A hybrid constraint propagation-cutting plane procedure for the RCPSP

Sophie Demassey, Christian Artigues, Philippe Michelon
Laboratoire d'Informatique d'Avignon - CNRS FRE 2487
Université d'Avignon et des Pays du Vaucluse
LIA-CERI, BP 1228,
84911 Avignon Cedex 9, France
email: `sophie.demassey@univ-avignon.fr`

Abstract

The resource-constrained project scheduling problem (RCPSP) is to minimize the makespan of a project, i.e. the total duration of a set of activities linked by precedence constraints and which execute on limited resources. In this paper, we present a new lower bound for this problem computed in a destructive way by both constraint programming and linear programming. The continuous relaxation of the classical integer linear formulation of the RCPSP with time-indexed variables is considered. The algorithm tests the validity of a given lower bound T by proving the infeasibility of this linear program tightened by the additional constraint that is, the makespan does not exceed $T-1$. In order to strengthen the linear relaxation, efficient constraint propagation techniques including shaving, are performed as preprocessing. As well as original constraint programming-based cutting planes are generated. The interest of such a cooperative method is demonstrated through a computational analysis on well-known sets of instances.

Keywords: resource-constrained project scheduling problem, cutting plane method, constraint propagation.

1 Introduction

An instance of the resource-constrained project scheduling problem (RCPSP) is made of:

- a set $\mathcal{R}$ of m resources with limited availabilities $R_k \in \mathbb{N}^*,\ \forall k \in \mathcal{R}$,
- a *project* i.e. a set V' of n activities. Each activity i must execute over $p_i \in \mathbb{N}^*$ time units and request during this period a constant amount $r_{ik} \in \mathbb{N}$ of each resource k. Moreover a partial order E' is given on the set V' representing precedence relations between the activities.

Two dummy activities 0 and $n+1$ (with duration and requests equal to 0) are also added to represent the start and the end of the project respectively. Hence let define $V = V' \cup \{0, n+1\}$ and $E = E' \cup \{(0,i),(i,n+1) \mid i \in V'\}$.

The objective of the problem is then to find a schedule S on V, i.e. an activity starting times vector $(S_0, S_1, \ldots, S_{n+1}) \in \mathbb{N}^V$ such that:

- S verifies the precedence constraints:
$$S_j \geq S_i + p_i \text{ for all } (i,j) \in E,$$
- S verifies the limitation resource constraints:
$$\sum_{j \in V_t} r_{jk} \leq R_k \text{ for any time } t \text{ and for any resource } k \in \mathcal{R},$$
where $V_t = \{j \in V' \mid S_j \leq t < S_j + p_j\}$ is the set of activities in progress at time t,
- the duration of the project or makespan S_{n+1} is minimized.

As a generalization of classical scheduling problems as the Job-Shop problem, RCPSP is clearly NP-hard and widely studied in the literature (see e.g. the survey [3]).

Numerous lower bounds on RCPSP have been proposed to be incorporated into branch-and-bound procedures, and especially linear programming (LP) based lower bounds. The most encountered integer linear formulation is based on time-indexed variables and has been treated by cutting-plane methods [9, 19] or by lagrangian relaxation [9, 17]. Another formulation have also been proposed in [16]. One of its relaxation has recently been enhanced by Brucker and Knust [4] applying column generation techniques.

Another way of bounding search trees is to apply constraint propagation rules to prove that no optimal schedule lie in a given node of the tree. Such constraint programming (CP) techniques have been used for the RCPSP (see e.g. [8, 1, 12]).

In [13], Klein and Scholl used CP techniques too, but in a destructive way to explicitly compute a lower bound. The principle of destructive approaches is to deduce that a given value T is a lower bound by proving there exists no feasible schedule with makespan lower than T. The lower bound by Brucker and Knust [4] is also computed in a destructive way. Moreover, for each tested value T, they use CP techniques at a first stage either to directly refute T or to tighten the linear program of their column generation procedure.

We inspired from this latter method with the aim of providing a deeper cooperation between constraint programming and linear programming. Hence in a preceding paper [11], we proposed two hybrid lower bounds for the RCPSP computed in a constructive way. The two bounds directly result of continuous relaxation of two different integer linear programs tightened by constraint propagation as preprocessing and by generation of original CP-based cutting planes. In this paper, we propose to still improve the best of these two algorithms, i.e. the one based on the time-indexed formulation, embedding it in a destructive procedure.

In section 2 we present the framework of this procedure. Section 3 gives the constraint programming rules implemented and section 4 gives the integer linear program preprocessed by CP. In section 5, we describe the valid linear inequalities for the linear program deriving from CP. Finally, section 6 presents some computational experiments on well-known sets of benchmark instances.

2 The destructive procedure

Let UB be an upper bound of the optimal makespan (for instance the sum of the duration activities) and LB be the current lower bound which is initialized at 0. By a dichotomizing search procedure, we look for the greatest value T between LB and UB such that the CP+LP algorithm proves there is no feasible schedule with makespan lower than T.

In the first stage of the algorithm, RCPSP is seen as a Constraint Satisfaction Problem with decision variables $S_j - S_i$. All the precedence constraints are taken into account as well as the additional constraint $S_{n+1} \leq T$. Some basic resource constraints are then propagated by means of classical consistency enforcing techniques, including shaving technique. Hence, they perform domain bound adjustments and detect new *disjunctions*, i.e. pairs of activities that cannot execute at the same time. The set of disjunctions is represented by a symmetrical relation D over V. The domain bounds are represented by a so-called *Start Start Distance-matrix* $B = (b_{ij}) \in \mathbb{Z}^{V \times V}$ where the *distance* b_{ij} is defined as a minimum time lag between start of activity i and start of activity j:

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

With this formalism, $[b_{ij}, -b_{ji}]$ is obviously a domain for $S_j - S_i$ such that bound adjustments consist of increasing distances b_{ij}.

The CP phase stops either when infeasibility of T is proved, that is if a variable domain becomes empty ($b_{ij} > -bji$), or when no more deduction is performed. In the second case, the RCPSP including constraint $S_{n+1} \leq T$ is modeled as an integer linear program taking into account the SSD-matrix B previously adjusted by CP. Its continuous relaxation is solved. Then series of valid inequalities deriving from disjunctions and shaving deductions are iteratively added to the program when they are violated by the current fractional solution.

Here again, process stops either if the linear program (and a fortiori T) is infeasible or when no violated inequality can be found or when no significant improvement of the lower bound has been made during a number of iterations. In the two latter cases the whole process CP+LP is run again with UB set to $T - 1$. Otherwise, as soon as the constraint $S_{n+1} \leq T$ is proved to be inconsistent, LB can be updated to $T + 1$ and the whole process is reiterated.

The destructive procedure is run until $LB > UB$ or until computational time limit is reached. At the end, LB is the lower bound.

3 Constraint programming as preprocessing

In this section, we briefly report the different consistency enforcing techniques applied in the CP phase. As explain in the preceding section, their goal is to detect new disjunctive pairs of activities and to increase entries of the SSD-matrix B. All these techniques are seen more in details in [10].

First B is initialized by taking into account all the precedence constraints and the

additional constraint $S_{n+1} \leq T$:

$$b_{ij} = \begin{cases} 0 & \text{if } i = j \\ p_i & \text{if } (i,j) \in E \\ -T & \text{otherwise.} \end{cases}$$

In turn, D is initialized to the set of pairs $\{i,j\}$ of activities such that $r_{ik} + r_{jk} > R_k$ for at least one resource k.

Then, a set of four local constraint propagation rules implemented with an $O(m^2 n^4)$ complexity is applied:

- the transitive closure of B,
- the symmetric triples rules by Brucker et al. [2],
- the immediate selection rule (see e.g. [5]),
- the primal and dual edge-finding rules of Carlier and Pinson [6].

We have also adapted the shaving technique [7, 15, 8] to the RCPSP in the way of removing inconsistent sequencing decisions:

For each pair of activities $\{i,j\}$ we test the validity of the three following constraints: $i \rightarrow j$ $(S_j \geq S_i + p_i)$, $j \rightarrow i$ $(S_i \geq S_j + p_j)$ and $i \parallel j$ $(S_j < S_i + p_i \wedge S_i < S_j + p_j)$, propagating it by means of the four local techniques described above, and getting then three new SSD-matrices $B^{i \rightarrow j}$, $B^{j \rightarrow i}$ and $B^{i \| j}$.

The inconsistency of one or two of these matrices results in global deductions on B or detection of new precedence, parallel or disjunction relation. For instance:

$$\begin{array}{rl} B := \min(B^{i \rightarrow j}, B^{i \| j}) & \text{if } \neg(j \rightarrow i) \\ i \rightarrow j \text{ and } B := B^{i \rightarrow j} & \text{if } \neg(i \parallel j) \text{ and } \neg(j \rightarrow i) \\ \{i,j\} \in D \text{ and } B := \min(B^{i \rightarrow j}, B^{j \rightarrow i}) & \text{if } \neg(i \parallel j) \end{array}$$

Obviously the infeasibility of the three "shaved" SSD-matrices for one pair of activities means the infeasibility of T.

Last, even if no infeasibility is detected, B can however be updated as follows:

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

Since shaving is mostly time consuming, we restrict its application to a reduced set of pairs of activities including especially pairs in disjunction.

At the end of the CP phase and if T is not already proved to be infeasible, linear programming is invoked. In order to tighten the linear program, deductions performed within the CP phase will be used: the adjusted SSD-matrix B to sharpen the integer program, the extended disjunction relation D and all the remaining consistent "shaved" SSD-matrices $B^{i \rightarrow j}, B^{j \rightarrow i}, B^{i \| j}$ to infer cutting planes.

4 The integer linear formulation

Our study relates to the formulation the most encountered for the RCPSP and given first by Pritsker et al. [18]. In this program, the decision variables are binary variables defined for each activity $j \in V$ and for each time period $t \in \{0, \ldots, T\}$ by:

$$y_{jt} = 1 \text{ if and only if activity } j \text{ starts at time } t.$$

Note that for each activity j, $S_j = \sum_{t=0}^{T} t y_{jt}$.

A first improvement allowed by CP preprocessing is to dratiscally reduce the number of variables since $y_{jt} = 0$ for any time t lower than the earliest starting time of j ($ES_j = b_{0j}$) or greater than its latest starting time ($LS_j = -b_{j0}$).

Hence, the RCPSP can be formulated as follows:

$$\min \sum_{t=ES_{n+1},\ldots,T} t y_{(n+1)t} \tag{1}$$

subject to:

$$\sum_{t=ES_j}^{LS_j} y_{jt} = 1 \qquad \forall\, j \in V \tag{2}$$

$$\sum_{t=ES_j}^{LS_j} t y_{jt} - \sum_{t=ES_i}^{LS_i} t y_{it} \geq b_{ij} \qquad \forall\, (i,j) \in V^2 \tag{3}$$

$$\sum_{j \in V} \Big(r_{jk} \cdot \sum_{\tau=\max(ES_j, t-p_j+1)}^{\min(LS_j,t)} y_{j\tau} \Big) \leq R_k \qquad \forall\, k \in \mathcal{R}, \forall\, t \in \{0, \ldots, T\} \tag{4}$$

$$y_{jt} \in \{0,1\} \qquad \forall\, j \in V, \forall\, t \in \{ES_j, \ldots, LS_j\} \tag{5}$$

Constraints (2) avoid preemption. Inequalities (3) derive immediately from CP deductions since they impose minimum time lags b_{ij} between start of i and start of j. They extend in particular the precedence constraints ($S_j - S_i \geq b_{ij} > p_i$ if $(i,j) \in E$) and include the additional constraint $(S_0) - S_{n+1} \geq b_{(n+1)0} = -T$. Constraints (4) are the resource constraints while constraints (5) enforce variables y_{jt} to be boolean.

The continuous relaxation of this program is solved first. Then cutting planes described in the following section are iteratively added within the solution process.

5 CP-based valid inequalities

We have generated two kinds of CP-based valid inequalities: The clique cuts (subsection 5.1) have a classical shape but they are here mostly deep since they rest on the disjunction relation D enhanced by CP. The shaving cuts (subsection 5.2) are original valid inequalities which translate shaving deductions.

5.1 Clique cuts

Clique cuts are straightforward inequalities stating that if C is a maximal set of mutually incompatible activities then, at any time t, at most one activity of C is executing:

$$\sum_{(j,\tau)\in C_t} y_{j\tau} \leq 1 \quad \forall\, t \in \{0,\ldots,T\} \tag{6}$$

where C_t is the set of couples (j,τ) verifying:

$$\begin{cases} j \in C \text{ and } ES_j \leq t < LS_j + p_j, \\ \tau \in \{\max(ES_j, t-p_j+1),\ldots,\min(LS_j,t)\}. \end{cases}$$

Inequalities (6) are considered for cliques over the disjunction+precedence relation which are generated using two heuristics proposed by Brucker et al. [2] and by Baptiste and Le Pape [1]. We build in average about n cliques with an $O(n^2)$ complexity.

5.2 Shaving cuts

Deductions performed by shaving have not all been propagated within the CP process. However they are stored within the shaved matrices. With shaving cuts, we aim to express such informations in terms of valid linear inequalities for the linear program.

weak 4-uple shaving cuts. The first shaving deduction we have considered is the following straightforward implication

$$S_j - S_i \geq p_i \implies S_l - S_h \geq b_{hl}^{i\to j}, \tag{7}$$

where $\{i,j\}$ and $\{h,l\}$ are two distinct pairs of activities.

To ensure that this deduction is not dominated by another constraint within the CSP formulation, we assume that $b_{hl}^{i\to j} > b_{hl}$ and that $b_{ij} \leq p_i - 1 < -b_{ji}$.

The inequality representing this implication can obviously be written as follows:

$$(-b_{ji} - p_i + 1)((S_l - S_h) - b_{hl}) \geq ((S_j - S_i) - p_i + 1)(b_{hl}^{i\to j} - b_{hl}) \tag{8}$$

as shows 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.

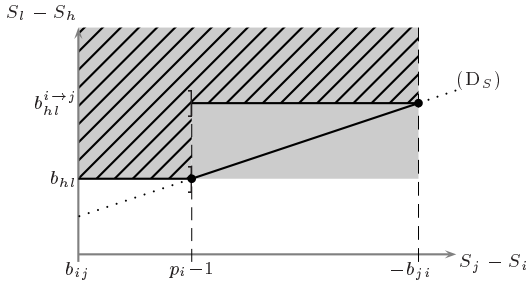


Figure 1: Projection of $\mathcal{S}$ in $(S_j - S_i, S_l - S_h)$-plane

Expressing it as a valid linear inequality for the program with time-indexed variables only consists in replacing variables S_j by $\sum_{t=ES_j}^{LS_j} t y_{jt}$ in inequality (8).

strong 4-uple shaving cuts. There is another way to design the initial relation (7) in a "disaggregated" shape.

Indeed Christofides et al. have proposed in [9] this other representation of precedence constraint $S_j - S_i \geq b_{ij}$:

$$\sum_{\tau=t}^{T} y_{i\tau} + \sum_{\tau=0}^{t+b_{ij}-1} y_{j\tau} \leq 1 \qquad \forall\ (i,j) \in V^2, \forall\ t \in \{0,\ldots,T\} \tag{9}$$

The set of inequalities (9) dominates the aggregated precedence constraint (3). On the other hand, the formulation with the disaggregated precedence constraints (9) take obviously more time to compute.

According to the disaggregated formalism, relation (7) can be formulated by each of the two following logical constraints:

$$S_j - S_i > p_i - 1 \ \Rightarrow \ \sum_{\tau=t}^{LS_h} y_{h\tau} + \sum_{\tau=ES_l}^{t+b_{hl}^{i\to j}-1} y_{l\tau} \leq 1 \quad \forall t \in \{0,\ldots,T\}$$

$$S_l - S_h < b_{hl}^{i\to j} \ \Rightarrow \ \sum_{\tau=t}^{LS_j} y_{j\tau} + \sum_{\tau=ES_i}^{t-p_j} y_{i\tau} \leq 1 \quad \forall t \in \{0,\ldots,T\}$$

The first implication can be designed by the next set of valid inequalities:

$$-b_{ji} - (S_j - S_i) \geq (-b_{ji} - p_i + 1)\left(\sum_{\tau=t}^{LS_h} y_{h\tau} + \sum_{\tau=ES_l}^{t+b_{hl}^{i\to j}-1} y_{l\tau} - 1\right)$$
$$\forall t \in \{\max(ES_h, ES_l - b_{hl}^{i\to j} + 1), \ldots, \min(LS_h, LS_l - b_{hl}^{i\to j} + 1)\} \tag{10}$$

And the second implication by:

$$(S_l - S_h) - b_{hl} \geq \left(\sum_{\tau=t}^{LS_j} y_{j\tau} + \sum_{\tau=ES_i}^{t-p_i} y_{i\tau}\right)(b_{hl}^{i\to j} - b_{hl})$$
$$\forall t \in \{\max(ES_j, ES_i + p_i), \ldots, \min(LS_j, LS_i + p_i)\} \tag{11}$$

3-uple Shaving cuts. In case h (or l) is equal to 0, the two corresponding valid inequalities (8) and (10) are dominated by the following one

$$-b_{ji} - (S_j - S_i) \geq (-b_{ji} - p_i + 1)\left(\sum_{\tau=ES_l}^{ES_l^{i\to j}-1} y_{l\tau} + \sum_{\tau=LS_l^{i\to j}+1}^{LS_l} y_{l\tau}\right) \tag{12}$$

since it models this stronger implication:

$$S_j - S_i \geq p_i \implies ES_l^{i\to j} \leq S_l \leq LS_l^{i\to j}.$$

6 Computational Experiments

We have tested our procedure on the Kolisch, Sprecher and Drexl RCPSP instances [14] with $m = 4$ resources. The whole algorithm is written in C++, using ILOG CONCERT 1.0, an LP library embedding CPLEX 7.0, for the linear programming phase. Our results were obtained using a Pentium III 800MHz. We compare them with the best known lower bounds computed by Brucker and Knust [4] as well as the lower bound computed with the proposed hybrid CP+LP algorithm in a constructive way [11].

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_{opt} of the lower bounds from the optimum. Lines 3 and 4 give the average and maximal CPU times in seconds. We also give the number of instances for which the 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:

(1) the destructive lower bound with complete CP preprocessing, clique cuts and strong shaving cuts,

(2) the constructive lower bound with complete CP preprocessing, clique cuts and strong shaving cuts [11],

(3) the best lower bound [4]

KSD30	hybrid		BK
	(1)	(2)	(3)
Average Δ_{opt}	0.68%	1.69%	1.50%
Maximal Δ_{opt}	15.2%	21.8%	11.1%
Av. CPU time (s.)	3.2	13.7	0.4
Max. CPU time (s.)	230	1129	4.3
# verified instances	403	376	318
# LP improves CP	28	35	71

Figure 2: Results on KSD30

Despite higher computational time, the quality of our destructive lower bound is undeniable since it is clearly better than the tightest known lower bound, in terms of both number of verified instances (403 against 318) and average deviation from the optimum (0.68% against 1.50%).

The power of destructive approaches is obviously demonstrated here, comparing for each criterion (quality and CPU time) results of the same algorithm used in a constructive way (column (2)) and in a destructive way (column (1)).

Finally, the CP algorithm seems to be almost self-sufficient for the KSD30 instances since the cutting-plane procedure is successfully invoked (i.e. at least one tested upper bound T has been refuted within the LP phase) for only 28 instances.

We have also tested our algorithm on the 480 KSD instances with 60 activities (see figure 3). To save computational time here, shaving is only applied to a reduced set of 500 pairs of activities including essentially pairs in disjunction. In our preliminary experiments on these instances, we had not yet implemented the cutting plane procedure. Hence in the array, column (1) (resp. column (2)) corresponds to the proposed lower bound with reduced shaving and without cutting planes computed in a destructive (resp. constructive) way. The maximal CPU time allowed to solve an instance is also set at

1800 seconds. Column (3) is again the best known lower bound by Brucker and Knust [4]. The two first rows give now the average and maximal deviation of the lower bounds from the trivial critical path lower bound.

KSD60	hybrid		BK
	(1)	(2)	(3)
Average Δ_{CPB}	7.69%	6.73%	7.75%
Maximal Δ_{CPB}	81.8%	81.8%	85.7%
Av. CPU time (s.)	115.5	45.0	5.0
Max. CPU time (s.)	1800	1238	62
# verified instances	360	355	341
# LP improves CP	61	57	92

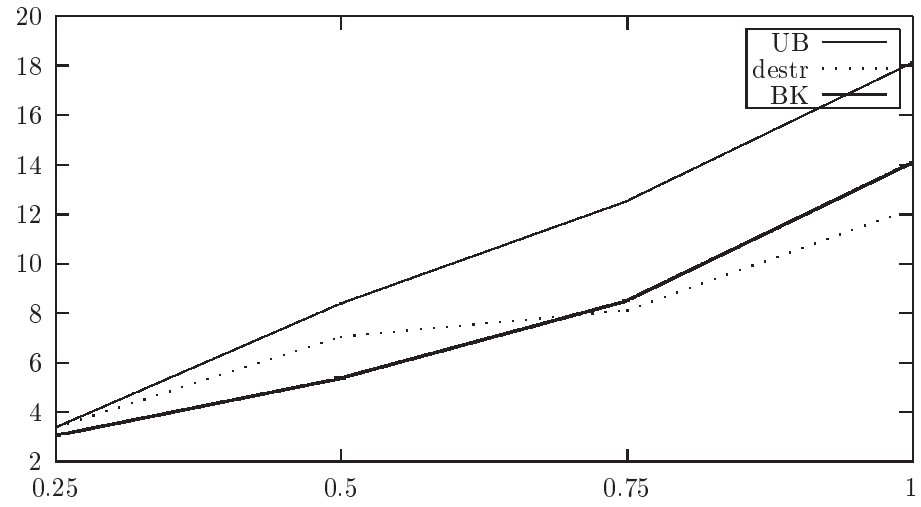
Figure 3: Results on KSD60

As already observed in [11] the shaving technique has a power of deduction as low as the size of the problem grows. On the other hand, linear programming improves the bound for a higher number of instances (61).

For KSD60 instances, the bound by Brucker and Knust is in average slightly better than our bound (7.75% against 7.69%), but again we verify more instances (360 against 341). However the current implementation of our procedure requires clearly much CPU time.

A deeper study of these latter results offers an interesting property of the proposed bound: The following graph represents the behavior of the average deviation Δ_{CPB} of: our lower bound (destr), the best known lower bound (BK) and the best known upper bound (UB), depending on the Resource Factor (see [14]) over the KSD60 instances. It shows that the performance of our algorithm is worse on instances with RF=0.75 and RF=1 i.e. when activities requires in average three or four resources (with $m = 4$).

KSD60: Average deviation from CPB / Resource Factor



7 Conclusion

In this paper, we have presented a new lower bound for the RCPSP resulting from a hybrid constraint programming-cutting plane procedure embedded in a destructive approach. Preliminar computational experiments are also given. This lower bound seems to be really competitive with the best known lower bounds despite rather high computational times. It obtains very good results especially on the KSD instances with 30 activities. In order to save computational time requirements, we are considering to implement a more classical CP procedure based on time windows and shaving of inconsistent starting times. By the way, we will also be able to derive more adapted CP-based cutting planes for the time-indexed linear program.

References

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

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

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

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

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

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

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

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

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

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

[11] S. Demassey, C. Artigues, P. Michelon, Comparing lower bounds for the RCPSP under a constraint-linear programming approach, Working Paper LIA, University of Avignon, France, 2001.

[12] U. Dorndorf, E. Pesch, T. Phan-Huy, A time-oriented branch-and-bound algorithm for project scheduling with generalised precedence constraints, Management Science 46 (2000), 1365–1384.

[13] R. Klein, A. Scholl, Computing lower bound by destructive improvement: An application to resource-constrained project scheduling, European Journal of Operational Research 112 (1999), 322–346.

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

[15] P. Martin, D.B. Shmoys, 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, 1996.

[16] A. Mingozzi, V. Maniezzo, S. Ricciardelli, L. Bianco, An exact algorithm for the multiple resource-constrained project scheduling problem based on a new mathematical formulation, Management Science 44 (1998), 714–729.

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

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

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