Chapter 1

LOWER BOUNDS FOR RESOURCE CONSTRAINED PROJECT SCHEDULING PROBLEM

Recent advances

Emmanuel Néron
LI, Université François-Rabelais de Tours
Polytech'Tours, 64 av. Jean Portalis
37200 Tours France
emmanuel.neron@univ-tours.fr

Christian Artigues
LIA - CERI
Université d'Avignon et des Pays de Vaucluse
339 chemin des Meinajariès, BP1228
84911 Avignon Cedex 9 France
Centre de recherche sur les transports
Université de Montral
C.P. 6128, succursale Centre-ville
Montréal, QC H3C 3J7 CANADA
christian.artigues@lia.univ-avignon.fr

Philippe Baptiste
CNRS LIX,
Ecole Polytechnique
91128 Palaiseau, France
Philippe.Baptiste@polytechnique.fr

Jacques Carlier
CNRS HeuDiaSyC,
Université de Technologie de Compiègne
Centre de recherches de Royallieu
60200 Compiègne, France
jacques.carlier@utc.fr

Jean Damay
CNRS LIMOS,
Université Blaise Pascal
Complexe scientifique des Céseaux
63177 Aubière Cedex, France.
damay@isima.fr

Sophie Demassey
Centre de Recherche sur les Transports,
Université de Montréal
C.P. 6128, succursale Centre-ville
Montréal, QC H3C 3J7, Canada
sophie.demassey@lia.univ-avignon.fr

Philippe Laborie
Ilog France
9, rue de Verdun BP 85,
94253 Gentilly Cedex, France
laborie@ilog.fr

Abstract We review the most recent lower bounds for the makespan minimization variant of the Resource Constrained Project Scheduling Problem. Lower bounds are either based on straight relaxations of the problems (e.g., single machine, parallel machine relaxations) or on constraint programming and/or linear programming formulations of the problem.

Keywords: Resource Constrained Project Scheduling Problem, lower bounds, constraint programming, linear programming.

Introduction: bounding RCPSP

In the Resource Constrained Project Scheduling Problem (RCPSP), non-preemptive activities requiring renewable resources subject to precedence and resource constraints have to be scheduled to minimize makespan. Our aim is to review recent advances in lower bounding techniques for this fundamental problem.

For a description of exact or heuristic resolution methods, we refer to the recent general state-of-the art surveys on the RCPSP, e.g., (Brucker et al., 1999) and (Demeulemeester and Herroelen, 2002).

Most of the bounds described in this paper are "destructive" bounds (Klein and Scholl, 1999). The mechanism of such bounds is rather simple: a trial value D is fixed for the makespan. If we are able to prove that the corresponding problem has no feasible solution, then $D+1$ is a valid lower bound. A binary search is performed on the trial value D to find the largest value for which we can prove that no feasible solution exists. Of course the key component of a destructive bound relies on the way we can prove that there is no solution.

This chapter is organized as follows:

- Section 1 is dedicated to classical relaxations of the RCPSP to single resource problems, they are based on the single machine problem, the identical parallel machine problem and the Cumulative Scheduling Problem. Bounds presented in this section may not be the most efficient ones but they are fast to compute.

- Section 2 presents constraint propagation techniques used to strengthen the problem. These techniques are useful especially if they are used as preprocessing for more complex lower bounds such as the ones based on some linear programming formulation.

- In Section 3 we present more sophisticated lower bounds that take into account simultaneously the precedence constraints and several resources. Most of these bounds are efficient but highly time consuming.

We do not develop in this chapter the description of the classical lower bounds based on longest paths computations in the project network and their extensions. We refer to (Demeulemeester and Herroelen, 2002) for a complete description of these bounds.

Notation used in this chapter are recalled in Table 1.1.

Table 1.1. Notation

Notation	Definition
Activity related data	
$\mathcal{A}$	set of activities
n	number of activities: $n = \lvert\mathcal{A}\rvert$
$G(\mathcal{A}, E)$	activity-on-node graph used to model classical precedence relationship
p_i	processing time of activity i
Resource related Data	
$\mathcal{R}$	set of renewable resources
m	number of resources: $\lvert\mathcal{R}\rvert$
$r_{i,k}$	number of units of resource k required by activity i
R_k	capacity of resource k
Other Notation	
ES_i	earliest starting time of activity i (release date)
LF_i	latest finishing time of activity i (deadline)
EF_i	earliest finishing time of activity i ($ES_i + p_i$)
LS_i	latest starting time of activity i ($LF_i - p_i$)
S_i	starting time of activity i
C_i	completion time of activity i
D	trial value used to compute destructive lower bound
q_i	tail of activity i: $q_i = D - LF_i$

1. Single resource problems

In this section, we present classical methods for bounding single resource problems that arise as relaxations of the RCPSP. Basically these new problems are obtained either by relaxing the precedence relations to time-windows for activities and/or by relaxing partially the resource constraints. Namely these problems are (1) the single machine problem, (2) the identical parallel machine problem and (3) the Cumulative Scheduling Problem (CuSP). All these problems involve release dates and tails (or deadlines). For these three relaxations, we first describe how they can be get from an initial RCPSP instance, and then we describe lower bounds and time-bound adjustments methods that can be used in turn to get lower bounds for the initial RCPSP.

1.1 Disjunctive based bounds

In the single machine problem, a set of activities has to be processed on a single machine. Each activity,i has a release date ES_i, a processing time p_i, a tail (q_i) or a deadline LF_i. This problem is known to be $\mathcal{NP}$-Hard in the strong sense. It plays a central role for solving the RCPSP as (1) we can derive release dates and tails of activities from the precedence constraints and (2) we can compute sets of activities that, due to machine or to precedence constraints, cannot be processed in parallel and can thus be "put"on a fictive single machine.

In this part we present both a method to build such single machine problems from an initial RCPSP and classical techniques either to compute lower bounds of the makespan ($\max_i(C_i + q_i)$, where C_i denotes the completion time of the activity i) or to derive satisfiability tests, and time-windows adjustments of activities.

1.1.1 Maximum clique computation. Consider the decision variant of the RCPSP. The aim of this section is to build a set of single machines (a machine is a resource with unit capacity), on which some activities have to be processed.

Such redundant machines are useful: on each redundant single machine, edge-finding constraint propagation can be applied (see Section 1.1.3) and thus, the time-windows of activities can be tightened.

To generate redundant single machines, we look for sets of activities that are known not to overlap in any feasible solution. Note that two activities i and j never overlap in time (1) if there is a precedence constraint between i and j or (2) if there is a resource such that the total amount of capacity required by i and j on the considered resource exceeds its capacity. In the following, two activities meeting conditions (1) and/or (2) are said to be "compatible". Any set of activities in which all activities are pairwise compatible is a candidate redundant machine.

We associate a binary variable $\Upsilon_i \in \{0, 1\}$ with each activity i (Υ_i equals 1 when i belongs to the single machine under construction, 0 otherwise). A vector Υ corresponds to a valid redundant machine if for all activities i, j that are not compatible, $\Upsilon_i + \Upsilon_j \leq 1$.

Since the edge-finding constraint propagation algorithm is costly in terms of CPU time, very few redundant machines can be generated. Hence, we have to heuristically select some of them. Our intuition is that "good" redundant machines are heavily loaded. So, we try to find a vector Υ that maximizes $\sum p_i \cdot \Upsilon_i$. The resulting problem is a MIP with n variables and at most n^2 constraints (much less in practice). In (Baptiste and Le Pape, 2000), a greedy heuristic is used to build a solution to a similar MIP. Initial experiments have shown that, in terms of final reduction of time-windows, it is much better to solve the MIP to optimality.

More precisely, we build one global redundant machine according to the above MIP and one redundant machine per initial RCPSP resource. For each cumulative resource, a redundant machine is created in which all the activities requiring more than half of the resource are put (such activities never overlap in time). We then try to add some extra activities on this redundant machine. To do so, the MIP is modified by replacing variable corresponding to activity h, i.e., Υ_h the variable corresponding to activity that is already put on the redundant machine by is replaced by 1. Then a "reduced" MIP, that is solved to optimality, is get.

1.1.2 One machine problem lower bound.

Several lower bounds have been proposed for such one machine problem, i.e., with release dates and tails or deadlines. The commonly used lower bound of the C_{max} for the single machine problem is

$$max_{(h,l)\in\mathcal{A}^2} ES_h + q_l + \sum_{i:ES_i \geq ES_h, q_i \geq q_l} p_i.$$

This maximum can be computed in $O(n \log n)$ steps thanks to Jackson Preemptive Schedule, the preemptive schedule associated with the Largest Tail First dispatching rule.

1.1.3 Edge-finding and time-bound adjustments.

Now, we consider the decision variant of the single machine problem as defined earlier. So, we have time-windows $[ES_i, LF_i]$ (release date / deadline) in which activities have to be processed.

Edge-Finding and time-bound adjustments (Carlier and Pinson, 1989, Applegate and Cook, 1991) consist of deducing that some activities from a given set Ω must, can, or cannot, be executed first (or last) in Ω. Such deductions lead to new ordering relations ("edges" in the graph representing the possible orderings of activities) and new time bounds, i.e., strengthened release dates and deadlines. The edge-finding algorithm is one of the most well known OR algorithm integrated in CP (Baptiste et al., 1999). It is a very efficient global constraint propagation algorithm for disjunctive non-preemptive scheduling.

In the following, ES_Ω denotes the smallest release date among the activities in Ω. Similarly, LF_Ω is the largest deadline among activities in Ω. Finally, let p_Ω be the sum of the processing times of the activities in Ω. Let $i \ll j$ ($i \gg j$) means that i is executed before (after) j and $i \ll \Omega$ ($i \gg \Omega$) means that i is executed before (after) all the activities in

Ω. Once again, variants exist (see Baptiste et al., 1999 for a review) but the following rules capture the "essence" of the Edge-Finding bounding technique:

$$\begin{aligned}
\forall\Omega, \forall i \notin \Omega, [LF_{\Omega\cup\{i\}} - ES_\Omega < p_\Omega + p_i] &\Rightarrow [i \ll \Omega] \\
\forall\Omega, \forall i \notin \Omega, [LF_\Omega - ES_{\Omega\cup\{i\}} < p_\Omega + p_i] &\Rightarrow [i \gg \Omega] \\
\forall\Omega, \forall i \notin \Omega, [i \ll \Omega] \Rightarrow [LF_i = \min(LF_i, \min_{\emptyset\neq\Omega'\subseteq\Omega}(LF_{\Omega'} - p_{\Omega'}))] \\
\forall\Omega, \forall i \notin \Omega, [i \gg \Omega] \Rightarrow [ES_i = \max(ES_i, \max_{\emptyset\neq\Omega'\subseteq\Omega}(ES_{\Omega'} + p_{\Omega'}))]
\end{aligned}$$

If n activities require the resource, there are a priori $O(n \cdot 2^n)$ pairs (i, Ω) to consider. An algorithm that performs all the time-bound adjustments in $O(n^2)$ is presented in (Carlier and Pinson, 1990). Another variant of the Edge-Finding technique is presented in (Carlier and Pinson, 1994). It runs in $O(n \log n)$ but requires much more complex data structures.

1.2 Cumulative problem based lower bounds

One simple way to extend single machine based lower bounds for the RCPSP is to consider that more than one activity can be in process at a time point. Then several lower bounds based on either identical parallel machine problem or Cumulative Scheduling Problem have been proposed in the literature. Some of these bounds are presented in this section.

1.2.1 Deducing identical parallel machine problems.

In the identical parallel machine problem, a set $\mathcal{A}$ of activities has to be processed on π identical parallel machines. Each activity, $i \in \mathcal{A}$, has a release date ES_i, a processing time p_i and a tail q_i or a deadline LF_i. These problems are known to be $\mathcal{NP}$-Hard in the strong sense. Preemptive or semi-preemptive (see Section 1.2.2) relaxations can be solved efficiently and then used to compute lower bounds for the RCPSP. Thus building relevant identical parallel machine problem, considering that release dates and tails of activities are deduced from the precedence constraints of the initial RCPSP, is useful for bounding RCPSP.

To build such π-machine problems, let us consider a resource k and a set $\mathcal{J} \subseteq \mathcal{A}$ of activities of the initial RCPSP such that $|\mathcal{J}| = \pi + 1$ and $\sum_{i\in\mathcal{J}} r_{i,k} > R_k$, then at most π activities of $\mathcal{J}$ can be in process at the same time. Thus $\mathcal{J}$ defines a π-machine problem, and the optimal makespan of this identical parallel machine problem is a lower bound of the makespan of the initial RCPSP.

In (Carlier and Latapie, 1991) π-machine problems are built such that:

- any activity satisfying $\left\lceil \frac{(\pi+1)r_{i,k}}{R_k} \right\rceil - 1 \geq \eta_i$, ($\eta_i \in \mathbb{N}$), is replaced by η_i identical activities in $\mathcal{J}$,
- activities such that $\sum_{i=1}^{\pi} r_{[i],k} > R_k$, (where $r_{[i],k}$ the $i-th$ smallest resource requirement of the activities of $\mathcal{J}$) are added to $\mathcal{J}$.

1.2.2 Bounding identical parallel machine problem. Here we consider lower bounds and satisfiability tests for the decision variant of the identical parallel machine problem with release dates and tails or deadlines. Satisfiability tests are used to prove that there exists no solution in which activities are processed within their time-windows $[ES_i, LF_i]$, without violating the machine constraint. Then they can be used to compute destructive lower bounds for the initial RCPSP instance.

Preemptive and semi-preemptive relaxations for identical parallel machine problem. In this section, we focus on two satisfiability tests for identical parallel machine decision problem: the first one is based on the preemptive relaxation, that can be solved in polynomial time, the second one uses the notion of mandatory parts of activities into the classical max-flow formulation. These methods are presented through an example.

$i \in \mathcal{A}, \pi = 2$	1	2	3
ES_i	23	0	0
p_i	14	45	74
LF_i	37	46	89

Table 1.2. Instance of identical parallel machine problem

It is well known that preemptive relaxation of the identical parallel machine problem is polynomially solvable and that an optimal solution can be found using a max-flow formulation (Horn, 1974): there exists a flow equal to $\sum_{i\in\mathcal{A}} p_i$, in graph G if and only if there exists a preemptive feasible schedule. We refer to (Brucker, 2002) for a formal description of the graph construction.

The figure 1.1 presents the graph associated with the instance presented in figure 1.2. The computation of this satisfiability test can be done in $O(n^3)$ time.

In a recent paper (Haouari and Gharbi, 2003) propose to build *semi-preemptive schedule.* It is based on a technical result to enforce some parts of an activity to be processed into a given time-interval. They proved that, if an activity $i \in \mathcal{A}$ verifies $LF_i - ES_i < 2 \cdot p_i$, then one

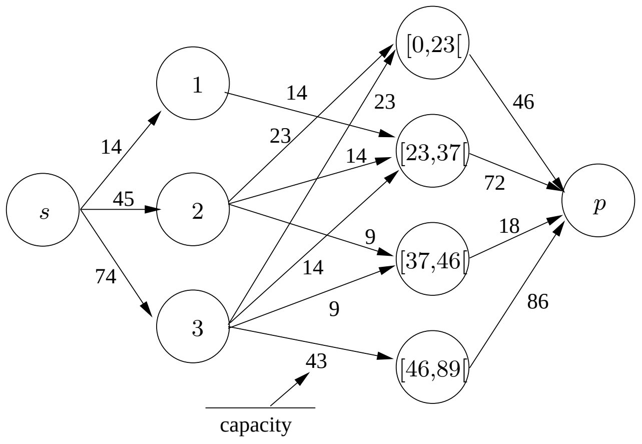


Figure 1.1. Graph for preemptive relaxation of π-machine problem

machine processes activity i during $[LF_i - p_i, ES_i + p_i]$. This constraint can be simply taken into account by adding minimum flow constraints on arcs between those activities and corresponding time-intervals, once relevant time-points $(LF_i - p_i, ES_i + p_i, \forall i \in \mathcal{A}$ s.t. $LF_i - ES_i < 2 \cdot p_i)$ have been added. Such a graph is presented below. If it does not exist a flow equal to $\sum_{i \in \mathcal{A}} p_i$ in graph G, then there does not exist a non-preemptive feasible schedule. The computation of this satisfiability test is still in $O(n^3)$ time, due to the fact that the number of nodes is still in $O(n)$. Such a graph corresponding to our previous example is presented in Figure 1.2.

Finally, (Tercinet et al., 2004) prove that the semi-preemptive relaxation can be replaced by mixing energetic reasoning and preemptive relaxation. The approach is quite similar to the one proposed by Haouari and Gharbi but works of activities for time-interval (see 1.2.4, for formal description), that are the mandatory parts of the activities that must be processed during this time-interval, are used as minimum capacities on the edges between activities and corresponding time-intervals. Notice that the authors also prove that this approach may be better than the separate use of preemptive relaxation and energetic reasoning based satisfiability tests.

Jackson Pseudo Preemptive Schedule (JPPS). (Carlier and Pinson, 1998) present a lower bound for the identical parallel machine problem, called Jackson Pseudo Preemptive Schedule ($JPPS$), (see example of Figure 1.3, for the instance with 2 machines given in the table

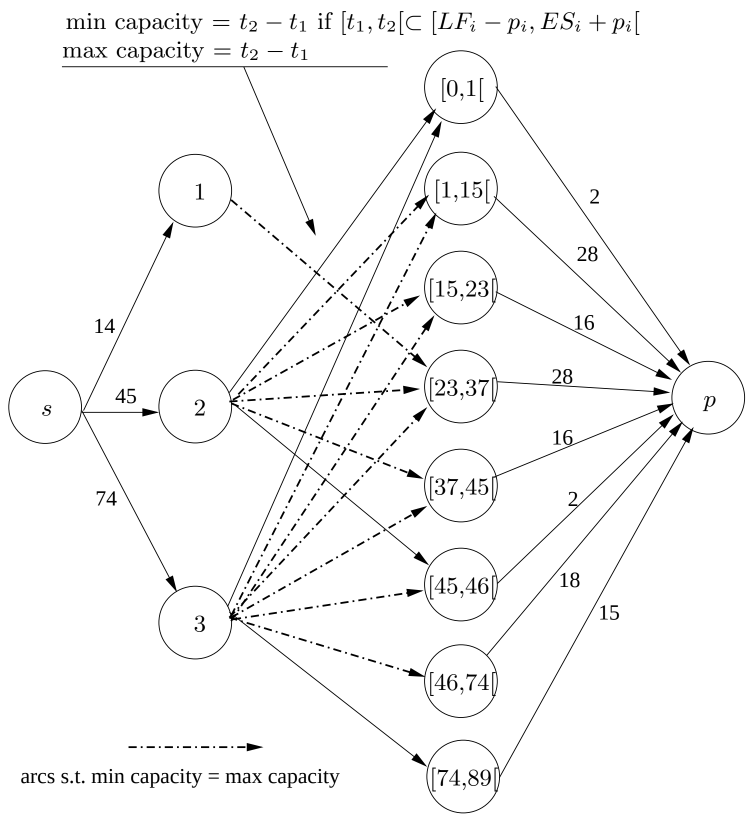

min capacity $= t_2 - t_1$ if $[t_1, t_2[\subset [LF_i - p_i, ES_i + p_i[$
max capacity $= t_2 - t_1$
s
1
2
3
p
14
45
74
[0,1[
[1,15[
[15,23[
[23,37[
[37,45[
[45,46[
[46,74[
[74,89[
2
28
16
28
16
2
18
15
arcs s.t. min capacity = max capacity

Figure 1.2. Graph for semi-preemptive relaxation of π-machine problem

below), in which the preemption of any activity is also allowed, and in which we assume that a machine can be shared by several activities (see machine 1 on $[4;5]$) and that an activity can be processed on several machines at a time (see activity 3 on $[3;4]$).

$i \in \mathcal{A}$	1	2	3	4	5
ES_i	0	1	2	3	3
p_i	3	2	3	1	1
q_i	4	4	1	0	0

Note that if deadlines rather than tails are associated with activities, we can rely on a destructive bound (see Introduction). Furthermore, (Carlier and Pinson, 2004) propose adjustments of release dates for this problem, that will be not retranscribed in this chapter.
To build $JPPS$, a list scheduling algorithm is used whose priority dis-

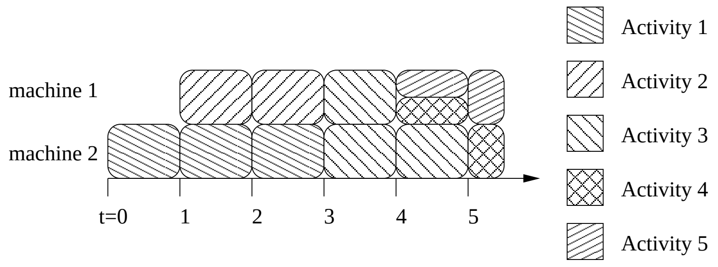


Figure 1.3. Example of JPPS for a 2-machine problem

patching rule at time t is the dynamic complete tail $c_i(t) = q_i + \xi_i(t)$, where $\xi_i(t)$ is the remaining processing time of activity i at time t in the current schedule. They divide the available activities in two classes: the *partially* available activities, whose part executed on $[ES_i; t]$ has been as large as possible ($\xi_i(t) = p_i - (t - ES_i)$), so that they can be processed at a rate $\alpha_i(t) \leq 1$, and the *totally* available activities, for which $\xi_i(t) > p_i - (t - ES_i)$, so that they can be processed at a rate $\alpha_i(t) \leq \pi$. Note that in this schedule we must respect at any time t, $\xi_i(t) \geq p_i - (t - ES_i)$.
The computation of the schedule blocks and of the next decision time are now presented. At time t, the algorithm schedules first the available activities with maximal complete tail at a maximal rate consistent with their status (partially or totally available). The events that can modify the current schedule block are one of the following types:

- a not in-process activity simply becomes available,
- an in-process activity is completed,
- a not in-process activity gets a higher priority than an in-process activity,
- a totally available activity, which is processed at a rate greater than 1, becomes partially available,
- a partially available activity, which is processed at a rate lower than 1, becomes totally available.

Several properties of JPPS are presented in (Carlier and Pinson, 1998) and (Carlier and Pinson, 2004). The makespan of JPPS can be computed in $O(n\ log\ n + n\ \pi\ log\ \pi)$ ($O(n^2 + n\ \pi^2)$, for building JPPS). So that it allows its intensive use in an enumerative process. JPPS does not systematically match the optimal preemptive solution. Then, the

McNaughton schedule issued from JPPS where the activities have a rate lower than or equal to 1 at any time t, is not a lower bound of the problem.
(Carlier and Pinson, 1998) propose a simple adaptation of JPPS for the CuSP (see Section 1.2.3 for a formal description), without additional computational effort, that confirms the interest of JPPS for computing lower bounds for the RCPSP.

1.2.3 Deducing redundant Cumulative Scheduling Problem. This section is dedicated to techniques used to build redundant Cumulative Scheduling Problem (CuSP) from an initial RCPSP. These methods are based on the notion of Linear Lower Bound and Redundant Resources. Lower bound for CuSP is presented in the next section. The method that we present is based on a sub-problem where release dates and deadlines are not taken into account: we focus on specific linear forms that are lower bounds of the makespan when exactly part κ_i of the activity i has to be processed. These linear forms are used to compute Cumulative Scheduling Problem, that can be used to compute efficient lower bounds.

The CuSP is made up of a set $\mathcal{A}$ of activities that has to be processed on a single renewable cumulative resource of capacity R. Each activity, $i \in \mathcal{A}$, has a release date ES_i, a processing time p_i and a deadline LF_i, and it requires r_i units of the resource to be processed. These problems are known to be $\mathcal{NP}$-Hard in the strong sense. Getting relevant CuSP, considering that release dates and tails of activities are deduced from the precedence constraints of the initial RCPSP, is useful for bounding RCPSP: CuSP can be seen as an extension of identical parallel machine problem where activities need more than one machine to be processed.

Linear Lower Bound definition. A Linear Lower Bound (LLB) is based on the relaxation of the RCPSP instance to one resource Cumulative Scheduling Problem (CuSP). A CuSP can be obtained from a RCPSP and a trial value D by ignoring all the resources but one, and by relaxing the precedence constraints to release dates and deadlines of activities, computed according to precedence relationship.

Definition 1.1. *The linear form: $(p_1, \ldots, p_n) \rightarrow \sum_{i \in \mathcal{A}} \lambda_i \cdot p_i$ is a Linear Lower Bound if, for any κ_i $(0 \leq \kappa_i \leq p_i)$, $\sum \lambda_i \cdot \kappa_i$ is a lower bound of the makespan when exactly part κ_i of the activity i has to be processed. (Carlier and Néron, 2003).*

We present here some of the classical lower bounds that can be expressed as Linear Lower Bound.

The basic bound. $(1/R)\sum_{i\in\mathcal{A}} r_i \cdot p_i$ is a LLB of the makespan of the CuSP. This basic bound is very useful due to energetic reasoning (see 1.2.4) that can be applied with it for getting adjustments of release dates and deadlines. We will see in the next paragraph, that any LLB is the basic bound of the redundant resource it is associated with, which explains the interest of redundant resources.

The critical path. Let $\Gamma = \{i_1, i_2, \ldots, i_r\}$ a path in the conjunctive graph of the initial RCPSP, that is: i_1 precedes i_2, i_2 precedes i_3 ..., etc. $\sum_{i\in\Gamma} p_i$, is a LLB. Moreover, if the path is optimal, it is equal to the value of the critical path.

The bound of Mingozzi et al.. Let us recall the linear programming technique of Mingozzi et al. (Mingozzi et al., 1998) in the case of a CuSP. Let J be a subset of activities that can be processed simultaneously, i.e., $\sum_{i\in J} r_i \leq R$. At first we associate with J the $\{0,1\}$ column vector defined by $\psi_J(j) = 1$ if $j \in J$. The bound of Mingozzi et al. is obtained by solving the linear program below where P is the vector of the processing times and τ the vector of the τ_J, (τ_J represents the duration in the schedule during which J is the subset of activities processed). Ψ is the matrix whose columns are the vector ψ_J

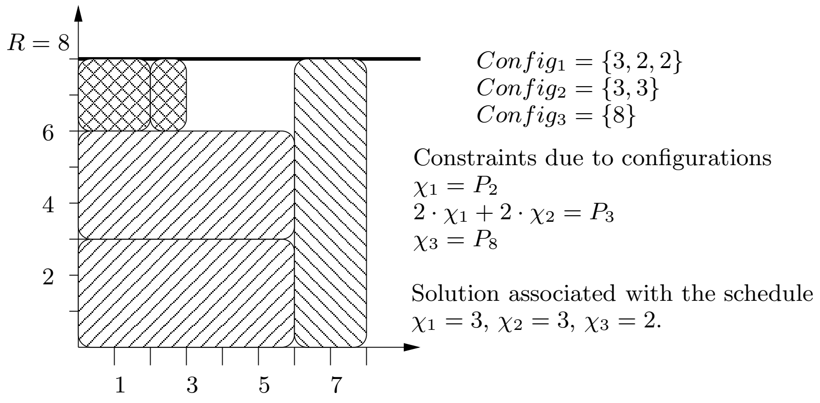
$$\min \sum_J \tau_J,\ \Psi \cdot \tau \geq P \wedge \tau \geq 0.$$

This bound can be improved by taking into account release dates and deadlines for defining J (Brucker and Knust, 2000). Indeed, this bound corresponds to the optimal makespan when preemption is allowed. The dual of the previous linear program is :

$$\max\ \upsilon^t \cdot P,\ \upsilon \cdot \Psi \leq 1 \wedge \upsilon \geq 0.$$

Let υ satisfying $\upsilon \cdot \Psi \leq 1$, $\upsilon \geq 0$, then $\upsilon^T \cdot P$ is a LLB. Moreover when $\upsilon = \upsilon^*$ (the optimal solution of the dual), it is equal to the Mingozzi bound. The corresponding LLB is a very strong one, although it is very difficult to compute it due to the large size of the linear program. An interesting property is that the polyhedra of the dual is independent of P. This property is used to get the Multiple Elastic Preemptive Bound.

The multiple elastic preemptive bound (Carlier and Néron, 2000). In the Multiple Elastic Preemptive (MEP) Relaxation, preemption is allowed, and several parts of the same activity can be processed simultaneously. At first, all the activities having the same resource requirement are merged. Thus, P_r is defined as the sum of the processing times of the activities that require r units of the resource during their processing: $P_r = \sum_{i\in\mathcal{A}/r_i=r} p_i$. $Congif^R = \{c_1, c_2, ..., c_K; c_k \in \mathbb{N}^*\}$ is called a *feasible configuration* for R if: $\sum_{k=1}^{K} c_k \leq R$ and $c_1 \geq c_2 \geq ... \geq c_K$. $\{Config_h^R; h \in \{1,\ldots,H\}\}$ denotes the set of feasible configurations for a resource capacity of R units, and $Config_h^R[r]$ is the number of resource requirements equal to $r \in \{1,\ldots,R\}$ in $Config_h^R$. Note that the number H of feasible configurations depends on R and can be very large. Figure



$R=8$
6
4
2
1
3
5
7
$Config_1 = \{3,2,2\}$
$Config_2 = \{3,3\}$
$Config_3 = \{8\}$
Constraints due to configurations
$\chi_1 = P_2$
$2 \cdot \chi_1 + 2 \cdot \chi_2 = P_3$
$\chi_3 = P_8$
Solution associated with the schedule
$\chi_1 = 3,\ \chi_2 = 3,\ \chi_3 = 2.$

Figure 1.4. A schedule for $R = 8$, $P_2 = 3$, $P_3 = 12$ and $P_8 = 2$

1.4 presents a non optimal schedule illustrating these notions. To simplify the presentation we have only considered four configurations, which are reported on the figure. According to the configuration constraints, it can been stated (Carlier and Néron, 2000), that an optimal solution of the preemptive problem where several parts of the same activity can be processed simultaneously, is given by the solution of (1.1): χ_h is the duration of the configuration $Config_h^R$, and then $\sum_{h=1}^{H} \chi_h$ is the makespan.

$$\min \sum_{h=1}^{H} \chi_h \text{ s.t. } \forall r \in \{1,\ldots,R\} \ \sum_{h=1}^{H} Config_h^R[r] \cdot \chi_h \geq P_r \,;\, \chi_h \geq 0 \quad (1.1)$$

(1.2) is the corresponding dual formulation. Its matrix depends only on the value of R.

$$\max \sum_{r=1}^{R} P_r \cdot o_r \text{ s.t. } \forall h \in \{1,\ldots,H\} \ \sum_{r=1}^{R} Config_h^R[r] \cdot o_r \leq 1 \qquad (1.2)$$

(1.1) and (1.2) are solved in (Carlier and Néron, 2000) for any value of R smaller than or equal to 10, i.e, all corresponding $LLBs$ have been explicitly enumerated by hand. The method for solving this linear program is based on the enumeration of the optimal solutions of the dual linear program for any P.

Indeed, let $S^g_{opt}(R)$, $g \in \{1, \ldots, G\}$ denote the g-th optimal solution of the dual LP for a given value of R. Then its associated cost $MEPB^g(R)$ is a LLB on P_r. Let $\lambda^g_1 \cdot P_1 + \lambda^g_2 \cdot P_2 + ... + \lambda^g_R \cdot P_R,\ g \in \{1, \ldots, G\}$ be these LLB. Due to linear programming properties, $\max_g MEPB^g(R)$ is the optimal value of (1.1) and (1.2), and so a valid lower bound of the makespan. The main advantage of this MEP relaxation is to take into account, the idle time that may occur due to the configurations involved in addition of the global work. Moreover all the $LLBs$ are independent of P and can be tabulated, as explained in (Carlier and Néron, 2000).

Redundant functions and redundant resources. Recently the works dealing with the MEP relaxation, have been generalized to enumerate the sets of valid LLB. This enumeration is based on the notion of *redundant functions* and *maximal redundant functions* (MRFs), and some fundamental properties of these MRF. We now present the notion of Redundant Resources that is used to get relevant one-resource instances from one initial RCPSP instance restricted to one of its resources and one Redundant Function. Let us recall that r_i is the resource requirement of activity i.

Definition 1.2. *A Redundant Function (RF) f is a discrete mapping of $\{0, \ldots, R\} \rightarrow \{0, \ldots, R'\}$ ($R \in \mathbb{N}, R' \in \mathbb{N}$) such that: $i_1 + i_2 + \ldots + i_k \leq R \Rightarrow f(i_1) + f(i_2) + \ldots + f(i_k) \leq R'$.*

RFs and LLBs are strongly linked: let $\sum_{i \in \mathcal{A}} a_i \cdot \rho_i$ be a LLB, and let us assume that all a_i are rational which $a_i = a'_i/R', \forall i \in \mathcal{A}$. Then an associated RF, f can be defined by $f : \{0, \ldots, R\} \rightarrow \{0, \ldots, R'\}, f(r_i) = a'_i$.

The authors present the link between this approach and the dual feasible solution introduced in (Johnson et al., 1974) for the bin-packing problem and used recently in (Fekete and Schepers, 1998), and (Martello and Toth, 1990).

Definition 1.3. *Let f be a redundant function $\{0, \ldots, R\} \rightarrow \{0, \ldots, R'\}$. The associated redundant resource has a capacity of R'. Moreover, activity i needs $f(r_i)$ units of the redundant resource.*

So the basic bound on this redundant resource is now equal to $\sum_{i \in \mathcal{A}} \frac{f(r_i)}{R'} \cdot p_i$. This new bound can be larger than the one computed on the initial resource, which confirms the interest of Redundant Resources.

Figure 1.5 shows how redundant resources are built for two MRFs.

Unfortunately, there are a lot of redundant functions for each couple (R, R'). To restrict the search to the most relevant ones we introduce the notion of *Maximal Redundant Function*: a Maximal Redundant Function

$i \in \mathcal{A}$	1	2	3	4	5	6
r_i	3	5	1	4	2	2
p_i	2	1	1	1	1	2
$f_1 : \{0, \dots, 5\} \to \{0, \dots, 1\}$, $f_1(r_i)$	1	1	0	1	0	0
$f_2 : \{0, \dots, 5\} \to \{0, \dots, 2\}$, $f_2(r_i)$	1	2	0	2	1	1

Table 1.3. An Example of Redundant Resources

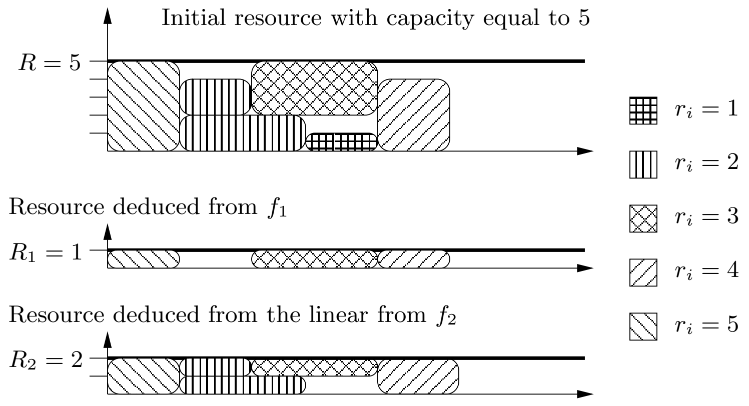
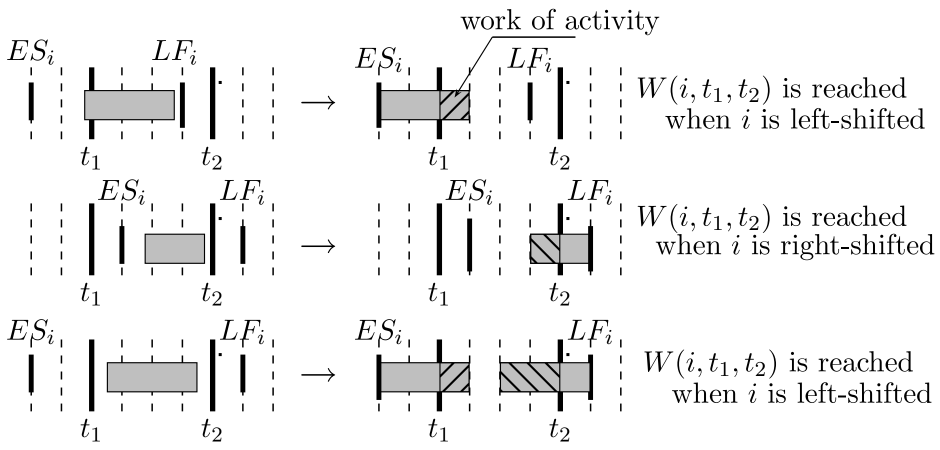


Figure 1.5. Example of feasible schedules for redundant resources, R = 5

(MRF) is a redundant function such that there exists no redundant function f' with $f' > f$, i.e., $\forall i \in \{0, \dots, R\}, f'(i) \geq f(i) \Rightarrow f' = f$.

Both an enumeration scheme to compute all MRF for fixed (R, R') and a linear programming formulation to detect the ones that are dominated, have been proposed. The notion of non-dominated MRFs corresponds to MRFs that may lead to interesting lower bounds. The fact that f is a non-dominated MRF corresponds to the existence of a set of processing times of activities such that the best lower bound for these processing times is given by the redundant resource corresponding to f.

1.2.4 Energetic reasoning for Cumulative Scheduling Problem. At first, we present a brief overview of the energetic reasoning adapted to the CuSP. Energetic reasoning aims to develop satisfiability tests that are used to prove that there exists no solution in which activities are processed within their time-windows $[ES_i, LF_i]$, without violating the machine constraint. It can be used to compute destructive lower bound for the initial RCPSP instance. This approach has been originally developed by (Erschler et al., 1991) and (Lopez et al., 1992)



work of activity
ES_i
LF_i
t_1
t_2
$W(i, t_1, t_2)$ is reached when i is left-shifted
$W(i, t_1, t_2)$ is reached when i is right-shifted
$W(i, t_1, t_2)$ is reached when i is left-shifted

Figure 1.6. Work of an activity through $[t_1, t_2]$.

to solve Cumulative Scheduling Problems. Recently (Baptiste et al., 1999), aim to develop satisfiability tests and time-bound adjustments to ensure that either a given schedule is not feasible or to derive some necessary conditions that every feasible schedule must satisfy. Further details and improvements can be found in (Schwindt, 2005).

Given a time-interval $[t_1, t_2]$, satisfiability tests are based on the computation of the part of the activity i, that must be processed between t_1 and t_2 where, without loss of generality, we assume that $t_1 < t_2$. The mandatory part of activity i is called its *work* in the time-interval $[t_1, t_2]$. To compute it, the activities are either *left-shifted* or *right-shifted* on their time-windows $[ES_i, LF_i]$, i.e., an activity either starts at ES_i or ends at LF_i (see Figure 1.6).

Using Figure 1.6, we formally define the work in interval $[t_1, t_2]$ as follows:

$$\begin{aligned}
W_{left}(i, t_1, t_2) &= r_i \cdot \min(t_2 - t_1, p_i, \max(0, ES_i + p_i - t_1)),\\
W_{right}(i, t_1, t_2) &= r_i \cdot \min(t_2 - t_1, p_i, \max(0, t_2 - LF_i + p_i)),\\
W(i, t_1, t_2) &= \min(W_{right}(t_1, t_2), W_{left}(t_1, t_2))\\
&= r_i \cdot \min(t_2 - t_1, p_i, \max(0, ES_i + p_i - t_1),\\
&\qquad \max(0, t_2 - LF_i + p_i)).
\end{aligned}$$

Property 1.4. Satisfiability Test: *If there exists some time interval* $[t_1, t_2]$, $\sum_{i \in \mathcal{A}} W(i, t_1, t_2) > R \cdot (t_2 - t_1)$ *then the CuSP has no solution.*

Based on the same idea, (Baptiste et al., 1999) propose a method to adjust the time-bounds of activities. We assume without loss of generality that the satisfiability tests have been done for all time-intervals. Let us now introduce $Sl(j, t_1, t_2)$, *the slack of* $j \in \mathcal{A}$ *over* $[t_1, t_2]$. Roughly speaking, this slack corresponds to the available *energy* that can be used to process j. Following the notation previously introduced, the slack can be defined as:

$$Sl(j, t_1, t_2) = \frac{R \cdot (t_2 - t_1) - \sum_{i \in \mathcal{A}, i \neq j} W(i, t_1, t_2)}{r_i}$$

If the right-work of the activity $j \in \mathcal{A}$ is strictly greater than $Sl(j, t_1, t_2)$, the activity cannot be right-shifted, i.e., it cannot end at its deadline. Hence, only a part of j, smaller than or equal to the slack $Sl(j, t_1, t_2)$ can be processed on $[t_1, t_2]$.

Property 1.5. Release Date Adjustments. *For any activity* $j \in \mathcal{A}$*,*

$$[W_{left}(j, t_1, t_2) > Sl(j, t_1, t_2)] \Rightarrow [ES_j \leftarrow \max(ES_j, t_2 - Sl(j, t_1, t_2))]$$

Property 1.6. Deadline Adjustments. *For any activity* $j \in \mathcal{A}$*,*

$$[W_{right}(j, t_1, t_2) > Sl(j, t_1, t_2)] \Rightarrow [LF_j \leftarrow \min(LF_j, t_1 + Sl(j, t_1, t_2))]$$

One crucial point to apply efficiently energetic reasoning is to determine what are the relevant time-intervals on which the above properties have to be applied. It is proved (see Baptiste et al., 1999) that for classical energetic reasoning there are only $O(n^2)$ relevant time-intervals, that are *the breaking points* of the work of activities depending on t_1 and t_2.

$$\begin{aligned}
t_1 &\in \{ES_i, i \in \mathcal{A}\} \cup \{LF_i - p_i, i \in \mathcal{A}\} \cup \{ES_i + p_i, i \in \mathcal{A}\} \\
t_2 &\in \Theta(t_1) \text{ with } \Theta(t) = ES_i + LF_i - t \\
t_2 &\in \{LF_i, i \in \mathcal{A}\} \cup \{ES_i + p_i, i \in \mathcal{A}\} \cup \{LF_i - p_i, i \in \mathcal{A}\} \\
t_1 &\in \Theta(t_2) \text{ with } \Theta(t) = ES_i + LF_i - t
\end{aligned}$$

2. Using constraint propagation to tighten the problem

This part is devoted to constraint programming based approaches that can be applied both to compute destructive lower bounds for the RCPSP and to adjust time-windows of activities. Three families of constraint propagation algorithms can be distinguished for dealing with renewable resources: *timetabling* techniques are based on the computation of an aggregation of the resource demand at every time-point; *edge finding* and *activity intervals* techniques rely on the analysis of the resource demand over time intervals whereas *conjunctive reasoning with temporal constraints* are based on an analysis of the current temporal constraint network. Those three families of techniques are described below more in detail.

2.1 Timetabling

Timetabling relies on the computation for every time-point t of the minimal resource usage at this time-point by the current activities in the schedule (Le Pape, 1994). This aggregated demand profile is maintained during the search and allows to restrict the domains of the start and end times of activities by removing the time-points that would necessarily lead to an over-consumption of the resource. Note that timetabling is also sometimes referred to as resource histograms (Caseau and Laburthe, 1996). Suppose a resource requirement $r_{i,k}$ on a given renewable resource k such that $LS_i < EF_i$, then we know surely that activity i will at least execute over the time interval $[LS_i, EF_i)$. Thus, it will surely require $r_{i,k}$ units of resource k all along this time interval. For each resource k, a curve $Req_k(t)$ is maintained that aggregates all these demands:

$$Req_k(t) = \sum_{i/LS_i \leq t < EF_i} r_{i,k}$$

It is clear that if there exists a time-point t such that $Req_k(t) > R_k$, the current schedule cannot lead to a feasible solution. Thus if D is the trial value (see Introduction) that has been used to compute deadline then $D+1$ is a valid lower bound. Furthermore, if there exists a resource requirement $r_{i,k}$ and a time-point t_0 such that:

$$EF_i \leq t_0 < LF_i \text{ and } \forall t \in [t_0, LF_i), Req_k(t) + r_{i,k} > R_k$$

then, activity i cannot end after time-point t_0. It would otherwise over-consume the resource. Indeed, remember that, as $EF_i \leq t_0$, i is never taken into account in the aggregation on the time interval $[t_0, LF_i)$. Thus, t_0 is a new valid upper bound for the end time of activity i. Similar reasoning can be applied to find new lower bounds on the start time of activities.

2.2 Edge finding and activities intervals

This section presents a variant of the edge-finding adjustments described in section 1.1.3 that works on cumulative machines rather than on one machine problem. Let $\Omega \subset \mathcal{A}$ a subset of activities that require a given resource k of capacity R_k. In addition to the notation introduced in section 1.1.3, let us denote:

- $EF_\Omega = \min_{j\in\Omega} EF_j$, the earliest finishing time of all activities in Ω, in a similar way, let us define $LF_\Omega = \max_{j\in\Omega} LF_j$, $ES_\Omega = \min_{j\in\Omega} ES_j$, and $LS_\Omega = \max_{j\in\Omega} LS_j$

- $w_\Omega = \sum_{j \in \Omega} p_j \cdot r_{j,k}$, the global energy required by Ω (on resource R_k).

The basic idea of edge-finding and activity interval techniques is to ensure that for any subset of activities Ω, resource R_k provides enough energy over the time interval $[ES_\Omega, LF_\Omega)$ to allow the execution of all the activities of Ω, that is: $w_\Omega \leq R_k \cdot (LF_\Omega - ES_\Omega)$. Constraint propagation is usually performed by applying the three following deduction rules, where $i \in \mathcal{A} \setminus \Omega$:

- If $R_k \cdot [LF_\Omega - ES_{\Omega \cup \{i\}}] < w_{\Omega \cup \{i\}}$, then i must finish after all activities in Ω, in particular $EF_i \leftarrow \max(EF_i, EF_\Omega)$.
- If $ES_\Omega < ES_i < EF_\Omega$ and $R_k \cdot [LF_\Omega - ES_\Omega] < w_\Omega + r_{i,k} \cdot [\min(LF_\Omega, EF_i) - ES_\Omega]$, then at least one activity j in Ω must precede i, in particular $ES_i \leftarrow EF_\Omega$.
- If $ES_i < ES_\Omega < EF_i$ and $R_k \cdot [LF_\Omega - ES_\Omega] < w_\Omega + r_{i,k} \cdot [EF_i - ES_\Omega]$, then i must finish after all activities in Ω, in particular $EF_i \leftarrow \max(EF_i, EF_\Omega)$.

These rules allow to update the earliest start or completion times of activities and symmetrical rules allow to update the latest start and completion times. Edge finding (Nuijten, 1994) and activity intervals propagation (Caseau and Laburthe, 1996) are very similar techniques that mainly differ in the way the propagation rules are triggered: edge-finding algorithms are global algorithms that perform all updates on a given resource whereas activity interval approaches are performed incrementally as soon as the time bound of an activity changes.

The propagation of edge-finding and activity interval can be strengthened by considering other time intervals than the one related with the time-bounds of activities as described in 1.2.4.

2.3 Distances between activities and shaving

(Brucker and Knust, 2000) and recently (Demassey et al., 2005) take advantage of the deductions performed by constraint propagation to preprocess linear programs, by fixing variables and strengthening constraints. In this paragraph B is the distance matrix between all couple of activities, and $b_{i,j}$ denotes the minimal distance between i and j (*i.e.* $S_j - S_i \geq b_{i,j}$)

The CP algorithm includes the classical filtering techniques described such as edge-finding, immediate selection (both are described in the previous subsection) and symmetric triples rules, but is run on an alternative CSP formulation based on sequencing variables: for each couple of activities (i, j), variable $X_{i,j}$ denotes the difference between the starting times S_i and S_j. Such a variable corresponds to the *distance* between activities i and j in the activity-on-node graph. Its domain is an interval $[b_{i,j}, -b_{j,i}]$. By ensuring *bound consistency*, constraint propagation allows to increase values $b_{i,j}$ but also to identify new *disjunctions* $i - j$ which are couples of mutually incompatible activities (i.e. forbidden sets of size 2, $\mathcal{F}_2$).

Additionally, Demassey et al. perform global filtering with a suited *shaving* technique, which follows the general principle of consistency enforcing techniques based upon probing: a new constraint c is temporarily added and constraint propagation is performed. If it leads to an infeasibility, then the opposite constraint $\rceil c$ is consistent with the problem. In this implementation, the validity of the three following constraints is tested, for each pair of activities $\{i, j\}$: $i \rightarrow j$ (j follows i), $j \rightarrow i$ (i follows j) and $i \parallel j$ (i and j are both executed in parallel during at least one time period). Shaving aims to prove if such sequencings are infeasible or necessary. To achieve this goal, it uses intermediate results that may be very helpful to deduce informations on the problem. For example, after fixing constraint $i \rightarrow j$, constraint propagation computes the minimal distance matrix $B^{i \rightarrow j}$ and a set $\mathcal{F}_2^{i \rightarrow j}$ of disjunctions that can be identified among all the feasible schedules such that i precedes j. Some of these informations can easily be exploited to derive valid linear inequalities.

The approach of (Demassey et al., 2005) which is described in section 3.2.1, is successfully applied to two well-known integer linear formulations for the RCPSP.

2.4 Conjunctive reasoning with temporal constraints

The propagation algorithms described above only reason on the time bounds of activities (ES_i, LS_i, EF_i, LF_i) and do not directly take into account the precedence constraints that may exist between activities. Some constraint propagation algorithms have been recently proposed that exploit the current temporal constraint network and proved to be very efficient especially when used in conjunction with a branching scheme that solves the scheduling problem by adding precedence constraints (see section 3.1).

These algorithms require that a temporal network representing the relations between the time-points (start and end) of all activities using the point algebra of (Vilain and Kautz, 1986) is maintained during the search. We denote $\{\emptyset, \prec, \preceq, =, \succ, \succeq, \neq, ?\}$ the set of qualitative relations between time points. The temporal network is in charge of maintaining the transitive closure of those relations. If S_i and C_i respectively denote the start and end time-point of activity i, the initial set of relations consist of the precedences $S_i \prec C_i$ for each activity i and $C_i \preceq S_j$ for each precedence constraint $(i, j) \in E$. During the search additional precedence relation can be added as decisions or as the result of constraint propagation.

The *energy precedence* propagation (Laborie, 2003) for an activity i on a resource k ensures that for each subset Ω of predecessor activities of activity i the resource provides enough energy to execute all activities in Ω between ES_Ω and S_i. More formally, it performs the following deduction rule:

$$\forall \Omega \subset \{j \in \mathcal{A}, C_j \preceq S_i\}, ES_i \leftarrow max(ES_i, ES_\Omega + \lceil w_\Omega / R_k \rceil)$$

The propagation of the energy precedence constraint can be performed for all the activities i, on a resource and for all the subsets Ω with a total worst-case time complexity of $O(n(p + log\ n))$ where n is the number of activities on the resource and p the maximal number of predecessors of a given activity in the temporal network ($p < n$).

On a renewable resource, the *balance constraint* (Laborie, 2003) can be defined as follows. The basic idea of the algorithm is to compute, for each activity i on a resource k, a lower bound on the resource usage at the start time of i (a symmetrical reasoning can be applied to perform some propagation based on a lower bound on the resource usage at the completion time of i). Using the temporal network a lower bound on the resource utilization at time-point $S_i + \epsilon$, i.e., just after the start time of i can be computed assuming that all the activities requiring the resource that do not necessarily overlap S_i will not overlap it:

$$L_k(i) = \sum_{j/(S_j \preceq S_i) \wedge (C_j \succ S_i)} r_{i,k}$$

Given this bound, the balance constraint is able to discover three types of information:

- **Dead ends**. Whenever $L_k(i) > R_k$, the resource will surely be over-consumed just after time-point S_i so the search has reached a dead end.
- **New bounds on time variables**. If $L_k(i) \leq R_k$, $\Delta(i) = R_k - L_K(i)$ represents a slack of capacity that must not be exceeded by all the resource requirements that, currently, do not necessarily overlap S_i but could overlap it. Let $\Pi(i) = \{j/(S_j \preceq S_i) \wedge \neg(C_j \succ$

$S_i)\}$. We suppose the activities $(j_1, \dots, j_u, \dots, j_p)$ in $\Pi(i)$ are ordered by decreasing earliest completion time EF_j. Let v be the index in $\{1, \dots, p\}$ such that:

$$\sum_{u=1}^{v-1} r_{j_u,k} \leq \Delta(x) < \sum_{u=1}^{v} r_{j_u,k}$$

If event S_i occurs at a time-point $S_i < EF_{j_v}$, not enough activity will be able to be completed strictly before S_i in order to ensure the resource is not over-consumed just after S_i as in this case, the consumed quantity will be at least $L_k(i) + \sum_{u=1}^{v} r_{j_u,k} > R_k$. Thus, EF_{j_v} is a valid lower bound of S_i.

- **New precedence relations**. Suppose that there exists activity h in $\Pi(i)$ such that:

  $$\sum_{l \in \Pi(i), C_l \succeq C_h} r_{l,k} > \Delta(x)$$

  Then, if we had $S_i \prec C_h$, we would see that again there is no way to avoid a resource over-consumption as it would consume at least:

  $$L_k(i) + \sum_{l \in \Pi(i), C_l \succeq C_h} r_{l,k} > R_k$$

  Thus, the necessary precedence relation: $C_h \preceq S_i$ can be deduced and added to the current temporal network.

The balance algorithm can be executed for all the activities i with a global worst-case complexity in $O(n^2)$ if the propagation that discovers new precedence relations is not turned on, in $O(n^3)$ for a full propagation. In practice, there are many ways to shortcut this worst case and, in particular, it is noticed that the algorithmic cost of the extra-propagation that discovers new precedence relations is in general negligible.

3. Multi-resource based lower bound

In this Section, we describe lower bounds based on relaxations that take the multi-resource context into account. They are based on several possible representations of the resource constraints. In Section 3.1, the forbidden set representation is discussed and lower-bounds based on this structure are introduced. In Section 3.2, the resource constraints are tackled explicitly by different integer linear programming formulations. The lower-bounds are then derived by LP relaxation of these programs and cooperation between constraint programming and cutting-plane generation. Last, in Section 3.3, linear and constraint programming are used to compute lower bounds on the basis of the feasible configuration representation of the resource constraints.

3.1 Using forbidden sets for bounding RCPSP

3.1.1 Forbidden sets. A subset of activities can be considered as a forbidden set (also called minimal critical set) if the activities could be executed simultaneously and if there exists a resource k such that the sum of resource requirement of these activities for resource k over-consumes the resource (Bartusch et al., 1988). Forbidden sets are a simple generalization to cumulative scheduling of the pairs of activities competing for the same unary resource in disjunctive scheduling. If ϕ is a subset of activities, we denote $r_k(\phi) = \sum_{i \in \phi} r_{i,k}$ the global consumption of resource k by ϕ.

Definition 1.7 (Forbidden set). *A forbidden set on a resource k is a subset of activities $\phi \subseteq \mathcal{A}$ such that:*
1. $R_k < r_k(\phi)$
2. $\forall \varphi \subsetneq \phi, r_k(\varphi) \leq R_k$
3. $\bigwedge_{(i,j) \in \phi \times \phi} S_i \prec C_j$ *is consistent with the current temporal network*

Informally, the different ways to resolve a forbidden set consist in fixing a precedence constraint between any two of its activities.

Definition 1.8 (Resolvers of a forbidden set). *If $\phi \subseteq \mathcal{A}$ is a forbidden set, the resolvers of ϕ consist of the set of temporal constraints $Res(\phi) = \{C_i \preceq S_j : (i,j) \in \phi \times \phi, i \neq j\}$.*

Forbidden sets can be exploited to compute lower bounds on the RCPSP considering a relaxation of the problem or directly be used in a complete search.

3.1.2 Relaxations based on forbidden sets. Starting from the remark that most of the forbidden sets in the hard instances of the PSPLIB benchmark (Kolisch and Sprecher, 1997) are of size 2 or 3, some relaxations are proposed in (Garaix et al., 2005) that only take into account those forbidden sets.

More precisely, when only forbidden sets of size 2 are considered (disjunctive relaxation $P(\mathcal{F}_2)$) and all the other ones are relaxed, the problem can be reformulated using unary resources only (disjunctive resources of capacity 1). Each maximal clique in the graph whose edges represent the forbidden sets corresponds to a unary resource in the relaxation. This relaxed problem can be solved using a complete classical disjunctive search. If the relaxed problem is shown to be unfeasible for a maximal makespan D, then, it means that $D+1$ is a legal lower bound of the original RCPSP.

This relaxation can be made tighter by considering forbidden sets of size 3 (relaxation $P(\mathcal{F}_3)$) and using a resource of capacity 2. Each activity in a forbidden set of size 3 requires 1 unit of the resource. From a modeling perspective, this resource of capacity 2 is represented as two unary resources and each activity in the forbidden set requires one of the two possible unary resources. In practice, as the number of forbidden sets of size 3 is very large, these forbidden sets are used as successive cuts to tighten the relaxation: a first try is performed using the disjunctive relaxation $P(\mathcal{F}_2)$ and in case a solution is found, only the forbidden sets of size 3 that are violated in this solution are added to the relaxed problem formulation. This process is repeated until a time limit is reached.

This approach allows improving several lower-bounds of the KSD instances reported on the PSPLIB web page together with some improvement reported in (Baptiste and Demassey, 2004) on the instances with 60 activities. The approach described in (Garaix et al., 2005) improves 13 lower bounds for the instances with 60 activities and 26 lower bounds for the instances with 90 activities.

3.1.3 Complete search based on forbidden sets. In the approach described in (Laborie, 2005), a complete search tree exploration is performed by selecting at each search node a forbidden set ϕ and branching on its possible resolvers in the children nodes until there is no more forbidden sets. This approach is clearly complete and can be used to compute lower bounds on the RCPSP by performing a complete search to prove the infeasibility of a particular makespan value and then used to compute destructive lower bounds (see Introduction).

As described in (Laborie and Ghallab, 1995), the set of resolvers $Res(\phi)$ of a forbidden set ϕ can be simplified so as to remove those resolvers $\rho \in Res(\phi)$ for which there exists another resolver $\rho' \in Res(\phi)$ such that $\rho \Rightarrow \rho'$ given the current temporal network. Indeed, in such case, the resolver ρ is redundant. In what follows, it is assumed that the set of resolvers of a forbidden set has been simplified.

As all the resolvers consist of temporal constraints of the form $\mathcal{T}_1 \preceq \mathcal{T}_2$ where $\mathcal{T}_1$ and $\mathcal{T}_2$ are two time-points (variable start or completion time of an activity), the estimation of the size of the search space after posting such a precedence constraint is particularly interesting to choose which forbidden set to solve at a given search node. The fraction of the search space that is preserved when adding a precedence constraint is estimated using the complementary of the commitment measure introduced in Laborie, 2003.

Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two time-points with respective lower and upper bound for time value: $[\mathcal{T}_1^{min}, \mathcal{T}_1^{max}]$ and $[\mathcal{T}_2^{min}, \mathcal{T}_2^{max}]$. The size of the search space is estimated by the Cartesian product of the domain of the two variables, that is, the area of the rectangle $[\mathcal{T}_1^{min}, \mathcal{T}_1^{max}], [\mathcal{T}_2^{min}, \mathcal{T}_2^{max}]$. The size of the search space that is preserved when adding the constraint $\mathcal{T}_1 \preceq \mathcal{T}_2$ is the part of that rectangle above the line $\mathcal{T}_1 = \mathcal{T}_2$. The fraction of the search space that is preserved $(preserved(\mathcal{T}_1 \preceq \mathcal{T}_2))$ can thus be estimated as the ratio between those two areas.

If ω is the size of the search space below the current search node, the size of the search space after posting a temporal constraint $\mathcal{T}_1 \preceq \mathcal{T}_2$ can be estimated by $\omega \cdot preserved(\mathcal{T}_1 \preceq \mathcal{T}_2)$. If ϕ is the forbidden set that is selected to be resolved at the current search node, the size of the search space to explore below the current node can thus be estimated as the sum of the sizes of the search space below each child node, that is: $\omega \cdot \sum_{\rho \in Res(\phi)} preserved(\rho)$. Therefore, $preserved(\phi) = \sum_{\rho \in Res(\phi)} preserved(\rho)$ estimates the fraction of the search space that is preserved when choosing ϕ as the next forbidden set to solve. The heuristic proposed in (Laborie, 2005) chooses to resolve next the forbidden set ϕ^* that minimizes $preserved(\phi)$ that is, the one that minimizes the estimation of the size of the explored search space. Once such a forbidden set has been selected, it is simplified and the search explores all of its resolvers $\rho \in Res(\phi^*)$ in the child nodes by decreasing order of $preserved(\rho)$. This order has no effect when the schedule is not feasible as in this case the complete search tree needs to be explored but it helps finding a solution quicker when a solution exists.

The above approach is implemented on top of ILOG SCHEDULER 6.1 using the *timetable*, *edge-finding*, *precedence energy* and *balance* constraints described in section 2. The approach is benchmarked on the KSD instances with 60, 90 and 120 activities (Kolisch and Sprecher, 1997) using the lower and upper bound reported on the PSPLIB web page as of May, 1st, 2005 together with some improvement reported in (Baptiste and Demassey, 2004) on the instances with 60 activities. Within a time-limit of 1800s CPU time, out of the 617 previously open instances, 197 lower-bounds are improved (that is more than 31% of the previously open instances) and 97 instances are closed (that is more than 15% of the previously open instances). Furthermore, for all the 943 previously closed instances but two, the approach finds and proves the optimal solution. On the instance set with 60 activities, the critical path lower bound is improved by 8.57% in average, on the instance set with 90 activities this lower bound is improved by 7.40% in average and on the instance set with 120 activities by 22.97%.

The same approach is used to close with a time-limit of 5s CPU time all the open instances of the open-shop benchmark of (Guéret and Prins, 1999).

3.2 Linear programming relaxations and cutting plane generation

Earliest exact solution methods for the RCPSP were mainly branch-and-bound procedures based on linear programming formulations of the problem. A lower bound is usually obtained by omitting the resource constraints and then by computing the longest path problem in the precedence graph. In order to improve this lower bound, some research has been carried out to solve less drastic linear program relaxations.

Unfortunately, the resource constraints for the RCPSP are not only hard to handle but also hard to model as linear inequalities. For example, the well-known time-indexed linear formulations for the RCPSP mostly contains numerous variables and have poor linear relaxations. In order to enhance such a linear relaxation, some cutting-planes were previously described by (Christofides et al., 1987) and (Sankaran et al., 1999). Recent approaches integrate linear relaxations, cutting plane generation and constraint programming to derive strong lower bounds.

3.2.1 Time-Indexed Linear Formulation. The most encountered integer linear formulation of the RCPSP was first given by Pritsker et al. (Pritsker et al., 1969) and is based on time-indexed 0-1 variables of the type: $y_{j,t} = 1$ if activity j starts at time t and $y_{j,t} = 0$ otherwise.

$$\min \quad \sum_{t=0}^{T} t \cdot y_{(n+1),t} \tag{D0}$$

$$s.t. \quad \sum_{t=0}^{T} y_{j,t} = 1 \qquad \forall\, j \in \mathcal{A} \tag{D1}$$

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

$$\sum_{j \in \mathcal{A}} r_{j,k} \sum_{t_0 = t - p_j + 1}^{t} y_{j,t_0} \leq R_k \qquad \forall\, k \in \mathcal{R}, \forall\, t \in \{0,\dots,T\} \tag{D3}$$

$$y_{j,t} \in \{0,1\} \qquad \forall\, j \in \mathcal{A}, \forall\, t \in \{0,\dots,T\} \tag{D4}$$

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. The size of this formulation is proportional to the value of the time horizon T and then may be very large. Furthermore, the optimal solutions of its linear relaxation are usually very fractional and their values give then weak lower bounds.

Preprocessing. To speed up the resolution of such a program, a good preprocessing is necessary. An efficient constraint propagation algorithm allows for example to drastically reduce its size by fixing numerous variables. Indeed, for each activity j in $\mathcal{A}$, a variable $y_{j,t}$ has to be defined only if t is a possible starting time for j. Hence, variables $y_{j,t}$ can be fixed to 0 for any t lower than the earliest starting time $ES_j = b_{0,j}$ of j, or greater than its latest starting time $LS_j = -b_{j,0}$.

Since $S_j = \sum_{t=ES_j}^{LS_j} t \cdot y_{j,t}$, precedence constraints may also be enhanced by taking into account the known minimal distances:

$$\sum_{t=ES_j}^{LS_j} t \cdot y_{j,t} - \sum_{t=ES_i}^{LS_i} t \cdot y_{i,t} \geq b_{i,j} \quad \forall\, (i,j) \in \mathcal{A}^2$$

Disaggregated precedence constraints. In (Christofides et al., 1987), Christofides et al. introduce a *disaggregated* variant of the precedence constraints. Constraints (D2) can be replaced by:

$$\sum_{t_0=t}^{LS_i} y_{i,t_0} + \sum_{t_0=ES_j}^{t+b_{i,j}-1} y_{j,t_0} \leq 1 \ \forall \ (i,j) \in \mathcal{A}^2 \ \forall t \in \{ES_j - b_{i,j} + 1, \dots, LS_i\} \tag{D2$_d$}$$

Despite of their larger number, these inequalities have two advantages. On one hand, they are together tighter than constraints (D2), giving then enhanced linear relaxation. On the other hand, the linear program obtained by replacing constraints (D2) by (D2$_d$) and by dropping resource constraints (D3) has the unimodularity property: its optimal fractional solutions are then integer.

Clique cuts. Clique cuts are well-known packing inequalities stating that if $\mathcal{C}$ is a maximal set of mutually incompatible activities (clique of disjunctions) then, at any time t, at most one activity of $\mathcal{C}$ is in process. Since additional disjunctions and conjunctions are likely to be detected by constraint programming, these clique cuts are expected to be stronger, after such a preprocessing, than the one used in classical implementations.

Shaving cuts. In this LP model, many informations deduced by shaving may be translated as linear inequalities. For example, the following relation is obviously valid: $S_j - S_i \geq p_i \implies S_l - S_h \geq b_{h,l}^{i\to j}$ ($b_{h,l}^{i\to j}$ is the distance between h and l if $i \to j$ is fixed). Furthermore, it is not dominated if the relative sequencing between activities i and j is yet unknown and if sequencing j after i improves the minimal distance between h and l. As for the precedence constraints (D2) and (D2$_S$), such a relation can be written according to both formalisms, aggregated or disaggregated. In the aggregated way, this can be modeled by the following inequality:

$$(-b_{j,i}-p_i+1)\Big(\sum_{t=0}^{T} t\cdot(y_{l,t}-y_{h,t})-b_{h,l}\Big) \geq \Big(\sum_{t=0}^{T} t\cdot(y_{j,t}-y_{i,t})-p_i+1\Big)\big(b_{h,l}^{i\to j}-b_{h,l}\big).$$

3.2.2 Computational experiments. We illustrate the quality of the lower bounds that can be obtained with the time indexed linear programming formulation by reporting the results of the constraint propagation based cutting plane procedure of (Demassey et al., 2002; Demassey et al., 2005) and the lagrangean relaxation approach proposed by (Möhring et al., 2003) on the 480 instances of (Kolisch and Sprecher,

1997) with 60 activities which is the smaller instance set with still open instances.

As reported by (Möhring et al., 2003) with their experiments on a Sun Ultra 2 with 200 MHz clock pulse and 512 MB of memory, solving the aggregated discrete LP relaxation without CP preprocessing nor cuts takes in average 3 seconds (279 seconds max) and improves on average the critical path lower bound by 5.2%. Adding the clique cuts gives an average deviation of 5.54% with an average CPU time of 6.4 seconds. To speed up the resolution of the linear relaxation (Möhring et al., 2003), following (Christofides et al., 1987), propose to dualize the resource constraints so as to obtain a lagrangian subproblem equivalent to a project scheduling problem with start-time dependent cost. They show that such a problem can be solved in polynomial time by minimum cut computations. Using a standard subgradient optimization to compute the optimal lagrangian multpliers, they reduce the computational time to 1.7 second in average (35 seconds max) without any loss of quality. (Demassey et al., 2002) show the benefit of incorporating constraint programming preprocessing and the above described shaving cuts by obtaining a deviation of 6.73% with, as a counterpart, an important increase of CPU times (45 seconds on average on an Pentium III cloked at 800 MHz). The results are still improved in Demassey et al., 2005 reaching a deviation of 7.72% by computing the lower bound in a destructive way. The CPU times however increase to 168 seconds on average (1963 max).

3.2.3 Continuous-time linear formulation. The classical Balas disjunctive formulation for the job-shop problem (Balas, 1970) is based on variables S_j modeling the starting time of activities j and mainly on disjunction variables $x_{i,j}$, stating if activity j starts or not after the completion of activity i. This formulation was extended to the RCPSP by Alvarez-Valdés and Tamarit (Alvarez-Valdés and Tamarit, 1993) making use of the concept of *forbidden sets* $\mathcal{F}$ (cf. definition 3.1.1):

$$\begin{array}{lllr}
\min & S_{n+1} & & \text{(C0)} \\
s.t. & x_{i,j} = 1 & \forall\ (i,j) \in E & \text{(C1)} \\
 & x_{i,j} + x_{j,i} \leq 1 & \forall\ (i,j) \in \mathcal{A}^2 & \text{(C2)} \\
 & x_{i,k} \geq x_{i,j} + x_{j,k} - 1 & \forall\ (i,j,k) \in \mathcal{A}^3 & \text{(C3)} \\
 & S_j - S_i \geq -M + (p_i + M)x_{i,j} & \forall\ (i,j) \in \mathcal{A}^2 & \text{(C4)} \\
 & \sum_{i,j \in F} x_{i,j} \geq 1 & \forall\ F \in \mathcal{F} & \text{(C5)} \\
 & x_{i,j} \in \{0,1\}, S_i \geq 0 & \forall\ (i,j) \in \mathcal{A}^2 & \text{(C6)}
\end{array}$$

Constraints (C1) give the precedence relations within the project. Constraints (C2) and (C3) ensure that no cycle will occur. Constraints (C4) model implications $x_{i,j} = 1 \Rightarrow S_j \geq S_i + p_i$ where M is a constant large enough to let S_i and S_j unrelated when $x_{i,j} = 0$. The resource constraints (C5) state that in any minimal forbidden set F, at least one sequencing decision must be taken.

Sequencing constraints and cuts. The implementation of this program is not realistic in general because of the possible exponential number of constraints (C5). In (Demassey et al., 2005), the authors tackle a relaxation of this program by dropping integrality constraints (C6) as well as constraints (C5) corresponding to minimal forbidden sets of cardinal strictly greater than 3. Nevertheless, their preprocessing algorithm described above, allow to identify much forbidden sets of size 2 which can directly be linearized as constraints (C5).

The CP formulation used in (Demassey et al., 2005) is close to this LP formulation. Indeed, distances $b_{i,j}$ are directly related to variables $x_{i,j}$:

$$\begin{array}{llll}
b_{i,j} \geq p_i & \Rightarrow & x_{i,j} = 1 & (j \text{ starts after the completion of } i) \\
b_{j,i} \geq 1 - p_i & \Rightarrow & x_{i,j} = 0 & (j \text{ starts before the completion of } i)
\end{array}$$

Such conditions are then useful to fix variables in the linear program.

Some shaving informations can also be efficiently used to derive valid linear inequalities. For instance, condition $b_{h,l}^{i \to j} \geq p_h$ means that l follows h in any feasible schedules such that j follows i. In turn, such a relation may be formulated by the valid linear inequality:

$$x_{h,l} \geq x_{i,j} \qquad \forall (i,j,h,l) \in \mathcal{A}^4 \mid b_{h,l}^{i \to j} \geq p_h.$$

Distance constraints and cuts. Another lack of the continuous-time formulation is the presence of "big-M" values which are well known to provide poor relaxations. Here again, the precomputed minimal distances are good estimations to refine M values, since M can obviously be replaced by $-b_{i,j}$ in inequality (C4).

According that the optimal schedule duration is the length of a path made of arcs (i, j) such that $x_{i,j} = 1$, it is tempting to generate "path cuts" of type:

$$S_l - S_i \geq \alpha + \beta x_{i,j} + \gamma x_{j,l} \quad \forall (i, j, l) \in \mathcal{A}^3$$

where coefficients α, β, γ correspond to default evaluations of the distance $S_l - S_i$ according to the different possible values of x_{ij} and x_{jl}. Such evaluations can easily be deduced from distances b_{il} computed by shaving on the sequencing of (i, j) and (j, l). Demassey et al. described how to obtain the deepest 3- or 4-activity path cuts according to any given evaluations.

Edge-finding cuts. Another kind of linear inequalities is closely related to the edge-finding rules described in section 1.1.3. They are also defined for any cliques $\mathcal{C}$ of activities pairwise in disjunction and aim at updating the starting time of one activity $j \in \mathcal{C}$ with respect to the other activities in the clique. Inequalities of that kind have already been proposed for the job-shop problem by Dyer and Wolsey (Dyer and Wolsey, 1990) and Applegate and Cook (Applegate and Cook, 1991) and can easily be adapted to the RCPSP. Demassey et al. proposed two variants based on the minimal distances in place of the earliest/latest starting times.

$$S_j \geq S_l + \sum_{i \in \mathcal{C} \setminus \{j\}} p_i \cdot x_{i,j} + \sum_{i \in \mathcal{C} \setminus \{l\}} b_{l,i} \cdot x_{i,l} \quad \forall j, l \in \mathcal{C}$$

As an example, the above cut states that the distance between two activities j and l of clique $\mathcal{C}$ is greater than the sum of the durations of the activities in $\mathcal{C}$ which are scheduled between j and l.

3.2.4 Computational experiments. The linear programming relaxation of the continuous time formulation is known to be extremely poor if the cuts are not used. The only experiments using this formulation are reported by (Demassey et al., 2005) on 264 non trivial KSD instances with 30 activities, i.e., on whose the critical path lower bound is not feasible. The lower bound is computed in a constructive way. The complete constraint propagation process, including shaving, yields a lower bound of 3.6% from the optimal solution on average and finds 155 optimal solutions. The cutting plane generation procedure on the continuous time formulation reduces the gap to 3.2% and finds 160 optimal solutions. In comparison, the discrete time formulation without cuts reduces the gap exactly to the same value.

3.3 Feasible configuration and linear programming methods

A subset of $\mathcal{A}$ is said to be *feasible* if neither precedence nor resource constraints are violated.

Definition 1.9. *A subset a of $\mathcal{A}$ is a* feasible subset *if and only if:*

1 Resource constraints: $\sum_{i \in a} r_{i,k} \leq R_k$, $\forall k \in \mathcal{R}$.

2 Precedence constraints: for all pairs $(i, j) \in a \times a$, there is no path in G from i to j or from j to i.

Activities of a feasible subset can be processed simultaneously. As described in Introduction, if we pretend to have an upper bound D of the makespan, constraint propagation techniques can be used to detect the infeasibility of any schedule of makespan lower or equal to D, and also to reduce the set A of all feasible subsets, as these techniques create new disjunctions between activities.

3.3.1 Initial formulation and relaxation. In (Mingozzi et al., 1998) a new exact formulation of the RCPSP is presented. This formulation uses $0-1$ discrete variables $\zeta_{i,t}$ that equal 1 if and only if activity i starts at time t, and $y_{l,t}$ that equal 1 if and only if the activities of the feasible subset a_l are in execution on time period $[t; t+1[$. Thus, precedence and no-preemption constraints can be simply formulated in spite of a very large number of variables (depending on the value D and on the cardinality of A). The resource constraints are included in the calculation of the feasible subsets. This formulation is used to derive new lower bounds.

By evading the non-preemption constraints and partially the precedence constraints (treated as disjunctions), (Mingozzi et al., 1998) also formulate several interesting LP-relaxations of the RCPSP. To do that, they introduce the continuous variable z_l which represents the total amount of time of execution of the activities of a_l in parallel, $z_l = \sum_{t=0}^{D} y_{l,t}$. One of their relaxation called (M) is presented below. Let us first define the subset A_{MAX} of the "undominated feasible subsets" of A such that $a_l \in A_{MAX}$ if and only if $\forall a_{l'} \in A \backslash a_l, a_l \not\subset a_{l'}$. With each set a_l is associated an incidence vector $A^l \in \{0,1\}^n$ ($A^l_i = 1$ iff $i \in a_l$).

$$\begin{aligned}
\min \quad & z(M) = \sum_{a_l \in A_{MAX}} z_l \\
& s.t. \quad \sum_{l \in A_{MAX}} A_i^l z_l \geq p_i & \forall i \in \mathcal{A} \\
& \qquad z_l \geq 0 & \forall a_l \in A_{MAX}
\end{aligned}$$

Due to the still very large number of variables (one for each undominated feasible subset) Mingozzi et al. approximate the optimal value of this LP by computing heuristic solutions of the dual program, that can be transformed into the weighted node packing problem of a non-oriented graph $\tilde{G} = (\mathcal{A}, \tilde{E})$ where $(i,j) \in \tilde{G}$ if and only if a feasible subset exists, containing both activities i and j. It actually consists in finding in $\tilde{G}$ an independent set of activities of maximum total processing time.

3.3.2 Column generation techniques and further improvements. In (Baar et al., 1998) and (Brucker and Knust, 2000), a new lower bound based on a destructive approach is presented (see Introduction). To detect infeasibility of trial value D, they first use constraint propagation techniques, including interval consistency, immediate selection, edge-finding and symetric triples (Brucker et al., 1998). If it is not sufficient to prove infeasibility, they try to prove that the solution of a LP-formulation called (BK) inspired from (M) (through a specific column generation algorithm) does not lead to a feasible schedule with makespan lower or equal to D. Note that the time window $[ES_i, LF_i]$ of each activity $i \in \mathcal{A}$ has been strengthened.
To present this LP-formulation, they first divide the time horizon $[0; D]$ into several contiguous subintervals: let $t_0 < t_1 < ... < t_L$ denote the ordered sequence of all different events ES_i and LF_i. For all $l \in \{1, \ldots, L\}$, γ_l denotes the total number of feasible subsets of $[t_{l-1}, t_l]$ and $a_{j,l}$ $(1 \leq j \leq \gamma_l)$ denotes all feasible subsets of the interval. They also associate with each set $a_{j,l}$ an incidence vector $A^{j,l} \in \{0,1\}^n$ ($A_i^{j,l} = 1$ *iff* $i \in a_{j,l}$). We have one variable $z_{j,l}$ per feasible subset in an interval $[t_{l-1}, t_l]$. It denotes the number of time units where all activities in $a_{j,l}$ are processed simultaneously. Non-negative artificial variables u_l, $l \in \{1, \ldots, L\}$ are also introduced in order to turn the decision problem into an optimization problem. If precedence and non-preemption constraints are relaxed, it is easy to see that the scheduling problem is feasible if and only if the following linear program has the optimal value zero.

$$
\begin{aligned}
\min z(BK) = \quad & \sum_{t=1}^{L} u_l \\
s.t. \quad & \sum_{l=1}^{L} \sum_{j=1}^{\gamma_l} A_i^{j,l} z_{j,l} \geq p_i && \forall i \in \mathcal{A} \\
& \sum_{j=1}^{\gamma_l} z_{j,l} - u_l \leq t_{l+1} - t_l && \forall l \in \{1, \ldots, L\} \\
& z_{j,l} \geq 0 && \forall l \in \{1, \ldots, L\}, \forall j \in \{1, \ldots, \gamma_l\} \\
& u_l \geq 0 && \forall l \in \{1, \ldots, L\}
\end{aligned}
$$

The pricing subproblem consists in scanning iteratively through the intervals $[t_{l-1}, t_l], 1 \leq l \leq L$ and searching for feasible subsets $a_{j,l}$ of executable activities in the current interval such that $\sum_{i=1}^{n} A_i^{jl} \theta_i - \varsigma_l > 0$, where θ_i and ς_l are the dual variables of the n first and the L next constraints of (BK), respectively. This involves the multidimensional knapsack-stable problem, that Brucker and Knust compute with a branch-and-bound procedure. This process generates several improving columns, and the solution of the restricted master problem is computed, before restarting it. If no improving column is found in each interval, then the optimal solution is found.
Baptiste and Demassey introduce in (Baptiste and Demassey, 2004) several cuts to be added to this linear formulation (the pricing subproblem is still the same, although they model it through a mixed integer program solved by CPLEX). As a matter of fact, this interval decomposition of $[0; D]$ allows some reflexions on the duration time an activity will be executed on each interval, because of *energetic*, non-preemptive or precedence reasons.
The "energetic reasoning" (Erschler et al., 1991) (Lopez et al., 1992), described in section 1.2.4, gives the minimum duration time of an activity i on an interval $[t_l; t_{l'}], l, l' \in \{1, \ldots, L\}, l < l'$: it is the minimum of:

1 the length of the interval;

2 the number of time units during which i is processed after time t_l if i is left-shifted, i.e. scheduled as soon as possible;

3 the number of time units during which i is processed before time $t_{l'}$ if i is right-shifted, i.e. scheduled as late as possible.

Then the non-preemption constraints can give an upper bound of the duration of an activity on several non-overlapping time intervals. For example, let i be an activity with $ES_i = 1$, $LF_i = 12$, and $p_i = 5$ and consider two intervals $[1, 4]$ and $[9, 11]$. In a non-preemptive schedule, i cannot overlap with both $[1, 4]$ and $[9, 11]$ since $9 - 4 \geq p_i$. Hence, i is processed during at most 3 time units in $[1, 4] \cup [9, 11]$.

Finally, Baptiste and Demassey use this interval decomposition (a sort of discretization) to deduce some consequences of the precedence constraints between activities. This involves the mid-points $mid_{i,l}$ of the activity i on each interval $[t_{l-1}, t_l]$, that can be related to the $z_{j,l}$ variables, where the feasible subsets $a_{j,l}$ contain activity i. As the global mid-point mid_i of activity i is the weighted average of all the mid-points of i, we have for each $(i, j) \in E$, $mid_j - mid_i \geq \frac{p_j - p_i}{2}$.

These three remarks are modeled as inequalities involving the current variables of (BK). They can be used as cutting planes in the LP-formulation, and will improve the first results of (Brucker and Knust, 2000) which were known as the best lower bounds of the RCPSP. Note first that Baptiste and Demassey perform more constraint propagation: they solve initially a mixed integer linear program where single redundant machines are added. Such a machine contains activities that cannot overlap in time, through precedence or resource constraints. Edge finding and specific branching schemes are designed and applied to this redundant machines model, in order to strengthen even more the disjunctions between activities.

The impact of these single constraint propagation techniques is important, insofar as 12 of the 119 still open instances of (Kolisch and Sprecher, 1997) with 60 activities are closed in an average CPU-time of 1 s, and the solution of the linear relaxation (BK) with this preprocessing step improves 34 other lower bounds. The new cutting planes presented above improve then 21 lower bounds of these instances, closing 4 of them.

3.3.3 Feasible configuration and Redundant Resources.

In (Carlier and Néron, 2003), the authors present a linear programming scheme for computing a lower bound for the RCPSP. It uses a set of K LLBs: $\{LLB_1, \ldots, LLB_K\}$(see section 1.2.3) and a segmentation of the time horizon into successive intervals. These LLBs are those associated with the initial resources, redundant resources and paths in the conjunctive graph. All the LLBs associated with the redundant resources are taken into account. Indeed, if $(i_1, i_2, \ldots, i_r)$ is such a path, the LLB given by $p_{i_1} + p_{i_2} + \ldots + p_{i_r}$ is valid. The LLBs corresponding to the 10 longest paths are used in the tests.

In the linear program below, the time horizon $[0, t_{L+1}]$ is divided into L intervals: $[t_1 = 0, t_2]$, $[t_2, t_3]$, $\ldots$, $[t_L, t_{L+1}]$. t_{L+1} is the makespan of the schedule. $\mu_{i,l}$ is that part of activity i which is processed in $[t_l, t_{l+1}]$:

$$\begin{array}{lll}
\min & t_{L+1} & \\
s.t. & \displaystyle\sum_{l=1}^{L} \mu_{i,l} = p_i & \forall i \in \mathcal{A}, \\
& \mu_{i,l} \leq t_{l+1} - t_l & \forall l \in \{1, \ldots L\}, \forall i \in \mathcal{A}, \\
& LLB_h(\mu_{1,l}, \mu_{2,l}, \ldots \mu_{n,l}) \leq (t_{l+1} - t_l) & \forall l \in \{1, \ldots, L\}, \forall h \in \{1 \ldots, K\} \\
& \mu_{i,l} \geq 0 & \forall l \in \{1, \ldots, L\}, \forall i \in \mathcal{A} \\
& t_1 = 0 \leq t_2 \leq t_3 \leq \ldots \leq t_{L+1} &
\end{array}$$

For these tests, authors have considered a set of time-points: $T = \{ES_i, LF_i, \forall i \in \mathcal{A}\} = \{t_1, t_2, \ldots, t_{L+1}\}$. They assume that T is sorted in a non-decreasing order and that all time-points are different. So we add the two following linear constraints:

$$\forall l \in \{1, \ldots, L\}, \forall i \in \mathcal{A}, \text{ if } LF_i \leq t_l \text{ then } \mu_{i,l} = 0$$
$$\forall l \in \{1, \ldots, L\}, \forall i \in \mathcal{A}, \text{ if } ES_i \geq t_{l+1} \text{ then } \mu_{i,l} = 0$$

This linear program has some similarities with that presented by (Brucker and Knust, 2000), including the segmentation of the time horizon. Brucker and Knust generate subsets of activities that cannot be processed simultaneously because of resource and precedence constraints. Their model appears slightly more efficient than the one presented above for a small number of activities, but its drawback is that it is based on column generation, so its complexity may become exponential. This method, on the other hand, is polynomial and therefore scalable.

Experiments prove the efficiency of this bounding method. It is benchmarked on the 480 KSD instances with 120 activities (Kolisch and Sprecher, 1997) using the lower and upper bound reported on the PSPLIB web page as of October, 1st, 2004.

- the average gap between the lower bound and the best reported one is very small (smaller than 0.34%)
- the average and maximum computation times are still reasonable even for large instances (average time: 14.4s max time: 99.5s). Moreover this time can be decreased if only a subset of LLB that are used to get these results is kept: our bound is scalable.

- the redundant resources are useful for both computing energetic satisfiability tests and energetic time bound adjustments: the gap between the best known lower bound and the destructive energetic lower bound decreases from 4.4 % to 0.75% if the redundant resources are used.

- the number of instances where the bound is worse than the best known bound decreases significantly from 156 to 145, if the linear programming formulation based on MRF is used.

4. Conclusion and further research directions

In this state-of-the-art survey, we have arbitrarily divided the lower bounds proposed in the literature for the RCPSP according to the way the resource constraints are considered.

Another classification has to be underlined. On one hand there are fast computable lower bounds that do not give the best relaxations but that can be included inside branch-and-bound exact methods. These are longest-path based lower bounds, one machine and identical parallel machine relaxations, constraint propagation techniques such as edge-finding and energetic reasoning, the node-packing bound of Mingozzi et al. On the other hand, there are more time-consuming methods that give better relaxations mainly based on the cooperation between linear and constraint programming and that can be applied only at the root node.

An important issue for future research is to close the gap between these two categories. This can be done by developing branch-and-cut or branch-and-price methods. Some interesting alternative recent work have been carried in this direction : lagrangean relaxation which significantly accelerates the LP resolution, Linear programming based on MRF, analysis of the forbidden set structure.

References

Alvarez-Valdés, Ramòn and Tamarit, José M. (1993). The project scheduling polyhedron: dimension, facets and lifting theorems. *European Journal of Operational Research*, 67:204–220.

Applegate, David and Cook, William (1991). A computational study of job-shop scheduling. *ORSA Journal on Computing*, 3(2):149–156.

Baar, Tonius, Brucker, Peter, and Knust, Sigrid (1998). Tabu-search algorithms and lower bounds for the resource-constrained project scheduling problem. In Voss, S., Martello, S., I.Osman, and C.Roucairol, editors, *Meta-heuristics: Advances and Trends in Local Search Paradigms for Optimization*, pages 1–18. Kluwer.

Balas, Egon (1970). Project scheduling with resource constraints. In Beale, E.M.L., editor, *Applications of Mathematical Programming Techniques*. American Elsevier.

Baptiste, Philippe and Demassey, Sophie (2004). Tight LP bounds for resource constrained project scheduling. *OR Spectrum*, 26:251–262.

Baptiste, Philippe and Le Pape, Claude (2000). Constraint propagation and decomposition techniques for highly disjunctive and highly cumulative project scheduling problems. *Constraints*, 5:119–139.

Baptiste, Philippe, Le Pape, Claude, and Nuijten, Wim (1999). Satisfiability tests and time-bound adjustments for cumulative scheduling problems. *Annals of Operations Research*, 92:305–333.

Bartusch, Martin, Möhring, Rolph H., and Radermacher, Franz J. (1988). Scheduling project networks with resource constraints time windows. *Annals of Operations Research*, 16:201–240.

Brucker, Peter (2002). *Scheduling algorithms*. Springer Verlag, Berlin.

Brucker, Peter, Drexl, Andreas, Möhring, Rolf, Neumann, Klaus, and Pesch, Erwin (1999). Resource-constrained project scheduling problem: Notation, classification, models and methods. *European Journal of Operational Research*, 112(1):3–41.

Brucker, Peter and Knust, Sigrid (2000). A linear programming and constraint propagation-based lower bound for the RCPSP. *European Journal of Operational Research*, 127:355–362.

Brucker, Peter, Knust, Sigrid, Schoo, Arno, and Thiele, Olaf (1998). A branch and bound algorithm for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 107:272–288.

Carlier, Jacques and Latapie, Bruno (1991). Une méthode arborescente pour résoudre les problèmes cumulatifs. *RAIRO-Recherche Opérationnelle*, 25(3):311–340.

Carlier, Jacques and Néron, Emmanuel (2000). A new LP based lower bound for the cumulative scheduling problem. *European Journal of Operational Research*, 127(2):363–382.

Carlier, Jacques and Néron, Emmanuel (2003). On linear lower bounds for the resource constrained project scheduling problem. *European Journal of Operational Research*, 149:314–324.

Carlier, Jacques and Pinson, Eric (1989). An algorithm for solving the job-shop problem. *Management Science*, 35:164–176.

Carlier, Jacques and Pinson, Eric (1990). A practical use of Jackson's preemptive schedule for solving the job-shop problem. *Annals of Operations Research*, 26:269–287.

Carlier, Jacques and Pinson, Eric (1994). Adjustment of heads and tails for the job-shop problem. *European Journal of Operational Research*, 78:146–161.

Carlier, Jacques and Pinson, Eric (1998). Jackson's pseudo preemptive schedule for the $pm/r_i, q_i/c_{max}$ scheduling problem. *Annals of Operations Research*, 83:41–48.

Carlier, Jacques and Pinson, Eric (2004). Jackson's pseudo preemptive schedule and cumulative scheduling problems. *Discrete Applied Mathematics*, 145:80–94.

Caseau, Yves and Laburthe, François (1996). Cumulative scheduling with task intervals. In Maher, Michael, editor, *Proceedings of the Joint International Conference and Symposium on Logic Programming, JCPSLP'96*, pages 363–377. The MIT Press, Cambridge, MA.

Christofides, Nicos, Alvarez-Valdés, Ramòn, and Tamarit, José M. (1987). Project scheduling with resource constraints: a branch and bound approach. *European Journal of Operational Research*, 29(3):262–273.

Demassey, Sophie, Artigues, Christian, and Michelon, Philippe (2002). A hybrid constraint propagation-cutting plane algorithm for the RCPSP. In *Proceedings of the 4th International Workshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, CPAIOR'02*, pages 321–331.

Demassey, Sophie, Artigues, Christian, and Michelon, Philippe (2005). Constraint-propagation-based cutting planes: an application to the resource-constrained project-scheduling problem. *INFORMS Journal on Computing*, 17(1).

Demeulemeester, Erik and Herroelen, Willy (2002). *Project scheduling. A research handbook.* Kluwer Academic Publishers.

Dyer, M. E. and Wolsey, L. A. (1990). Formulating the single machine sequencing problem with release dates as a mixed integer program. *Discrete Applied Mathematics*, 26:255–270.

Erschler, Jacques, Lopez, Pierre, and Thuriot, Catherine (1991). Raisonnement temporel sous contraintes de ressources et problèmes d'ordonnancement. *Revue d'Intelligence Artificielle*, 5:7–32.

Fekete, Sandor P. and Schepers, Jörg (1998). New classes of lower bounds for bin packing problems. *Lecture Notes in Computer Science*, 1412:257–270.

Garaix, Thierry, Artigues, Christian, and Demassey, Sophie (2005). Bornes inférieures et supérieures pour le RCPSP. In *Proceedings of 6ième congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision, ROADEF 2005*, pages 219–240, Tours, France.

Guéret, Christelle and Prins, Christian (1999). A new lower bound for the open-shop problem. *Annals of Operations Research*, 92:165–183.

Haouari, Mohamed and Gharbi, Anis (2003). An improved max-flow-based lower bound for minimizing maximum lateness on identical parallel machines. *Operations Research Letters*, 31:49–52.

Horn, W.A. (1974). Some simple scheduling algorithms. *Naval Research Logistic Quarterly*, 21:177–185.

Johnson, David S., Demers, Alan J., Ullman, Jeffrey D., M.R., Michael R. Garey, and Graham, Ronald L. (1974). Worst case performance bounds for simple one-dimensional packing algorithms. *SIAM Journal of Computing*, 3:299–325.

Klein, Robert and Scholl, Andreas (1999). Computing lower bound by destructive improvement: An application to resource-constrained project scheduling. *European Journal of Operational Research*, 112:322–346.

Kolisch, Rainer and Sprecher, Arno (1997). PSPLIB - a project scheduling problem library. *European Journal of Operational Research*, 96:205–216.

Laborie, Philippe (2003). Algorithms for propagation of resource constraints in AI planning and scheduling: Existing approaches and new results. *Artificial Intelligence*, 143:151–188.

Laborie, Philippe (2005). Complete MCS-based search: Application to resource constrained project scheduling. In *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI-05.*

Laborie, Philippe and Ghallab, Malik (1995). Planning with sharable resource constraints. In *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI-95*.

Le Pape, Claude (1994). Implementation of resource constraints in ILOG SCHEDULE: A library for the development of constraint-based scheduling systems. *Intelligent Systems Engineering*, 3(2):55–66.

Lopez, Pierre, Erschler, Jacques, and Esquirol, Patrick (1992). Ordonnancement de tâches sous contraintes: une approche énergétique. *R.A.I.R.O. APII*, 26(5–6):453–481.

Martello, Silvano and Toth, Paolo (1990). Lower bound and reduction procedure for the bin-packing problem. *Discrete Applied Mathematics*, 28:59–70.

Mingozzi, Aristide, Maniezzo, Vittorio, Ricciardelli, Salvatore, and Bianco, Lucio (1998). An exact algorithm for the multiple resource-constrained project scheduling problem based on a new mathematical formulation. *Management Science*, 44:714–729.

Möhring, Rolph H., Schultz, Andreas, Stork, Frederik, and Uetz, Marc (2003). Solving project scheduling problems by minimum cut computations. *Management Science*, 49:330–350.

Nuijten, Wim (1994). *Time and resource constrained scheduling: A constraint satisfaction approach*. PhD thesis, Eindhoven University of Technology.

Pritsker, Alan A., Watters, Laurence J., and Wolfe, Philip M. (1969). Multi-project scheduling with limited resources: a zero-one programming approach. *Management Science*, 16:93–108.

Sankaran, Jarayam K., Bricker, Dennis L., and Juang, Schuw-Hwey (1999). A strong fractionnal cutting-plane algorithm for resource-constrained project scheduling. *International Journal of Industrial Engineering: Applications and Practice*, 6(2):99–111.

Schwindt, Christoph (2005). *Resource Allocation in Project Management*. GOR-Publications.

Tercinet, Fabrice, Lenté, Christophe, and Néron, Emmanuel (2004). Mixed satisfiability tests for multiprocessor scheduling with release dates and deadlines. *Operations Research Letters*, 32(11):326–330.

Vilain, Marc and Kautz, Henry (1986). Constraint propagation algorithms for temporal reasoning. In *Proceedings of Fifth National Conference on Artificial Intelligence*, pages 377–382.