INFORMS Journal on Computing
Vol. 17, No. 1, Winter 2005, pp. 52–65
ISSN 0899-1499 | EISSN 1526-5528 | 05 | 1701 | 0052

DOI 10.1287/ijoc.1030.0043

Constraint-Propagation-Based Cutting Planes: An Application to the Resource-Constrained Project Scheduling Problem

Sophie Demassey, Christian Artigues, Philippe Michelon
Laboratoire d'Informatique d'Avignon, 339, Chemin des Meinajariés, Agroparc, BP 1228, 84911 Avignon Cedex 9, France
{sophie.demassey@lia.univ-avignon.fr, christian.artigues@lia.univ-avignon.fr, philippe.michelon@lia.univ-avignon.fr}

We propose a cooperation method between constraint programming and integer programming to compute lower bounds for the resource-constrained project scheduling problem (RCPSP). The lower bounds are evaluated through linear-programming (LP) relaxations of two different integer linear formulations. Efficient resource-constraint propagation algorithms serve as a preprocessing technique for these relaxations. The originality of our approach is to use additionally some deductions performed by constraint propagation, and particularly by the shaving technique, to derive new cutting planes that strengthen the linear programs. Such new valid linear inequalities are given in this paper, as well as a computational analysis of our approach. Through this analysis, we also compare the two considered linear formulations for the RCPSP and confirm the efficiency of lower bounds computed in a destructive way.

Key words: resource-constrained project scheduling problem; cutting plane; constraint propagation; shaving
History: Accepted by John W. Chinneck, Area Editor; received February 2001; revised December 2001, December 2002, March 2003; accepted May 2003.

1. Introduction

The resource-constrained project scheduling problem (RCPSP) is one of the most general scheduling problems that is extensively studied in the literature (Brucker et al. 1999). It consists of scheduling a *project*, i.e., a set of activities linked by precedence constraints, on a set of resources with limited availabilities. The objective is to minimize the total duration of the project, or *makespan*.

Being strongly NP hard, the exact resolution of this problem has most often been tackled by branch-and-bound procedures; see, e.g., Baptiste and Le Pape (2000), Brucker et al. (1998), Demeulemeester and Herroelen (1997), Dorndorf et al. (2000), Mingozzi et al. (1998), and Sprecher (2000). Consequently, some research focuses on the computation of good lower bounds. Among them we can mention the ones based on linear programming, like the cutting-plane algorithm over the time-indexed linear formulation presented in Christofides et al. (1987) and in Sankaran et al. (1999), the lagrangian relaxation of this same formulation by Christofides et al. (1987) and its enhancement by Möhring et al. (2003) solving a minimum cut problem, the preemptive linear relaxations of a new formulation based on the concept of feasible subsets proposed in Mingozzi et al. (1998), the recent improvement of one of these relaxations in Brucker and Knust (2000) applying column-generation techniques, and the linear lower bound in Carlier and Néron (2000) computed from the quick resolution by a parametric approach of a new linear multi-elastic preemptive relaxation of the problem based on the concept of feasible configurations.

In a second category, we can group together constraint-programming-based lower bounds like the ones proposed by Klein and Scholl (1999), Caseau and Laburthe (1996), Baptiste and Le Pape (2000), or Dorndorf et al. (2000). In fact, among all these authors, Klein and Scholl are the only ones actually to compute a lower bound of the optimal makespan. They make use of a *destructive* procedure: constraint-propagation rules are applied in order to prove that no feasible schedule with makespan lower than T exists, yielding a fortiori that $T + 1$ is a lower bound. On the other hand, in the three other papers, constraint-propagation rules are directly applied to prune a search tree by proving that no optimal schedule can be reached from a given node.

The bound proposed in Brucker and Knust (2000) belongs to both categories since they use constraint-propagation techniques to preprocess their linear program.

Our objective is also to propose lower bounds for the RCPSP based on cooperation between linear programming (LP) and constraint programming

(CP). We first use constraint-propagation algorithms as a preprocessing technique as in Brucker and Knust (2000) and we compute a lower bound by solving to optimality the linear program without the integrality constraints (i.e., the LP relaxation). Such a preprocessing of linear programs by constraint programming is relatively well-known. However, in contrast to other hybrid methods, including the Brucker and Knust one, we aim to exploit the deductions performed by constraint propagation in a deeper way. Indeed, we derive from theses deductions new valid linear inequalities that are added to the LP relaxation to strengthen the LP-based bound, if they *cut* the current solution. To our knowledge, the latter experiment, which is an actual cooperation, has not been carried out yet for this problem. However, approaches based on such cooperation are increasingly being reported as successful in the literature for various combinatorial optimization problems, including scheduling problems (Harjunkoski et al. 2000, Hooker 2000). We apply this hybrid constraint-linear programming approach on two different linear formulations of the RCPSP, one based on a continuous-time representation and the other based on a discrete-time representation. In both cases, the same preprocessing phase is used. It is composed of constraint-propagation algorithms and it includes an original shaving technique. Then we propose new CP-based valid inequalities for each of the two linear programs. Last, for a further improvement, we embed the best of these two approaches into a destructive procedure.

The paper is organized as follows. Section 2 gives definitions and notation for the RCPSP. In §3 we review two integer linear formulations for the RCPSP, as well as their relaxations. We report in §4 the different rules implemented in our constraint-propagation algorithm at the preprocessing stage and, in particular, the global shaving rule. In §5, we explain for each formulation, how the information provided by constraint programming is used to derive valid linear inequalities within the linear program resolution procedure. Finally, §6 presents some computational results, including an experimental comparison between the two linear formulations and the results of the destructive approach applied to the time-indexed formulation.

2. Definitions and Notation

An instance of the RCPSP is composed of:

- a set $\mathcal{R}$ of m renewable resources with limited availabilities $R_k \in \mathbb{N}^*,\ \forall k \in \mathcal{R}$;
- a project or a set V' of n activities. Each activity i must execute over $p_i \in \mathbb{N}^*$ time units and requests 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.

It is assumed that two dummy activities 0 and $n+1$ (with null duration and requests) are added to represent the start and the end of the project, respectively. Let $V = V' \cup \{0, n+1\}$ 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 $(S_0, S_1, \ldots, S_{n+1}) \in \mathbb{N}^V$ in such a way that $S_0 = 0$, and:

- S satisfies the precedence constraints: If i and j are two activities linked by $(i,j) \in E$ then j cannot start before the completion of i, i.e., $S_j \geq S_i + p_i$.
- S satisfies the limited-resource constraints: At any time t and for any resource k, the capacity of k must not be exceeded by the total request of the activities in progress at time t, i.e., $\sum_{j \in V_t} r_{jk} \leq R_k$, where $V_t = \{j \in V' \mid S_j \leq t < S_j + p_j\}$.
- The completion time of the project (makespan) S_{n+1} is minimized.

Finally, let T denote an upper bound on the optimal makespan.

3. Integer Linear Programs and Relaxations

There are two usual ways to model scheduling problems as integer linear programs: by using continuous-time variables or time-indexed variables. Our study is related to one formulation in each category. The first one, presented in §3.1, follows the disjunctive graph approach by Balas (1970). The second one, in §3.2, was presumably given first in Pritsker et al. (1969). Once an integer program is formulated for the RCPSP, the exact resolution of any of its relaxations, in particular of its LP relaxation (i.e., dropping the integrality requirements on the variables), provides a lower bound on 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 time variables S_i, was extended to the RCPSP by Alvarez-Valdés and Tamarit (1993) making use of the concept from Radermacher (1985) of *minimal forbidden sets* (i.e., any subset F of activities not linked by any precedence path in E, satisfying $\sum_{j \in F} r_{jk} > R_k$ for some resource $k \in \mathcal{R}$ and minimal for inclusion). To model resource constraints, additional variables are defined: for any pair of activities (i,j), let x_{ij} be 1 if j starts after the completion of i, and 0 otherwise.

The RCPSP is formulated in Alvarez-Valdés and Tamarit (1993) as follows:

$$\begin{aligned} &\min\ S_{n+1} \\ &\text{subject to: } x_{ij} = 1 \quad \forall (i,j) \in E \end{aligned} \tag{C1}$$

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

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

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

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

$$x_{ij} \in \{0,1\} \quad \forall (i,j) \in V \times V \tag{C6}$$

$$S_i \geq 0 \quad \forall i \in V. \tag{C7}$$

Constraints (C1) give the precedence relations within the project. Constraints (C2) and (C3) avoid cycles. Constraints (C4) model implications $x_{ij} = 1 \Rightarrow S_j \geq S_i + p_i$, where M is some large constant. The resource constraints (C5) state that in any minimal forbidden set F, at least one sequencing decision must be taken. Finally, constraints (C6) and (C7) state that decision variables x_{ij} are Boolean and that the variables S_i are nonnegative, respectively.

Note that the implementation of the LP relaxation of this program is not practical because of the possible exponential number of constraints (C5). Hence, in our relaxation, besides the integrality constraints (C6), we also drop all constraints (C5) with minimal forbidden sets of cardinality strictly greater than 3. Hence, it is clearly essential to tighten this linear program, and in particular to adjust the value of M inside the constraints (C4), to take into account the missing resource constraints implicitly. We will see in §5.1 how constraint-programming preprocessing allows this.

3.2. Time-Indexed Variables

The most frequently encountered integer linear formulation of the RCPSP (Christofides et al. 1987, Möhring et al. 2003, Pritsker et al. 1969) is based on time-indexed Boolean variables y_{jt} where $y_{jt} = 1$ if and only if activity j starts at time t, for each activity $j \in V$ and for each time period $t = 0, \ldots, T$. Given these variables, the RCPSP can be formulated as follows:

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

$$\text{subject to: } \sum_{t=0}^{T} y_{jt} = 1 \quad \forall j \in V \tag{D1}$$

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

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

$$y_{jt} \in \{0,1\} \quad \forall j \in V,\ \forall t \in \{0, \ldots, T\}, \tag{D4}$$

where constraints (D1) state that each activity must be started exactly once over the planning horizon T. Inequalities (D2) and (D3) represent precedence and resource constraints, respectively. Constraints (D4) enforce variables y_{jt} to be 0-1.

Christofides et al. (1987) introduce a variant where precedence constraints are presented in a disaggregated (strongest) way:

$$\sum_{\tau=t}^{T} y_{i\tau} + \sum_{\tau=0}^{t+p_i-1} y_{j\tau} \leq 1 \quad \forall (i,j) \in E,\ \forall t = 0, \ldots, T. \tag{D2$_S$}$$

Constraints (D2$_S$) state that for any precedence relation (i,j) in E, if activity i starts at time t or later then activity j cannot start before time $t + p_i$, and conversely.

We have implemented the LP relaxation for these two variants with time-indexed variables despite the relatively large number of constraints (D1), (D2), (D2$_S$), and (D3). Preprocessing is especially useful to fix a maximal number of variables y_{it}.

4. Constraint Programming as Preprocessing

In the first phase of the proposed approach, a constraint-propagation algorithm is applied to the problem, given a feasible upper bound T on the optimal makespan. We detail in this section the constraint-propagation rules we have implemented, including an original shaving technique for the RCPSP. First, we present the way the rules are propagated within the shaving process.

As in Brucker and Knust (2000), our algorithm is implemented using the *start-start distance (SSD)-matrix* formalism. An SSD-matrix $B = (b_{ij})_{V \times V}$ is any integer matrix satisfying, for any feasible schedule S,

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

The major interest in this notion is to reflect the sequencing relations for any pair $\{i,j\}$ of activities. For instance, if $b_{ij} \geq p_i$ then i precedes j (which is denoted by $i \to j$) in any feasible schedule. On the other hand, if $b_{ji} \geq 1 - p_i$, then j starts before the completion of i ($i \nrightarrow j$) and if, moreover, $b_{ij} \geq 1 - p_j$, then i and j are executed in parallel ($i \parallel j$) in any feasible schedule.

With this formalism, the RCPSP can easily be seen as a constraint-satisfaction problem (CSP) with variables $(S_j - S_i)$ and domains approximated by intervals $[b_{ij}, -b_{ji}]$. The bounds on these domains are initialized taking the precedence constraints E and the planning horizon T into account:

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

In this CSP, we also take resource constraints partially into account by computing all the minimal forbidden sets (see §3.1) with two ($\mathcal{F}_2$) or three ($\mathcal{F}_3$) activities. The constraint-programming algorithm also maintains, besides the matrix B, a symmetric relation D over the set of activities, the *disjunction* relation defined by $(i, j) \in D$ if activities i and j cannot be executed in parallel ($i \to j$ or $j \to i$). Obviously, D may be initialized to the set $\mathcal{F}_2$ of minimal forbidden pairs.

The constraint-programming algorithm consists of four local consistency-enforcing techniques (§4.1) embedded in an original shaving framework (§4.2). The objective is to deduce some additional relations (parallel, conjunction, or disjunction), deriving in turn bound adjustments on the variable domains (i.e., increases of some entries in B). During this process, $b_{0(n+1)}$ is an actual lower bound (CPLB) of the optimal makespan.

4.1. Local Constraint Propagation

The local constraint-propagation algorithm we use is inspired by Brucker and Knust (2000). A series of four CP algorithms is applied iteratively until no more adjustments can be performed. A single execution of these four algorithms has an $\mathcal{O}(m^2n^4)$ time complexity. But note that in practice, they are applied only a couple of times. We refer to Brucker et al. (1998) for more details about the four local techniques enumerated below:

Path Consistency. The first local constraint-propagation rule can be implemented in $O(n^3)$ time by the Floyd-Warshall algorithm, which computes the transitive closure of the matrix B by setting $b_{il} := \max_{j \in V}(b_{ij} + b_{jl})$. Hence it reflects the transitivity property $S_l - S_i = (S_l - S_j) + (S_j - S_i)$. Note also that whenever only one entry b_{hl} is updated in a transitively closed matrix, the path consistency is run in $O(n^2)$ time by setting $b_{ij} := \max(b_{ij}, b_{ih} + b_{hl} + b_{lj})\ \forall (i, j) \in V \times V$. This is the case in the shaving process described in §4.2.

The immediate selection algorithm (see, e.g., Carlier and Pinson 1989) is a simple $O(|D|)$ algorithm that replaces each disjunction $\{i, j\} \in D$ by the precedence constraint $i \to j$ whenever $b_{ij} \geq 1 - p_j$ (i.e., $j \nrightarrow i$).

Symmetric triples rules deduce new disjunctions considering forbidden sets of three activities. For example, let $(i, j, k) \in \mathcal{F}_3$ 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. (1998).

Edge-finding rules of Carlier and Pinson (1990) also deduce new precedence relations but consider *cliques of disjunctions* that are sets in which each pair of activities are in disjunction. We use primal (respectively dual) edge-finding to detect whether an activity of the clique has to execute after (respectively before) all the other activities in the clique. For instance, primal edge-finding tests for each activity j in a clique C, if the condition $\min_{i \in C} b_{0i} + \sum_{i \in C} p_i > \max_{i \in C,\, i \neq j}(-b_{i0} + p_i)$ is satisfied. In the positive case, the earliest start time of j is updated by setting $b_{0j} := \max\{b_{0j}, \max_{C' \subseteq C \setminus \{j\}}(\min_{i \in C'} b_{0i} + \sum_{i \in C'} p_i)\}$ and the latest start time of any activity $i \in C$, $i \neq j$ is updated by setting $b_{i0} := \max\{b_{i0}, b_{j0} + p_i\}$.

We also perform additional adjustments on the lower bound $b_{0(n+1)}$ for any computed clique C and all its sub-cliques:

$$b_{0(n+1)} := \max\left\{b_{0(n+1)}, \min_{i \in C} b_{0i} + \sum_{i \in C} p_i + \min_{i \in C} b_{i(n+1)}\right\}.$$

Note, however, that this additional constraint does not propagate. The version of edge-finding we have implemented runs in $\mathcal{O}(|C|^2)$ time (Nuijten 1994), as well as the latter adjustment to the lower bound. To compute cliques we have implemented two heuristics, one proposed in Brucker et al. (1998) and the other proposed in Baptiste and Le Pape (2000). Since this can be done in $\mathcal{O}(n^2)$ time, the overall algorithm of clique generation and edge-finding runs in $O(n^4)$ time.

The entries of B are eventually increased by all these propagation techniques. Furthermore, infeasibility may be detected if some variable domain remains empty ($b_{ij} > -b_{ji}$): that is, if no feasible schedule of total duration lower than T exists.

4.2. Shaving

To improve the constraint-propagation process, we apply an adapted shaving technique. *Shaving* (Carlier and Pinson 1994, Caseau and Laburthe 1996, Martin and Shmoys 1996) follows the general principle of consistency-enforcing techniques based on refutation for a CSP: A new constraint c is temporarily added and constraint propagation is performed. If it leads to an infeasibility, then the opposite constraint $\neg c$ is valid. We have adapted the shaving technique to the RCPSP with the objective of generating sequencing constraints. For each pair of activities $\{i, j\}$, we test the validity of the three following constraints: $i \to j$, $j \to i$, and $i \parallel j$, propagating them separately on the overall problem by means of the four local techniques described in §4.1. We obtain three new SSD-matrices $B^{i \to j}$, $B^{j \to i}$, and $B^{i \parallel j}$ of the same RCPSP instance in which the corresponding constraint is added. If a matrix B^c is inconsistent, that is, if $b^c_{ij} > -b^c_{ji}$ for some activities (i, j) and $c \in \{i \to j, j \to i, i \parallel j\}$, then constraint c is 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.

For instance, let $\{i, j\}$ be a pair of activities such that $B^{j\to i}$ is inconsistent, but neither $B^{i\to j}$ nor $B^{i\|j}$ is inconsistent. This implies that in any feasible schedule either i precedes j, or i and j are in parallel. This information is stated by

$$B := \min\{B^{i\to j}, B^{i\|j}\}.$$

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

$$B := \min\{B^{i\to j}, B^{j\to i}, B^{i\|j}\}.$$

Such a global operation is powerful but also very time-consuming since it calls the local constraint-propagation algorithm for each unresolved sequencing decision. We investigate two ways of keeping reasonable CPU times, on the one hand, by reducing the local constraint-propagation algorithm within the shaving process (essentially by suppressing symmetric triples rules) and, on the other hand, by restricting shaving to a reduced set of pairs of activities.

5. Valid Inequalities Inferred from Constraint Propagation

At the end of the preprocessing, and if the lower bound obtained by constraint propagation CPLB = $b_{0(n+1)}$ has not reached the upper bound T, we resort to the LP relaxations reported in §3, augmented by cutting planes. First, some data computed within the preprocessing phase are fixed and stored: on the one hand, the SSD-matrix B and the disjunction relation D and, on the other hand, the cliques of disjunctions computed for the edge-finding and all the remaining consistent "shaved" SSD-matrices $B^{i\to j}$, $B^{j\to i}$, and $B^{i\|j}$ for each pair $\{i, j\}$ of activities not yet sequenced. The distance and disjunction matrices are used, before the resolution, to sharpen the linear programs by fixing variables and tighten linear inequalities, while shaving deductions and cliques of disjunctions permit us to infer some strong cutting planes. In this section, we detail how deductions performed by constraint propagation enhance the linear programs corresponding to each formulation with continuous-time variables (§5.1) and with time-indexed variables (§5.2).

5.1. Continuous-Time Variables

The constraint-programming algorithm described in the previous section is obviously directed toward sequencing decisions for pairs of activities. Hence, it is especially suited to tighten the linear program in continuous-time variables. In §§5.1.1 and 5.1.2 we describe how constraint programming is used for preprocessing. Then, we derive from constraint propagation roughly two kinds of valid inequalities. The first ones express shaving deductions as described in §§5.1.3, 5.1.4, and 5.1.5. The second ones, described in §5.1.6, translate and extend edge-finding-like rules.

5.1.1. Fixing Variables. Indeed, numerous variables can be fixed before resolution by considering the SSD-matrix B since the following equalities hold for any feasible schedule:

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

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

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

$$S_j - S_i \geq b_{ij} \quad \forall (i, j) \in V \times V \mid b_{ij} \geq p_i \qquad \text{(C4}'\text{)}$$

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

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

These inequalities express the minimal distance, performed within the preprocessing phase, between the beginning of two activities $S_j - S_i \geq b_{ij}$. They also represent the sequencing relations $x_{ij} = 1 \Leftrightarrow S_j - S_i \geq p_i$ and $x_{ij} = x_{ji} = 0 \Rightarrow S_j - S_i \geq 1 - p_j$.

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

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

$$\sum_{u, v \in \{i, j, k\}} x_{uv} \geq 1 \quad \forall \text{minimal forbidden set } (i, j, k) \in \mathcal{T}_3. \qquad \text{(C5}''\text{)}$$

5.1.3. Four-Tuple Shaving Cuts Based on Sequencing. The four-tuple shaving cuts based on sequencing link the relative sequencing of two activities (i, j) with the relative sequencing of two other activities (h, l). Such a link is implicitly represented in the shaved SSD matrices. For instance, the condition $b_{ij}^{h\to l} \geq p_i$ represents the relation $h \to l \Rightarrow i \to j$. Then the linear constraint $x_{ij} \geq x_{hl}$ is clearly valid in this case.

Following this idea, we generate all the dominant cutting planes linking variables x_{ij}, x_{ji}, x_{hl}, and x_{lh} according to the different values of b_{ij}, b_{ji}, b_{hl}, and b_{lh} in the matrices B, $B^{i\to j}$, $B^{h\to l}$, $B^{i\|j}$, $B^{h\|l}$, $B^{j\to i}$, and $B^{l\to h}$.

Another example can give theoretical insight of this approach. Suppose that no information about the relative ordering of i and j has been deduced within the CP phase (i.e., if $b_{ji} < 1 - p_i$ and $b_{ij} < 1 - p_j$). However, the CP has detected that h cannot be scheduled after l (that is, if $b_{hl} \geq 1 - p_l$ and $b_{lh} < 1 - p_h$). Hence, x_{lh} is set

to 0 whereas x_{ij}, x_{ji}, and x_{hl} are undetermined. Since $x_{ij} + x_{ji} \leq 1$, the projection P' of the fractional solution space on $\{(x_{ij}, x_{ji}, x_{hl})\} \subseteq [0,1]^3$ is included in the convex hull of

$$X = \left\{ \begin{pmatrix}0\\0\\0\end{pmatrix}, \begin{pmatrix}1\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\end{pmatrix}, \begin{pmatrix}0\\0\\1\end{pmatrix}, \begin{pmatrix}1\\0\\1\end{pmatrix}, \begin{pmatrix}0\\1\\1\end{pmatrix} \right\}.$$

Because each of these vertices is associated with a necessary existence condition, this set can be reduced through an analysis of the shaved SSD matrices. The proposed cuts are the facets of the convex hull of the subset obtained after removing one or several vertices from X.

For example, the point $\left(\begin{smallmatrix}1\\0\\1\end{smallmatrix}\right)$ can be removed from P' if the following condition is satisfied:

$$b_{lh}^{i \to j} \geq 1 - p_h \quad \text{or} \quad b_{ji}^{h \to l} \geq 1 - p_i. \tag{a}$$

Indeed, this corresponds to the implication $i \to j \Rightarrow \neg(h \to l)$. Furthermore, $x_{hl} \leq 1 - x_{ij}$ is a facet of $\text{conv}\left(X \setminus \left\{\left(\begin{smallmatrix}1\\0\\1\end{smallmatrix}\right)\right\}\right)$. Consequently, it is a valid inequality of P'. Suppose that, in addition, we have,

$$b_{lh}^{i \| j} \geq 1 - p_h \quad \text{or} \quad b_{ij}^{h \to l} \geq p_i \quad \text{or} \quad b_{ji}^{h \to l} \geq p_j. \tag{b}$$

In other words, there cannot simultaneously be $i \| j$ and $h \to l$. Then

$$P' \subseteq X' = \text{conv}\left(X \setminus \left\{ \begin{pmatrix}1\\0\\1\end{pmatrix}, \begin{pmatrix}0\\0\\1\end{pmatrix} \right\} \right)$$

and the deeper cut $x_{ji} \leq x_{hl}$, which is a facet of X', can be generated. Figure 1 gives an illustration of the proposed cuts and shows that using jointly deductions (a) and (b) clearly gives better results than treating them separately.

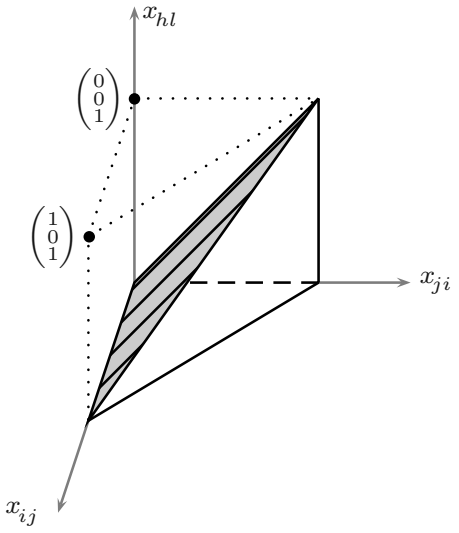

Figure 1 **Cut $x_{ji} \leq x_{hl}$ as a Facet of X'**

The general framework for finding the dominant sequencing cuts is as follows. For each four-tuple (i, j, h, l), we consider the polyhedron of Figure 1. We remove the maximal number of infeasible extreme points by performing an analysis of the distance matrices similar to the above-described one. Such a removal generates one or two facets corresponding to the sequencing cut(s).

5.1.4. Four-Tuple Shaving Cut Based on Distance. A four-tuple shaving cut based on distance is defined for any activities (i, j, h, l) such that $i \neq j$ and $h < l$. It links starting time variables with sequencing variables and is directly derived from the shaved distances, following the "lifting" principle for linear inequalities:

$$S_j - S_i \geq b_{ij}^{h \| l} + (b_{ij}^{h \to l} - b_{ij}^{h \| l}) x_{hl} + (b_{ij}^{l \to h} - b_{ij}^{l \| h}) x_{lh}. \tag{C7}$$

The validity of this cut is straightforward by testing the three admissible values for the pair (x_{hl}, x_{lh}), which are $\left(\begin{smallmatrix}0\\0\end{smallmatrix}\right)$, $\left(\begin{smallmatrix}1\\0\end{smallmatrix}\right)$, and $\left(\begin{smallmatrix}0\\1\end{smallmatrix}\right)$. Note that, in the general case, there is no dominance between this cut and the shaving cuts based on sequencing.

5.1.5. Path Cuts. Obviously, the optimal solution of the RCPSP is equal to the length of a path made of arcs (i, j) such that $x_{ij} = 1$. Hence, it is tempting to generate some "path cuts" with three activities:

$$S_l - S_i \geq \alpha + \beta x_{ij} + \gamma x_{jl} \quad \forall (i, j, l) \in V^3, \tag{C8}$$

where the coefficients α, β, and γ are computed from default evaluations of the distance between S_i and S_l according to the different values of x_{ij} and x_{jl}. Here again the shaved SSD matrices provide some tight evaluations, allowing generation of deeper cutting planes. We detail hereafter the lifting technique that can be used to compute some (α, β, γ) triples such that the resulting cuts $C8(\alpha, \beta, \gamma)$ are dominant.

If $\mathcal{S}(P)$ denotes the set of the feasible solutions of the initial problem P formulated in §3.1, let us define the following notation:

$$b_{il}^{00} = \min\{S_l - S_i \mid S \in \mathcal{S}(P, x_{ij} = 0, x_{jl} = 0)\}$$
$$b_{il}^{10} = \min\{S_l - S_i \mid S \in \mathcal{S}(P, x_{ij} = 1, x_{jl} = 0)\}$$
$$b_{il}^{01} = \min\{S_l - S_i \mid S \in \mathcal{S}(P, x_{ij} = 0, x_{jl} = 1)\}$$
$$b_{il}^{11} = \min\{S_l - S_i \mid S \in \mathcal{S}(P, x_{ij} = 1, x_{jl} = 1)\}.$$

Hence, by enumerating the four possible values for the pair of variables (x_{ij}, x_{jl}) we obtain the following lemma:

LEMMA 1. *$C8(\alpha, \beta, \gamma)$ is a valid inequality for P if and only if $\alpha \leq b_{il}^{00}$, $\alpha + \beta \leq b_{il}^{10}$, $\alpha + \gamma \leq b_{il}^{01}$, and $\alpha + \beta + \gamma \leq b_{il}^{11}$.*

Unfortunately, the computation of b_{il}^{00}, b_{il}^{10}, b_{il}^{01}, and b_{il}^{11} is itself as difficult as is the original problem P. Suppose however that we have four minorants of these values (say A, B, C, and D, respectively); then there is a dominance relation between the valid inequalities of type $C8(\alpha, \beta, \gamma)$:

Proposition 2. *Any valid inequality* $C8(\alpha, \beta, \gamma)$ *satisfying* $\alpha \le A$, $\alpha+\beta \le B$, $\alpha+\gamma \le C$, *and* $\alpha+\beta+\gamma \le D$ *is dominated by the conjunction of two inequalities* $PC1$ *and* $PC2$ *where*,

(i) *if* $A+D \ge B+C$ *then* $PC1 = C8(A, B-A, C-A)$, *and* $PC2 = C8(B+C-D, D-C, D-B)$, *or*

(ii) *if* $A+D \le B+C$ *then* $PC1 = C8(A, D-C, C-A)$, *and* $PC2 = C8(A, B-A, D-B)$.

A proof is in the appendix.

The proposed path cuts with three activities are deepest if A, B, C, and D are close to the optimal coefficients b_{il}^{00}, b_{il}^{01}, b_{il}^{10}, and b_{il}^{11}. A way of computing good minorants is to use the shaving principle of constraint propagation, running the local constraint-propagation algorithms after posting each of the corresponding constraints

$$A = b_{il}^{\neg(i\to j)\wedge\neg(j\to l)}, \qquad B = b_{il}^{(i\to j)\wedge\neg(j\to l)},$$
$$C = b_{il}^{\neg(i\to j)\wedge(j\to l)}, \quad \text{and} \quad D = b_{il}^{(i\to j)\wedge(j\to l)}.$$

However, to save computational time, we perform a weaker approximation, using only the shaved distance matrices stored during the CP phase:

$$A = \max\Big\{\min\{b_{il}^{i\|j}, b_{il}^{j\to i}\}, \min\{b_{il}^{j\|l}, b_{il}^{l\to j}\}, \min\{b_{ij}^{j\|l}, b_{ij}^{l\to j}\} + \min\{b_{jl}^{i\|j}, b_{jl}^{j\to i}\}\Big\}$$
$$B = \max\Big\{b_{il}^{i\to j}, \min\{b_{il}^{j\|l}, b_{il}^{l\to j}\}, \min\{b_{ij}^{j\|l}, b_{ij}^{l\to j}\} + b_{jl}^{i\to j}\Big\}$$
$$C = \max\Big\{\min\{b_{il}^{i\|j}, b_{il}^{j\to i}\}, b_{il}^{j\to l}, b_{ij}^{j\to l} + \min\{b_{jl}^{i\|j}, b_{jl}^{j\to i}\}\Big\}$$
$$D = \max\Big\{b_{il}^{i\to j}, b_{il}^{j\to l}, b_{ij}^{i\to j} + b_{jl}^{j\to l}, b_{ij}^{j\to l} + b_{jl}^{i\to j}\Big\}.$$

Given these coefficients, we obtain dominant path cuts with three activities in the sense of Proposition 2.

By similar arguments, we also obtain dominant cuts for four-activity paths starting in 0 or ending in $n+1$:

$$S_l(-S_0) \ge \alpha + \beta x_{ij} + \gamma x_{jl} \quad \forall (i,j,l) \in V^3 \qquad (C8')$$
$$S_{n+1} - S_i \ge \alpha + \beta x_{ij} + \gamma x_{jl} \quad \forall (i,j,l) \in V^3. \qquad (C8'')$$

Finally, we have generated some path cuts with four nondummy activities without attempting to find all dominant ones:

$$S_l - S_i \ge \alpha + \beta x_{ij} + \gamma x_{jh} + \delta x_{hl} \quad \forall (i,j,h,l) \in V^4. \quad (C9)$$

5.1.6. Clique Cuts. The valid inequalities presented in this section are all defined for any clique of disjunctions and aim at updating the starting time of one of its activities j with respect to the other activities in the clique. For this reason, they can be considered as translations of some edge-finding rules in terms of cutting planes. Each of these cuts has two symmetric expressions. One corresponds to a lower bound of the distance between starting time $S_0 = 0$ of the project and starting time S_j of j. The other corresponds to a lower bound of the distance between completion time S_{n+1} of the project and completion time $S_j + p_j$ of j. We provide the symmetric counterpart only for the first cut since the mechanism for obtaining it is straightforward.

In the remaining, C is any clique of disjunctions, and j and l are two distinct activities in C.

The first cut we have implemented is the "half cut" proposed in Applegate and Cook (1991) for the job-shop problem. It states that each activity $j \in C$ has to be scheduled after all activities $i \in C$ if $x_{ij} = 1$:

$$S_j \ge \min_{i\in C} b_{0i} + \sum_{i\in C\setminus\{j\}} p_i x_{ij} \quad \forall j \in C. \qquad (C10)$$

Let $q_i = b_{i(n+1)} - p_i$ denote the *tail* of activity i, then the symmetric inequality is

$$S_{n+1} - S_j \ge p_j + \sum_{i\in C\setminus\{j\}} p_i x_{ji} + \min_{i\in C} q_i. \qquad (C'10)$$

The second cut from Dyer and Wolsey (1990) is called a "late job cut" and has also been introduced for the job-shop problem. It modifies a 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:

$$S_j \ge b_{0l} + \sum_{i\in C\setminus\{j\}} p_i x_{ij} + \sum_{i\in C\setminus\{l\}} \min\{0, b_{0i} - b_{0l}\} x_{il} \quad \forall j, l \in C. \qquad (C11)$$

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 starting time. Whenever an activity $i \in C$ has to be scheduled before l, S_l is replaced by $S_l + b_{li} \le S_i$:

$$S_j \ge S_l + \sum_{i\in C\setminus\{j\}} p_i x_{ij} + \sum_{i\in C\setminus\{l\}} b_{li} x_{il} \quad \forall j, l \in C. \qquad (C12)$$

Finally, we generate another kind of cut that tightens (C12) if activity l is known to precede all activities of C (such a condition may be detected by the dual edge-finding rule):

$$S_j \ge S_l + \sum_{i\in C\setminus\{j\}} p_i x_{ij} + \min_{i\in C\setminus\{l\}} (b_{li} - p_l) \quad \forall j, l \in C \,|\, l \to C\setminus\{l\}. \quad (C13)$$

To generate all the cuts involving a clique of disjunctions, we have used the heuristic clique generation algorithm of the CP phase (see §4.1).

5.2. Time-Indexed Variables

For the two formulations in time-indexed variables, the aggregated one and the disaggregated one, we relax the only integrality constraints (D4) as seen in §3.2. In §§5.2.1 and 5.2.2, we describe how CP is used to preprocess the time-indexed linear program. Some cutting planes have already been proposed for the time-indexed formulations (see, e.g., Christofides et al. 1987, Sankaran et al. 1999). Among them, we use the clique cuts as described in §5.2.3. In §§5.2.4 and 5.2.5, we also propose a new category of cuts, shaving cuts.

5.2.1. Fixing Variables. As for the continuous formulation, before the resolution, the huge number of variables can be drastically reduced thanks to the preprocessed SSD-matrix B. Indeed, for each activity j in V, we only have to define the variables y_{jt} for t bounded by the earliest starting time $ES_j = b_{0j}$ of j and its latest starting time $LS_j = -b_{j0}$.

5.2.2. Strengthening Linear Constraints. The weak and the strong precedence constraints can be more efficiently implemented as follows:

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

$$\sum_{\tau=t}^{LS_i} y_{i\tau} + \sum_{\tau=ES_j}^{t+b_{ij}-1} y_{j\tau} \leq 1$$

$$\forall (i,j) \in V^2,\ \forall t \in \{ES_j - b_{ij} + 1, \ldots, LS_i\}. \quad \text{(D2'}_S\text{)}$$

5.2.3. Clique Cuts. Clique cuts are packing 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 in process:

$$\sum_{(j,\tau) \in C_t} y_{j\tau} \leq 1 \quad \forall t \in \{0, \ldots, T\}, \qquad \text{(D5)}$$

where $C_t = \{(j,\tau) \in C \times \{\max\{ES_j, t - p_j + 1\}, \ldots, \min\{LS_j, t\}\} \mid ES_j \leq t < LS_j + p_j\}$.

These inequalities are considered for all cliques of disjunctions which were generated during the preprocessing phase and which are maximal for inclusion. Hence, we can expect that these clique cuts are stronger than the ones used in classical implementations since numerous additional disjunctions and conjunctions are likely to be detected by constraint programming.

5.2.4. Four-Tuple Shaving Cuts. With the aim of using shaving deductions, we have first translated the implication $S_j - S_i \geq p_i \Rightarrow S_l - S_h \geq b_{hl}^{i \to j}$ for two distinct pairs of activities (i,j) and (h,l), into linear inequalities by means of the time-indexed variables y_{it}. 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 $b_{ij} \leq p_i - 1 < -b_{ji}$.

For better readability, let z_{ij} denote

$$\sum_{t=ES_j}^{LS_j} t y_{jt} - \sum_{t=ES_i}^{LS_i} t y_{it},$$

i.e., $S_j - S_i$. As for the precedence constraints (D2) and (D2$_S$), we can write the relation according to both formalisms, aggregated or disaggregated:

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

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

The first implication can be modeled by the inequality:

$$(-b_{ji} - p_i + 1)(z_{hl} - b_{hl}) \geq (z_{ij} - p_i + 1)\left(b_{hl}^{i \to j} - b_{hl}\right), \quad \text{(D6)}$$

as shown in Figure 2, where solutions of the integer program lie in the cross-hatched zone and solutions of the linear relaxation lie in the gray zone.

The second implication can be represented by the next set of inequalities:

$$-b_{ji} - z_{ij} \geq (-b_{ji} - p_i + 1)\left(\sum_{\tau=t}^{LS_h} y_{h\tau} + \sum_{\tau=ES_l}^{t+b_{hl}^{i\to j}-1} y_{l\tau} - 1\right)$$

$$\forall t \in \left\{\max\left\{ES_h, ES_l - b_{hl}^{i\to j} + 1\right\}, \ldots, \min\left\{LS_h, LS_l - b_{hl}^{i\to j} + 1\right\}\right\}. \quad \text{(D6}_S\text{)}$$

There is another way to write the initial relation in a disaggregated shape. Indeed, the equivalent relation $S_l - S_h < b_{hl}^{i \to j} \Rightarrow S_i - S_j \geq 1 - p_i$ can be written:

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

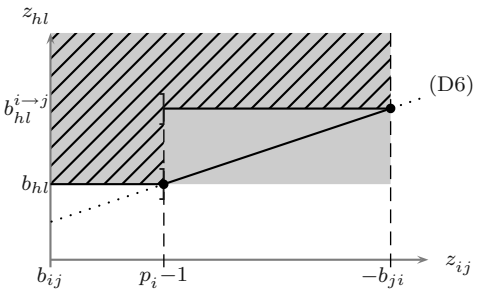

Figure 2 **Projection of $\mathcal{S}$ in the (z_{ij}, z_{hl})-Plane**

Consequently, the following inequalities are also valid:

$$z_{hl} - b_{hl} \geq \left(\sum_{\tau=t}^{LS_j} y_{j\tau} + \sum_{\tau=ES_i}^{t-p_i} y_{i\tau} \right) \left(b_{hl}^{i \to j} - b_{hl} \right)$$

$$\forall t \in \{\max\{ES_j, ES_i + p_i\}, \ldots, \min\{LS_j, LS_i + p_i\}\}. \quad (D6'_S)$$

Obviously, selecting cutting planes ($D6_S$) (or ($D6'_S$)) rather than (D6) amounts to choosing between the strong, but more numerous, precedence constraints ($D2_S$) and the weak ones (D2).

5.2.5. Triple Shaving Cuts. If h (or l) is equal to 0, the two corresponding valid inequalities (D6) and ($D6_S$) are dominated by

$$-b_{ji} - z_{ij} \geq (-b_{ji} - p_i + 1) \left(\sum_{\tau=ES_l}^{ES_l^{i \to j} - 1} y_{l\tau} + \sum_{\tau=LS_l^{i \to j}+1}^{LS_l} y_{l\tau} \right) \quad (D7)$$

since it models the stronger implication $S_j - S_i \geq p_i \Rightarrow ES_l^{i \to j} \leq S_l \leq LS_l^{i \to j}$.

6. Computational Experiments

We have tested our lower bounds on the ProGen instances (Kolisch et al. 1995) with 30, 60, 90, and 120 activities. The constraint propagation, shaving, and cutting-plane algorithms were all written in *C++*, using ILOG CPLEX 7.0 as the LP solver. The experiments were carried out on a Pentium III 800 MHz system, under Linux and using *g++* 2.95.4. Our procedures are essentially compared to the strongest currently available one on these instances, i.e., (BK) proposed by Brucker and Knust (2000), who obtained their results on a Sun Ultra 2 workstation 167 MHz.

We first tested our CP-LP hybrid method on the two linear models in a constructive way. In §6.1 we present variants of these two algorithms and make an experimental comparison of their efficiency for computing lower bounds. To enhance our method, we then selected whichever of these two algorithms that obtained the best lower bounds, and we embedded it into a destructive procedure. We explain the principle of this destructive procedure and report its results compared to the bound by Brucker and Knust (2000) in §6.2.

6.1. Constructive Lower Bounds

The constructive lower bounds are computed according to the following scheme: given a feasible upper bound T, the CP algorithm including shaving is applied until no more deductions are found. Then the constraint-programming lower bound $\text{CPLB} = b_{0(n+1)}$ is obtained. The LP phase is invoked if $\text{CPLB} < T$. Starting from the linear relaxation, the different pools of cuts are successively added in a cyclic way. 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 dual simplex, and the nonbinding cuts are removed from the LP. The on-the-fly cutting-plane procedure stops when no significant improvement of the lower bound has been made during a certain number of iterations, when no violating inequality can be found, or when the computation time allowed is elapsed.

In Table 1, we report experiments on the 264 KSD *nontrivial* instances with 30 activities, which are the instances for which the optimal value is not the length of the critical path within the precedence graph. Rows 1 and 2 give the average and maximal deviation Δ_{opt} from the optimal solution of our lower bounds. Rows 3 and 4 give the average and maximal CPU times. We also give the number of instances for which the optimal value is reached (Row 5) and the number of instances for which LP improves constraint propagation (Row 6). Each column corresponds to a specific lower bound obtained from:

- Columns 2 and 3: the constraint-programming process alone, "LCP" local (i.e., CP without shaving), and "CCP" complete (including shaving).
- Columns 4 and 5: the resolution of the "weak discrete" linear program (i.e., in time-indexed variables with aggregated precedence constraints) with either LCP or CCP preprocessing.
- Columns 6 and 7: the resolution of the weak discrete linear program with CCP preprocessing and cutting planes including either aggregated ("weak") cuts (D6) or disaggregated ("strong") cuts ($D6_S$) and ($D6'_S$).

Table 1 Results on the Nontrivial KSD30 Instances

	CP		Discrete (aggregated)				Continuous
KSD30 264 Instances	LCP	CCP	LCP + LP	CCP + LP	CCP + weak	CCP + strong	CCP + cuts
Average Δ_{opt} (%)	5.8	3.6	5.3	3.2	3.1	3.0	3.2
Maximal Δ_{opt} (%)	33.7	31.3	25.0	25.0	21.8	21.8	29.7
Average time (s.)	0.0	2.3	1.0	3.0	10.2	35.6	4.9
Max time (s.)	0.0	17.3	49.5	31.1	601	1,296	37.6
No. of LB = opt	95	155	96	157	159	160	160
No. of LP > CP	—	—	24	17	35	42	47

• Column 8: the resolution of the "continuous" linear program (i.e., in continuous-time variables) with CCP preprocessing and with cutting planes.

For each of these instances, T is set to the optimal makespan, which is known. In terms of the quality of the bound, the results on the KSD30 instances are very good for both formulations that prove the optimality of 160 instances out of 264. Compared to the continuous formulation, the discrete formulation using our strong cuts and the complete CP process reaches a slightly better average deviation under the optimum (3.0% versus 3.2%) but requires much more average CPU time (35.6 seconds versus 4.9 seconds).

Within the CP phase, the shaving technique greatly improves the local rules (3.6% versus 5.8%) at the expense of extra computational times (2.3 seconds versus 0 seconds). The cuts derived from CP succeed in improving significantly the CP bound: Of the 109 instances not solved by CCP alone, the proposed cutting planes are useful on 18 instances for the discrete formulation with aggregated precedence cuts, 25 instances for the discrete formulation with disaggregated precedence cuts, and 47 instances for the continuous formulation.

The interest in the cooperation between CP and LP is enlightened by this experiment. Indeed, the average deviation from the optimum obtained with cooperation is between 3.0% and 3.1% (Columns 6 and 7), while the CP phase alone (including complete shaving) obtains 3.6% (Column 3), and discrete LP relaxation without shaving preprocessing obtains 5.3%.

We have also tested our algorithm on the 184 nontrivial KSD instances with 60 activities (see Table 2). Some of them are still open (not solved to optimality). We use then for T the best known upper bounds to date. Furthermore, we compare our bounds over their average deviations above the trivial critical path lower bound LB0 (Row 1: Δ_{LB0}). Here, RCP denotes the reduced CP process where shaving is only applied to the pairs of activities in disjunction. Finally, the processing time is limited to 30 minutes.

For the KSD60 instances, preprocessing through CP is less efficient than for KSD30. Because of the size of the problems, the shaving technique has obviously a lower power of deduction. This also holds for the cutting planes derived from the CP. However, use of a reduced version of the CP algorithm appears to be really advantageous since it allows more reasonable CPU times (27.7 seconds versus 62.1 seconds) with only a slight deterioration of the results (9.5% versus 9.6%).

The method based on the continuous formulation proves optimality for one more instance than does the method based on the discrete formulation. However, the average deviation from LB0 stays dramatically low for the continuous model. A basic reason is that, within this procedure, the only resource constraints that are taken into account are the ones involving fewer than three activities at a time.

On the other hand, the discrete LP model performs remarkably well for this criterion, improving by more than 8% the results of the CP phase. The weak and the strong cuts increase the number of times LP improves CP (by 5 and 7, respectively) and perform better in terms of average deviation above LB0 than does the LP relaxation without cuts (17.7% versus 17.5%). However the efficiency of the cutting planes is rather disappointing since they have considerably less impact on the quality of the bound than does the basic LP model itself. Actually, their computation requires a great deal of CPU time. In particular, the disaggregated precedence cutting planes that dominate, in theory, the aggregated ones, do not significantly improve the average deviation above LB0 since their huge number really slows down the entire procedure.

The interest in the cooperation between CP and LP is again underlined by the improvement this cooperation brings to both methods used separately. As an illustration, we improve on the bound proposed in Möhring et al. (2003) based on the discrete-time formulation without CP preprocessing.

As a conclusion to this experimental comparison, it appears that the discrete-time formulation outperforms the continuous-time formulation as the problem size increases. As a counterpart, the cuts we have proposed bring more improvement for the continuous-time formulation than for the discrete-time formulation.

We follow up our experiments by considering only the best algorithm (based on the discrete-time formulation with aggregated precedence cutting planes) and by including it into a destructive procedure.

Table 2 Results on the Nontrivial KSD60 Instances

	CP			Discrete (aggregated)			Continuous
KSD60 184 Instances	LCP	RCP	CCP	RCP + LP	RCP + weak	RCP + strong	RCP + cuts
Average Δ_{LB0} (%)	7.7	9.5	9.6	17.5	17.7	17.7	10.0
Average time (s.)	0.0	27.7	62.1	81.8	243	771	257
Max time (s.)	0.1	130.8	297	904	1,800	1,800	919
No. of LB = opt	41	58	59	58	58	58	59
No. of LP > CP	—	—	—	57	62	64	51

6.2. Destructive Lower Bound

To improve the constructive algorithm based on the discrete formulation, we incorporated it into a destructive procedure. Indeed, Klein and Scholl (1999) reported the efficiency of destructive approaches to compute lower bounds, and the quality of the destructive bound of Brucker and Knust (2000) seems to confirm this conclusion. Moreover, the implementation of a destructive procedure embedding the algorithm presented in the preceding section is quite easy.

Starting again from an upper bound UB of the optimal makespan (e.g., the sum of the duration activities), we look, via a dichotomizing search procedure, for the greatest value T between 0 and UB $-$ 1 such that the constructive CP + LP algorithm proves that there is no feasible schedule with makespan less than or equal to T. Our destructive lower bound is then $T+1$. In both phases of the constructive algorithm, the infeasibility of the planning horizon T can be detected: if $b_{0,n+1} > T$ within the CP phase, or if the set of feasible solutions becomes empty within the cutting-plane-generation phase.

In Table 3, we give the results of our destructive lower bound (destr: Column 6) compared with the trivial critical path bound (LB0: Column 3), with the tightest bound to date on the KSD instances proposed by Brucker and Knust (2000) (BK: Column 4) and with the best available lower bound for each instance (best: Column 5) on the KSD instances with 30, 60, 90, and 120 activities (available at http://www.bwl.uni-kiel.de/Prod/psplib/).

For the KSD60, KSD90, and KSD120 instance sets, we give from top to bottom in Table 3 the average deviation of the lower bounds from the critical path bound (LB0), the average and maximal deviation from the available best upper bound (UB), the average and maximal CPU time in seconds (note that the two bounds were not computed on the same machine), the number of instances for which the optimal value is reached (LB = UB), the number of times we perform better than the previous best-known lower bound, and the number of new optima we prove. Since the optimal value is known for all instances with 30 activities, results on KSD30 presented in Table 3 are only statistics on the deviation from the optimal value, the CPU time, and the number of instances for which the optimal value is reached.

Finally, for each instance set, we adapted our algorithm to save computation time: for the 480 instances with 30 activities, the constructive algorithm was run with CCP preprocessing (i.e., complete shaving), clique cuts, and aggregated precedence cuts. For the KSD60 instances, the algorithm was run with RCP preprocessing (where shaving was applied only to a reduced set of 500 pairs of activities including pairs in disjunction), clique cuts, and aggregated precedence cuts, but within computation time limited to 30 minutes. For the 480 instances with 90 activities and for the 600 instances with 120 activities, we just ran the algorithm with RCP preprocessing and LP without generating cutting planes and with computation time limited to 1 hour.

Table 3 Results on the KSD Instance Sets

No. of Act	LB0	BK	best	destr
30 (480)				
Av. Δ_{opt} (%)	—	1.5	—	**0.7**
Max (%)	—	11.1	—	15.2
Av. CPU*	—	**0.4**	—	3.2
Max	—	4.3	—	229.9
No. of LB = opt	—	318	—	**403**
60 (480)				
Av. Δ_{LB0} (%)	—	7.8	7.9	7.7
Av. Δ_{UB} (%)	7.1	1.9	1.8	**1.8**
Max (%)	50.0	14.7	13.7	17.9
Av. CPU*	—	**5**	—	168
Max	—	3,720	—	1,963
No. of LB = UB	296	341	356	**360**
No. of LB > best	—	—	—	43
No. of new opt	—	—	—	**9**
90 (480)				
Av. Δ_{LB0} (%)	—	7.2	7.2	7.0
Av. Δ_{UB} (%)	6.6	**1.8**	1.8	**1.8**
Max (%)	50.0	12.7	12.7	23.4
Av. CPU*	—	**72**	—	379
Max	—	9,900	—	3,606
No. of LB = UB	334	350	351	**364**
No. of LB > best	—	—	—	28
No. of new opt	—	—	—	**13**
120 (600)				
Av. Δ_{LB0} (%)	—	21.4	21.4	19.1
Av. Δ_{UB} (%)	16.2	**3.8**	3.8	4.8
Max (%)	66.1	17.4	17.4	33.2
Av. CPU*	—	21,300**	—	**1,388**
Max	—	259,200	—	3,836
No. of LB = UB	178	208	208	**229**
No. of LB > best	—	—	—	60
No. of new opt	—	—	—	**21**

*"BK" was computed on a Sun Ultra 2 workstation at 167 MHz and "destr" on a Pentium III at 800 MHz.

**Refers only to 481 of the 600 instances. For the remaining instances the computation was carried out until the limit of 259,200 seconds (72 hours).

Despite the higher computation time, our destructive lower bound is comparable to, and even improves upon, the tightest known lower bound (BK), in terms of both the number of solved instances and the average deviation from the upper bound for the KSD30, KSD60, and KSD90 instance sets. For the biggest instances with 120 activities, we proved optimality for more instances but the average deviation from the upper bound is higher. However, an advantage of our bound is the possibility to limit the computation time, which is not possible for the column-generation process. Hence, for the KSD120 set, our procedure seems to be on average faster

Table 4 Compared Results on the "Hard" KSD30 Instances

Group	NC	RF	RS	LB0	LB1	destr
21	1.8	0.50	0.20	25.67	13.55	**0.00**
25	1.8	0.75	0.20	36.68	9.75	**1.07**
29	1.8	1.00	0.20	41.56	11.66	**5.67**
30	1.8	1.00	0.50	8.88	5.34	**3.17**
31	1.8	1.00	0.70	2.66	**0.74**	0.95
41	2.1	0.75	0.20	37.24	8.21	**0.11**
45	2.1	1.00	0.20	36.52	8.26	**0.36**

(although the tests have not been run on the same machine).

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 (Table 1, Column 6 for KSD30, and Table 2, Column 6 for KSD60) and in a destructive way (Table 3, Column 6).

Finally, note that for each challenging instance set, we found new lower bounds for 43 of 124 nonsolved instances, 28 of 129 and 60 of 392 for the 60, 90, and 120 activity instance sets, respectively. Among these new bounds, we closed the optimality gap for 9, 13, and 21 instances, respectively.

We now propose a deeper analysis of these computational results by evaluating the performance of our bound with respect to the characteristics of the instances. The KSD instances were generated by a controlled design of specified parameters (Kolisch et al. 1995). These characteristics are the network complexity, the resource factor and the resource strength. The network complexity $\mathrm{NC} \in [0, 1]$ gives the ratio of nonredundant precedence constraints. The resource factor $\mathrm{RF} \in [0, 1]$ describes the average number of resources required by a job: $\mathrm{RF} = 1$ means that any nondummy activity requires each of the m resources. The resource strength $\mathrm{RS} \in [0, 1]$ measures the tightness of the resource constraints. Hence, $\mathrm{RS} = 0$ means that the availability R_k of any resource k is set to the minimal feasible value, whereas $\mathrm{RS} = 1$ corresponds to an unconstrained problem where the resource availabilities are set such that the CPM schedule is resource-feasible. The KSD30, KSD60, and KSD90 sets are divided into 48 groups of ten instances sharing the same triple (NC, RF, RS). The KSD120 set is divided into 60 groups of ten similar instances.

Table 6 Compared Results on KSD90 Instances with RS = 0.2

Groups	RS	RF	LB0 (%)	BK (%)	destr (%)	No. of destr > best	No. of UB > best
1,17,33	0.20	0.25	8.23	1.65	**0.36**	**15**	18
5,21,37	0.20	0.50	22.17	8.56	**8.27**	10	30
9,25,41	0.20	0.75	33.13	**8.81**	10.69	0	30
13,29,45	0.20	1.00	37.79	**7.16**	8.47	1	30

For the 30 activity set, Mingozzi et al. (1998) exhibited groups of instances that the branch and bound algorithm of Demeulemeester and Herroelen (1992) was not able to solve. In Table 4, we give the average deviation in percent from the optimum of our destructive bound on these hard instances compared with the LB0 bound and the best LP-based lower bound LB1 of Mingozzi et al. (1998).

Our bound performs better than LB1 on almost all hard instance groups (21, 25, 41, 45, 29, 30), but it does not perform as well on Group 31. In fact, it seems that the "hard" instances identified by Mingozzi et al. (1998) do not correspond at all to the ones for which our method is less efficient. For example, our lower bound reaches the optimal value for all instances of Group 21, while its deviation from the optimal value is on average 8.21% for Group 13.

Actually, Groups 13 and 29 correspond to a high resource factor ($\mathrm{RF} = 1$) and low resource strength ($\mathrm{RS} = 0.2$), which are characteristics that are together not favorable to our bound. On the other hand, the quality of our bound is very good for low resource factors ($\mathrm{RF} = 0.25$ or 0.5) as in Group 21. To show more precisely how these characteristics affect our bound, we give in Tables 5, 6, and 7 experimental results on all instance groups of the 60, 90, and 120 activity sets with $\mathrm{RS} = 0.2$. As mentioned by Brucker and Knust (2000), these are the hardest instances since they correspond to scarce resources and have numerous disjunctions. Among these instances, we have grouped the ones having the same resource factor.

In Tables 5, 6, and 7, Columns 4, 5, and 6 give the average deviation from the best upper bound for, respectively, LB0, BK, and destr lower bounds. In Column 7, we report how many times destr improves upon the best known lower bound among the instances for which the optimality gap is not closed, i.e., instances with the best known upper

Table 5 Compared Results on the KSD60 Instances with RS = 0.2

Groups	RS	RF	LB0 (%)	BK (%)	destr (%)	No. of destr > best	No. of UB > best
1,17,33	0.20	0.25	8.82	0.99	**0.08**	**3**	3
5,21,37	0.20	0.50	23.40	8.23	**3.43**	**27**	30
9,25,41	0.20	0.75	31.51	**9.38**	10.43	8	30
13,29,45	0.20	1.00	39.90	**7.36**	11.55	0	30

Table 7 Compared Results on KSD120 Instances with RS = 0.2

Groups	RS	RF	LB0 (%)	BK (%)	destr (%)	No. of destr > best	No. of UB > best
1,21,41	0.20	0.25	18.67	4.25	**1.34**	**25**	28
6,26,46	0.20	0.50	41.69	**10.23**	11.72	3	29
11,31,51	0.20	0.75	53.29	**10.69**	19.85	0	30
16,36,56	0.20	1.00	58.82	**8.61**	21.44	0	30

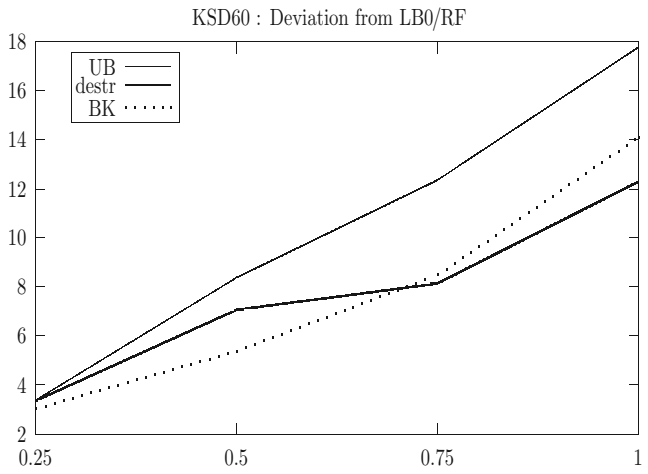

Figure 3 Comparisons on the KSD60 Instances with RS = 0.2

bound strictly greater than the best known lower bound (Column 8).

It appears that our results on the hard instances are not homogeneous and that the quality of our bound strongly depends on the resource-factor characteristic. Indeed, we significantly improve upon the BK bound for instances where RF = 0.25, where we find a large number of new best lower bounds (43 of 49 nonsolved instances) and also for RF = 0.50 (40 new best lower bounds of 89). On the other hand, our procedure is less efficient for instances with RF = 0.75 and RF = 1, i.e., when activities require on average from three to four resources out of four. Again, the CP phase, which appears to be crucial in our scheme, is inefficient for such problem characteristics and the cuts cannot close the gap. This is not surprising since most of the cuts we proposed are based on the shaved distance matrices computed by CP.

This property of the proposed bound is enlightened by Figure 3, which represents the behavior of the average deviation Δ_{LB0} of our lower bound (destr), of the best-known lower bound (BK), and of the best-known upper bound (UB), depending on the resource factor over the KSD60 instances.

7. Conclusion

In this paper, we have presented a new method to compute lower bounds for the RCPSP, based on close cooperation between CP and LP. We have first proposed a new shaving technique from which we derive original cutting planes for linear models of the RCPSP. In order to investigate this method, we have performed two parallel studies by considering two different linear formulations of the RCPSP. In both cases, computational experimentations show that the proposed cuts are able to improve the initial lower bound on some hard RCPSP instances. An experimental comparison between the two proposed lower bounds confirms that the formulation with discrete-time variables is clearly more efficient than the one with continuous-time variables. Indeed, the bound obtained from this first program is quite competitive with the best known lower bounds, despite rather high computation times. To speed up and also to improve this algorithm, we embedded it into a destructive procedure. The results obtained are satisfactory and confirm the power of destructive approaches to compute lower bounds.

A significant number of new lower bounds has been found on each challenging instance set. Although the proposed bound is rather time consuming and significantly weaker on average for the KSD120 instance set than the best current bound designed by Brucker and Knust (2000), it is in turn quite competitive compared to the latter bound for the KSD30, KSD60, and KSD90 sets.

To make a better compromise between time requirements and bound quality, it would be of great interest to combine the lagrangian approach of Möhring et al. (2003), which would speed up the linear program resolution, with our cuts.

It must also be noted that our algorithm is not adapted to difficult highly cumulative instances like the ones proposed by Baptiste and Le Pape (2000). This should motivate the search for constraint-propagation algorithms and cutting planes able to tackle such problem characteristics. For that, we can modify the preprocessing phase of our algorithm (initially developed for the linear program in continuous-time variables), by implementing a more classical CP procedure based on time windows and shaving of inconsistent starting times. We could then derive cutting planes better suited to the discrete-time program while saving space and computational time requirements. It would also be interesting to add energetic reasoning (Lopez et al. 1992) within the CP phase or to generate energetic-reasoning-based cutting planes.

The scope of the experiments we have performed was to validate the approach. This is mostly achieved in this study. Further necessary research lies in the optimization of the management of the cuts inside the cutting-plane-generation algorithm, which is still purely enumerative. Moreover, the global scheme introduced in this paper (cutting-plane generation through CP) can probably be adapted on many other problems.

Appendix

Proof of Proposition 2. In both cases (i) and (ii), $PC1$ and $PC2$ are valid since they satisfy the conditions of Lemma 1. Let (α, β, γ) satisfy the conditions of the proposition. Let $\delta_\alpha = A - \alpha$, $\delta_\beta = B - A + \delta_\alpha - \beta$, and $\delta_\gamma = C - A + \delta_\alpha - \gamma$. Then δ_α, δ_β, and δ_γ are nonnegative and satisfy $\delta_\alpha - \delta_\beta - \delta_\gamma \leq A + D - B - C$.

(i) Suppose that $A + D \geq B + C$ and let $\bar{S} \in \mathcal{S}(P)$.

(i1) if $\bar{S}$ is such that $\bar{x}_{ij} + \bar{x}_{jl} \leq 1$, then if inequality $PC1$ holds for $\bar{S}$,

$$\begin{aligned}\alpha + \beta\bar{x}_{ij} + \gamma\bar{x}_{jl} &= A + (B-A)\bar{x}_{ij} + (C-A)\bar{x}_{jl} \\ &\quad + \delta_\alpha(\bar{x}_{ij} + \bar{x}_{jl} - 1) - \delta_\beta\bar{x}_{ij} - \delta_\gamma\bar{x}_{jl} \\ &\leq A + (B-A)\bar{x}_{ij} + (C-A)\bar{x}_{jl}.\end{aligned}$$

(i2) Otherwise, if $\bar{S}$ is such that $\bar{x}_{ij} + \bar{x}_{jl} > 1$, then if inequality $PC2$ holds for $\bar{S}$,

$$\begin{aligned}&\alpha + \beta\bar{x}_{ij} + \gamma\bar{x}_{jl} \\ &\quad = (B + C - D) + (D - C)\bar{x}_{ij} + (D - B)\bar{x}_{jl} \\ &\qquad + (\delta_\alpha - \delta_\beta - \delta_\gamma - (A + D - B - C))(\bar{x}_{ij} + \bar{x}_{jl} - 1) \\ &\qquad + \delta_\beta(\bar{x}_{jl} - 1) + \delta_\gamma(\bar{x}_{ij} - 1) \\ &\quad \leq (B + C - D) + (D - C)\bar{x}_{ij} + (D - B)\bar{x}_{jl}.\end{aligned}$$

Hence, if $A + D \geq B + C$, then any solution satisfying both inequalities $PC1$ and $PC2$, also satisfies any inequality $C8(\alpha, \beta, \gamma)$ satisfying the conditions of the proposition.

(ii) It can be symmetrically demonstrated that if $A + D \leq B + C$, then any valid inequality $PC(\alpha, \beta, \gamma)$ is dominated by either $PC1$ or by $PC2$. □

References

Alvarez-Valdés, R., J. M. Tamarit. 1993. The project scheduling polyhedron: Dimension, facets and lifting theorems. *Eur. J. Oper. Res.* **67** 204–220.

Applegate, D., W. Cook. 1991. A computational study of job-shop scheduling. *ORSA J. Comput.* **3** 149–156.

Balas, E. 1970. Project scheduling with resource constraints. E. M. L. Beale, ed. *Appl. Math. Programming Tech.* The English Universities Press, London, U.K., 187–200.

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

Brucker, P., A. Drexl, R. Möhring, K. Neumann, E. Pesch. 1999. Resource-constrained project scheduling problem: Notation, classification, models and methods. *Eur. J. Oper. Res.* **112** 3–41.

Brucker, P., S. Knust. 2000. A linear programming and constraint propagation-based lower bound for the RCPSP. *Eur. J. Oper. Res.* **127** 355–362.

Brucker, P., S. Knust, A. Schoo, O. Thiele. 1998. A branch and bound algorithm for the resource-constrained project scheduling problem. *Eur. J. Oper. Res.* **107** 272–288.

Carlier, J., E. Néron. 2000. A new LP based lower bound for the cumulative scheduling problem. *Eur. J. Oper. Res.* **127** 363–382.

Carlier, J., E. Pinson. 1989. An algorithm for solving the job-shop problem. *Management Sci.* **35** 164–176.

Carlier, J., E. Pinson. 1990. A practical use of Jackson's preemptive schedule for solving the job-shop problem. *Ann. Oper. Res.* **26** 269–287.

Carlier, J., E. Pinson. 1994. Adjustment of heads and tails for the job-shop problem. *Eur. J. Oper. Res.* **78** 146–161.

Caseau, Y., F. Laburthe. 1996. Cumulative scheduling with task intervals. M. Maher, ed. *Proc. Joint Internat. Conf. Sympos. Logic Programming, JCPSLP'96*. MIT Press, Cambridge, MA, 363–377.

Christofides, N., R. Alvarez-Valdés, J. M. Tamarit. 1987. Project scheduling with resource constraints: A branch and bound approach. *Eur. J. Oper. Res.* **29** 262–273.

Demeulemeester, E., W. Herroelen. 1992. A branch-and-bound procedure for the multiple-resource constrained single project scheduling problem. *Management Sci.* **38** 1803–1818.

Demeulemeester, E., W. Herroelen. 1997. New benchmark results for the resource-constrained project scheduling problem. *Management Sci.* **43** 1485–1492.

Dorndorf, U., E. Pesch, T. Phan-Huy. 2000. A branch-and-bound algorithm for the resource constrained project scheduling problem. *Math. Methods Oper. Res.* **52** 413–439.

Dyer, M., L. A. Wolsey. 1990. Formulating the single machine sequencing problem with release dates as mixed integer program. *Discrete Appl. Math.* **26** 255–270.

Harjunkoski, I., V. Jain, I. Grossmann. 2000. Hybrid mixed integer/constraint logic programming strategies for solving scheduling and combinatorial optimization problems. *Comput. Chemical Engrg.* **24** 337–343.

Hooker, J. N. 2000. *Logic-Based Methods for Optimization: Combining Optimization and Constraint Satisfaction*. Wiley, New York.

Klein, R., A. Scholl. 1999. Computing lower bound by destructive improvement: An application to resource-constrained project scheduling. *Eur. J. Oper. Res.* **112** 322–346.

Kolisch, R., A. Sprecher, A. Drexl. 1995. Characterization and generation of a general class of resource-constrained project scheduling problems. *Management Sci.* **41** 1693–1703.

Lopez, P., J. Erschler, P. Esquirol. 1992. Ordonnancement de tâches sous contraintes: une approche énergétique. *Revue Française Automatisme Informatique Rech. Oper. APII* **26** 453–481.

Martin, P., D. B. Shmoys. 1996. A new approach to computing optimal schedules for the job-shop scheduling problem. W. H. Cunningham, S. T. McCormick, M. Queyranne, eds. *Proc. 5th Internat. Conf. Integer Programming Combin. Optim., IPCO'96*. Vancouver, British Columbia, Canada, 389–403.

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

Möhring, R. H., A. Schultz, F. Stork, M. Uetz. 2003. Solving project scheduling problems by minimum cut computations. *Management Sci.* **49** 330–350.

Nuijten, W. 1994. Time and resource constrained scheduling: A constraint satisfaction approach. Ph.D. thesis, University of Technology, Eindhoven, The Netherlands.

Pritsker, A., L. Watters, P. Wolfe. 1969. Multi-project scheduling with limited resources: A zero-one programming approach. *Management Sci.* **16** 93–108.

Radermacher, F. 1985. Scheduling of project networks. *Ann. Oper. Res.* **4** 227–252.

Sankaran, J. K., D. L. Bricker, S-H. Juang. 1999. A strong fractional cutting-plane algorithm for resource-constrained project scheduling. *Internat. J. Indust. Engrg.: Appl. Practice* **6** 99–111.

Sprecher, A. 2000. Scheduling resource-constrained projects competitively at modest memory requirements. *Management Sci.* **46** 710–723.