using a Branch-and-Bound algorithm combined with a column generation procedures based on Mingozzi formulation.

Each line of the three following tables corresponds to the average values obtained over the 480 instances of the PSPLIB containing 30 activities. The first column of the table gives the formulation name.

Table 1 presents the linear relaxation $\widetilde{\mathcal{P}}_1$, $\widetilde{\mathcal{P}}_2$ and $\widetilde{\mathcal{P}}_3$ of the three formulations. The entries of Table 1 are as follows.

Ropt	:	the average optimal linear relaxation value,
Ming. Gap	:	the average relative error between the optimal linear relaxation value and the optimal value of the Mingozzi linear formulation,
NBbest	:	the number of instances for which the optimal linear relaxation value is strictly greater than the value of the Mingozzi linear relaxation [15] over the 480 instances,
TT	:	the average total CPU time in seconds to obtain the relaxation value.

From Table 1, we can first remark that the relaxations of the three formulations can be obtained within a few seconds and often give better lower bounds than the optimal values of the Mingozzi linear formulation. In particular, formulation $\mathcal{P}_3$ gives a strictly greater lower bound for 144 instances of the 480 ones, with 0.54% improvement in average.

Since every solution of the antichain formulation $\mathcal{P}_1$ corresponds to a solution of the two-index formulation $\mathcal{P}_2$, the relaxation of formulation $\mathcal{P}_2$ is better than the one of formulation $\mathcal{P}_1$, and we can notice from Table 1 that this latter value is often strictly greater. Recall that formulation $\mathcal{P}_3$ is obtained from $\mathcal{P}_2$ by adding valid inequalities (5.2) and (5.3). We can remark that these inequalities give rise to a small improvement of the linear relaxation.

TABLE 2. Experimental results within a time limit of 3 h.

	Copt	LB	IntG	OptG	NBcol	NBcut	NBnode	TT	TTprice	TTsep	NBopt
$\mathcal{P}_1$	59.45	57.20	4.10	2.43	336.7	152450.2	3948.10	4759.6	523.2	3481.5	278
$\mathcal{P}_2$	59.38	57.40	3.66	2.41	1862.5	232209.7	23909.02	5266.1	3211.5	654.9	253
$\mathcal{P}_3$	58.74	57.47	2.38	1.27	1629.8	194707.0	16914.11	4217.8	2551.5	456.3	306

TABLE 3. Experimental results within a time limit of 5 min.

	Copt	LB	IntG	OptG	NBcol	NBcut	NBnode	TT	TTprice	TTsep	NBopt
$\mathcal{P}_1$	59.95	57.09	5.13	3.26	222.3	33245.3	400.39	153.26	47.77	99.8	246
$\mathcal{P}_2$	60.03	57.27	4.92	3.44	917.4	13107.5	1014.85	163.5	128.47	28.6	225
$\mathcal{P}_3$	59.37	57.29	3.82	2.38	811.1	14915.1	876.54	152.81	115.4	22.0	252

Table 2 (resp. Tab. 3) gives the experimental results for the BCP algorithms for the three formulation within a CPU time limit of 3 h (resp. 5 min). Their entries are as follows.

Copt	:	the average best obtained value,
LB	:	the average best obtained lower bound,
IntG	:	the average relative error between the best obtained value and the best obtained lower bound,
OptG	:	the average relative error between the best obtained value and the optimal solution obtained in [16],
NBcol	:	the average number of generated columns,
NBcut	:	the average number of generated circuit inequalities,
NBnode	:	the average number of nodes in the branching tree,
TT	:	the average total CPU time in seconds,
TTprice	:	the average total CPU time to solve the pricing subproblems in seconds,
TTsep	:	the average total CPU time to solve the separation problems in seconds,
NBopt	:	the number of instances solved to optimality over the 480 instances.

Within a time limit of 3 h, the two BCP algorithms presented in this article succeed in solving to optimality more than the half of the instances and propose good solutions with a small gap from the optimal solutions.

The antichain formulation $\mathcal{P}_1$ generates few columns during the pricing phase and then consumes less time in the pricing phase than the other formulation. However, the separation phase for $\mathcal{P}_1$ is much more time consuming since the graph on which the separation problem is solved have to be dynamically constructed at every iteration. We can notice than the branching tree is quite smaller for $\mathcal{P}_1$ than for the other formulation. Consequently, the antichain formulation $\mathcal{P}_1$ is able to solve more instances than the two-index formulation $\mathcal{P}_2$, but unfortunately propose a lower average gap from the optimal solution.

Finally, formulation $\mathcal{P}_3$ is clearly better with an average gap of 1.27% from the optimal solutions and 306 instances over 480 solved to optimality. Indeed, adding inequalities (5.2) and (5.3) to formulation $\mathcal{P}_2$ permits to generate less nodes in the branching tree, less columns and less cuts.

Since Table 3 presents the same statistics than Table 2 for the same algorithm within 5 min instead of 3 h, the number of solved instances is of course lower in Table 3 than in Table 2. However, it can be noticed that half of the instances have been solved to optimality and that the optimality gap of $\mathcal{P}_3$ is still small. Then our BCP algorithms can be used efficiently to produced good solutions within 5 min.

TABLE 4. Details of the experimental results for $\mathcal{P}_3$ within a time limit of 3 h.

Inst.	NC	RS	RF	IntG	OptG	NBcol	NBcut	NBnode	TT	NBopt
J30-01	1.50	0.20	0.25	12.00	6.85	7420.6	932 069.8	31 268.9	9833.4	1
J30-05	1.50	0.20	0.50	10.78	3.63	4968.3	745 881.9	26 662.6	10 841.5	0
J30-09	1.50	0.20	0.75	8.00	3.03	1217.7	592 285.5	30 620.5	9721.0	1
J30-13	1.50	0.20	1.00	6.50	4.28	1329.6	803 604.1	19 213.4	9943.3	1
J30-02	1.50	0.50	0.25	4.44	2.48	6323.9	564 924.4	28 226.0	6545.9	4
J30-06	1.50	0.50	0.50	9.55	6.50	6578.5	651 090.9	33 942.4	10 800.0	0
J30-10	1.50	0.50	0.75	9.20	7.78	6382.4	767 458.9	19 800.6	9725.7	1
J30-14	1.50	0.50	1.00	9.01	7.28	4509.6	473 665.4	26 550.0	9758.1	1
J30-03	1.50	0.70	0.25	2.15	0.60	1167.8	242 346.7	24 431.1	4321.1	6
J30-07	1.50	0.70	0.50	3.13	2.71	3017.3	280 174.1	20 474.0	5245.1	6
J30-11	1.50	0.70	0.75	0.93	0.83	2106.9	130 797.1	7608.3	3038.1	8
J30-15	1.50	0.70	1.00	0.89	0.89	1687.6	115 949.0	4775.9	2263.3	8
J30-04	1.50	1.00	0.25	0.00	0.00	27.6	0.0	1.0	0.0	10
J30-08	1.50	1.00	0.50	0.00	0.00	28.5	0.0	1.0	0.0	10
J30-12	1.50	1.00	0.75	0.00	0.00	17.5	0.0	1.0	0.0	10
J30-16	1.50	1.00	1.00	0.00	0.00	27.3	0.0	1.0	0.0	10
J30-17	1.80	0.20	0.25	5.28	0.77	2367.3	235 428.9	33 659.8	6549.0	4
J30-21	1.80	0.20	0.50	1.94	0.66	1252.3	148 649.5	30 809.3	6653.3	5
J30-25	1.80	0.20	0.75	1.74	0.03	916.3	159 127.3	28 207.5	7847.7	3
J30-29	1.80	0.20	1.00	2.12	0.55	603.4	285 874.9	28 419.9	8674.0	2
J30-18	1.80	0.50	0.25	4.35	1.89	2946.7	591 367.6	74 411.0	8640.0	2
J30-22	1.80	0.50	0.50	4.38	2.03	2605.6	128 464.8	15 588.9	5473.9	5
J30-26	1.80	0.50	0.75	1.40	0.76	1590.4	190 219.1	12 074.3	4396.7	6
J30-30	1.80	0.50	1.00	5.03	3.75	2787.1	350 062.2	24 403.6	10 202.3	1
J30-19	1.80	0.70	0.25	0.79	0.62	2128.7	51 973.6	14 281.0	2831.8	8
J30-23	1.80	0.70	0.50	1.18	0.42	1199.5	168 171.8	23 602.5	4968.2	6
J30-27	1.80	0.70	0.75	0.18	0.18	1542.9	50 237.3	12 629.9	2055.1	9
J30-31	1.80	0.70	1.00	0.17	0.01	1007.7	32 975.8	9474.4	2874.2	8
J30-20	1.80	1.00	0.25	0.00	0.00	23.0	0.0	1.0	0.0	10
J30-24	1.80	1.00	0.50	0.00	0.00	25.0	0.0	1.0	0.0	10
J30-28	1.80	1.00	0.75	0.00	0.00	35.7	0.0	1.0	0.0	10
J30-32	1.80	1.00	1.00	0.00	0.00	28.0	0.0	1.0	0.0	10
J30-33	2.10	0.20	0.25	2.15	0.15	1733.8	180 981.9	51 636.6	6488.0	4
J30-37	2.10	0.20	0.50	1.82	0.00	975.2	99 935.3	40 328.9	5209.4	6
J30-41	2.10	0.20	0.75	0.29	0.00	458.4	38 029.9	15 719.4	3150.1	8
J30-45	2.10	0.20	1.00	0.00	0.00	206.4	16 081.8	2711.4	663.9	10
J30-34	2.10	0.50	0.25	0.49	0.00	288.9	16 799.3	7201.5	1089.1	9
J30-38	2.10	0.50	0.50	1.13	0.07	1194.0	88 040.8	29 486.3	5152.1	6
J30-42	2.10	0.50	0.75	0.71	0.55	876.9	25 338.4	9802.4	2592.0	8
J30-46	2.10	0.50	1.00	0.45	0.24	1091.5	81 065.4	15 965.3	5233.3	7
J30-35	2.10	0.70	0.25	0.21	0.17	401.4	4610.0	14 818.3	1105.8	9
J30-39	2.10	0.70	0.50	0.24	0.00	1011.0	17 987.9	17 291.9	1861.3	9
J30-43	2.10	0.70	0.75	1.33	1.15	1096.7	60 083.5	16 095.0	4326.3	6
J30-47	2.10	0.70	1.00	0.46	0.21	936.8	24 179.3	9672.5	2379.3	8
J30-36	2.10	1.00	0.25	0.00	0.00	22.5	0.0	1.0	0.0	10
J30-40	2.10	1.00	0.50	0.00	0.00	22.9	0.0	1.0	0.0	10
J30-44	2.10	1.00	0.75	0.00	0.00	23.8	0.0	1.0	0.0	10
J30-48	2.10	1.00	1.00	0.00	0.00	19.9	0.0	1.0	0.0	10

We can notice that most of the columns and cuts are produced during the first minutes. This implies that the branching phase is really time consuming: it will then be very interesting to study how to reinforce the master problem of formulation $\mathcal{P}_3$ in order to reduce the time spent in the branching phase.

In order to give more insights concerning our BCP algorithm performance, Table 4 gives some statistical details over the instances. Indeed, the PSPLIB instances, containing 30 activities and 4 resources, proposed in [12] are divided within 48 groups of 10 instances, where each group is characterized by the three parameters NC, RF and RS. Parameter NC (network complexity) is the average number of non-redundant arcs per node in the graph corresponding to poset G: three values are considered for NC: NC$\in \{1.50, 1.80, 2.10\}$. Parameter RS (Resource Strength) is a scaling parameter expressing the resource availability of a resource and parameter RF (Resource Factor) reflects the average portion of one resource used by an activity. Four values are considered for RS and RF: RS$\in \{0.20, 0, 50, 0.70, 1.00\}$ and RF$\in \{0.25, 0.50, 0.75, 1.00\}$. Indeed, each of the instance groups corresponds to a distinct triplet over the 48 possible ones.

Table 4 only presents the experimental results obtained by the BCP algorithm for formulation $\mathcal{P}_3$ as this formulation is clearly the better one. Each line gives the average values over the 10 instances of the group whose name and parameters are given in the four first columns. Groups are sorted by values of the triplet NC, RS and RF. It appears that this order almost corresponds to the case where the groups are sorted by their difficulties to be solved. We can first notice that instances having a denser poset are easier to solve by our BCP. This can be explained by the fact that the poset density directly impacts the number of R-antichains in the problem and then the number of potential columns. Indeed the number of generated columns decrease with NC. We can also remark that instances having a high amount of resources (*i.e*, when RS is high) are easier to solve. This can be explained by the fact that, in this case, more activities can be set in parallel in the schedule leading our BCP algorithm to directly generate R-antichains that will be in the optimal solution. For instance, when RS equals to 1, the problem is directly solved without using the branching phase.

7. Conclusion

In this paper, we have studied two MIP formulations for the preemptive Resource-Constrained Precedence Scheduling Problem, respectively called antichain and two-indices formulation. We have developed two different Branch-and-Cut-and-Price algorithms for solving the two formulations. From our experiments, it appears that the antichain formulation needs less variables than the two-indices formulation to solve the same instances. However, the latter has performed better. Moreover, using additional valid inequalities, the two-indices formulation could solve most of the 30 activities instances of the PSPLIB within 3 h time limit and give a solution with a gap lower than 1.27% gap for each of these instances.

In [16], these instances have been solved to optimality within a few seconds using a Branch-and-Bound approach. Since this approach consists in enumerating every acyclic subgraph of graph Γ^2, this method cannot be used for larger instances. However our BCP approach can be improved in order to solve bigger instances. In particular, it will be interesting to devise a polyhedral approach for the master problem of the two-indices formulation to produce additional valid inequalities. Another improvement for this BCP approach is to study further strategies to speed up the convergence of the pricing phase.

References

[1] R. Alvarez-Valdés and J.M. Tamarit Goerlich. The project scheduling polyhedron: dimension, facets, and lifting theorems. *Eur. J. Oper. Res.* **67** (1993) 204–220

[2] P. Baptiste and S. Demassey. Tight LP bounds for resource constrained project scheduling. *OR Spectrum* **26** (2004) 251–262

[3] P. Brucker, A. Drexl, R. Möhring, K. Neumann and E. Pesch. Resource-constrained project scheduling: notation, classification, model, and methods. *Eur. J. Oper. Res.* **112** (1999) 3–41.

[4] P. Brucker and S. Knust. A linear programming and constraint propagation-based lower bound for the rcpsp. *Eur. J. Oper. Res.* **127** (2000) 355–362.

[5] J. Damay, A. Quilliot and E. Sanlaville. Linear programming based algorithms for preemptive and non-preemptive rcpsp. *Eur. J. Oper. Res.* **182** (2007) 1012–1022.

[6] M.R. Garey and D.S. Johnson, Computers and Intractability: a Guide to the Theory of NP-Completeness, edited by W.H. Freeman (1979).

[7] S. Gualandi and F. Malucelli. Exact solution of graph coloring problems via constraint programming and column generation. *INFORMS J. Comput.* **24** (2011) 1–20.
[8] P. Hansen, M. Labbé and D. Schindl. Set covering and packing formulations of graph coloring: slgorithms and first polyhedral results. *Discret. Optim.* **6** (2009) 135–147.
[9] J.R. Hardin, G.L. Nemhauser and M.W.P. Savelsbergh, Strong valid inequalities for the resource-constrained scheduling problem with uniform resource requirements. *Discret. Optim.* **5** (2008) 19–35.
[10] W. Herroelen, E. Demeulemeester and B. De Reyck, An integrated classification scheme for resource scheduling. Technical report, Department of applied economics K.U.Leuven, (1999).
[11] R. Klein, *Scheduling of Resource Constraints Projects*. Kluwer Academic Publishers, Boston (1999).
[12] R. Kolish and A. Sprecher, http://www.om-db.wi.tum.de/psplib/.
[13] E. Malaguti, M. Monaci and P. Toth, An exact approach for the vertex coloring problem. *Discret. Optim.* **8** (2010) 174–190
[14] A. Mehrotra and M. Trick, A column generation approach for graph coloring. *INFORMS J. Comput.* **8** (1996) 344–354.
[15] A. Mingozzi, V. Maniezzo, S. Ricciardelli and L. Bianco, An exact algorithm for the resource-constrained project scheduling based on a new mathematical formulation. *Manag. Sci.* **44** (1998) 714–729.
[16] A. Moukrim, A. Quilliot and H. Toussaint, Branch and price for preemptive resource constrained project scheduling problem based on interval orders in precedence graphs, in *6th Workshop on Computational Optimization, Kraków, Poland* (2013).
[17] J.H. Patterson and W.D. Huber, A horizon-varying, zero-one approach to project scheduling problem. *Manag. Sci.* **20** (1974) 990–998.
[18] J.H. Patterson and G.W. Roth, Scheduling a project under multiple resource constraints: a zero-one programming approach. *AIIE Trans.* **8** (1976) 449–455.
[19] A.A. Pritsker, L.J. Watters and P.M. Wolfe, Multi-project scheduling with limited resources: a zero-one programming approach. *Manag. Sci.* **16** (1969) 93–108.