Event-based MILP models for resource-constrained project scheduling problems

Oumar Koné[1,2,3], Christian Artigues[1,2], Pierre Lopez[1,2], Marcel Mongeau[3,4]

[1] CNRS ; LAAS ; 7 avenue du colonel Roche, F-31077 Toulouse, France ;

[2] Université de Toulouse ; UPS, INSA, INP, ISAE ; LAAS ; F-31077 Toulouse, France ;

[3] Université de Toulouse ; UPS, INSA, UT1, UTM ; Institut de Mathématiques de Toulouse ;

[4] CNRS ; Institut de Mathématiques de Toulouse UMR 5219 ; F-31062 Toulouse, France.

e-mails: {okone,artigues,lopez}@laas.fr, mongeau@math.univ-toulouse.fr

Abstract

In this paper we make a comparative study of several mixed integer linear programming (MILP) formulations for resource-constrained project scheduling problems (RCPSPs). First, we present three discrete and continuous time MILP formulations issued from the literature. Second, instead of relying on the traditional discretization of the time horizon, we propose two original MILP formulations for the RCPSP based on the concept of *event* : the *Start/End* formulation and the *On/Off* formulation. These formulations present the advantage of involving fewer variables than the formulations indexed by time. Because the variables of this type of formulations are not function of the time horizon, we have a better capacity to deal with instances of very large scheduling horizon. We also illustrate our contribution with a series of tests on various types of instances with the three MILP formulations issued from the literature together with our two new formulations, and we draw some conclusions on their use.

Keywords: Resource-constrained project scheduling, mixed integer linear programming, project management.

Introduction

In this paper we consider the resource constrained project scheduling problem (RCPSP). A project involves activities, and resources (renewable or non-renewable), generally available in limited quantities. The processing of an activity requires throughout its duration, one or more units of one or more resources. The RCPSP deals with organizing in time the realization of activities, taking into account a number of precedence constraints, and constraints on the use and availability of the resources needed. A schedule is a solution that describes resource allocation over time, and aims at satisfying one or more objectives.

There exist strongly NP-hard problems that can be solved reasonably well in practice. However, the RCPSP belongs to the class of *really* hard optimization problems. To illustrate this statement, instances of the minimum lateness one-machine scheduling problem with release dates involving thousands of jobs can be solved to optimality by Carlier's algorithm [9]. This is unfortunately not the case for the RCPSP for which today's exact methods are still unable to solve problems dealing with more than 60 activities [14]. A way to compare several exact and heuristic methods proposed to solve an NP-hard optimization problem involves comparing the results they obtain in terms of consumed CPU time, memory requirement, and objective-function value on a common set of problem instances. Since the late sixties, a wide variety of benchmark RCPSP instances has been proposed. The best exact methods to date for solving the RCPSP [14, 20] are specialized branch-and-bound methods, taking advantage of the problem structure to solve the RCPSP.

Besides these powerful specific methods, it is of theoretical and practical interest to study the performance of standard Mixed Integer Linear Programming (MILP) for solving the problem. Indeed, MILP solvers are often the only available to practitioners.

Hence, the purpose of this paper is to compare the performance of classical and new MILP formulations.

There exist a large number of MILP formulations for the RCPSP. Among others, we can cite the formulations involving an exponential number of variables such as the *discrete time formulation* of Mingozzi *et al.* [22] which considers that all *feasible sets* of activities (all activities involved in this set can be processed simultaneously) of the problem are given, and the *continuous time* formulation of Alvarez *et al.* [1] that assumes that all the *forbidden sets* (distinct sets whose activities involved in them are not authorized to be processed simultaneously) are known. On the other hand, there also exist some formulations involving a polynomial number of variables, known as *compact*, and other formulations involving a pseudo-polynomial number of variables. We restrict our study to these two latter categories because the purpose of this article is to study the formulations that allow solving problems directly with an MILP solver.

There have been previously theoretical and experimental comparison of MILP formulations of the RCPSP [4, 11, 13, 22, 26] but mainly in terms of the quality of the LP relaxation.

However, when direct solving through a MILP solver is involved, the best results are not necessarily obtained by the strongest MILP formulations. For the RCPSP, one reason for that is that the strongest formulations known to date involve a pseudo-polynomial number of variables, which makes necessary the study of more compact formulations.

This paper is divided into four sections. We briefly describe the RCPSP in the first section. Section 2 presents three MILP formulations of the RCPSP issued from the literature: the basic discrete-time formulation proposed by Pritsker *et al.* [24], the disaggregated discrete-time formula-

tion proposed by Christofides *et al.* [11], and the flow-based continuous-time formulation proposed by Artigues *et al.* [3]. In Section 3, we propose two new MILP formulations of the RCPSP based on the concept of events: the *Start/End* formulation and the *On/Off* model. Then, we perform in Section 4, a series of tests on various instances to assess these new formulations, both in terms of the calculation of the lower bound obtained by their linear relaxation, and in terms of the exact resolution. We conclude this study by drawing conclusions about the tests we carried out together with some tips about how to use these new formulations.

1 Resource-constrained project scheduling

Formally, the RCPSP is a particular combinatorial optimization problem, i.e. it is defined by a solution space $\mathcal{X}$, which is discrete or which can be reduced to a discrete set, and by a subset of feasible solutions $\mathcal{Y} \subseteq \mathcal{X}$ associated with an objective function $f : \mathcal{Y} \to \mathbb{R}$. A combinatorial optimization problem aims at finding a feasible solution $y \in \mathcal{Y}$ such that $f(y)$ is optimized (minimized or maximized). A resource-constrained project scheduling problem (RCPSP) is a combinatorial optimization problem defined by a tuple (V, p, E, R, B, b), where V is a set of *activities*, p is a vector of *durations*, E is a set of *precedence relations*, R is a set of renewable *resources*, B is a vector of *resource availabilities*, and b is a matrix of *demands*.

1.1 Problem description

Let n be the number of activities to be scheduled, and m be the number of available resources. Activities constituting the project are identified by a set $\{0, \ldots, n+1\}$. Activity 0 represents by convention the start of the schedule, and activity $n+1$ represents symmetrically the end of the schedule. The set of *non-dummy* activities is identified by $A = \{1, \ldots, n\}$.

The durations are represented by a vector p of $\mathbb{N}^{n+2}$ whose ith component, p_i, is the duration of activity A_i, with the special values: $p_0 = p_{n+1} = 0$.

The precedence relations are given by a set E of index pairs such that $(i, j) \in E$ means that activity i must precede activity j. We assume that we are given a *precedence activity-on-node* graph $G(V, E)$ whose nodes correspond to activities $V = A \cup \{0, n+1\}$, and arcs correspond to precedence relations. We shall identify in the sequel each activity with the corresponding node of the precedence graph. We assume that G contains no cycle, otherwise the precedence relations are obviously inconsistent. Since precedence is a transitive binary relation, the existence of a path in G from node i to node j means also that activity i must precede activity j. Hence, all precedence graphs having the same transitive closure define the same precedence constraints. Taking into account the preceding remark, we assume that E is such that 0 is a predecessor of all other activities and $n+1$ is a

successor of all other activities.

The renewable resources are formalized by set $R = \{1, \ldots, m\}$.

The availabilities of the resources are represented by a vector B of $\mathbb{N}^m$ such that B_k denotes the availability of resource k. In particular, a resource k such that $B_k = 1$ is called a *unary* or *disjunctive* resource.

The demands of the activities for resources are abstracted by b, an $(n+2) \times m$ integer matrix, such that entry b_{ik} represents the amount of resource k used per time period during the execution of activity i. Note that $b_{0k} = 0$ and $b_{n+1,k} = 0$, for all $k \in R$.

A *schedule* is a point S of $\mathbb{R}^{n+2}$ such that its ith component, S_i, represents the start time of activity i. S_0 is a reference point for the start of the project. Here we assume that $S_0 = 0$. A solution S is said *feasible* if it is compatible with the precedence constraints

$$S_j - S_i \geq p_i \quad \forall (i,j) \in E, \tag{1}$$

and the resource constraints

$$\sum_{i \in A_t} b_{ik} \leq B_k \quad \forall k \in R, \forall t \in H, \tag{2}$$

where $A_t = \{i \in A \mid S_i \leq t < S_i + p_i\}$ represents the set of non-dummy activities in process at time t. $H = \{0, 1, \ldots, T\}$ is the *scheduling horizon*, and T (the length of the scheduling horizon) is some upper bound for the makespan.

The *makespan* of a schedule S is equal to S_{n+1}, the start time of the end activity. The above-defined set A_t and constraints state that an activity cannot be interrupted once it is started. This is referred to as not allowing *preemption*. The RCPSP can then be stated as follows: The RCPSP is the problem of finding a non-preemptive schedule S of minimal makespan S_{n+1} subject to precedence constraints (1) and resource constraints (2).

1.2 Complexity

According to the computational complexity theory [16], the RCPSP is one of the most intractable combinatorial optimization problems. Indeed, the RCPSP belongs to the class of problems that are *NP-hard in the strong sense*. The complexity theory states that an optimization problem is NP-hard in the strong sense if its decision version is *NP-complete* in the strong sense. Garey and Johnson [15] have shown that the decision variant of the RCPSP with a single resource and no precedence constraints, called the resource-constrained scheduling problem, is NP-complete in the strong sense by reduction from the 3-partition problem. NP-hardness can be shown by a simpler observation made by Blazewicz *et al.* [7] yielding even worse negative results [26].

2 MILP formulations for the RCPSP

The oldest work conducted on the exact resolution of RCPSP used Mixed Integer Linear Programming (MILP). There are in the literature several formulations of the RCPSP based on MILP. We concentrate our study on three of these formulations.

2.1 Basic discrete-time formulation (DT)

In 1969, Pritsker *et al.* [24] gave a formulation of the RCPSP containing only one type of binary decision variable, x_{it}, indexed by both activities and time. We call it the basic discrete-time formulation, noted DT. This variable is defined so that $x_{it} = 1$ if activity i starts at time t, and $x_{it} = 0$ otherwise. Thus, this formulation can be written as follows:

$$\min_x \sum_{t \in H} t x_{n+1,t} \tag{3}$$

$$\sum_{t \in H} t x_{jt} \geq \sum_{t \in H} t x_{it} + p_i \qquad \forall (i,j) \in E \tag{4}$$

$$\sum_{i=1}^{n} b_{ik} \sum_{\tau = t - p_i + 1}^{t} x_{i\tau} \leq B_k \qquad \forall t \in H, \forall k \in R \tag{5}$$

$$\sum_{t \in H} x_{it} = 1 \qquad \forall i \in A \cup \{0, n+1\} \tag{6}$$

$$x_{it} \in \{0,1\} \qquad \forall i \in A \cup \{0, n+1\}, \forall t \in H, \tag{7}$$

Since $S_i = \sum_{t \in H} t x_{it}$, for all $i \in A \cup \{0, n+1\}$, we remark that constraints (4) and (5) are simple translations of precedence constraints (1) and resource constraints (2), respectively. Constraints (6) and (7) impose non-preemption of the project activities.

This formulation includes $(n+2)(T+1)$ binary variables, and $|E| + (T+1)m + n + 2$ constraints.

2.2 Disaggregated discrete-time formulation (DDT)

In 1987, Christofides *et al.* [11] proposed a formulation which is very similar to the DT formulation. The two formulations mainly differ in how they formulate the precedence constraints. Indeed, the precedence constraints formulated by Christofides

$$\sum_{\tau = t}^{T} x_{i\tau} + \sum_{\tau = 0}^{t + p_i - 1} x_{j\tau} \leq 1, \quad \forall t \in H, \forall (i,j) \in E, \tag{8}$$

are disaggregated expressions of DT precedence constraints (4). We note it DDT for disaggregated discrete-time formulation. The number of binary variables in these formulations indexed by time increases proportionally with T, the length of the scheduling horizon. Formulation DDT involves

the same number, $(n+2)(T+1)$, of binary variables as DT but DDT requires more constraints ($(T+1)(m+|E|)+n+2$ constraints). Note that since (4) $\Rightarrow$ (8) for $0 \leq x \leq 1$, the LP relaxation of DDT is at least as good as the LP relaxation of DT.

2.3 Flow-based continuous-time formulation (FCT)

Inspired by the work by Balas *et al.* [5], Artigues *et al.* [3] proposed a flow-based continuous-time formulation (noted FCT in the sequel) of the RCPSP using three types of variables. First, the usual starting-time continuous variables S_i, for each activity i. Second, *sequential* variables x_{ij} which are binary and indicate whether activity i is processed before activity j. Finally, continuous *flow* variables f_{ijk} to denote the quantity of resource k that is transferred from activity i (at the end of its processing) to activity j (at the start of its processing). Note that $\tilde{b}_{ik} = b_{ik}$ for all $i \in A$ and $\tilde{b}_{0k} = \tilde{b}_{n+1,k} = B_k$, since activity 0 acts as a resource source while activity $n+1$ acts as a resource sink. Thus, formulation FCT can be written as follows:

$$\min_{S,f,x} S_{n+1} \tag{9}$$

$$x_{ij} + x_{ji} \leq 1, \qquad \forall (i,j) \in (A \cup \{0, n+1\})^2, i < j \tag{10}$$

$$x_{ik} \geq x_{ij} + x_{jk} - 1 \qquad \forall (i,j,k) \in (A \cup \{0, n+1\})^3 \tag{11}$$

$$S_j - S_i \geq -M + (p_i + M)x_{ij} \qquad \forall (i,j) \in (A \cup \{0, n+1\})^2 \tag{12}$$

$$f_{ijk} \leq \min(b_{ik}, b_{jk})x_{ij} \qquad \forall (i,j) \in (A \cup \{0\} \times A \cup \{n+1\}), \forall k \in R \tag{13}$$

$$\sum_{j \in A \cup \{0, n+1\}} f_{ijk} = \tilde{b}_{ik} \qquad \forall i \in A \cup \{0, n+1\}, \forall k \in R \tag{14}$$

$$\sum_{i \in A \cup \{0, n+1\}} f_{ijk} = \tilde{b}_{jk} \qquad \forall i \in A \cup \{0, n+1\}, \forall k \in R \tag{15}$$

$$x_{ij} = 1 \qquad \forall (i,j) \in E \tag{16}$$

$$f_{ijk} \geq 0 \qquad \forall (i,j) \in (A \cup \{0, n+1\})^2, \forall k \in R \tag{17}$$

$$f_{(n+1)0k} = B_k \qquad \forall k \in R \tag{18}$$

$$S_i \geq 0 \qquad \forall i \in A \cup \{0, n+1\} \tag{19}$$

$$x_{ij} \in \{0, 1\} \qquad \forall (i,j) \in (A \cup \{0, n+1\})^2 \tag{20}$$

where M is some large enough constant. M can be set to any valid upper bound of the makespan (e.g. $M = \sum_{i=1}^{n} p_i$). Constraints (10) state that for two distinct activities, either i precedes j, or j precedes i, or i and j are processed in parallel. Constraints (11) express the transitivity of the precedence relations. Constraints (25) are so-called disjunctive contraints linking the start time of i and j w.r.t. variable x_{ij}. The constraint is active when $x_{ij} = 1$ (i precedes j) and, in that case, enforces the precedence relation $S_j \geq S_i + p_i$. If $x_{ij} = 0$, the constraint is always satisfied.

Constraints (26) link flow variables and x_{ij} variables. If i precedes j, the maximum flow sent from i to j is set to $\min\{b_i, b_j\}$ while if i does not precede j the flow must be zero. Constraints (14) and (15) are resource flow conservation constraints. Constraint (18) ensures the conservation of the flow. Constraints (16) set the preexisting precedence constraints.

Applegate and Cook [2] showed in their computational study of the job-shop problem (which can be seen as a particular case of the RCPSP) that this formulation yields poor relaxations, which is due to the big-M constant in constraints (12). However, to solve a problem involving a large time-horizon, FCT can be preferable to DT and DDT. Formulation FCT involves $(n+2)^2$ binary variables, $m(n+2)^2+n+2$ continuous variables, and $n^3+(m+(15/2))n^2+(4m+(35/2))n+5m+13$ constraints.

3 Event-based MILP models for the RCPSP

In contrast to the formulations using the variables indexed by time (like DT and DDT), we propose here two new formulations that use variables indexed by events. This is inspired by work by Grossmann on batch process problems [23] and on the formulation by Dauzère-Pérès and Lasserre for flow-shop problems [12], as well as on the former polyhedral study of machine scheduling by Lasserre and Queyranne [21]. We consider that an event occurs when an activity starts or ends. In any left-shifted schedule for the RCPSP, the start time of an activity is either 0 or coincides with the end time of some other activity. Furthermore, it can be simply shown that the set of left-shifted (or semi-active) schedules is dominant. Consequently, the number of events can be restricted to the number of activities plus one. Let $\mathcal{E} = \{0, 1, \ldots, n\}$ be the index set of the events. In fact, event-based formulations do not involve the use of dummy activities. Consequently, the number of activities is n instead of $n+2$ for all the preceding formulations. Event-based formulations (as well as FCT) have the marginal advantage of being able to deal with instances containing some non-integer activity processing times. More importantly, for instances with long-enough scheduling horizon, event-based models involve fewer variables compared to the models indexed by time. Remark also that the event-based formulations we are introducing in this paper do *not* involve any big-M constant.

3.1 Start/End Event-based formulation (SEE)

The first event-based formulation we present involves two types of binary variable and two types of continuous variable. Variable x_{ie} (respectively y_{ie}) is equal to 1 if activity i starts (respectively ends) at event e. Thus, x variables set the start times and y variables set the finish time of each activity. The continuous variable t_e represents the date of event e. The continuous variable r_{ek}

represents the quantity of resource k required immediately after event e. This yields the Start/End Event-based formulation (noted SEE):

$$\min_{r,t,x,y} t_n \tag{21}$$

$$t_0 = 0 \tag{22}$$

$$t_f \geq t_e + p_i x_{ie} - p_i(1 - y_{if}) \qquad \forall (e,f) \in \mathcal{E}^2, f > e, \forall i \in A \tag{23}$$

$$t_{e+1} \geq t_e \qquad \forall e \in \mathcal{E}, e < n \tag{24}$$

$$\sum_{e \in \mathcal{E}} x_{ie} = 1 \qquad \forall i \in A \tag{25}$$

$$\sum_{e \in \mathcal{E}} y_{ie} = 1 \qquad \forall i \in A \tag{26}$$

$$\sum_{e'=e}^{n} y_{ie'} + \sum_{e'=0}^{e-1} x_{je'} \leq 1 \qquad \forall (i,j) \in E, \forall e \in \mathcal{E} \tag{27}$$

$$r_{0k} = \sum_{i \in A} b_{ik} x_{i0} \qquad \forall k \in K \tag{28}$$

$$r_{ek} = r_{(e-1)k} + \sum_{i \in A} b_{ik} x_{ie} - \sum_{i \in A} b_{ik} y_{ie} \qquad \forall e \in \mathcal{E}, e \geq 1, k \in R \tag{29}$$

$$r_{ek} \leq B_k \qquad \forall e \in \mathcal{E}, k \in R \tag{30}$$

$$x_{ie} \in \{0,1\}, y_{ie} \in \{0,1\} \qquad \forall i \in A \cup \{0, n+1\}, \forall e \in \mathcal{E} \tag{31}$$

$$t_e \geq 0 \qquad \forall e \in \mathcal{E} \tag{32}$$

$$r_{ek} \geq 0 \qquad \forall e \in \mathcal{E}, k \in R. \tag{33}$$

The objective function is given by (21). Constraint (22) stipulates that event 0 starts at time 0. Inequalities (23) ensure that if activity i starts at event e and ends at event f, then $t_f \geq t_e + p_i$. Constraints (24) ensure that if event e precedes event f, then e must start before f. Constraints (25) (respectively (26)) require that a start event (respectively end event) has a single occurrence. Constraints (27) describe the precedence relation between activities. If $i < j$ then i ends at event e or after, j cannot start before event e. Constraints (28) give the total consumption of the activities that start at event 0. Constraints (29) are resource conservation constraints that imply that for each resource k, its consumption immediately after event e is equal to its consumption immediately after the previous event $e-1$, plus the consumption required by the activities that start at event e, minus the consumption required by the activities that end at event e. Constraints (30) limit the consumption of resources at each event to the availability of resources.

Note that variables r_{ek} can all be replaced by their expression in function of variables x_{ie}, starting by (28) and making substitutions with (29). So they are not counted below.

Formulation SEE involves $2n^2+2n$ binary variables, $(n+1)$ continuous variables, and $(1/2)n^3 + n^2 + (3+|E|+m)n + |E| + m + 1$ constraints. Compared to DT and DDT, formulation SEE has a

polynomial number of variables and constraints. Compared to FCT, SEE does not involve big-M constraints but has unfortunately a larger number (about twice more) of binary variables.

3.2 On/Off Event-based formulation (OOE)

The second event-based formulation we introduce in this paper is a variant of SEE that uses only one type of binary variable per event, and one type of continuous variable. Variable z_{ie} is set to 1 if activity i starts at event e or if it still being processed immediately after event e. Thus, z_{ie} remains equal to 1 for the duration of the process activity i. That is why we call this model, the On/Off Event-based formulation (noted OOE). In this model, the number of events is exactly equal to the number of activities n. The continuous variable t_e represents, as in SEE, the date of event e. Moreover, with formulation OOE, the resource constraints are modelled in very simple way. Here is the OOE formulation:

$$\min_{z,t,C_{\max}} C_{\max} \tag{34}$$

$$C_{\max} \geq t_e + (z_{ie} - z_{i(e-1)})p_i \qquad \forall e \in \mathcal{E}, \forall i \in A \tag{35}$$

$$t_0 = 0 \tag{36}$$

$$t_f \geq t_e + ((z_{i-e} - z_{i(e-1)}) - (z_{if} - z_{i(f-1)}) - 1)p_i \qquad \forall (e,f,i) \in \mathcal{E}^2 \times A, f > e \neq 0 \tag{37}$$

$$t_{e+1} \geq t_e \qquad \forall e \neq n-1 \in \mathcal{E} \tag{38}$$

$$\sum_{e'=0}^{e-1} z_{ie'} \geq e(1 - (z_{ie} - z_{i(e-1)})) \qquad \forall e \neq 0 \in \mathcal{E} \tag{39}$$

$$\sum_{e'=e}^{n-1} z_{ie'} \geq e(1 + (z_{ie} - z_{i(e-1)})) \qquad \forall e \neq 0 \in \mathcal{E} \tag{40}$$

$$\sum_{e \in \mathcal{E}} z_{ie} \geq 1 \qquad \forall i \in A \tag{41}$$

$$z_{ie} + \sum_{e'=0}^{e} z_{je'} \leq 1 + (1 - z_{ie})e \qquad \forall e \in \mathcal{E}, \forall (i,j) \in E \tag{42}$$

$$\sum_{i=0}^{n-1} b_{ik} z_{ie} \leq B_k \qquad \forall e \in \mathcal{E}, \forall k \in R \tag{43}$$

$$t_e \geq 0 \qquad \forall e \in \mathcal{E} \tag{44}$$

$$z_{ie} \in \{0,1\} \qquad \forall i \in A, \forall e \in \mathcal{E} \tag{45}$$

Constraint (35) gives the makespan ($C_{max} \geq t_e + p_i$ if i is in process at event $e-1$ but not at event e). Constraints (37) link the binary optimization variables z_{ie} to the continuous optimization variables t_e, and ensures that the processing time of an activity is equal to the processing time of this activity. $t_f \geq t_e + p_i$ if activity i starts immediately after event e and ends at event f. Constraints (39) and (40), called *contiguity constraints*, ensure the adjacency of the events during

which an activity being processed. Constraint (41) ensures that each activity is processed at least once during the project. Constraint (42) is the precedence constraint and constraint (43) is the resource constraint.

Formulation OOE involves n^2 binary variables (twice as less for SEE), $(n+1)$ continuous variables, and $(1/2)n^3 - (1/2)n^2 + (3 + |E| + m)n - 2$ constraints.

3.3 Example

In order to better understand the new SEE and OOE models, let us consider an illustrative instance of the RCPSP. It involves 10 activities and 2 resources. Durations (processing times) and availabilities are displayed in Table 1. Figure 1 shows the Gantt chart of a feasible schedule (solution).

i	1	2	3	4	5	6	7	8	9	10
p_i	7	3	5	5	6	4	5	4	3	7
b_{1i}	0	2	3	3	2	1	1	1	1	3
b_{2i}	2	1	3	2	1	0	3	1	1	1
Successors	3	6,7	4,9	11	1	1	5,8	10	4	9

Table 1: An illustrative RCPSP instance

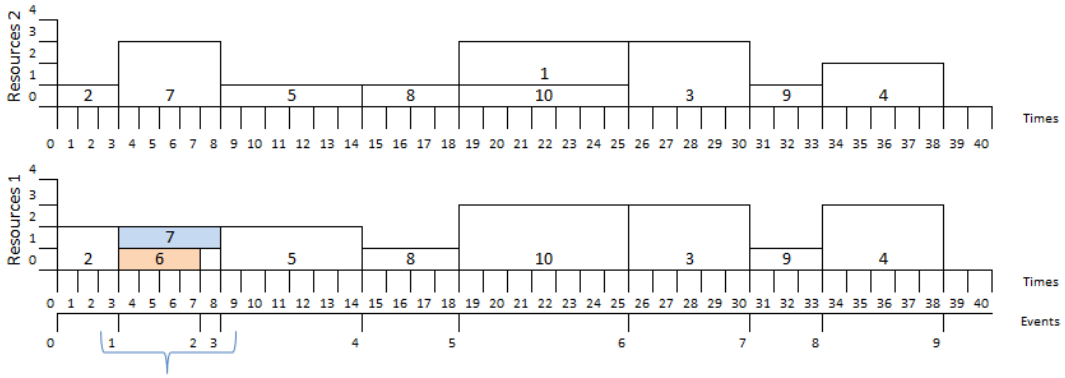

Figure 1: Gantt chart of a feasible solution with discrete time and events

Table 2 displays corresponding values of the optimization variables related to activities 6 and 7. The x_{it}'s are the time-indexed variables used in DT and DDT models (center), while the x_{ie}, y_{ie} and z_{ie} (left-hand side) are the event-indexed variables involved in SEE and OOE models. Finally, the S_i (right-hand side) are the continuous variables used in FCT formulation.

e	0	1	2	3
x_{6e}	0	1	0	0
y_{6e}	0	0	1	0
z_{6e}	0	1	0	0
x_{7e}	0	1	0	0
y_{7e}	0	0	0	1
z_{7e}	0	1	1	0

t	2	3	4	5	6	7	8
x_{6t}	0	1	0	0	0	0	0
x_{7t}	0	1	0	0	0	0	0

e	0	1	2	3
z_{6e}	0	1	0	0
z_{7e}	0	1	1	0

Table 2: Some values of variables for the feasible solution

4 Computational comparison

In this section, we first describe the instances we shall use in our numerical comparisons; second, we characterize them through some instance indicators; and finally, we perform a computational comparison of the five MILP models, first in terms of linear programming relaxation, and then, in terms of integer solving.

4.1 Instances and instance indicators

Among the large number of instances found in the literature, we shall concentrate on the following:

KSD: PSPLIB (from Kolisch *et al.* [19]) is a scheduling library reachable through a web site [25], that contains the most used RCPSP instances. Among these instances, we choose the 480 instances that involve $n = 30$ activities (noted KSD30). The groups of instances have been generated using different values of the NC, RF, and RS indicators (see below).

BL: Baptiste and Le Pape [6] proposed a set of 39 instances, among which 19 comprise $n = 20$ activities, and 20 involve $n = 25$ activities. Each activity requires $m = 3$ resources with a randomly generated demand ranging from 0 to 60% of the total availability. For the 20-activity instances, $|E| = 15$ precedence constraints were randomly generated, while $|E| = 45$ precedence constraints were generated for the 25-activity instances.

Pack: Carlier and Néron [10] proposed a set of 55 instances with a small number ($|E|$) of precedence relations. The number of activities varies from 17 to 35 (with an average of $n = 25$ activities), and there are $m = 3$ resources. Resource availability ranges from 5 to 10. There are two instance categories. In the first one, the activity demand b_k is randomly generated between 0 and B_k ($k = 1, \ldots, m$), which may generate disjunctions. In the second one, any activity demand cannot exceed half of the resource availability, which implies the absence of disjunctions and consequently yields highly cumulative instances.

The instances described in the literature to test the formulations proposed for the RCPSP, are

often being subject of some criticism. In fact for the most used instances, the KSD ones, it appears that the harder instances all have a small RS indicator (see below) implying a high disjunction ratio. This motivated the generation of the BL and Pack instances, both characterized by small disjunction ratios. Regardless of these criticisms, we also note that most of the instances above involve relatively short durations, which could represent an advantage for the MILP formulations indexed by time. Thus, in order to enhance representativeness, we created two new types of instances (noted Pack_d and KSD15_d), which are modified versions of instances Pack and KSD30. These types of instances can typically be found in the process industry.

Hereafter is a more precise description of these new instance sets we are introducing.

Pack_d: These instances are obtained by multiplying by 50 the processing time of a randomly selected part of the activities. Thus, the processing times vary from a few units of time to hundreds of units of time.

KSD15_d: These are instances obtained by modifying KSD30 as follows. The processing time of a randomly selected part of the activities are multiplied by 15, and each instance now only involves 15 activities (instead of 30 for KSD30). Both Pack_d and KSD30_d are publicly available[1].

Since the sixties, indicators have emerged to characterize the RCPSP instances. They can be roughly classified into four categories: precedence-oriented (e.g. OS and NC), time-oriented (e.g. RF), resource-oriented, and hybrid (e.g. RS, #FS, ACUFS) [4]. Let us now recall briefly the established relationships between these indicator values and instance tractability. Table 3 displays for each instance the range of each indicator.

Table 3: Average tractability indicator values for five instance sets

	KSD30	BL	Pack	KSD15_d	Pack_d
$\|V\|$	32	22 - 27	17 - 35	17	17 - 35
$\|R\|$	4	3	2 - 5	4	2 - 5
T	34 - 130	14 - 34	23 - 139	187 - 999	644 - 3694
OS	0.34 - 0.69	0.25 - 0.45	0.13 - 0.48	0.34 - 0.64	0.13 - 0.48
NC	1.5 - 2.13	1.45 - 2	1.5 - 1.72	1.18 - 1.82	1.50 - 1.72
RF	0.25 - 1.0	0.5 - 0.77	1 - 1	0.25 - 1	1
RS	0.14 - 1	0.16 - 0.55	0.08 - 0.53	0.18 - 1	0.08 - 0.48
DR	0.36 - 0.9	0.25 - 0.45	0.19 - 0.94	0.35 - 0.90	0.19 - 0.94
PR	10	5	19	250	1138

- **OS**: *Order strength* is defined as the density of the transitive closure of the precedence graph

[1] See `http://www.laas.fr/~okone/`

$(0 \leq \text{OS} \leq 1)$, where $\text{OS} = 0$ corresponds to a total parallelism whereas $\text{OS} = 1$ means that the activities are totally ordered.

- **NC**: *Network complexity* corresponds to the average number of precedence arcs per activity (assuming E includes no redundant arcs). Globally, the hardness of instances decreases as NC and OS increase.
- **RF**: *Resource factor* is defined as the average number of required resources. It is generally experienced that instance hardness increases as RF increases.
- **RS**: *Resource Strength* defines resource features incorporating also time features. The required CPU time varies in function of RS according to a continuous bell-shaped easy-hard-easy pattern (with instances close to $\text{RS} = 0$ being far harder than the ones close to $\text{RS} = 1$).
- **DR** *Disjunction Ratio* integrates precedence and resource features. It is used to distinguish between cumulative instances (with a low disjunction ratio) and disjunctive instances (with a high disjunction ratio).
- **PR** *Process Range* is simply defined as $PR = \dfrac{\max p_i}{\min p_i}$.

For more details on instance indicators, see [4]. In [4], the Pack instances were shown to be the hardest to solve with state-of-the-art exact and heuristic methods. These instances are characterized by a smaller disjunction ratio and a small number of precedence constraints (as suggested by the range of OS and NC).

In the next subsection, we shall compare the different formulations of the RCPSP on the five instance sets KSD30, BL, Pack, KSD15_d, and Pack_d.

4.2 Results

We perform two series of tests. The first series deals with the calculation of lower bounds obtained through a linear relaxation of each of the five formulations DT, DDT, FCT, SEE, and OOE. The second series tests the exact solving on various instances using each of the five formulations. These tests were carried out on a XEON 5110 biprocessor Dell PC clocked at 1.6Ghz with 4GB RAM, and running Linux FEDORA as operating system. The formulations are coded in C++, in an ILOG-Concert (version 26) environment. The solver used is ILOG-Cplex (version 11). We limit the resolution time of each instance to 500 seconds.

We set the big-M value (involved in formulation FCT) to $\sum_{i=1}^{n} p_i$.

4.2.1 Time window preprocessing

As already mentioned, formulations DT and DDT are highly sensitive to time horizon. To moderate this characteristic, we perform a preprocessing phase aiming at reducing activity time windows by standard precedence and resource constraint propagation. The horizon T is set to an upper bound obtained by the parallel schedule scheme heuristic with the minimum latest finishing time rule [18]. Then starting with $[0, T]$ the operation time windows are reduced by using the constraint propagation algorithms described in [8] until no more adjustment can be detected. Such preprocessing allows a high reduction of the number of binary variables and strengthen the relaxation of formulations DT and DDT [13].

4.2.2 LP relaxations

In this series of tests we compare the lower bounds obtained by linear relaxation of each of the five MILP formulations. The results are displayed in Table 4, where we use the following abbreviations:

Opt. Sol.: Instances for which the relaxation could be solved in 500 seconds.

%: Percentage of instances for which the relaxation could be solved in 500 seconds.

% Gap: Percentage of average deviation from the earliest start time produced by time window preprocessing.

Time: Average CPU time required, in seconds.

By order of importance, the criteria of performance are % Opt. Sol., % Gap, and Time. The time-indexed formulations present better performances on the instances involving relatively short scheduling horizon (KSD30, Pack and BL). On these instances, the best lower bounds are produced by DDT, followed by DT.

These results allow us to say unequivocally that formulation DDT provides better lower bounds on instance sets KSD30 and Pack, followed by DT. The classification of formulations in descending order, in terms of quality of lower bounds, is as follows:

- for instances KSD30, Pack, and BL: DDT>DT>OOE,SEE>FCT.

No conclusion can be drawn concerning instances KSD15_d and Pack_d, because the event-based formulations which solve most of these instances, return only the earliest start time produced by the time window preprocessing (as mentioned before).

4.2.3 Exact solving

The second series of tests involves exact resolutions, i.e. computing optimal solutions for each instance. The results are display in Table 5, where are added the following abbreviations:

Table 4: Linear relaxation results

Instances	Formulations	Opt. Sol.		
		%	% Gap	Time
KSD30	DT	100	0.04	0.04
	DDT	100	0.31	0.78
	FCT	97	0.0	6.94
	SEE	100	0.0	3.01
	OOE	100	0.0	0.39
Pack	DT	82	10.43	0.11
	DDT	82	15.26	1.14
	FCT	82	0.0	4.17
	SEE	82	0.0	1.54
	OOE	82	0.0	0.24
BL	DT	100	5.53	0.05
	DDT	100	13.09	0.12
	FCT	97	0.0	2.25
	SEE	100	0.0	1.09
	OOE	100	0.0	0.17
KSD15_d	DT	71	0.0	0.28
	DDT	15	22.05	0.52
	FCT	100	0.0	0.10
	SEE	100	0.0	0.05
	OOE	100	0.0	0.04
Pack_d	DT	0	0.0	0.00
	DDT	0	0.0	0.00
	FCT	100	0.0	3.79
	SEE	100	0.0	0.41
	OOE	100	0.0	0.15

% Gap: Percentage of average deviation from the optimal (or the best known) objective-function value.

Non-opt. Sol.: Found solution for which optimality is not proven;

Total Sol.: Total of solutions (including both Opt. Sol. and Non-opt. Sol.);

% No Sol.: Percentage of instances for which the LP relaxation could not be solved within 500 seconds.

We can reasonably define a priority order between these criteria of performance as follows: % Opt. Sol., Time, % Non-opt. Sol., and % Gap.

On the instance sets KSD30, Pack, and BL, formulation DDT presents the best results, because it solves to optimality most of the instances. It is followed by DT. Note that DDT solves, in less than 500 seconds, 82% of instances while the branch and bound of Demeulemeester and Herroelen [14] solves all instances to optimality much more quickly. However, although the gap is not closed with specific methods, this relative good result underlines the progress of MILP solvers. On the other hand, on instance sets KSD15_d (involving very large scheduling horizon), formulation FCT yields the best performances. Finally, on instance sets Pack_d (which have a large duration range but also, as mentioned before, are very highly cumulative instances) formulation OOE obtains the best results, although only a small percentage of instances are solved to optimality (18%). For these instances the performance of DDT and DT decreases dramatically. Consequently, solving exactly instances with a large range of durations through MIP is an actual challenge.

To summarize, the classification of formulations, in descending order of quality, is as follows:

- For the instances KSD30, and Pack: DDT>DT>FCT>OOE>SEE ;
- For the instances Pack: DDT>DT>OOE>FCT>SEE ;
- For the instances BL: DDT>DT>OOE>SEE>FCT ;
- For KSD15_d: FCT>OOE>SEE>DT>DDT ;
- For instances Pack_d: OOE>FCT,SEE>DDT,DT.

Conclusions

In this article, we proposed two new MILP formulations for the RCPSP: Start/End Event-based formulation and On/Off Event-based formulation. These formulations have the features of using variables indexed by events (not by time), and they involve limited complexity in terms of number of binary variables. We also compared these new MILP formulations, together to classical ones, both in terms of linear-relaxation lower bounds, and in terms of exact resolution.

Table 5: Exact resolution results

Instances	Formulations	Opt. Sol.		Non-opt. Sol.		Total Sol.	No Sol.
		%	Time	%	% Gap	%	%
KSD30	DT	78	12.76	8	6	86	14
	DDT	82	10.45	9	5	91	9
	FCT	52	33.81	4	2	56	44
	SEE	3	123.62	0	4	3	97
	OOE	24	112.62	9	5	33	67
Pack	DT	64	37.32	9	2	73	27
	DDT	73	61.09	22	127	95	5
	FCT	0	0.00	2	13	2	98
	SEE	0	0.00	0	0	0	100
	OOE	27	20.63	18	127	45	55
BL	DT	100	37.93	0	0	100	0
	DDT	100	13.68	0	0	100	0
	FCT	0	0.00	0	0	0	100
	SEE	0	0.00	8	13	8	92
	OOE	0	0.00	49	0	49	51
KSD15_d	DT	55	6.34	1	0	56	44
	DDT	5	1.65	0	0	5	95
	FCT	95	7.87	4	0	99	1
	SEE	76	10.95	18	1	94	6
	OOE	82	2.96	18	0	100	0
Pack_d	DT	0	0.00	0	0	0	100
	DDT	0	0.00	0	0	0	100
	FCT	4	7.58	0	0	4	96
	SEE	4	215.08	0	0	4	96
	OOE	18	75.58	42	0	60	40

RCPSP problems involving a wide range of processing times are now common in industry but are not represented in classical benchmarks. We proposed in this study new instance sets involving such features, KSD15_d and Pack_d, and we observed the event-based formulations we proposed are very promising for such problems.

Thus, compared on various types of instances, with three other formulations issued from the literature, which two of them (DT and DDT) use variables indexed by time and the last (FCT) uses sequential variables, we obtain that the formulation proposed by Christofides *et al.* (DDT) yields better results for the exact resolution on traditional instances KSD30, BL, and Pack.

This is consistent with the superiority of the formulation in terms of linear relaxation. However, our subsequent experiments show that, when exact solving through a commercial solver is involved, no formulation dominate the other ones (including ours) and that the accurate formulation has to be selected depending on instance characteristics.

Indeed, the formulations based on the events (more particularly On/Off formulation), as well as FCT, have the advantage of solving more easily the instances involving very large scheduling horizons (KSD15_d). This is not the case for the formulations using variables indexed by time. When these instances involving very large scheduling horizons are highly cumulative (Pack_d), On/Off formulation presents better performances compared to all other MILP formulations. We concluded that to solve highly cumulative RCPSP instances involving very large scheduling horizon, our event-based On/Off formulation seems the most appropriate.

Finally, remark that another feature of the event-based formulations we introduced is that they are theoretically able to process instances with non-integer processing times.

References

[1] R. Alvarez-Valdès and J. M. Tamarit, "The project scheduling polyhedron: dimension, facets and lifting theorems", *European Journal of Operational Research*, 67(2): 204–220, 1993.

[2] D. Applegate and W. Cook, "A computational study of job-shop scheduling", *ORSA Journal on computing*, 3(2): 149–156, 1991.

[3] C. Artigues, P. Michelon, and S. Reusser, "Insertion techniques for static and and dynamic resource-constrained project scheduling", *European Journal of Operational Research*, 149(2): 249–267, 2003.

[4] C. Artigues, O. Koné , P. Lopez, M. Mongeau, E. Néron, and D. Rivreau, "Computational Experiments", in C. Artigues, S. Demassey, and E. Néron, (Ed.), *Resource-Constrained*

Project Scheduling Models, algorithms, extensions and applications, ISTE/Wiley, pages 98–102, 2008.

[5] E. Balas, "Project scheduling with resource constraints", in E. M. L. Beale, (Ed.), *Applications of mathematical programming techniques*, pages 187–200, American Elsevier, 1970.

[6] P. Baptiste and C. Le Pape, "Constraint propagation and decomposition techniques for highly disjunctive and highly cumulative project scheduling problems", *Constraints*, 5(1-2): 119–139, 2000.

[7] J. Blazewicz, J. Lenstra, and A. Rinnooy, "Scheduling subject to resource constraints: Classication and complexity", *Discrete Applied Mathematics*, 5(1): 11–24, 1983.

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

[9] J. Carlier, "The one-machine sequencing problem", *European Journal of Operational Research*, 11(1): 42–47, 1982.

[10] J. Carlier and E. Néron, "On linear lower bounds for resource constrained project scheduling problem", *European Journal of Operational Research*, 149: 314–324 ,2003.

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

[12] S. Dauzère-Pérès and J.B. Lasserre, "A new mixed-integer formulation of the flow-shop sequencing Problem", *2nd Workshop on models and algorithms for planning and scheduling problems*, Wernigerode, Allemagne, may 1995.

[13] S. Demassey, C. Artigues, and P. Michelon, "Constraint propagation based cutting planes: an application to the resource-constrained project scheduling problem", *INFORMS Journal on Computing*, 17(1): 52–65, 2005.

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

[15] M. Garey and D. Johnson, "Complexity results for multiprocessor scheduling under resource constraints", *SIAM Journal on Computing*, 4(4): 397–441, 1975.

[16] M. Garey and D. Johnson, *Computers and intractibility. A guide to the theory of NP-Completeness*, W.H. Freeman and Company, New York, 1979.

[17] J. Jozefowska and J. Weglarz, *Perspectives in modern project scheduling*, Springer, 2006.

[18] R. Kolisch, "Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation", *European Journal of Operational Research*, 90(2): 320–333, 1996.

[19] R. Kolisch and A. Sprecher, "PSPLIB - A project scheduling library", *European Journal of Operational Research*, 96(1): 205–216, 1997.

[20] P. Laborie, "Complete MCS-based search: Application to resource constrained project scheduling", *IJCAI*, pages 181–186, 2005.

[21] J.B. Lasserre and M. Queyranne, "Generic scheduling polyhedra and a new mixed-integer formulation for single-machine scheduling, integer programming and combinatorial optimization", *Proceedings of the 2nd International IPCO Conference*, pages 136–149, 1992.

[22] A. Mingozzi, V. Maniezzo, S. Ricciardelli, and L. Bianco, "An exact algorithm for the multiple resource-constrained project scheduling problem based on a new mathematical formulation", *Management Science*, 44(5): 714–729, 1998.

[23] J. M. Pinto and I. E. Grossmann, "A continuous time MILP model for short term scheduling of batch plants with pre-ordering constraints", *Industrial & Engineering Chemistry Research*, 34(9): 3037–3051, 1995.

[24] A. Pritsker, L. Watters, and P. Wolfe , "Multi-project scheduling with limited resources: A zero-one programming approach", *Management Science*, 16: 93–108, 1969.

[25] PSPLIB. `http://129.187.106.231/psplib/`.

[26] M. Uetz, Algorithms for Deterministic and Stochastic Scheduling, PhD thesis, Technische Universität Berlin, 2001.