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 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.

Finally, we illustrate our contribution with a series of tests on various types of instance with the MILP formulations issued from the literature, together with our new formulations. We also compare our results with a recent RCPSP-specific exact method. We show that, in terms of exact solving, no MILP formulation class dominates the other ones and that a state-of-the art specialized (non MILP) method for the RCPSP is even outperformed by MILP on a set of hard instances. Furthermore, on another set of hard "highly cumulative" RCPSP instances with a high processing time range, our On/Off formulation outperforms all the others MILP formulations and obtains results close to the ones of the specialized method.
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 [15]. 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 [15, 35, 25] 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 software available to practitioners and, most of the best lower bounds ever obtained on the KSD instance set [24] were obtained by a hybrid method [14] involving constraint propagation and the MILP formulation of Christofides *et al.* [11]. A branch-and-cut method based on the latter formulation was developed by Zhu *et al.* [39], to solve the multimode RCPSP and yielded very competitive results on benchmark problems. Surprisingly, no computational study on standard benchmark instances is available in the literature for exact solving of the RCPSP through MILP. Hence, the purpose of this paper is to propose new MILP formulations and compare their performance with that of classical MILP formulations and specialized RCPSP methods on standard and new benchmark instance sets.

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.* [27] which considers that all *feasible sets* of activities (all activities involved in such a 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 elements are activities which 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.

Previous theoretical and experimental comparison of MILP formulations of the RCPSP [4, 11, 14, 27, 37] were mainly restricted to evaluating the quality of the Linear Programming (LP) relaxation obtained by the various formulations. However, when direct solving through an MILP solver is involved, the best results are not necessarily obtained by the MILP formulation involving the strongest LP relaxation. For the RCPSP, one reason for that is that the strongest formulations known to date involve a pseudo-polynomial number of variables, such as the formulation of Christofides *et al.* [11] involving time-indexed variables. The KSD instance set is tractable by these formulations since it involves a small processing time range (from 1 to 10). This feature is far from being representative of actual applications, especially in the process industry. For instance, Pinto and Grossmann [29] provide a real example from a plastic compounding plant where the processing times vary from 0.783 to 11.250 days. The discrete-time formulations are not able to tackle this class of problems unless by rounding the processing time or decomposing empirically the time horizon into intervals, which only yields approximate solutions, as the approaches proposed in [19] and [28]. Consequently, the design of more compact formulations is a promising research area. In this spirit, continuous time compact formulations were proposed for the RCPSP or some of its extensions [3, 34, 38]. There are two general classes of continuous time formulation. The first class involves sequencing variables [3, 34], the second one involves event variables [38]. In [38], Zapata *et al.*, with again process engineering applications in background, conclude that, when CPU time is less limited than memory, the event-based formulations they propose to solve a multimode

mutiproject RCPSP is more advantageous than other formulations.

In this paper we propose two new event based formulations for the RCPSP. The first one, the *Start/End* formulation, is a variant of that of Zapata *et al.* [38] involving fewer variables. The second one, the *On/Off* formulation, deeply differs in the way precedence and resource constraints are modeled. We will show that it involves much fewer binary variables and brings drastic improvements with respect to computational experiments.

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.* [30], the disaggregated discrete-time formulation proposed by Christofides *et al.* [11], and the flow-based continuous-time formulation proposed by Artigues *et al.* [3]. In Section 3, we propose the *Start/End* and the *On/Off* formulations. 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 exact solving. We compare these results with those obtained with the standard MILP formulations and the recent specialized exact method of Laborie [25]. We conclude in Section 5 by drawing conclusions about the tests we carried out together with some tips about how to use the new formulations.

1 Resource-constrained project scheduling

Formally, the resource-constrained project scheduling problem (RCPSP) belongs to the set of combinatorial optimization problems, *i.e.,* it is characterized 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). The RCPSP is a particular 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. The 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 $p \in \mathbb{N}^{n+2}$ whose ith component, p_i, is the duration of activity 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 the execution of activity i must precede that of 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 the set $R = \{1, \ldots, m\}$.

The availabilities of the resources are represented by a vector $B \in \mathbb{N}^m$ 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 $S \in \mathbb{R}^{n+2}$ 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. 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, the set $H = \{0, 1, \ldots, T\}$ is the *scheduling horizon*, and T (the length of the scheduling horizon) is some given 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 [18], 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 [17] 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 [37].

1.3 Time windows

As already mentioned, some MILP formulations are highly sensitive to time horizon. To moderate this characteristic, we associate with each activity i an earliest start time ES_i and a latest start time LS_i, which may be calculated independently by preprocessing. More precisely, ES_i corresponds to the date before which activity i will not start, while LS_i is the date before which it should start. Hence, $[ES_i, LS_i]$ represents the time window to start activity i. Note that if each arc (i,j) of G is valuated by p_i, valid activity time windows can be polynomially computed with a longest path algorithms by setting ES_i to the length of the longest in path G from 0 to i, and LS_i to T minus the length of the longest path in G from i to $n+1$. In Section 4.2.1, we present more complex preprocessing techniques yielding tighter time windows.

2 MILP formulations for the RCPSP

The oldest work conducted on exact solving for the 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.* [30] 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. The decision 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:

$$\min \sum_{t=ES_i}^{LS_i} t x_{n+1,t} \tag{3}$$

$$\sum_{t=ES_j}^{LS_j} t x_{jt} \geq \sum_{t=ES_i}^{LS_i} t x_{it} + p_i \qquad \forall (i,j) \in E \tag{4}$$

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

$$\sum_{t=ES_i}^{LS_i} x_{it} = 1 \qquad \forall i \in A \cup \{n+1\} \tag{6}$$

$$x_{00} = 1 \tag{7}$$

$$x_{it} = 0 \qquad \forall i \in A \cup \{n+1\}, t \in H \setminus \{ES_i, LS_i\} \tag{8}$$

$$x_{it} \in \{0,1\} \qquad \forall i \in A \cup \{n+1\}, \forall t \in \{ES_i, LS_i\} \tag{9}$$

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 (9) impose non-preemption of the project activities. Since ES stands for the earliest start time, and LS the latest start time, constraints (8) mean that the start time of any activity occurs between its earliest start time and its latest start time. This formulation involves $\sum_{i=1}^{n+1}(LS_i - ES_i)$ binary variables, and $|E| + (T+1)m + n + 1$ 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

$$\sum_{\tau=t}^{LS_i} x_{i\tau} + \sum_{\tau=ES_j}^{\min(LS_j, t+p_i-1)} x_{j\tau} \leq 1, \quad \forall (i,j) \in E, \forall t \in \{ES_i, LS_i\} \tag{10}$$

are disaggregated expressions of DT precedence constraints (4). We note the corresponding formulation: 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 of binary variables as DT, but DDT requires $m\sum_{i=1}^{n+1}(LS_i - ES_i)$ more constraints. It is well known that since [(6) and (10)] $\implies$ (4), even if variables x_{it} are fractional, the LP relaxation of DDT is tighter than the one of DT. We observe that both DDT and DT require pseudo-polynomial numbers of variables and constraints.

2.3 Flow-based continuous-time formulation (FCT)

The first category of continuous time formulations involve sequencing variables. In [1], a formulation of this category is proposed, involving an exponential number of constraints. Inspired by the work of Balas *et al.* [5], Artigues *et al.* [3] proposed a compact flow-based continuous-time formulation (noted FCT in the sequel) of the RCPSP using three types of variables. Another formulation based on rectangle packing was recently proposed in [34] for a multimode model. However this formulation is only valid for $m \leq 2$.

The formulation of Artigues *et al.* [3] involves three types of decision variables. First, the starting-time continuous variables S_i are defined for each activity i. Second, *sequential* binary variables x_{ij} are required to indicate whether activity i is processed before activity j. Finally, continuous *flow* variables f_{ijk} are introduced 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). We define $\tilde{b}_{ik} := b_{ik}$ for all $i \in A$ and $\tilde{b}_{0k} := \tilde{b}_{n+1,k} := B_k$, meaning that activity 0 acts as a resource source while activity $n+1$ acts as a resource sink. Thus, formulation FCT can be written as follows:

$$\min S_{n+1} \tag{11}$$

$$x_{ij} + x_{ji} \leq 1, \qquad \forall (i,j) \in (A \cup \{0, n+1\})^2, i < j \tag{12}$$

$$x_{ik} \geq x_{ij} + x_{jk} - 1 \qquad \forall (i,j,k) \in (A \cup \{0, n+1\})^3 \tag{13}$$

$$S_j - S_i \geq -M_{ij} + (p_i + M_{ij})x_{ij} \qquad \forall (i,j) \in (A \cup \{0, n+1\})^2 \tag{14}$$

$$f_{ijk} \leq \min(\tilde{b}_{ik}, \tilde{b}_{jk})x_{ij} \qquad \forall (i,j) \in (A \cup \{0\} \times A \cup \{n+1\}), \forall k \in R \tag{15}$$

$$\sum_{j \in A \cup \{0,n+1\}} f_{ijk} = \tilde{b}_{ik} \qquad \forall i \in A \cup \{0, n+1\}, \forall k \in R \tag{16}$$

$$\sum_{i \in A \cup \{0,n+1\}} f_{ijk} = \tilde{b}_{jk} \qquad \forall i \in A \cup \{0, n+1\}, \forall k \in R \tag{17}$$

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

$$x_{ij} = 1 \qquad \forall (i,j) \in TE \tag{19}$$

$$x_{ji} = 0 \qquad \forall (i,j) \neq TE \tag{20}$$

$$f_{ijk} \geq 0 \qquad \forall (i,j) \in (A \cup \{0, n+1\})^2, \forall k \in R \tag{21}$$

$$S_0 = 0 \tag{22}$$

$$ES_i \leq S_i \leq LS_i \qquad \forall i \in A \cup \{n+1\} \tag{23}$$

$$x_{ij} \in \{0, 1\} \qquad \forall (i,j) \in (A \cup \{0, n+1\})^2 \tag{24}$$

where M_{ij} is some large enough constant, which can be set to any valid upper bound for $S_i - S_j$ (*e.g.,* $M_{ij} = ES_i - LS_j$), and TE is the transitive closure of E. Constraints (12) state that for two distinct activities, either i precedes j, or j precedes i, or i and j are processed in parallel.

Constraints (13) express the transitivity of the precedence relations. Constraints (14) are so-called disjunctive constraints linking the start time of i and j with respect to 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 (15) link flow variables and x_{ij} variables. If i precedes j, the maximum flow sent from i to j is set to $\min\{b_{ik}, b_{jk}\}$ while if i does not precede j the flow must be zero. Constraints (16-18) are resource flow conservation constraints. Constraints (19) and (20) set the preexisting precedence constraints. Constraints (23) restrict the start time of any activity $i \in A \cup \{0, n+1\}$ to lie between its earliest start time (ES_i) and its latest start time (LS_i).

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 big-M constant in constraints (14). However, to solve a problem involving a large time-horizon, FCT can be preferable to DT and DDT. Indeed, DT and DDT both have a pseudopolynomial number of variables and constraints, depending on the scheduling horizon while FCT involves polynomial numbers of variables and constraints. Let $\overline{G}(V, \overline{TE})$ denote the complement of $G(V, TE)$, ignoring arc orientation. Formulation FCT involves $2|\overline{TE}|$ binary variables (with $2|\overline{TE}| \leq n^2$), $m(n+2)^2 + n + 1$ continuous variables, and $(n+2)^2 + (n+1)^2 m + 2(n+1)n$ non-redundant constraints (14-17).

3 Event-based continuous time formulations 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 the work of Pinto and Grossmann on batch process problems [29] and on the formulation by Dauzère-Pérès and Lasserre for flow-shop problems [13], as well as on the former polyhedral study of machine scheduling by Lasserre and Queyranne [26]. Events correspond to start or end times of activities. In any left-shifted schedule for the RCPSP, with finish-to-start precedence relations, with zero time lag, 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 [36]) 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 also the 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.

Extension of previously-existing event-based models to the RCPSP is not immediate. Previously proposed event-based approaches were based on sequential variables on machines (*i.e.,* resource availability is equal to 1). This allows the definition of x_{iem} variables stating that activity i is scheduled at event e on machine m. All events are totally ranked on each machine, and activities assigned to distinct events cannot overlap in time. For resource of availability greater than 1 as in the RCPSP, there is only a *partial* order between activities. Our propositions are fundamentally different since we consider a global sequence of events, instead of a sequence per machine. Consequently, two activities associated to distinct events may overlap in time.

3.1 Start/End Event-based formulation (SEE)

Zapata *et al.* [38] propose such an event-based formulation for a multimode resource-constrained multiproject scheduling problem, an extension of the RCPSP. Their formulation considers that an event occurs when an activity starts or ends. Transposed to the RCPSP, this model involves n^2 binary variables to identify the activity start events, n^2 binary variables for the activity end events, and n^2 binary variables to identify the events where the activities are in process, yielding a total of $3n^2$ binary variables. In this section, we introduce an improved formulation for the RCPSP, involving only two types of binary variables. A decision 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 times of activities. A continuous variable t_e is introduced to represent the date of event e, and a continuous variable r_{ek} is used for the quantity of resource k required immediately after event e. This yields the Start/End Event-based formulation (noted SEE):

$$\min t_n \tag{25}$$

$$t_0 = 0 \tag{26}$$

$$t_f \geq t_e + p_i x_{ie} - p_i(1 - y_{if}) \qquad \forall (e,f) \in \mathcal{E}^2, f > e, \forall i \in A \tag{27}$$

$$t_{e+1} \geq t_e \qquad \forall e \in \mathcal{E}, e < n \tag{28}$$

$$\sum_{e \in \mathcal{E}} x_{ie} = 1 \qquad \forall i \in A \tag{29}$$

$$\sum_{e \in \mathcal{E}} y_{ie} = 1 \qquad \forall i \in A \tag{30}$$

$$\sum_{e'=e}^{n} y_{ie'} + \sum_{e'=0}^{e-1} x_{je'} \leq 1 \qquad \forall (i,j) \in E, \forall e \in \mathcal{E} \tag{31}$$

$$r_{0k} = \sum_{i \in A} b_{ik} x_{i0} \qquad \forall k \in R \tag{32}$$

$$r_{ek} = r_{e-1,k} + \sum_{i \in A} b_{ik} x_{ie} - \sum_{i \in A} b_{ik} y_{ie} \qquad \forall e \in \mathcal{E}, e \geq 1, k \in R \tag{33}$$

$$r_{ek} \leq B_k \qquad \forall e \in \mathcal{E}, k \in R \tag{34}$$

$$ES_i x_{ie} \leq t_e \leq LS_i x_{ie} + LS_{n+1}(1 - x_{ie}) \qquad \forall i \in A, \forall e \in \mathcal{E} \tag{35}$$

$$ES_{n+1} \leq t_n \leq LS_{n+1} \tag{36}$$

$$(ES_i + p_i) y_{ie} \leq t_e \leq (LS_i + p_i) y_{ie} + LS_{n+1}(1 - y_{ie}) \qquad \forall i \in A, \forall e \in \mathcal{E} \tag{37}$$

$$t_e \geq 0 \qquad \forall e \in \mathcal{E} \tag{38}$$

$$r_{ek} \geq 0 \qquad \forall e \in \mathcal{E}, k \in R. \tag{39}$$

$$x_{ie} \in \{0,1\}, y_{ie} \in \{0,1\} \qquad \forall i \in A \cup \{0, n+1\}, \forall e \in \mathcal{E} \tag{40}$$

The objective function is given by (25). Constraint (26) stipulates that event 0 starts at time 0. Inequalities (27) ensure that if activity i starts at event e and ends at event f (*i.e.*, $x_{ie} = 1$ and $y_{if} = 1$), then $t_f \geq t_e + p_i$. Any other combination of values for x_{ie} and y_{if} yield either $t_f \geq t_e$ or $t_f \geq t_e - p_i$, which are redundant with constraints (28). Constraints (28) order the events. Constraints (29) (respectively (30)) require that a start event (respectively an end event) has a single occurrence. Constraints (31) describe the precedence relation between activities. If $(i,j) \in E$ and i ends at or after event e, then j cannot start before event e. Constraints (32) give for each resource $k \in R$ the total resource demand r_{0k} of the activities that start at event

0. Constraints (33) are resource conservation constraints. More precisely, for each resource k, its demand immediately after event e (r_{ek}) is equal to its demand immediately after the previous event $e-1$ ($r_{e-1,k}$), plus the demand required by the activities that start at event e, minus the demand required by the activities that end at event e. Constraints (34) limit the demand of resources at each event to the availability of resources.

Note that the variables r_{ek} can all be replaced by their expression in function of variables x_{ie}, starting with (32) and substituting further using (33). The resulting substitution has not been displayed for better readability.

Constraints (35-37) are valid inequalities based on activity time windows. Constraints (35) state that the start time of activity i must occur after its earliest start time ES_i, and before its latest start time LS_i. Symmetrically, constraints (37) impose that its finish time must be between its earliest start time plus its processing time ($ES_i + p_i$), and its latest start time plus its processing time ($LS_i + p_i$).

Formulation SEE involves $2n(n+1)$ binary variables, $(n+1)$ continuous variables, and $(n+1)(m + n^2/2 + |E|) + 3n$ non-redundant constraints (27-34). Compared with DT and DDT, our formulation SEE involves a polynomial number of variables and constraints. Compared with the previously-proposed event-based formulation [38], our formulation involves n variables fewer since in-process variables are not necessary in SEE. Finally, compared with to FCT, SEE does not involve big-M constraints. However, SEE has a larger number of binary variables than FCT.

3.2 On/Off Event-based formulation (OOE)

The second event-based formulation we introduce in this paper uses only one type of binary variable per event. We define a decision variable z_{ie} to be set to 1 if activity i starts at event e or if it is still being processed immediately after event e. Thus, z_{ie} remains equal to 1 for the duration of the process of 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. A continuous variable t_e represents, as in SEE, the date of event e. We also use one single extra continuous variable: C_{max} (makespan). With formulation OOE, the resource constraints are

modeled in a very simple way. Here is the OOE formulation:

$$\min C_{\max} \tag{41}$$

$$\sum_{e\in\mathcal{E}} z_{ie} \geq 1 \qquad \forall i \in A \tag{42}$$

$$C_{\max} \geq t_e + (z_{ie} - z_{i,e-1})p_i \qquad \forall e \in \mathcal{E}, \forall i \in A \tag{43}$$

$$t_0 = 0 \tag{44}$$

$$t_{e+1} \geq t_e \qquad \forall e \neq n-1 \in \mathcal{E} \tag{45}$$

$$t_f \geq t_e + ((z_{ie} - z_{i,e-1}) - (z_{if} - z_{i,f-1}) - 1)p_i \qquad \forall (e,f,i) \in \mathcal{E}^2 \times A, f > e \tag{46}$$

$$\sum_{e'=0}^{e-1} z_{ie'} \leq e(1 - (z_{ie} - z_{i,e-1})) \qquad \forall e \in \mathcal{E} \setminus \{0\} \tag{47}$$

$$\sum_{e'=e}^{n-1} z_{ie'} \leq (n-e)(1 + (z_{ie} - z_{i,e-1})) \qquad \forall e \in \mathcal{E} \setminus \{0\} \tag{48}$$

$$z_{ie} + \sum_{e'=0}^{e} z_{je'} \leq 1 + (1 - z_{ie})e \qquad \forall e \in \mathcal{E}, \forall (i,j) \in E \tag{49}$$

$$\sum_{i=0}^{n-1} b_{ik} z_{ie} \leq B_k \qquad \forall e \in \mathcal{E}, \forall k \in R \tag{50}$$

$$ES_i z_{ie} \leq t_e \leq LS_i(z_{ie} - z_{i,e-1}) + LS_n(1 - (z_{ie} - z_{i,e-1})) \qquad \forall e \in \mathcal{E}, \forall i \in A \tag{51}$$

$$ES_{n+1} \leq C_{\max} \leq LS_{n+1} \qquad \forall e \in \mathcal{E}, \forall i \in A \tag{52}$$

$$t_e \geq 0 \qquad \forall e \in \mathcal{E} \tag{53}$$

$$z_{ie} \in \{0,1\} \qquad \forall i \in A, \forall e \in \mathcal{E} \tag{54}$$

Constraints (42) ensure that each activity is processed at least once during the project. Constraints (43) link the makespan to the event dates: $C_{max} \geq t_e + p_i$ if i is in process at event e but not at event $e-1$, *i.e.,* if i starts at event e. Constraints (44) and (45) ensure event sequencing. Constraints (46) link the binary optimization variables z_{ie} to the continuous optimization variables t_e, and ensure that, if activity i starts immediately after event e and ends at event f, then the date of event f is at least equal to the date of event e plus the processing time of activity i ($t_f \geq t_e + p_i$). The validity of these constraints follows the same logic as for constraints (27). For coping with the case of $e = 0$, we define $z_{i,-1} := 0$.

Proposition 1. Constraints (47) and (48), called *contiguity constraints*, ensure non-preemption (the events after which a given activity is being processed are adjacent).

Proof: Constraints (47) are obtained from the following condition

$$[z_{i(e-1)} = 0 \wedge z_{ie} = 1] \implies [\sum_{v=0}^{e-1} z_{iv} = 0], \quad \forall e \in \mathcal{E} \setminus \{0\}, \forall i \in A. \tag{55}$$

Roughly speaking: if activity i starts at event e, then i cannot be processed before e.

By definition of binary variables we have: $z_{ie} - z_{i(e-1)} \in \{-1,0,1\}$ and $1 - (z_{ie} - z_{i(e-1)}) \in \{0,1,2\}$. Condition (55) can thus be rewritten as:

$$[1 - (z_{ie} - z_{i,e-1,}) = 0] \implies [\sum_{v=0}^{e-1} z_{iv} = 0 \quad \forall e \in E \setminus \{0\}], \forall i \in A.$$

We also know that: $\sum_{v=0}^{e-1} z_{iv} \leq e$, since z_{ie} is a binary variable. This allows us to build constraints (47) by *lifting* the preceding inequality. In the same way, constraints (48) are obtained from the following condition:

$$[z_{i(e-1)} = 1 \wedge z_{ie} = 0] \implies [\sum_{v=e}^{n-1} z_{iv} = 0], \quad \forall e \in E \setminus \{0\}, \forall i \in A. \quad \square$$

Constraints (49) describe each precedence constraint $(i, j) \in E$, modeling the expression $[z_{ie} = 1] \implies [\sum_{e'=0}^{e} z_{je} = 0]$ for each event e.

Constraints (50) are the resource constraints limiting the total demand of activities in process at each event.

Constraints (51) and (52) set the start time of any activity i between its earliest start time ES_i and its latest start time LS_i.

Formulation OOE involves n^2 binary variables (twice as few as SEE, and the same number as FCT), $(n + 1)$ continuous variables, and $(n - 1)(3 + |E| + m + n^2/2) + n^2 + n$ non-redundant constraints (42-50).

3.3 Example

In order to illustrate 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

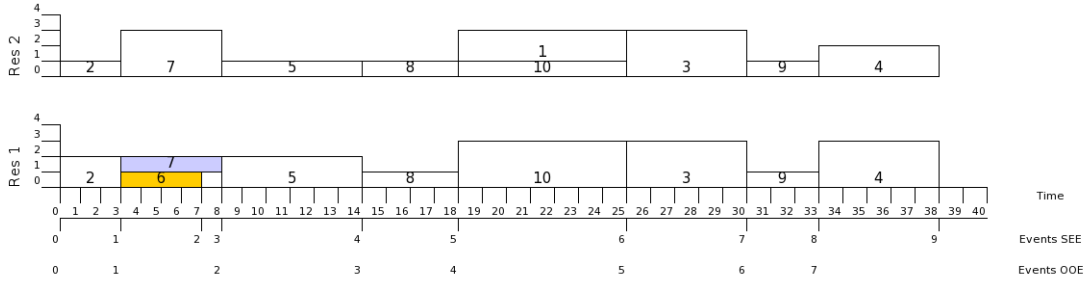

Figure 1: Gantt chart of a feasible solution with discrete time and with events

Tables 2, 3, and 4 display corresponding values of the optimization variables related to activities 6 and 7. The x_{it}'s are the time-indexed variables used in the DT and DDT models (Table 2), while the x_{ie}'s and the y_{ie}'s are the event-indexed variables involved in the SEE model (Table 3). Finally, the z_{ie}'s (Table 4) are the event-indexed variables related to the OOE model.

Note that 39 time points are necessary for the DT and DDT formulations, while only 10 events are necessary for the SEE formulation, and 8 events for the OOE formulation. The advantage of

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

Table 2: Some values of variables x_{it} of DT and DDT models, for the above feasible solution

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

Table 3: Some values of variables x_{ie} and y_{ie} of SEE model, for the above feasible solution

the OOE formulation over the SEE formulation appears clearly since OOE is based on *intervals* during which the resource demand is assumed to be constant, rather than on start and end events.

3.4 Preprocessing for the OOE formulation

The number of binary variables of the event-based formulations depends on the number of events and the number of events that may be assigned to each activity. Although there are generally significantly fewer binary variables than in the time indexed formulations, this number can still yield impracticable MILPs. Therefore, we now present three preprocessing methods for the OOE formulation in order to provide guidelines on how the number of binary variables can be reduced. Some of these methods yield approximate formulations (only yielding feasible solutions without optimality guarantee).

OOE_Prec This formulation is obtained by removing, from the set of possible events for an activity, all the first events during which the activity cannot or does not need to be in process because of its predecessors. Symetrically, we also remove the last events during which the activity cannot or does not need to be in process because of its successors. Let $A(i)$ denote the set of ancestors of i in G. Les $D(i)$ denote the set of its descendents.

Proposition 2. There exists an optimal solution of OOE such that for each activity i: $\sum_{e=0}^{|A(i)|} z_{ie} = 0$ and $\sum_{e=n-|D(i)|+1}^{n} z_{ie} = 0$.

Proof: Since there are n events, there always exist an optimal solution such that each activity starts at a different event, *i.e.,* for two distinct activities i and j,

$$z_{ie} - z_{i,e-1} = 1 \wedge z_{jf} - z_{j,f-1} = 1 \implies e \neq f.$$

(Remark that this does not forbid $t_e = t_f$.)

Furthermore, if j is an ancestor of i in G, then the event assigned to j is strictly smaller than

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

Table 4: Some values of variables z_{ie} of OOE model, for the above feasible solution

the event assigned to i. The property follows from these two observations. □

Consequently, we obtain OOE_Prec by setting to 0 the following variables, yielding a correct formulation for the RCPSP :

$$z_{ie} = 0, \quad i \in A, e \in \{0, \ldots, |A(i)|\} \cup \{n - |D(i)| + 1, \ldots, n\}. \tag{56}$$

OOE_Level This formulation introduces heuristic variable setting to reduce the search space, yielding an approximate formulation. The variable setting method is based on the topological ordering of the activities. Let v denote the number of levels of G; v is the length of the longest path (in number of arcs) from 0 to $n+1$. Let $l^-(i)$ denotes the earliest level of activity i in graph G, *i.e.,* the longest path length from 0 to i. Let $l^+(i)$ denote its latest level, equal to v minus the length of the longest path from i to $n+1$. Intuitively, one may expect that the activities of a level l start after the activities of level $l-1$, although this may lead to suboptimal solutions. Let $N^-(i) = \{j \in A | l^+(j) < l^-(i)\}$ and $N^+(i) = \{j \in A | l^-(j) > l^+(i)\}$. We define formulation OOE_Level by setting the following variables to 0 (note that we have $A(i) \subseteq N^-(i)$ and $D(i) \subseteq N^+(i)$):

$$\sum_{e=0} z_{ie} = 0, \quad i \in A, e \in \{0, \ldots, |N^-(i)|\} \cup \{n - |N^+(i)| + 1, \ldots, n\}. \tag{57}$$

OOE_Reduced This formulation simply decreases the number of events by taking as the maximal number of events, the number n' of distinct start times of a feasible solutions obtained by a heuristic. We have obviously $n' < n$ but we do not know whether an optimal solution involving n' events exists.

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. Finally, we perform a computational comparison of the five MILP models, first in terms of linear programming relaxation, and then, in terms of exact (integer) solving.

4.1 Instances and instance indicators

Since the sixties, indicators have emerged to characterize the RCPSP instances [21, 12]. They can be roughly classified into four categories: precedence-oriented (*e.g.,* OS and NC), time-oriented, resource-oriented, and hybrid (*e.g.,* RS, #FS, ACUFS, RF) [4]. Let us now define briefly some indicators and the established relationships between these indicator values and instance tractability.

- **OS**: *Order strength* is defined as the density of the transitive closure of the precedence graph ($0 \leq \text{OS} \leq 1$), 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 [16, 33]. Note that instances close to RS = 0 are far harder than the ones close to 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 $PR = \frac{\max_i p_i}{\min_i p_i}$.

For more detail on instance indicators, see [4].

Among the large number of instances found in the literature, we shall concentrate on three instance sets named KSD, BL and Pack. We also generated two new sets: KSD15_d and Pack_d. Table 5 displays for each instance set the range of each indicator. We describe below each instance set:

Table 5: 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

KSD: PSPLIB (from Kolisch *et al.* [24]) is a scheduling library reachable through a web site [31], 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 for the NC, RF, and RS indicators.

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 category, any activity demand cannot exceed half of the resource availability, which implies the absence of disjunctions

and consequently yields highly cumulative instances. 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).

The instances described in the literature to test the formulations proposed for the RCPSP, are often being subject to some criticism. In fact for the most used instances, the KSD ones, it appears that the hardest instances all have a small RS indicator 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 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 [29].

Hereafter is a more precise description of the method used to generate a new set of instances (B) from an existing one (A). Let x, y and a denote the parameters of the method. We select the first x non-dummy activities of the instance set A, we remove the remaining ones and their precedence constraints. We connect activities without predecessors to 0 and activities without successor to activity $x+1$. We select randomly y activities among the x non-dummy activities. We multiply the duration of each selected activity by $a+b$ where b is a random number uniformly generated between 0 and 1, and we round the obtained duration to the nearest integer.

Pack_d: We set x to the initial number of activities, y to 10 and a to 50. Thus, we obtain a set of 55 instances with the same number of activities as in Pack with processing times ranging from a few units of time to hundreds of units of time.

KSD15_d: We set x to 15, y to 7 and a to 25. Hence, we obtain a set of 480 small-sized instances ($n = 15$) and the processing times may vary from 1 to approximately 250.

Both Pack_d and KSD30_d are publicly available [32].

In the next subsection, we shall compare the different formulations of the RCPSP and the specialized MCS-based search method [25] 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 solving time of each instance to 500 seconds.

4.2.1 Time-window preprocessing

We perform a preprocessing phase aiming at reducing activity time windows by standard precedence and resource constraint propagation, which allows reducing the number of x_{it} variables in the discrete time formulations and strengthens the valid inequalities involving ES_i and LS_i.

To that purpose, the horizon T is first set to an upper bound obtained by the parallel schedule

scheme heuristic with the minimum latest finishing time rule [22]. 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 was already used in [14] to strengthen the relaxation of formulations DT and DDT, so we refer to the paper of Demassey *et al.* [14] for a precise description of this step.

4.2.2 Comparison of LP relaxations

We compare the lower bounds obtained by linear relaxation of each of the five MILP formulations. The results are displayed in Table 6, where we use the following abbreviations: %Solved gives the percentage of instances for which the relaxation could be solved within 500 seconds. ΔCPM gap provides the average deviation in percent for solved instances from the critical-path method lower bound. This indicator constitutes an absolute reference for comparison with other methods. ΔES_{n+1} gives average deviation for solved instances from the lower bound on the makespan produced by time-window preprocessing, in percent. This indicator shows whether the LP relaxation brings useful information or not. Column Time displays the Average CPU time required, in seconds. The methods are ranked according to criteria %Solved, %CPM gap, Time, by order of importance.

Table 6: Linear relaxation results

Instances	**Models**	**%Solved**	**ΔCPM**	ΔES_{n+1}	**Time (s)**
KSD30	DDT	100	8.29	0.31	0.78
	DT	100	7.99	0.04	0.04
	OOE	100	7.94	0.00	0.39
	SEE	100	7.94	0.00	3.01
	FCT	97	7.96	0.00	6.94
PACK	DDT	100	148.88	17.99	1.47
	DT	100	142.23	12.57	0.18
	OOE	100	128.94	0.00	0.22
	SEE	100	128.94	0.00	1.32
	FCT	100	128.94	0.00	3.53
BL	DDT	100	13.09	13.09	0.12
	DT	100	5.53	5.53	0.05
	OOE	100	0.00	0.00	0.17
	SEE	100	0.00	0.00	1.09
	FCT	97	0.00	0.00	2.25
KSD15_d	OOE	100	7.69	0.00	0.04
	SEE	100	7.69	0.00	0.05
	FCT	100	7.69	0.00	0.10
	DT	71	6.78	0.00	0.28
	DDT	15	21.22	22.05	0.52
Pack_d	OOE	100	80.55	0.00	0.15
	SEE	100	80.55	0.00	0.41
	FCT	100	80.55	0.00	3.79
	DT	0	-	-	-
	DDT	0	-	-	-

Table 6 reveals that the time-indexed formulations clearly 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. The event-based and flow formulation all have very poor relaxations, since the lower bound obtained by preprocessing is never improved. These results are consistent with previous studies of lower bounds using discrete and continuous time relaxations [37, 14]. On the new instance sets KSD15_d, the performance of formulation DDT collapses as only 15% of LP relaxations can be computed within 500 seconds. Here, formulation DT is still able to compute the relaxations of 71% of the instances, and becomes the best formulation. On the new instance set Pack_d, the LP relaxations of the time indexed formulation cannot be computed anymore because of the LP size increase. Although they are able to compute all the LP relaxations for the KSD15_d and Pack_d sets, the time indexed and flow formulations perform poorly. Hence, computing good LP-based lower bounds for such classes of instance is a challenging perspective.

4.2.3 Exact solving

The second series of tests involves exact solving, *i.e.,* computing optimal solutions for each instance with the five MILP formulation and the specialized MCS-based search exact method of Laborie [25]. The results are displayed in Table 7, where we use the following notation: %Integer gives the percentage of instances for which integer (feasible) solutions were found (including both optimal and suboptimal solutions). %Opt gives the percentage of optimal solutions found and proved by the branch and bound within 500 seconds. Gap provides the average deviation from the optimal (or the best known) makespan (upper bound) in percent. ΔCPM Gap gives the average deviation from critical-path based lower bound in percent. This value is given to ease comparison with other methods as it is a stable reference value. Time Opt displays the average solving time in seconds when optimality could be proved (other instances were solved in 500 seconds).

The methods are listed for each instance set according to the %Integer criterion decreasing values. Note that the MCS-based search method issues only a lower bound if the optimal solution is not found. Hence, it is displayed separately as a reference for its performance on the number of optimal solutions found. For each instance set, the best result in terms of number of optimal solutions (%Opt) found is displayed in bold while the best result for %opt, among the MILP formulations is underlined.

There are clearly two groups of instance sets concerning the results of the MILP formulations:

- For the KSD30, Pack, and BL instance sets (instances with a low processing time range), the best MILP formulations for the %Integer, %Opt and Time Opt criteria is DDT, followed by DT. On the KSD30 and BL sets, MCS always proves the largest number of optima. However, one remarkable result is that for the Pack instances (the hardest ones for all methods) both the DDT and the DT formulations outperform MCS. Indeed, the DDT formulation solves three times as many instances as MCS to optimality. This unexpected result underlines the progress of MILP solvers and by itself comforts the pursuit of researches of MILP-based methods for the RCPSP. The FCT formulation is the best continuous-time formulation for the KSD30 instances while it yields extremely poor results on the Pack and BL instances. The SEE formulation is always ranked last. The OOE formulation and its preprocessed OOE_Prec variant find an integer solution for about half of the instances with an acceptable gap. The preprocessing is much more efficient for the KSD30 set which involves more precedence constraints.

- For the KSD15_d and the Pack_d instance sets (instances with a high processing time range),

Table 7: Exact solving (branch and bound) results

Instances	**Formulations**	**%Integer**	**%Opt**	**Gap**	**ΔCPM**	**Time Opt (s)**
KSD30	DDT	91	82	0.47	8.91	10.45
	DT	86	78	0.55	6.74	12.76
	FCT	67	62	0.16	3.76	22.66
	OOE_Prec	46	30	1.69	13.65	52.31
	OOE	33	24	1.22	7.00	112.62
	SEE	3.1	2.9	0.24	0.61	123.62
	MCS	-	**97**	0.00	11.48	7.39
PACK	DDT	95	**76**	1.08	199.02	63.39
	DT	85	55	0.49	203.58	48.24
	OOE_Prec	55	5	3.25	227.19	18.92
	OOE	49	9	2.89	231.29	61.78
	FCT	2	0	1.28	14.49	-
	SEE	0	0	-	-	-
	MCS	-	25	0.00	149.81	115.88
BL	DDT	100	100	0.00	32.40	13.68
	DT	100	100	0.00	32.40	37.93
	OOE_Prec	54	0	7.26	40.30	-
	OOE	49	0	7.90	41.65	-
	FCT	21	3	6.14	30.64	310.58
	SEE	8	0	12.81	29.96	-
	MCS	-	**100**	0.00	32.40	3.29
KSD15_d	OEE_Prec	99.8	86	0.00	10.02	6.49
	FCT	99	94	0.02	9.02	12.06
	OEE	99	83	0.01	10.14	4.68
	SEE	92	76	0.15	9.86	13.04
	DT	55	54	0.23	4.31	12.10
	DDT	1	1	0.00	2.63	3.34
	MCS	-	**100**	0.00	10.18	0.07
PACK_d	OEE	60	18	1.26	120.13	75.58
	OOE_Prec	60	14	1.62	117.56	54.35
	FCT	7	7	0.00	0.00	60.88
	SEE	4	4	0.00	0.00	215.08
	DT	0	0		-	-
	DDT	0	0		-	-
	MCS	-	**38**	0.00	50.59	72.34

the performance of the discrete time formulation crashes, as expected. Since it involves fewer constraints, the DT formulations becomes better than DDT. The specialized MCS-based search method solves the largest number of instances to optimality. If we consider the %Opt criterion, the best method is FCT for the KSD15_d set. However, we can also note the excellent performance of the OOE_Prec formulation, which is the best formulation for the number and quality of feasible solutions found. All optimal solutions found by OOE_Prec were reached (without proof of optimality in some cases) in twice as less time than FCT. On the Pack_d set which is much harder to solve (MCS reaches only 38% for the %Opt criterion), OOE and OOE_Prec are the best formulations according to all criteria. Formulation SEE is consistently outperformed by the OOE formulation, although it obtains acceptable results on the KSD_15 set. This underlines the drastic improvement brought by our formulation compared to the formulation proposed by Zapata *et al.* in [38] (recall that SEE already dominates the formulation of Zapata *et al.*. As a side remark, the Pack_d instances are easier than the Pack instances for the event-based and the MCS methods. Introducing variability in the processing times may not necessarily yield harder instances.

Although these results should be taken with care as we performed experiments on a limited number of instances and a given allowed CPU time, we summarize our findings in Table 8. We denote OOE_x either OOE or its preprocessed variant OOE_Prec. When exact solving through MILP is considered, formulation DDT should be used if the processing time range is low. For instances with a high processing time range, if the instances have a high disjunction ratio, the flow based formulation FCT or the event based formation have equivalent performances, although the On-Off event-based formulation was faster with better solutions found. If the instances are highly cumulative, the On-Off event-based formulation should be preferred.

Table 8: Synthesis of the experiments

	High DR	Low DR
Low PR	DDT $\succ$ DT $\succ$ FCT $\succ$ OOE_x $\succ$ SEE	DDT $\succ$ DT $\succ$ OOE_x $\succ$ FCT $\succ$ SEE
High PR	FCT,OOE_x $\succ$ SEE $\succ$ DT $\succ$ DDT	OOE_x $\succ$ FCT $\succ$ SEE $\succ$ DT $\succ$ DDT

4.2.4 Comparison of variants of the OOE formulation

In Table 9, the results of the OOE variants presented in Section 3.4 are compared in terms of both number and quality of integer solutions found within 500 seconds. The simple heuristic preprocessing techniques (OOE_Level and OOE_Reduced) are able to increase the number of integer solutions found without decreasing significantly the gap (or even improving it). OOE_Reduced obtains its best results on the BL instances, while OOE_Level is efficient on the KSD30 and Pack_d instances. A promising research direction would be to find more efficient heuristics based on the OOE_x formulations.

Conclusions

In this paper, we proposed two new MILP formulations for the RCPSP: the Start/End Event-based (SEE) formulation (an adaptation to the RCPSP of the formulation proposed in [38]), and the On/Off Event-based (OOE) 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

Table 9: Results of OOE variants

Instances	Formulations	%Integer	Gap	ΔCPM
KSD30	OOE_Level	52	1.68	14.72
	OOE_Reduced	48	1.31	10.08
	OOE_Prec	46	1.69	13.65
	OOE	33	1.22	7.00
PACK	OOE_Level	55	4.57	206.66
	OOE_Prec	55	3.25	227.19
	OOE_Reduced	53	4.19	212.62
	OOE	49	2.89	231.29
BL	OOE_Reduced	77	3.84	36.68
	OOE_Level	72	7.08	41.32
	OOE_Prec	54	7.26	40.30
	OOE	49	7.90	41.65
KSD15_d	OEE_Prec	99.8	0.00	10.02
	OEE_Level	99.6	0.02	10.05
	OEE_Reduced	99.4	0.01	10.05
	OEE	98.6	0.01	10.14
PACK_d	OOE_Level	69	1.44	123.38
	OOE_Prec	60	1.62	117.56
	OEE	60	1.26	120.13
	OEE_Reduced	54	1.93	122.97

variables. We also compared these new MILP formulations, together with classical ones, both in terms of linear-relaxation lower bounds, and in terms of exact solving.

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 numerically that the event-based formulations we proposed are very promising for such problems.

We compared the event-based formulations with three other formulations issued from the literature, two of which (DT and DDT) use variables indexed by time, and a third formulation (FCT) uses sequential variables. The overall five formulations are representative of the three main classes of MILP formulations.

We obtained from these experiments that the formulation proposed by Christofides *et al.* (DDT) yields better results for exact solving on the 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 class dominates the other ones, and that the appropriate formulation has to be selected depending on instance characteristics. Indeed, the formulations based on events (more particularly the 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), the On/Off formulation presents better performances than all the other MILP formulations. We conclude that to solve highly cumulative RCPSP instances involving very large scheduling horizon, our event-based On/Off formulation appears to be the most appropriate. Furthermore, the On/Off formulation consistently outperformed the Start/End formulation on

all tested instance sets. Thus, our contribution brings a major improvement to the event-based formulation class, which could in turn has a positive impact on the targeted applications in the process industry. Further work should concentrate on in designing efficient heuristics embedding event-based MILP formulations, as well as extensions to practical constraints and/or objectives.

As a side and unexpected result, we observed that the standard MILP discrete-time formulations yield better results than a specialized state-of-the art RCPSP exact solving method on a set of hard instances. In accordance with the results of Zhu *et al.* [39], we thereby hope to motivate further research in MILP-based solution strategies for scheduling.

Acknowledgment

This work was supported by the Interdisciplinary Program "Energie" of the French National Center for Scientific Research (CNRS) - GIMEP project (2008-2009).

The authors are indebted to anonymous referees whose comments really improve a first version of the paper.

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, (Eds.), *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.H.G. Rinnooy Kan, "Scheduling subject to resource constraints: Classification 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] B. de Reyck and W. Herroelen, "On the use of the complexity index as a measure of complexity in activity networks", *European Journal of Operational Research*, 91(2): 347–366, 1996

[13] 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, Germany, May 1995.

[14] 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.

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

[16] S.E. Elmaghraby and W.S. Herroelen, "On the measurement of complexity in activity networks" *European Journal of Operational Research*, 5: 223–234, 1980.

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

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

[19] O. Icmeli and W. O. Rom, "Solving the resource constrained project scheduling problem with optimization subroutine library", *Computers & Operations Research*, 23(8): 801–817, 1996.

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

[21] R. Kolisch, A. Sprecher, and A. Drexl, "Characterization and generation of a general class of Resource-Constrained Project Scheduling Problems", *Management Science*, 41: 1693–1703, 1995

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

[23] R. Kolisch and S. Hartmann, "Experimental evaluation of heuristics for the resource-constrained project scheduling problem: An update", *European Journal of Operational Research*, 174: 23–37, 2006.

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

[25] P. Laborie, "Complete MCS-based search: Application to resource constrained project scheduling", *International Joint Conferences on Artificial Intelligence*, pages 181–186, 2005.

[26] 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 Integer Programming and Combinatorial Optimization Conference*, pages 136–149, 1992.

[27] 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.

[28] J. Carlier and E. Néron, "On linear lower bounds for the resource constrained project scheduling problem", *European Journal of Operational Research* 149(2): 314–324, 2003.

[29] 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.

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

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

[32] High-duration RCSPSP instances. `http://www2.laas.fr/laas/files/MOGISA/RCPSP-instances/high_duration_range.zip` .

[33] B. De Reyck and W. Herroelen, "On the use of the complexity index as a measure of complexity in activity networks", *European Journal of Operational Research*, 91: 347–366, 1996.

[34] M. Sabzehparvar and S. M. Seyed-Hosseini, "A mathematical model for the multi-mode resource-constrained project scheduling problem with mode dependent time lags", *Journal of Supercomputing*, 44(3): 257–273, 2007.

[35] A. Sprecher, "Solving the RCPSP efficiently at modest memory requirements", *Management Science*, 46(5): 710–723, 2000.

[36] A. Sprecher, R. Kolisch, and A. Drexl, "Semi-active, active, and non-delay schedules for the resource-constrained project scheduling problem", *European Journal of Operational Research*, 80: 94–102, 1995.

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

[38] J. C. Zapata, B. M. Hodge, and G. V. Reklaitis, "The multimode resource constrained multiproject scheduling problem: Alternative formulations", *AIChE Journal*, 54(8): 2101–2119, 2008.

[39] G. Zhu, J.F. Bard, and G. Ju, "A branch-and-cut procedure for the multimode resource-constrained project-scheduling problem", *INFORMS Journal on Computing*, 18(3): 377–390, 2006.