Comparing lower bounds for the RCPSP under a hybrid constraint-linear programming approach

S. Demassey, C. Artigues, P. Michelon

Laboratoire d'Informatique d'Avignon,
339, chemin des Meinajariés, Agroparc, BP 1228,
84911 Avignon Cedex 9, France
email: sophie.demassey@lia.univ-avignon.fr

Abstract

We propose a cooperation method between constraint programming and integer programming to compute lower bounds for the resource-constrained project scheduling problem. Lower bounds are evaluated through linear relaxations of the two main integer program formulations of the RCPSP. We show how these bounds can be improved by performing (i) efficient preprocessing and (ii) on-the-fly generation of cutting planes, both by using constraint propagation algorithms. The interest of such a cooperation is demonstrated through a preliminary computational analysis.

Keywords: Resource-constrained project scheduling problem, lower bounds, constraint propagation rules, linear programming, cutting-planes.

1 Introduction

The resource-constrained project scheduling problem (RCPSP) is one of the most general scheduling problems and is extensively studied in the literature (for a survey of the RCPSP, we refer the reader to the article of Brücker *et al.*[6]). It consists in scheduling a project, i.e. a set of activities linked by precedence constraints by means of a set of limited resources while minimizing the total duration of the project. Being strongly NP hard, this problem has most often been tackled by branch and bound techniques, see e.g.[14, 5]. Consequently some research focuses on the computation of good lower bounds. Our objective is to compare new lower bounds for the RCPSP, based on a cooperation between linear programming and constraint programming.

Constraint programming has been widely applied in the area of scheduling. Especially, numerous deductive "constraint propagation" techniques have yet been successfully implemented. To compute lower bounds, these techniques are generally used at a preprocessing stage to sharpen an integer program formulation of the scheduling problem before evaluating its relaxation. They are very useful to decrease the program size, fixing variables for example and strengthening the linear constraints. Such a cooperation has already been implemented for the RCPSP in [7].

We follow this approach but we lead also the cooperation between constraint programming and integer programming further. Actually, we study here lower bounds resulting of the resolution at optimality of some linear relaxations of the two main integer program formulations of the RCPSP, with natural date variables or with time-indexed variables. With the aim of comparing these different evaluations, the linear programs are preprocessed by a same constraint propagation scheme, including our own shaving technique. Furthermore, we enhance the relaxations of the integer programs, deriving in two manners new cutting-planes from constraint propagation. The cutting-planes of the first class are basically translations of some constraint programming rules in terms of linear inequalities, while the second class of cutting-planes makes use of some deductions implicitly performed by preprocessing. To our knowledge, the latter experiment, which is an actual cooperation, has not been carried out yet for the RCPSP, although similar hybrid approaches are more and more successful in the literature for various combinatorial optimization problems, including scheduling problems [16, 17].

The paper is organized as follows:

Section 2 gives the definitions and notations of the considered problem. In section 3 we present the two existing integer program formulations most encountered for the RCPSP, as well as their respective relaxations being considered. We report in section 4 the different rules implemented in our constraint propagation algorithm at the preprocessing stage. In section 5, we explain for each formulation, how the preprocessing deductions are used within the linear program resolution process. Finally, section 6 presents our preliminary computational experiments on a well-known set of benchmark instances.

2 Definitions and notations

A project is made of a set of activities linked by precedence constraints represented by an activity-on-node (AON) network $G = (V, E)$ where each node in V represents an activity and each arc in E represents a precedence constraint. It is assumed that $|V| = n + 2$ where 0 and $n + 1$ are dummy activities representing the start and the end of the project, respectively. The processing time of activity j is denoted by p_j with $p_0 = p_{n+1} = 0$.

A set $\mathcal{R}$ of renewable resources is considered, each resource $k \in \mathcal{R}$ having a number R_k of units available on the entire planning horizon T. Each activity $j \in V$ requires a positive amount r_{jk} of each resource $k \in \mathcal{R}$ and must be executed without being interrupted.

The objective of the RCPSP is to find a precedence and resource-feasible schedule with a minimal duration, in other words to determine starting times S_i for the activities $i \in V$ in such a way that:

- the precedence constraints are fulfilled, i.e. $S_j \geq S_i + p_i$ for all $(i,j) \in E$,
- at each time t on the entire planning horizon T, and for each resource $k \in \mathcal{R}$, if $S(t)$ denotes the set of activities j executed at time t, i.e. verifying $S_j \leq t < S_j + p_j$, then $\sum_{j \in S(t)} r_{jk} \leq R_k$,
- the total duration, or makespan, $\max_{i \in V} S_i + p_i = S_{n+1}$ is minimized.

3 Integer program formulations and relaxations for the RCPCP

There are two common ways to model scheduling problems as integer programs. The first ones use continuous time variables, the second ones use time indexed variables. Our study relates to two formulations for the RCPSP, one of each kind. The first one, presented in section 3.1, follows Balas' disjunctive graph approach [3]. The second one, in section 3.2, was presumably first inspired by Pritsker *et al.*[21]. Once an integer program is formulated for the RCPSP, any of its relaxations, in particular its linear programming (LP) relaxation that is the relaxation of the integrality requirements, provides a lower bound of the optimal makespan. We present here the two models and the way we use their relaxations.

3.1 Continuous time variables

The classical Balas' disjunctive model for the job-shop problem, based on the natural starting date variables, has been extended for the RCPSP by Alvarez-Valdés and Tamarit [2] making use of the concept of *minimal forbidden set* (i.e. any subset F of activities, minimal for the inclusion and verifying $\sum_{j \in F} r_{jk} > R_k$ for some resource $k \in \mathcal{R}$). To model resource constraints, additional variables are defined: for all couple of activities (i,j), let x_{ij} be 1 if j starts after the completion of i, and 0 otherwise.

The problem in [2] is formulated as follows:

$\min S_{n+1}$

subject to:

$$x_{ij} = 1 \qquad \forall\ (i,j) \in E \tag{C1}$$

$$x_{ij} + x_{ji} \leq 1 \qquad \forall\ (i,j) \in V^2 \tag{C2}$$

$$x_{ik} \geq x_{ij} + x_{jk} - 1 \qquad \forall\ (i,j,k) \in V^3 \tag{C3}$$

$$S_j - S_i \geq -M + (p_i + M)x_{ij} \qquad \forall\ (i,j) \in V^2 \tag{C4}$$

$$\sum_{i,j \in F} x_{ij} \geq 1 \qquad \forall \text{ minimal forbidden set } F \tag{C5}$$

$$x_{ij} \in \{0,1\} \qquad \forall\ (i,j) \in V^2 \tag{C6}$$

Constraints (C1) give the precedence relations within the project. Constraints (C4) model implications $x_{ij} = 1 \Rightarrow S_j \geq S_i + p_i$. Here M is an arbitrarily big constant (greater than T for example). The resource constraints (C5) state that in any minimal forbidden set F, at least one sequencing decision must be taken.

Note that the implementation of the only linear relaxation of this program is not realistic because of the eventual exponential number of constraints (C5). Hence in our relaxation, besides constraints of integrity (C6) we also withdraw all constraints (C5) for minimal forbidden sets of cardinality strictly greater than 3. For that reason, it is clearly essential to tighten this linear program, and in particular to adjust values M of constraints (C4), implicitly taking into account the missing resource constraints. We will see in section 5.1 how constraint programming preprocessing allows it.

3.2 Time-indexed variables

The most encountered integer linear formulation of the RCPSP [21, 12, 20] is based on time indexed boolean variables x_{jt} defined by: $x_{jt} = 1$ if and only if activity j starts at time t, for each activity $j \in V$ and for each time period $t = 0, \dots, T$. Given these variables, the RCPSP can be formulated as follows:

$$\min \sum_{t=0,\dots,T} t x_{(n+1)t}$$

subject to:

$$\sum_{t=0}^{T} x_{jt} = 1 \qquad \forall\, j \in V \tag{D1}$$

$$\sum_{t=0}^{T} t(x_{jt} - x_{it}) \geq p_i \qquad \forall\, (i,j) \in E \tag{D2}$$

$$\sum_{j \in V} r_{jk} \sum_{\tau=t-p_j+1}^{t} x_{j\tau} \leq R_k \qquad \forall\, k \in \mathcal{R}, \forall\, t \in \{0, \dots, T\} \tag{D3}$$

$$x_{jt} \in \{0,1\} \qquad \forall\, j \in V, \forall\, t \in \{0, \dots, T\} \tag{D4}$$

where inequalities (D2) and (D3) represent precedence and resource constraints respectively.

Christofides *et al.* propose in [12] the same formulation but where precedence constraints are presented in a disaggregated strongest way:

$$\sum_{\tau=t}^{T} x_{i\tau} + \sum_{\tau=0}^{t+p_i-1} x_{j\tau} \leq 1 \qquad \forall\, (i,j) \in E, \forall\, t = 0, \dots, T \tag{D2s}$$

Constraints (D2s) state that for all couple (i,j) in E, if activity i starts at or after time t then activity j cannot start before time $t + p_i$ and conversely.

For the two formulations, we have implemented their linear relaxation: constraints (D1), (D2), (D2s) et (D3) being easily computable despite of their large number. Preprocessing will be, here especially, useful to fix a maximal number of variables.

4 Constraint programming as preprocessing

In the first phase, a constraint propagation algorithm is applied to the problem given a feasible upper bound T.

As Brücker and Knust [7], our algorithm has been implemented using the *start-start distance (SSD)-matrix* formalism. A SSD-matrix $B = (b_{ij})$ is any matrix of integers indexed by V^2 and verifying:

$$S_j - S_i \geq b_{ij}, \quad \forall (i,j) \in V^2, \ \forall \text{ optimal schedule } S.$$

The goal of our preprocessing is then to adjust with more close the interval domains $[b_{ij}, -b_{ji}]$ of the variables $S_j - S_i$, in other words to increase entries b_{ij} of the matrix.

First, note that in terms of start-start distance:

- $b_{0(n+1)}$ is a lower bound (*CPLB*) of the optimal value of the RCPSP, and
- the relative order between two activities i and j is explicit:

$$\begin{array}{lcll}
b_{ij} \geq p_i & \iff & i \rightarrow j \ (i \text{ precedes } j) & \forall \text{ optimal schedule} \\
b_{ji} \geq 1 - p_i & \iff & i \nrightarrow j \ (i \text{ does } \textbf{not} \text{ precede } j) & \forall \text{ optimal schedule} \\
\left.\begin{array}{l} b_{ji} \geq 1 - p_i \\ b_{ij} \geq 1 - p_j \end{array}\right\} & \iff & i \parallel j \ (i \text{ and } j \text{ in parallel}) & \forall \text{ optimal schedule.}
\end{array}$$

Note also that a first SSD-matrix is easily computable for a given instance of RCPSP with a planning horizon T. Floyd algorithm applied to the oriented valuated graph $G' = (V, E \cup (n+1, 0))$ (arc $(n+1, 0)$ being valuated by $-T$) computes such a matrix with a $O(n^3)$ complexity, setting for each $i, j \in V$, b_{ij} to the longest path length in G' between i and j.

The propagation of any increase of entries b_{ij} on the overall matrix B will be also provided by the same algorithm, which computes the **transitive closure** of the matrix B ($b_{ik} := \max_{k \in V}(b_{ij} + b_{jk})$), hence reflecting the transitivity property:

$$S_k - S_j \geq b_{jk} \wedge S_j - S_i \geq b_{ij} \implies S_k - S_i \geq b_{ij} + b_{jk}.$$

Besides this algorithm, we have implemented the three following local constraint propagation techniques. They use (and maintain for the first one) a symmetric relation, namely the *disjunction* relation D, over the set V of activities, defined as follows:

$(i,j) \in D$ (or $i - j$) if and only if, in all optimal schedule, i and j are not executed simultaneously.

Symmetric triples algorithm deduces new disjunctions considering forbidden sets of three activities. For example, let $(i, j, k) \in V^3$ be a forbidden set, then $k \parallel i$ and $k \parallel j$ imply that i and j are in disjunction. Other relations are deduced considering an additional activity l related to such a symmetric triple (i, j, k). We have implemented the $O(m^2n^4)$ algorithm proposed by Brucker *et al.* [5].

Immediate selection algorithm (see e.g. [8]) is a simple $O(|D|)$ algorithm that states implication: $i - j$ and $i \nrightarrow j \implies j \rightarrow i$.

Edge-finding algorithm of Carlier and Pinson [9] deduces also new precedence relations but considering *cliques of disjunction* that are sets of activities pairwise in disjunction. To compute maximal cliques we have implemented two heuristics, the one proposed by Brücker *et al.* [5] the other proposed by Baptiste and Le Pape [4]. The overall algorithm of clique generation and edge-finding runs in $O(n^4)$.

By all these propagation techniques the entries of B are eventually increased. Furthermore, infeasibility may be detected if for some activity i, $b_{ii} > 0$ occurs, that says: no feasible schedule of total duration less than T exists. To still improve the constraint propagation process, we use this last property applying an adapted shaving technique. *Shaving* [10, 19, 11] follows the general principle of consistency enforcing techniques based upon refutation for the Constraint Satisfaction Problem: A new constraint c is temporarily added and constraint propagation, for example, is performed. If it leads to an infeasibility, then the negative constraint $\rceil c$ is posted. We have adapted the shaving techniques to the RCPSP in the way of solving sequencing decisions:

Shaving algorithm. For each pair of activities $\{i, j\}$ we test the validity of the three following constraints: $i \rightarrow j$, $j \rightarrow i$ and $i \parallel j$, propagating it on the overall problem by means of the four local techniques described above. We obtain then the new SSD-matrices $B^{i\rightarrow j}$, $B^{j\rightarrow i}$ and $B^{i\|j}$ of the same RCPSP instance in which is added the corresponding constraint. If one or two among this three matrices is proved inconsistent (that is if an entry of its diagonal is strictly positive) then the corresponding constraints are refuted and global deductions on B can be done. Moreover, if the constraint $i \parallel j$ is refuted then the disjunction $i - j$ is added to D. One example: let $\{i, j\}$ a pair of activities such that $B^{j\rightarrow i}$ is inconsistent but not $B^{i\rightarrow j}$ and $B^{i\|j}$. It implies that in any optimal schedule (and in any feasible schedule S such that $S_{n+1} \leq T$), either i precedes j or i and j are in parallel. This information is stated by the global increase of entries of B:

$$B := \min(B^{i\rightarrow j}, B^{i\|j}).$$

Going further, even if no infeasibility is detected, the distance matrix may however be updated as follows

$$B := \min(B^{i\rightarrow j}, B^{j\rightarrow i}, B^{i\|j})$$

Such a global operation is mostly powerful but also very time consuming since it reproduces the local constraint propagation algorithm for each unresolved sequencing decision between two activities. We have been exploring two manners of keeping reasonable CPU times, on the one hand reducing the local constraint propagation algorithm within the shaving (essentially by suppressing symmetric triples rules), on the other hand restricting shaving to some pairs of activities (the disjunctive pairs).

5 Valid inequalities inferred from constraint propagation

At the end of the preprocessing, and if the latter did not reach alone the upper bound, that is if $b_{0(n+1)} < T$, some computed data are fixed and stored in order to tighten the linear programs reported in section 3. These data are

(i) the SSD-matrix B and the disjunction relation D (they will be useful to sharpen the integer programs),

(ii) the maximal cliques of disjunction calculated for the edge-finding and all of the remaining "shaved" distance matrices $B^{i\to j}, B^{j\to i}, B^{i\|j}$ for each pair $\{i, j\}$ of activities not yet sequenced. Indeed they may infer some strong cutting-planes.

In the two next sections we detail how all these results of constraint propagation enhance linear programs corresponding to each formulation with continuous time variables (5.1) or with time-indexed variables (5.2).

5.1 Continuous time variables

As stated in section 3.1, for the integer formulation in continuous time variables, we have chosen to relax, besides the integrity constraints, a number of resource constraints (C5). Some of these constraints have been considered within the preprocessing. So the cooperation allows us to take these dropped constraints implicitly into account in the linear programming stage.

5.1.1 Fixing variables

First before resolution, numerous variables can be fixed considering the SSD-matrix B since the following equalities yield for all optimal schedule

$$x_{ij} = 1 \qquad \forall (i,j) \in V^2 \text{ such that } b_{ij} \geq p_i \tag{C1'}$$

$$x_{ij} = 0 \qquad \forall (i,j) \in V^2 \text{ such that } b_{ji} \geq 1 - p_i \tag{C1''}$$

5.1.2 Strengthening linear constraints

In the same way, we can obviously replace the "big M" value in constraint (C4) by b_{ij}. In order to strengthen the precedence constraints, we replace (C4) by

$$S_j - S_i \geq b_{ij} \qquad \forall (i,j) \in V^2 \mid b_{ij} \geq p_i \tag{C4'}$$

$$S_j - S_i \geq b_{ij} + (p_i - b_{ij})x_{ij} \qquad \forall (i,j) \in V^2 \mid 1 - p_j \leq b_{ij} < p_i \tag{C4''}$$

$$S_j - S_i \geq (1 - p_j) + (p_i + p_j - 1)x_{ij} + (b_{ij} + p_j - 1)x_{ji} \qquad \forall (i,j) \in V^2 \mid b_{ij} < 1 - p_j \tag{C4'''}$$

All these inequalities model the relation $x_{ij} = 1 \Leftrightarrow S_j - S_i \geq p_i$ with more close.

With the new disjunctions deduced by the symmetric triples and the shaving techniques, we can enlarge the definition of forbidden sets of cardinal 2 at all the pair of activities in disjunction, hence increasing the number of remaining constraints (C5).

$$x_{ij} + x_{ji} = 1 \qquad \forall (i,j) \in D \tag{C5'}$$

$$\sum_{u,v \in \{i,j,k\}} x_{uv} \geq 1 \qquad \forall \text{ minimal forbidden set } (i,j,k) \tag{C5''}$$

5.1.3 Generating cutting-planes

We derived from constraint propagation roughly two kinds of valid inequalities. Some fully use shaving deductions, other translate and extend edge-finding techniques in terms of linear inequalities. All of them are seen more in details in [13].

4-uple shaving cuts link the relative sequencing of two activities (i,j) with the relative sequencing of two activities (h,l). Such a link is implicitly represented by the shaved matrices. For instance the inequality $b_{ij}^{h\to l} \geq p_i$ represents the relation $h \to l \;\Rightarrow\; i \to j$. Then the linear constraint $x_{ij} \geq x_{hl}$ is clearly valid.

Following this idea, we generated all the deepest cutting-planes linking variables x_{ij}, x_{ji}, x_{hl}, x_{lh} according to the different values of b_{ij}, b_{ji}, b_{hl}, b_{lh} in the matrices B, $B^{i\to j}$, $B^{h\to l}$, $B^{i||j}$, ... We give here a single example of such a cut when, for all optimal schedules where h precedes l, i and j are necessarily executed in parallel:

$$x_{ij} + x_{ji} \leq 1 - x_{hl} \qquad \text{if } \begin{cases} b_{lh}^{i\to j} \geq 1 - p_h \text{ or } b_{ji}^{h\to l} \geq 1 - p_i \text{ and,} \\ b_{lh}^{j\to i} \geq 1 - p_h \text{ or } b_{ij}^{h\to l} \geq 1 - p_j \end{cases}$$

We have also obtained good results in generating the straightforward valid inequality for all activities i,j,h,l with $i \neq j$ and $h < l$:

$$S_j - S_i \geq b_{ij}^{h||l} + (b_{ij}^{h\to l} - b_{ij}^{h||l})x_{hl} + (b_{ij}^{l\to h} - b_{ij}^{l||h})x_{lh} \tag{C$_S$}$$

Path cuts. Since the optimal solution of the RCPSP is equal to the length of a path made of arcs (i, j) such that $x_{ij} = 1$, we have generated 3-activities and 4-activities path cuts:

$$S_l - S_i \geq \alpha + \beta x_{ij} + \gamma x_{jl} \qquad \forall (i,j,l) \in V^3 \qquad (\mathrm{C}_{3P})$$

$$S_l - S_i \geq \alpha + \beta x_{ij} + \gamma x_{jh} + \delta x_{hl} \qquad \forall (i,j,h,l) \in V^4 \qquad (\mathrm{C}_{4P})$$

The coefficients α, β, γ, δ are calculated from default evaluations of the distance between S_i and S_l in all optimal schedules, according to the different values of x_{ij} and x_{jl} (or of x_{ij}, x_{jh} and x_{hl}). Here again the shaved SSD-matrices provide some tight evaluations, allowing then to generate deeper cutting-planes.

Clique cuts. Under this name, we pool valid inequalities used to update the starting time of an activity of a clique of disjunction with respect to the other activities of the clique. For this reason, these inequalities can be seen as a translation of the edge-finding rules in terms of linear constraints. The maximal cliques C computed and stored within the preprocessing (c.f. section 4) are reused here to apply the clique cuts.

In the remaining, C is any clique of activities all being pairwise in disjunction, and j and l two different activities in C.

The first cuts that we have implemented are those that Applegate and Cook [1] have already used for the job-shop problem under the name "half cuts". They state that each activity $j \in C$ has to be scheduled after all activities $i \in C$ such that $x_{ij} = 1$:

$$S_j \geq \min_{i \in C} b_{0i} + \sum_{i \in C \setminus \{j\}} p_i x_{ij} \quad \forall j \in C \qquad (\mathrm{C}_H)$$

The second cuts from Dyer and Wolsey [15] are called "late job cuts". It modifies half cut by assuming that another activity $l \in C$ is scheduled at the first position. A penalty is then added whenever another activity has to be scheduled before l.

$$S_j \geq b_{0l} + \sum_{i \in C \setminus \{j\}} p_i x_{ij} + \sum_{i \in C \setminus \{l\}} \min(0, b_{0i} - b_{0l}) x_{il} \quad \forall j, l \in C \qquad (\mathrm{C}_{LJ})$$

We propose our own version of the late-job cut by introducing the actual starting time S_l of activity l instead of its earliest start time. Whenever an activity $i \in C$ has to be scheduled before l, S_l is replaced by $S_l + b_{li} \leq S_i$.

$$S_j \geq S_l + \sum_{i \in C \setminus \{j\}} p_i x_{ij} + \sum_{i \in C \setminus \{l\}} b_{li} x_{il} \quad \forall j, l \in C \qquad (\mathrm{C}_{LJ2})$$

Finally, we generate other cuts that tighten (C_{LJ2}) in the case where activity l is known, by CP, to precede all activities of C.

$$S_j \geq S_l + \sum_{i \in C \setminus \{j\}} p_i x_{ij} + \min_{i \in C \setminus \{l\}} (b_{li} - p_l) \quad \forall j, l \in C \qquad (\mathrm{C}_{LJ3})$$

Note that each of these cuts has a symmetric expression that we have also implemented. Actually, the cuts (C_H), (C_{LJ}), (C_{LJ2}) and (C_{LJ3}), compute a lower bound of the distance between the starting time $S_0 = 0$ of the project and the starting time S_j. Their symmetric counterparts compute a lower bound of the distance between the finishing time S_{n+1} of the project and finishing time $S_j + p_j$.

5.2 Time-indexed variables

For the two formulations in time-indexed variables, the weak one and the strong one, we relax only the integrity constraints (D4) as seen in section 3.2.

5.2.1 Fixing variables

As for the continuous formulation, before the resolution, the possibly huge number of variables can be drastically reduced thanks to the SSD-matrix B of the preprocessing. Indeed, for each activity i in V, we only have to define the variables x_{it} for t bounded by the earliest starting time $ES_j = b_{0i}$ of i, and its latest starting time $LS_j = -b_{i0}$.

5.2.2 Strengthening linear constraints

Obviously, precedence constraints may be enhanced as follows:

$$\sum_{t=ES_j}^{LS_j} tx_{jt} - \sum_{t=ES_i}^{LS_i} tx_{it} \geq b_{ij} \quad \forall\ (i,j) \in V^2 \tag{D2'}$$

$$\sum_{\tau=t}^{LS_i} x_{i\tau} + \sum_{\tau=ES_j}^{t+b_{ij}-1} x_{j\tau} \leq 1 \qquad \forall\ (i,j) \in V^2, \forall t \in \{ES_j - b_{ij} + 1, \dots, LS_i\} \tag{D2's}$$

5.2.3 Generating cutting-planes

Numerous cutting-planes have already been proposed for the time-indexed formulations (see e.g. [12, 22]). We reuse some of them, the clique cuts, and propose new ones, the shaving cuts.

Clique cuts. As for the continuous formulation, the well-known clique cuts can be easily implemented for each maximal clique of disjunction precalculated within the constraint programming process.

$$\sum_{i \in C_t} x_{it} \leq 1 \quad \forall \text{ maximal clique } C, \forall\ t \in \{0, \dots, T\} \tag{D$_C$}$$

where $C_t = \{i \in C \mid ES_i \leq t \leq LS_i\}$.

Shaving cuts. With the aim of using shaving results, we propose new valid inequalities, the 4-uple shaving cuts: Let see the example of translating the straightforward implication

$$S_j - S_i > p_i - 1 \Rightarrow S_l - S_h \geq b_{hl}^{i\to j},$$

for two distinct pairs of activities (i,j) and (h,l), by means of the time-indexed variables x_{it}. Since such a relation has an interest only if the current solution of the linear relaxation does not already verify neither $S_l - S_h \geq b_{hl}^{i\to j}$ nor $S_j - S_i < p_i$, we assume that $b_{hl}^{i\to j} > b_{hl}$ and $b_{ji} < 1 - p_i$. Doing so we also ensure that $i \to j$ is not already known by CP, i.e. that $b_{ij} < p_i$.

For more readability, let y_{ij} denote $\sum_{t=ES_j}^{LS_j} tx_{jt} - \sum_{t=ES_i}^{LS_i} tx_{it}$. As for the precedence constraints (D2) and (D2s), we can write the relation according to both formalisms, aggregated or disaggregated:

$$y_{ij} > p_i - 1 \Rightarrow y_{hl} \geq b_{hl}^{i\to j}$$

$$y_{ij} > p_i - 1 \Rightarrow \sum_{\tau=t}^{ES_h} x_{h\tau} + \sum_{\tau=ES_l}^{t+b_{hl}^{i\to j}-1} x_{l\tau} \leq 1 \quad \forall t \in \{0,\dots,T\}$$

The inequality representing the first implication can be written as follows:

$$(-b_{ji} - p_i + 1)y_{hl} \geq (b_{hl}^{i\to j} - b_{hl})y_{ij} + b_{hl}^{i\to j}(1-p_i) - b_{hl}b_{ij} \qquad (\mathrm{D}_S)$$

as show the figure 1, where solutions of the integer program lie in the hatching zone and solutions of the linear relaxation lie in the gray zone.

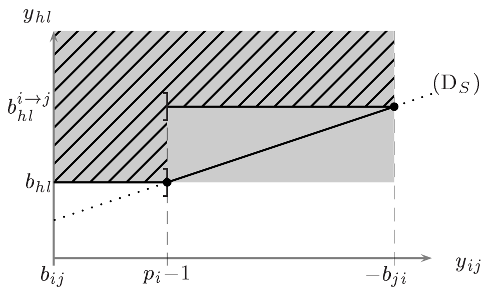

Figure 1: Projection of $\mathcal{S}$ in (y_{ij}, y_{hl})-plane

The second implication can be designed by the next constraints:

$$y_{ij} + b_{ji} \leq (-b_{ji} - p_i + 1)\left(1 - \sum_{\tau=t}^{ES_h} x_{ht} - \sum_{\tau=ES_l}^{t+b_{hl}^{i\to j}-1} x_{lt}\right)$$
$$\forall t \in \{\max(ES_h, ES_l - b_{hl}^{i\to j} + 1), \dots, \min(LS_h, LS_l - b_{hl}^{i\to j} + 1)\} \qquad (\mathrm{D}_{Ss})$$

Obviously, choosing cutting-planes (D_{Ss}) rather than (D_S) amounts to the same thing as choosing between the strong, but more numerous, precedence constraints (D2s) and the weak ones (D2).

6 Preliminary Computational Experiments

We have tested the proposed lower bounds on the Kolisch, Sprecher and Drexl RCPCP instances [18]. The local constraint propagation, shaving and cutting-planes generation algorithms have been written in C++, using ILOG CONCERT 1.0, a LP library embedding CPLEX 7.0.

The lower bounds are built in a constructive way: starting from a feasible upper bound T, the local CP and eventually shaving (total or reduced to the pairs of disjunction) algorithms are applied until no more deductions are found. Then *CPLB* is obtained. The linear programming phase is invoked if $CPLB < T$.

We have implemented the proposed CP based cutting planes for the continuous LP formulation only. For the discrete one, we generate the strong precedence constraints as cuts.

Starting from the LP relaxation, the different pools of cuts are successively added. At each iteration, all the inequalities of a single group are tested inside an enumerative procedure but only the ones violating the current fractional solution are generated and included in the LP. The LP relaxation is solved with the dual simplex and the non-binding cuts are removed from the LP. The on-the-fly cutting plane generation procedure stops when no significant improvement of the lower bound has been made during a certain number of iterations, or when no violating inequality can be found.

Our results were obtained using a Pentium III 800MHz. We compare them with the best known lower bounds computed by Brücker and Knust [7].

In figure 2, we report experiments on the 480 KSD instances with 30 activities. Lines 1 and 2 give the average and maximal deviation Δ_T from optimum of our lower bounds. Lines 3 and 4 give the average and maximal CPU times. We give also the number of instances for which optimal value is reached (line 5) and the number of instances for which linear programming improves constraint propagation (line 6). Each column corresponds to a specific lower bound obtained from:

(1) the local constraint programming (LCP) process alone (*i.e.* constraint programming without shaving),

(2) the complete constraint programming process (including shaving),

(3) the resolution of the weak formulation in time-indexed variables (without cuts) with only LCP preprocessing,

(4) the resolution of the weak formulation in time-indexed variables with complete CP preprocessing and strong precedence cuts,

(5) the resolution of the formulation in continuous time variables with cutting-planes and with the complete CP preprocessing.

For these instances T is set to the optimal solutions which are known. In terms of the quality of the bound, the results on the KSD30 instances are very good.

KSD30	CP		discrete		continuous
	(1)	(2)	(3)	(4)	(5)
Average Δ_T	3.6%	2.1%	3.3%	1.8%	1.9%
Maximal Δ_T	38.0%	34.3%	25.0%	23.9%	31.3%
Av. CPU time (s.)	0.0	0.9	0.3	2.9	2.2
Max. CPU time (s.)	0.0	11.1	7.5	672.8	33.6
# verified instances	307	367	308	369	373
# LP improves CP	-	-	25	39	57

Figure 2: Results on KSD30

With the continuous formulation, we prove the optimality of 373 instances out of 480 whereas the lower bound of Brücker and Knust verifies 318 instances. With the discrete formulation, we verify less instances but the average deviation under the optimum is slightly better (1.8%). However we do not outperform for this criterion the result (1.5%) of Brücker and Knust.

In terms of constraint programming, the shaving technique greatly improves the local rules at the expense of extra computational times. The LP cuts derived from constraint programming succeed in improving significantly the CP bound, especially for the continuous model (57 instances).

We have also tested our algorithm on the 480 KSD instances with 60 activities (see figure 3). For some of them, optimal values are not known. We use then for T the best known upper bounds to date. Here, RCP means the reduced constraint programming process where shaving is only applied to the pairs of activities in disjunction. Columns of the array correspond to:

(1) the LCP process,

(2a) the RCP process,

(2b) the complete CP process,

(3) the resolution of the weak formulation in time-indexed variables with only LCP preprocessing,

(4) the resolution of the weak formulation in time-indexed variables with the RCP preprocessing and strong precedence cuts,

(5) the resolution of the formulation in continuous time variables with cutting-planes and with the RCP preprocessing.

KSD60	CP			discrete		continuous
	(1)	(2a)	(2b)	(3)	(4)	(5)
Average Δ_T	5.5%	4.9%	4.8%	3.3%	2.8%	4.7%
Maximal Δ_T	47.1%	47.1%	47.1%	24.8%	20.5%	47.1%
Av. CPU time (s.)	0.0	1.2	18.8	4.6	164.8	34.8
Max. CPU time (s.)	0.1	27.7	355.3	305.2	1500	591
# verified instances	337	348	354	337	348	349
# LP improves CP	-	-	-	66	70	52

Figure 3: Results on KSD60

For the KSD60 instances, the preprocessing through CP is less efficient than for the previous ones. Because of the size of the problem, the shaving technique has obviously a lower power of deduction. This holds also for the cuts derived from the CP. The continuous formulation still verifies more instances (349) than the discrete formulation (348), and requires much less CPU time. Furthermore we still perform better than Brücker and Knust (340). However the average deviation from the best known solution increases dramaticaly for the continous model. On the other hand, the discrete LP model performs remarkably well for this criterion, improving by 2% the results of the CP phase. This seems to indicate that the size of the problem has less impact on the performance of the discrete time LP relaxation than on the performance of the continuous one. The deviation under the best known lower bound (Brücker and Knust) is of 1.1%.

The interest of the cooperation between CP and LP is enlightened by this experiment. Indeed, the average deviation from the best solution obtained with the cooperation is 2.8% (column 4), while the CP phase alone (including shaving) obtains 4.8% (column 2b) and discrete LP relaxation without any preprocessing obtains 4.3% (not diplayed in the array).

7 Conclusion

We take care not to conclude prematurely on any comparison between our different lower bounds before more complete tests. However we can be encouraged by the first results. Our hybrid approach seems to be very competitive with the best known methods although our computational times are rather high. We aim at developping new shaving cuts for the discrete LP model, since they show their efficiency in the continuous formulation.

We project too, considering the other existing lower bounds, to embed our algorithm in a destructive method with the goal to always find the better lower bound with a minimal computing effort, in view of an exact resolution method for the RCPSP.

References

[1] Applegate D., Cook W., 1991. A computational study of the job-shop scheduling problem. ORSA Journal on Computing, 3(2), 149–156.

[2] Alvarez-Valdés R., Tamarit J.M., 1993. The project scheduling polyhedron: dimension, facets and lifting theorems. European Journal of Operational Research 67, 204-220.

[3] Balas E., 1970. Project scheduling with resource constraints, in E.M.L. Beale (ed.), Applications of Mathematical Programming Techniques, American Elsevier.

[4] Baptiste P., Le Pape C., 2000. Constraint propagation and decomposition techniques for highly disjunctive and highly cumulative project scheduling problems, Constraints, 5, 119–139.

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

[6] Brücker P., Drexl A., Möhring R., Neumann K., Pesch E., 1999. Resource-constrained project scheduling: Notation, classification, models, and methods, European Journal of Operational Research, 112, 3–41.

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

[8] Carlier J., Pinson E., 1989. An algorithm for solving the job-shop problem. Management Science 35, 164–176.

[9] Carlier J., Pinson, E., 1990. A practical use of Jackson's preemptive schedule for solving the job-shop problem. Annals of Operational Research, 26, 269–287.

[10] Carlier J., Pinson E., 1994. Adjustment of heads and tails for the job-shop problem. European Journal of Operational Research, 78, 146-61.

[11] Caseau Y., Laburthe F., 1996. Cumulative scheduling with task intervals. In Michael Maher, editor, Proceedings of the Joint International Conference and Symposium on Logic Programming, MIT Press, 363–377.

[12] Christofides N., Alvarez-Valdés R., Tamarit J.M., 1987. Project scheduling with resource constraints: a branch and bound approach, European Journal of Operational Research, 29(3), 262–273.

[13] Demassey S., Artigues C., Michelon P., 2000. Constraint propagation based cutting planes: an application to the resource-constrained project scheduling problem, Technical report LIA-237, University of Avignon.

[14] Demeulemeester E., Herroelen W., 1997. New benchmark results for the resource-constrained project scheduling problem, Management Science, 43(11), 1485–1492.

[15] Dyer M., Wolsey L.A., 1990. Formulating the single machine sequencing problem with release dates as mixed integer program, Discrete Applied Mathematics, 26, 255–270.

[16] Hooker J., 2000. Logic-based methods for optimization, Wiley, New-York.

[17] Harjunkoski I., Jain V., Grossmann I.E., 2000. Hybrid mixed integer/constraint logic programming strategies for solving scheduling and combinatorial optimization problems. Computers and Chemical Engineering, 24, 337-343.

[18] Kolisch R., Sprecher A., Drexl A., 1995. Characterization and generation of a general class of RCPSP, Management Science, 41, 1693–1703.

[19] Martin P., Shmoys D.B., 1996. A New Approach to computing optimal schedules for the job-shop scheduling problem. in Proceedings of the 5th conference on integer programming and combinatorial optimization Vancouver, British Columbia.

[20] Möhring R.H., Schulz A.S., Stork F., Uetz M., 2000. Solving project scheduling problems by minimum cut computations. Research Report 680/2000, Technische Universitat Berlin, 2000.

[21] Pritsker A.A., Watters L.J., Wolfe P.M., 1969, Multi-project scheduling with limited resources: a zero-one programming approach, Management Science, 16, 93–108.

[22] Sankaran J.K., Bricker D.L., Huang S.-H., 1999. A strong fractional cutting-plane algorithm for resource-constrained project scheduling, International Journal of Industrial Engineering, 6(2), 99–111.