Comparison of mixed integer linear programming models for the resource-constrained project scheduling problem with consumption and production of resources

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

[1] Laboratoire de Mathématiques et Informatique, UFR-SFA, Université d'Abobo - Adjamé, BP 801 Abidjan 02, Côte d'Ivoire.
[2] CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France.
[3] Univ de Toulouse, LAAS, F-31400 Toulouse, France.
[4] École Nationale de l'Aviation Civile, 7 av. É.-Belin - BP 54005, 31055 Toulouse cedex 4, France.
e-mails: mr.okone@gmail.com, {artigues,lopez}@laas.fr, marcel.mongeau@enac.fr

Abstract

This paper addresses an extension of the resource-constrained project scheduling problem that takes into account storage resources which may be produced or consumed by activities. To solve this problem, we propose the generalization of two existing mixed integer linear programming models for the classical resource-constrained project scheduling problem, as well as one novel formulation based on the concept of event. Computational results are reported to compare these formulations with each other, as well as with a reference method from the literature. Conclusions are drawn on the merits and drawbacks of each model according to the instance characteristics.

Keywords: Resource-constrained project scheduling, mixed integer linear programming, consumption and production of resources, event-based on/off formulation.

1 Introduction

The resource-constrained project scheduling problem (RCPSP) is one of the best-known cumulative scheduling problems due to the interest from the operational research community, and to its numerous industrial applications. In this article we are concerned with the extension of the RCPSP that, beyond renewable resources considered in its basic version, also allows resources that can be produced and consumed during the execution of activities. This extension is called the *RCPSP with consumption and production of resources*, noted *RCPSP/CPR* throughout the paper.

There exist in the literature some works relating to the RCPSP/CPR. Among others, we can cite Neumann and Schwindt [25] who formalize the problem of project scheduling with inventory constraints and generalized precedence constraints, including both renewable and storage resources. The authors propose a branch-and-bound approach associated with a beam-search heuristic to solve the problem. Carlier *et al.* [12] also include generalized precedence relations and propose a new list algorithm. Laborie [24] defines the concept of *resource temporal network*, offering a powerful modeling and providing a support for other authors such as Bouley *et al.* [9]. For more detail on the state of the art in RCPSP/CPR, we refer the reader to [9, 24, 12].

Throughout this paper, we are interested in designing mixed integer linear programming (MILP) formulations for the problem, which are passed to an MILP off-the-shelf branch-and-bound solver.

The solutions obtained with MILP formulations cannot generally compete with the above-cited methods, all based on specialized scheduling models and algorithms, aiming at exploiting the problem structure within the solving scheme. However, the interest of proposing (efficient) MILP formulations is double. On the one hand, MILP solvers have made gigantic progress in the last years. In [22], the authors exhibit a class of specific but particularly hard standard RCPSP instances for which MILP formulations yield in average better results than the best-known specialized exact method. On the other hand, MILP solvers are often the only software available to practitioners facing industrial applications.

There exist roughly three categories of MILP formulations for the standard resource-constrained project scheduling problem. Note that we avoid considering here the formulations that explicitly involve an exponential number of variables and/or constraints, which can only be solved through branch-and-price and/or branch-and-cut techniques. This falls out of the scope of the paper, since we are only concerned in this paper with direct solving via MILP solvers.

The first category gathers $0-1$ time-indexed formulations which involve binary variables x_{it}, where i is the activity index while t is the time index, such that $x_{it} = 1$ if and only if activity i starts at time t. We assume integer values for both activity durations and possible start times. The time index t varies from 0 to $T-1$, where T is an upper bound of the schedule length. Time-indexed formulations were proposed in Pritsker *et al.* [30] and refined in Christofides *et al.* [14]. They generally yield good LP relaxations but, as a counterpart, they involve a pseudo-polynomial number of variables, since T depends on activity durations. Remark that if the durations are arbitrarily large, the number of variables grows exponentially. However, we distinguish these models from the models that involve a number of variables and constraints which is also exponential in the numeric value of the input.

The second category of formulations are called *sequence-based*, or *disjunctive*, MILP formulations as they involve binary variables y_{ij} and continuous start-time variables S_i, i and j being activity indices, such that $y_{ij} = 1$ if and only if $S_i + p_i \leq S_j$ where p_i is the duration of task i. Originally, these formulations were proposed for machine or "disjunctive" scheduling [6, 3] where all activities sharing the same machine must be fully sequenced. Additional variables modeling resource flows are necessary to extend these formulations to the RCPSP (see [4]). That is why these formulations are also named *flow-based* formulations. These formulations are *compact*, in the sense that they involve a polynomial number of variables and constraints. On the other hand, they yield poor LP relaxations. This is notoriously due to the big-M constraints needed to linearize the disjunctive constraint.

The third category of formulations are named *event-based* formulations. They involve binary variables z_{ie} indicating whether activity i is in process (or not) immediately after event e. Each event corresponds to the start or the end time of at least one activity. There is a continuous variables t_e to indicate the date of each event e. As there is a polynomial number of events, these formulations are also compact. Originally proposed for single-machine scheduling in [23], they were extended to more complex problems including flow-shop [16], batch plant scheduling [28, 13], and variants of the RCPSP [34, 22]. Although these formulations do not involve big-M constraints, the experiments carried out in [22] show that they yield poor LP relaxations.

For (mixed-)integer exact solving, it is shown in [22] that, despite this drawback, flow-based and event-based formulations can be an alternative for RCPSP instances when the time horizon (i.e. the largest t such that a variable x_{it} must be defined) is so large that the time-indexed formulations become intractable even when the number of activities remains of reasonable magnitude. This happens if there are activities with a large duration while the greatest common divisor of all durations is relatively small. We provide from the literature two classes of practical applications

where this situation occurs. A first class concerns problems with a high duration range. According to [29], this is a characteristic of complex manufacturing processes such as semiconductor manufacturing. The authors mention that, in this industry, the durations may range from less than 15 minutes up to more than twelve hours. A second class of practical applications gathers problems where accuracy of the processing time estimation requires a fine time discretization. Pinto and Grossmann [28] provide a real example from a plastic compounding plant where the processing times vary from 0.736 to 11.250 days. Consequently, in the area of process scheduling, a significant amount of research has been carried out to propose alternatives to time-indexed formulations [17], although Castro and Grossmann [13] state that time-indexed formulations associated with duration rounding give the best approximations. Furthermore, the process industry frequently involves an integrated management of production and consumption of utilities [1]. The RCPSP/CPR model is well suited for such applications.

Specific MILP formulations were proposed for batch plant scheduling, including time-indexed, sequence-based, and event-based formulations (see among others [28, 17, 13, 34]). Schwindt and Trautmann [32] and Neumann *et al.* [26] first proposed the application of a generalized RCPSP/CPR model to the batch scheduling problem in the process industries, and then compared the performance of a time-indexed MILP formulation with the performance of a combinatorial branch-and-bound algorithm. However, to our knowledge, no experiments have been carried out yet in the literature to compare the performance of various MILP formulations for the RCPSP/CPR. In this paper, we propose an extension to consumption and production of resources for each of the three categories of MILP formulations, and we carry out experimental comparison on instances involving low and high duration ranges. We also evaluate the impact of the problem characteristics on the performance of the models. Last, we compare the proposed MILP models to the constraint programming method proposed by Laborie [24].

We describe in detail in Section 2 the RCPSP/CPR. Section 3 is dedicated to the proposed MILP formulations. Section 4 reports experimental results on the performance of all the proposed models. We conclude in Section 5.

2 Problem description

Let V be a set of activities, p a vector of processing times, E a set of precedence relations, R a set of renewable resources, B a vector of capacity (availability of resources), and b a matrix of resource usage. Typically, the RCPSP is a combinatorial optimization problem defined by the tuple (V, p, E, R, B, b), aiming primarily at scheduling activities on resources available in limited quantities. Its general character, due to many potential applications in the industry, yields numerous possible extensions and variants that we cannot evoke here in detail. However, we will focus on its basic version and a variant involving specific resources that can be consumed and produced during the processing of activities.

2.1 Basic version (RCPSP)

Let n be the number of activities to schedule, and m the number of resources. The project under study consists of $n+2$ activities defined by the set $\{0, \ldots, n+1\}$, where activities 0 and $n+1$ are dummy activities representing by convention the beginning and the end of the project, respectively. The set of non-dummy activities, noted $A = \{1, \ldots, n\}$, must be scheduled on the available renewable resources belonging to a set $R = \{1, \ldots, m\}$. The processing times are represented by a vector p of $\mathbb{N}^{n+2}$, where the i^{th} component, p_i, is the processing time of activity i with the special values

$p_0 = p_{n+1} = 0$ for the dummy activities. Each activity i also demands b_{ik} amount of each resource k during its processing. Each resource k is available in quantity B_k. Let us define the decision variable S_i for indicating the starting time of activity i (with $S_0 = 0$), for $i = 0, 1, 2, \ldots, n+1$. Note that S_{n+1} is the date of the project completion time, also called *makespan*.

The precedence relations (precedence constraints) 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. This can be formulated as follows:

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

Resource constraints dictate that at any time the sum of demands of the activities being processed does not exceed the resource availability:

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

where: $A_t = \{i \in A | S_i \leq t < S_i + p_i\}$ represents all non-dummy activities in process at time t taking into account the non-preemption of the activities; $H = \{0, 1, \ldots, T\}$ is the *scheduling horizon*, and T is its length (which may be regarded as an upper bound for the makespan).

A *schedule* S (with i^{th} component S_i), is said *feasible* if it is compatible with both the precedence constraints and the resource constraints. The objective of the RCPSP consists in finding a non-preemptive (with non interrupted activities) schedule S of minimal makespan subject to precedence constraints and resource constraints. According to the computational complexity theory, the RCPSP is NP-hard in the strong sense [8, 33].

2.2 RCPSP with consumption and production of resources (RCPSP/CPR)

The particularity of the RCPSP with consumption and production of resources is that, in addition to using the renewable resources described above, it also involves specific *storage* resources. These resources, described sometimes as *cumulative* by some authors [27], can be consumed (or not) at the start time of an activity in a certain amount and/or then produced in another amount at the completion time of this activity. More specifically, an activity i consumes $c^-_{i\rho}$ units of resource ρ at the beginning of its processing, and produces $c^+_{i\rho}$ units at the end of its processing, where both $c^-_{i\rho}$ and $c^+_{i\rho}$ are (given) input data of the problem. Furthermore, the total amount of each resource must remain non-negative throughout the scheduling horizon. Note that, in contrast to the earlier problem definitions by [25] and [24], no capacity restriction is imposed for the storage resources.

Let P be the set of such resources, and C_ρ represent the level of initial stock of resource $\rho \in P$. We can distinguish the following cases:

1. If $c^-_{i\rho} = c^+_{i\rho}$, then ρ is simply a renewable resource, as in the basic RCPSP.
2. If $c^-_{i\rho} \neq c^+_{i\rho}$:
   (a) if $c^-_{i\rho} = 0$ and $c^+_{i\rho} > 0$ (case of resource production). This case is usually encountered in material transformation industries and sometimes in power production. In general, the resource production is preceded by the consumption, in a given amount, of another resource.
   (b) if $c^-_{i\rho} < c^+_{i\rho}$, then we first consume a quantity of a given product before producing more. This constraint appears frequently in the process industry where an intermediate product

or a utility is produced in larger quantity than it is consumed by an activity. Of course, there are other resources required by the activity that are consumed and transformed into the considered utility.

(c) if $c^-_{i\rho} > c^+_{i\rho}$, then ρ is a non-renewable resource such as a project budget, raw materials, etc.

3 Extended formulations & proposal

In this section, we introduce first an extension of two famous (time-indexed) MILP formulations of the RCPSP to the RCPSP/CPR case. Second, we propose an extension of a flow-based continuous-time formulation. Finally, we introduce an MILP formulation based on the concept of event [28, 22, 34].

Since some MILP formulations are highly sensitive to the cardinality of the scheduling horizon, it is of great importance to develop efficient mechanisms to moderate its impact on the general performance of such formulations. For each activity i, an earliest start time ES_i (the date before which activity i will certainly not start) and a latest start time LS_i (the date before which it must start) can be calculated (in polynomial time) by preprocessing using standard resource constraint propagation techniques, for example *edge-finding* (more detail in [15, 19]). Hence, the time interval $[ES_i, LS_i]$ represents the time window during which activity i can start. We only use the constraint propagation mechanism described in [22] involving only renewable resource- and temporal-constraint propagation techniques. Furthermore, as we do not have any procedure for finding an initial feasible solution, we use the sum of activity durations as initial upper bound. As a consequence, we obtain wide time windows.

3.1 Time-indexed formulations

Discrete-time formulations are characterized by the use of variables indexed by discrete times. Among these formulations, we can cite the basic discrete-time formulation (DT) introduced by Pritsker in 1969 [30], and the disaggregated discrete-time formulation (DDT) proposed by Christofides in 1987 [14]. These two formulations are very similar. The major difference between them lies in the formulation of the precedence constraints. The DT formulation involves only one type of binary decision variable, x_{it}, indexed by both activities and time. Variable $x_{it} = 1$ if activity i starts at time t; $x_{it} = 0$ otherwise. Here is the DT formulation of the RCPSP:

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

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

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

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

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

The values of the above-mentioned starting-time variables S_i's can be readily recovered through the relation: $S_i = \sum_{t \in H} t x_{it}, \quad i \in A \cup \{0, n+1\}$. Constraints (4) and (5) simply express the precedence constraints (1) and the resource constraints (2), respectively. Constraints (6) and (7) impose non-preemption of the project activities (Values $x_{00} = 1$ and $x_{it} = 0$, $\forall i \in A \cup \{n+1\}, t \in H \setminus \{ES_i, \ldots, LS_i\}$ are preset). This formulation involves $\sum_{i=1}^{n+1} (LS_i - ES_i)$ binary variables, and $|E| + (T+1)m + n + 1$ constraints.

To take into account resource consumption/production in this model, we introduce continuous decision variable s_{tp} indicating the level of each resource $p \in P$ at each time t. The additional constraints required to model the RCPSP/CPR are:

$$s_{0\rho} = C_\rho - \sum_{i=1}^{n} x_{i0} c_{i\rho}^- \qquad \forall \rho \in P \tag{8}$$

$$s_{t\rho} = s_{t-1,\rho} + \sum_{i=1}^{n} x_{i,t-p_i} c_{i\rho}^+ - \sum_{i=1}^{n} x_{it} c_{i\rho}^- \qquad \forall (t,\rho) \in H \times P, t > 0 \tag{9}$$

$$s_{t\rho} \geq 0 \qquad \forall (t,\rho) \in H \times P. \tag{10}$$

Constraints (8) state that the level of resource ρ at time 0 is equal to its initial value C_ρ minus the sum of consumption of activities starting at time 0. Constraints (9) require at each time t and for each resource $\rho \in P$ that the level of stock ($s_{t\rho}$) is equal to the level of ρ at the previous time $(t-1)$, plus the sum of output ($\sum_{i=1}^{n} x_{i,t-p_i} c_{i\rho}^+$) of resource ρ for activities ending their processing at time t, and decreased by the sum of consumption ($\sum_{i=1}^{n} x_{it} c_{i\rho}^-$) of activities starting their processing at the same time t. Constraints (10) enforce non negativity for each resource level.

Note that the intermediate decision variables $s_{t\rho}$'s can therefore be explicitly substituted out by their value as a function of the x_{it}'s.

As previously announced, the DDT model proposed by Christofides [14] is very similar to DT, but differs in the formulation of the precedence constraints. Typically, while the DT model defines one constraint for each pair of activities and for each precedence relation, the DDT model involves one constraint for each pair of activities, for each precedence relation, *and* for every time of the scheduling horizon:

$$\sum_{\tau=t}^{LS_i} x_{i\tau} + \sum_{\tau=ES_j}^{\min(LS_j, t+p_i-1)} x_{j\tau} \leq 1, \qquad \forall (i,j) \in E, \forall t \in \{ES_i, \ldots, LS_i\}. \tag{11}$$

All the other constraints remain identical. Thus, as for DT, the extension of DDT to the RCPSP/CPR that we propose simply requires adding s_{tp} variables and constraints (8) to (10).

These two models are known to involve a pseudo-polynomial number of variables and constraints. As a consequence, they yield disastrous performance when solving problems dealing with a very broad time horizon. However, they are known (mainly DDT) to provide fairly good results and interesting linear-relaxation bounds on the classical instances of the RCPSP (see [5]).

3.2 Flow-based continuous-time formulation

Inspired by the work of Balas *et al.* [6], and on the basis of the formulation of Alvarez-Valdes and Tamarit [2], Artigues *et al.* [4] proposed a flow-based continuous-time (FCT) model for the

RCPSP that uses flow variables to manage the resources. The idea is as follows. All resources are available and stored at the dummy activity 0. Each activity uses resources at the beginning of its execution, and once completed, it transfers these resources to the activities that follow according to the precedence constraints. The last activities to be executed finally forwards the resources to the dummy activity $n+1$. In the FCT model, 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). Sequential binary variables x_{ij} are required to indicate whether activity i is processed before activity j. Finally, a continuous start-time variable S_i is also needed for each activity i.

Since activity 0 acts as a resource source, and activity $n+1$ acts as a resource sink, we define for each resource k: $\tilde{b}_{ik} := b_{ik}$ for all $i \in A$ and $\tilde{b}_{0k} := \tilde{b}_{n+1,k} := B_k$.

We first recall below the flow-based formulation for the standard RCPSP, and then we extend it to the RCPSP/CPR.

$$\min S_{n+1} \tag{12}$$

$$x_{ij} + x_{ji} \leq 1, \qquad \forall (i,j) \in (A \cup \{0, n+1\})^2, i < j \tag{13}$$

$$x_{ik} \geq x_{ij} + x_{jk} - 1 \qquad \forall (i,j,k) \in (A \cup \{0, n+1\})^3 \tag{14}$$

$$S_j - S_i \geq p_i x_{ij} - M_{ij}(1 - x_{ij}) \qquad \forall (i,j) \in (A \cup \{0, n+1\})^2 \tag{15}$$

$$f_{ijk} \leq \min(\tilde{b}_{ik}, \tilde{b}_{jk}) x_{ij} \qquad \forall (i,j) \in (A \cup \{0\}) \times (A \cup \{n+1\}), \forall k \in R \tag{16}$$

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

$$\sum_{i \in A \cup \{0, n+1\}} f_{ijk} = \tilde{b}_{jk} \qquad \forall j \in A \cup \{0, n+1\}, \forall k \in R \tag{18}$$

$$f_{n+1,0,k} = B_k \qquad \forall k \in R \tag{19}$$

$$f_{ijk} \geq 0 \qquad \forall (i,j) \in (A \cup \{0, n+1\})^2, \forall k \in R \tag{20}$$

$$S_0 = 0 \tag{21}$$

$$ES_i \leq S_i \leq LS_i \qquad \forall i \in A \cup \{n+1\} \tag{22}$$

$$x_{ij} \in \{0, 1\} \qquad \forall (i,j) \in (A \cup \{0, n+1\})^2, \tag{23}$$

where M_{ij} is some large-enough constant, which can be set to $ES_i - LS_j$, and TE is the transitive closure of E. Constraints (13) and (14) are redundant constraints (valid inequalities expressing logical conditions on sequencing variables). Constraints (15) link the start-time variables and the sequential binary variables. Constraints (16) link flow variables and x_{ij} variables. Constraints (17-19) are resource flow-conservation constraints. Constraints (22) are also redundant; they 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). Furthermore, in a preprocessing we set $x_{ij} = 1$ and $x_{ji} = 0$ for every pair (i,j) of activities in TE, the transitive closure of E.

Applegate and Cook [3] show that this formulation, ranked among the *compact* models, produces bad linear-relaxation bounds due to the use of big-M constants in constraints (15). On the contrary, this formulation may be preferable to time-indexed formulations to solve problems involving a large time horizon. However, the use of sequential binary variables could be the cause of too many symmetries, thereby affecting the performance of the model for solving strongly *cumulative* problems (in the sense that many tasks can be processed in parallel).

Here is how we adapt this model to solve the RCPSP with consumption and production of resources. We define a new continuous variable D_{ijp} representing the amount of resource $p \in P$ that activity i (at the end of its processing) sends to activity j (at the beginning of its processing). As expressed by constraints (24) below, this quantity cannot exceed the minimum between the quantity produced by activity i and the quantity consumed by activity j, if activity i precedes activity j. Thus, in accordance with the principle of this model, the storage resources are also managed through flows, as follows:

$$D_{ij\rho} \leq \min(c^+_{i\rho}, c^-_{j\rho}) x_{ij} \qquad \forall (i,j) \in A^2, \rho \in P \tag{24}$$

$$\sum_{i \in A \cup \{0\}} D_{ij\rho} = c^-_{j\rho} \qquad \forall j \in A, \rho \in P \tag{25}$$

$$\sum_{j \in A \cup \{n+1\}} D_{ij\rho} = c^+_{i\rho} \qquad \forall i \in A \cup \{0\}, \rho \in P. \tag{26}$$

Constraints (24) link our new flow variables $D_{ij\rho}$'s with the sequencing variables x_{ij}'s by enforcing the resource ρ flow from i to j to be zero as soon as $x_{ij} = 0$ (i does not precede j). If $x_{ij} = 1$ (i precedes j), the flow of resource ρ from activity i to activity j cannot be larger than the minimum between the amount produced by i and the amount consumed by j.

Constraints (25) require that the total amount of resource $\rho \in P$ received by activity j from previous activities, is equal to the amount this activity consumes during its processing. Note that we do not impose any constraint on the amount received by activity $n+1$, as unused produced (or initially available) resource amount can be stocked. Conversely, constraints (26) state that the quantity of resources sent by activity i to other activities, is equal to the amount it produces.

We set the resource production of activity 0 as: $c^+_{0\rho} = C_\rho$ for all $\rho \in P$, to model the resource initial levels.

3.3 On/off event-based formulation (OOE)

In contrast with time-indexed formulations, we now propose a formulation for the RCPSP/CPR that uses variables indexed by *events*. Inspired by previous papers on batch process problems [28] or on flow-shop problems [16], an extension of event-based formulation was proposed in [34, 22] for the RCPSP. We recall below some characteristics of the two existing event-based formulations for the standard RCPSP. Then, we shall introduce its extension to the RCPSP/CPR.

Zapata *et al.* [34] propose such an event-based formulation for a multimode RCPSP. Their formulation considers that an event occurs when an activity starts or ends. Transposed to the RCPSP, this model involves three types of binary variables per activity and per event. The proposition of [22] uses only *one* type of binary variable per activity and per event: a decision variable z_{ie} such that $z_{ie} = 1$ if and only if activity i starts at event e or is still in process at event e. This variable z_{ie} is similar to the variable introduced by Bowman [10] for the discrete-time based problem. A continuous variable t_e represents the date of event e, and one single extra continuous variable, $C_{\max}$, is used for the makespan. This model is called the *on/off event-based* formulation (noted **OOE**). When compared with the formulation of [34], the number of binary variables is divided by 3, which drastically reduces the search space for integer solving. The OOE formulation involves fewer variables when compared with the models indexed by time. Furthermore, it does *not* require as many big-M constants as the flow-based formulation. Finally, the OOE model involves a number of events that is lower than or equal to the number of activities: $|\mathcal{E}| \leq n$, where $\mathcal{E}$ denotes the set of events.

To illustrate this, consider Figure 1 which displays the schedule of $n = 5$ activities (1, 2, 3, 4, and 5) with a single resource. Here, *a priori* no more than $n = 5$ events are needed with the OOE model. In fact, it requires only three events (events 0, 1, and 2) for the specific feasible solution displayed in Figure 1. Since no activity starts at the end of the processing of activities 2, 3 and 5, it is not necessary to associate events to these times. Note that this minimal number of events cannot be known a priori unless we are able to prove that activities 1, 2 and 4 can start at the same time in the optimal solution. The number of events could be restricted *a priori* as a heuristic to accelerate the search but when no information is available, the worst-case situation still only involves n events.

For the illustrated solution, the event variables associated to activity 1 have values $z_{10} = z_{11} = 1$ and $z_{12} = 0$. Remark furthermore that in the CPR context, any resource produced at the end of activity 2 can be considered available only at the subsequent events (from event 1 on).

Figure 1: *On/off* event-based formulation: a feasible solution and its associated events

Let us now introduce an extension of the OOE model for the RCPSP/CPR (we shall continue to call OOE this extension). First, we model the renewable resource constraints in a straightforward manner. The management of storage resources, still ensured by the material-balance constraints, requires the introduction of the following continuous variables:

- $s_{e\rho}$: stock level of resource $\rho \in P$ at event e;
- $u_{ie\rho}$: the amount of resources $\rho \in P$ *consumed* by activity i at event e;
- $v_{ie\rho}$: the quantity of resources $\rho \in P$ *produced* by activity i at event e.

Here is the OOE formulation we are proposing:

$$\min C_{\max} \tag{27}$$

$$\sum_{e\in\mathcal{E}} z_{ie} \geq 1 \qquad \forall i \in A \tag{28}$$

$$C_{\max} \geq t_e + (z_{ie} - z_{i,e-1})p_i \qquad \forall e \in \mathcal{E}, \forall i \in A \tag{29}$$

$$t_0 = 0 \tag{30}$$

$$t_{e+1} \geq t_e \qquad \forall e \in \mathcal{E} : e \neq n-1 \tag{31}$$

$$t_f \geq t_e + ((z_{ie} - z_{i,e-1}) - (z_{if} - z_{i,f-1}) - 1)p_i \qquad \forall (e,f,i) \in \mathcal{E}^2 \times A, f > e \tag{32}$$

$$\sum_{e'=0}^{e-1} z_{ie'} \leq e(1 - (z_{ie} - z_{i,e-1})) \qquad \forall i \in A, \quad \forall e \in \mathcal{E} : e \neq 0 \tag{33}$$

$$\sum_{e'=e}^{n-1} z_{ie'} \leq (n-e)(1 + (z_{ie} - z_{i,e-1})) \qquad \forall i \in A, \quad \forall e \in \mathcal{E} : e \neq 0 \tag{34}$$

$$\sum_{e'=0}^{e} z_{je'} \leq (e+1)(1 - z_{ie}) \qquad \forall e \in \mathcal{E}, \forall (i,j) \in E \tag{35}$$

$$\sum_{i=0}^{n-1} b_{ik} z_{ie} \leq B_k \qquad \forall e \in \mathcal{E}, \forall k \in R \tag{36}$$

$$v_{ie\rho} \geq 0 \qquad \forall (e,i,\rho) \in \mathcal{E} \times A \times P \tag{37}$$

$$v_{ie\rho} \geq c^+_{i\rho}(z_{i,e-1} - z_{ie}) \qquad \forall (e,i,\rho) \in \mathcal{E} \times A \times P \tag{38}$$

$$v_{ie\rho} \leq c^+_{i\rho} z_{i,e-1} \qquad \forall (e,i,\rho) \in \mathcal{E} \times A \times P \tag{39}$$

$$v_{ie\rho} \leq c^+_{i\rho}(1 - z_{ie}) \qquad \forall (e,i,\rho) \in \mathcal{E} \times A \times P \tag{40}$$

$$u_{ie\rho} \geq 0 \qquad \forall (e,i,\rho) \in \mathcal{E} \times A \times P \tag{41}$$

$$u_{ie\rho} \geq c^-_{i\rho}(z_{ie} - z_{i,e-1}) \qquad \forall (e,i,\rho) \in \mathcal{E} \times A \times P \tag{42}$$

$$u_{ie\rho} \leq c^-_{i\rho} z_{ie} \qquad \forall (e,i,\rho) \in \mathcal{E} \times A \times P \tag{43}$$

$$u_{ie\rho} \leq c^-_{i\rho}(1 - z_{i,e-1}) \qquad \forall (e,i,\rho) \in \mathcal{E} \times A \times P \tag{44}$$

$$s_{e\rho} = s_{e-1,\rho} + \sum_{i\in A} v_{ie\rho} - \sum_{i\in A} u_{ie\rho} \qquad \forall (e,\rho) \in \mathcal{E} \times P, e > 0 \tag{45}$$

$$s_{0\rho} = C_\rho - \sum_{i\in A} u_{i0\rho} \qquad \forall \rho \in P \tag{46}$$

$$s_{e\rho} \geq 0 \qquad \forall (e,\rho) \in \mathcal{E} \times P \tag{47}$$

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

First, remark that $z_{ie} - z_{i,e-1}$ is equal to 1 if and only if activity i *starts* at event e, and it is -1 if and only if i *ends* at e. Constraints (28) are used to ensure that each activity is processed at least once during the project. Constraints (29) link the makespan with the event dates. Constraints (30) and (31) impose the event sequencing. The duration constraints (32) are used to link the binary optimization variables z_{ie}'s to the continuous optimization variables t_e's. They also ensure that, if activity i starts at event e and ends at event f, then the time difference between events f and e is at least the processing time of activity i ($t_f \geq t_e + p_i$). Constraints (33) and (34) are

called *contiguity constraints*. They ensure non-preemption since they force the events after which an activity is being processed to be adjacent. We refer to [22] for a detailed explanation of the contiguity constraints. Constraints (35) describe precedence constraints, modeling the implication $(z_{ie} = 1) \implies (\sum_{e'=0}^{e} z_{je} = 0)$ for each event e and for each $(i, j) \in E$. Constraints (36) are the renewable resource constraints limiting the total demand of activities in process at each event.

Constraints (37) to (40) amount to require, for any resource $\rho \in P$, that the value of variable $v_{ie\rho}$ must be equal to $c^{+}_{i\rho}$, if activity i finishes its process at event e, and is zero otherwise. Similarly, the resource consumption variable $u_{ie\rho}$ is determined by constraints (41) to (44). Finally, balance constraints (45) determine the stock level of $p \in P$ at event e, taking into account productions and consumptions at event e. Constraints (46) set the level of stock at event 0 to the initial level minus the consumed amount for each resource $\rho \in P$. Constraints (47) ensure that the level of each resource remains non negative.

We now propose **OOE_Prec**, a preprocessed variant of OOE, directly extended from the preprocessed variant introduced in [22] for the standard RCPSP. Roughly speaking, it is obtained from OOE 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. Symmetrically, we remove the last events during which the activity cannot, or does not need to, be in process because of its successors.

More precisely, let $A(i) = \{j \in A | (j, i) \in TE\}$ be the set of predecessors of activity i, and $D(i) = \{j \in A | (i, j) \in TE\}$ be the set of its successors.

Proposition 1 (extended from [22]) *There is 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: Considering n events, there is always an optimal solution for which activities begin at distinct events. In other words, for two distinct activities i and j, and for any pair of events e and f, we have:

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

It is important to note that this does not preclude $t_e = t_f$. Moreover, if j is a predecessor of i in E, then the event f assigned to j will occur strictly prior to the event e assigned to i: $t_e > t_f$. Proposition 1 is a consequence of these two observations. Thus, in our OOE model, we can set the following variables to 0:

$$z_{ie} = 0, \quad i \in A, e \in \{0, \ldots, |A(i)|\} \cup \{n - |D(i)| + 1, \ldots, n\}. \tag{48}$$

□

Thus, equations (48) can be used to eliminate decision variables before setting up the OOE formulation. We call this resulting formulation OOE_Prec.

4 Computational results

In this section, we compare the results obtained by the different formulations on a set of randomly generated problem instances.

The most popular instances used to test classical RCPSP propositions are KSD [20], BL [7], and PACK [11]. Among these instances, the most used are the KSD instances, available on the

PSPLIB web site [21]. Among them, we only focus on the 480 KSD30 instances ($n = 30$ activities). However, the KSD30 instances are generally not considered sufficiently cumulative. More precisely, in the KSD instance set, the hard-to-solve instances are all highly disjunctive, in the sense that there are many pairs of activities that cannot be processed in parallel. The highly cumulative instances of this set (where many activities can be processed in parallel) are known to be easy to solve [7]. The BL instances constitute a set of 39 instances involving between 19 to 25 activities, for 3 resources with demands generated randomly ranging from 0 to 60% of the total availability. The number of precedence relations varies from 15 to 45 ($|E| \in [15, 45]$). These instances are more "cumulative" but still easy to solve due to their size. Last, the PACK instances constitute a set of 55 instances with a small number of precedence relations, from 17 to 35 activities, and 3 resources. These are commonly considered as difficult-to-solve highly cumulative instances [22].

These three classes of instances contain short durations, very moderate scheduling horizons and homogeneous processing times. Thus, we shall also consider here the modified instance sets PACK_d and KSD15_d (see [22]) obtained from the PACK and KSD30 instances by increasing the range of processing times, since large scheduling horizons can be encountered in practical applications, as mentioned in Section 1. The authors of [22] obtained these modified instance sets by proceeding as follows.

To generate an instance B from an existing instance A, consider the following parameters a, b, x and y, such that $y \leq x$ and:

1. One selects the first x non-dummy activities of instance A (leaving aside other activities and the precedence constraints that are adjacent).
2. The selected activities without predecessors are connected to the dummy activity 0, and similarly, activities without successors are also connected to activity $x + 1$.
3. One randomly selects y of the x non-dummy activities, and their duration is multiplied by a coefficient $a + b$, where b is a randomly generated number between 0 and 1, and a is a multiplying factor duration.

For KSD15_d, the values for these parameters are: $x = 15$; $y = 7$; $a = 25$, and, for PACK_d they are $x = n$; $y = 10$; $a = 50$. The resulting durations are rounded to the nearest integer.

The resulting instance sets KSD15_d and PACK_d are publicly available (see [31]).

Since benchmark instances for the RCPSP/CPR are not available, we used the RCPSP instances (KSD30, PACK, BL, and KSD15_d, PACK_d), to which we made some further modifications to obtain the corresponding instance sets KSD30-CPR, PACK-CPR, BL-CPR, and KSD15_d-CPR, PACK_d-CPR. Typically, for each of these instances, the changes consist mainly in generating three new storage resources for each activity. Each of these new resources is characterized by the quantity consumed, $c^-_{i\rho}$ (respectively the quantity produced, $c^+_{i\rho}$) at the beginning (respectively end) of the processing of each activity, both randomly generated between 0 and 10. Each of these resources is also associated with an initial stock (original capacity) C_ρ. Given randomly-generated resource productions and consumptions, the tightness of the storage resource constraints can be further controlled by the chosen values of the initial stock. In [25], the tightness of the storage resource constraints is controlled by the resource strength indicator (that we denote $\mathrm{RS}^{\mathrm{CPR}}$ for storage resources) continuously varying from 0 to 1. The initial stock is given as a function of $\mathrm{RS}^{\mathrm{CPR}}$ by the following formula:

$$-C_\rho = \mathrm{RS}^{\mathrm{CPR}} \times \min_t \tilde{c}_\rho(ES, t) + (1 - \mathrm{RS}^{\mathrm{CPR}}) \times \sum_{i \in A} (c^+_{i\rho} - c^-_{i\rho}),$$

where $\tilde{c}_\rho(ES, t)$ denotes the stock level at time t when the activities are processed according to the earliest start schedule ES and no initial stock is available. It can be seen from this expression that when $\mathrm{RS}^{\mathrm{CPR}} = 0$, the initial stock has the minimum value required to accommodate the difference between the total consumption and the total production of a resource. Oppositely, when $\mathrm{RS}^{\mathrm{CPR}} = 1$, the expression ensures that the earliest start schedule is feasible with respect to storage constraints. Moreover, the renewable resource constraints are kept in all instances so that each instance involves the two considered resource types. In our setting, we used values of $\mathrm{RS}^{\mathrm{CPR}}$ described in Table 1 through its minimum value, its maximum value, the average value and the coefficient of variation (the ratio of the standard deviation to the mean, noted CV). The RCPSP/CPR instance sets that we just described have a CPR suffix in Table 1 and are available online [31]. The $\mathrm{RS}^{\mathrm{CPR}}$ values have been selected such that the average value is consistent with the values selected in the instances proposed in [25].

Table 1: Distribution of the storage resource strength values of the generated instances

$\mathrm{RS}^{\mathrm{CPR}}$	KSD30-CPR	BL-CPR	PACK-CPR	KSD15_d-CPR	PACK_d-CPR
min	0.7	0.73	0.73	0.7	0.73
max	1	1	1	1	0.97
av.	0.85	0.85	0.85	0.85	0.84
CV	7.8%	8.25%	7.67%	7.36%	7.6%

Tests were performed with IBM Cplex and Concert (version 12.2) on a dual core Intel x86-64 Xeon processor 5110 (1.60 GHz) with 1.96 GB RAM under GNU/Linux Centos release 5.8.

Furthermore, we also tested the constraint programming (CP) method proposed by Laborie [24]. This method initially solves satisfaction problems. Thus, we embedded it in a dichotomic search using the same initial lower and upper bounds as those of the MILP formulations. By reference to [24], we tested the method with criticality function 3, order criticality function B, and arc-consistency for temporal constraint enforcement. As before, 500 seconds are allocated to the method.

For each instance, the CP method was able either to find a feasible solution or to prove infeasibility. Hence, we separate the presentation of the results between the feasible and the infeasible instances. Table 2 displays the results we obtained on the feasible instances. Below each instance set name, we display the number of feasible instances.

In Table 2, *%Integer* is the percentage of instances for which a (non-necessarily optimal) integer solution was found within 500 seconds of CPU time. The percentage of instances for which an optimal solution was found is noted *%Optimal*. Column *%Dev Best* provides the average deviation of the obtained makespan (in the case where an integer solution is found) from the best solution among all formulations, in percentage of the best solution. Column *%Gap* provides the difference between the best upper bound and best lower bound found by the considered model (when an integer solution is found) in percentage of the lower bound. Column *#Nodes* gives the average number of nodes in the branch-and-bound tree for the case where an integer solution is found. Last, *Time* is the average time (in seconds) when a solution is found, which cannot exceed the time limit fixed to 500 seconds per instance. In the case where ILP solving fails due to memory overflow, "mem" is displayed in the column.

We first compare the performance of the different MILP formulations independently of the CP method. We observe that, in terms of number of (not necessarily optimal) integer solutions found, the OOE model or its variant OOE_Prec outperform the other formulations on instance sets KSD30_d-CPR and PACK_d-CPR. For the KSD15_d-CPR set, OOE, OOE_Prec and FCT always

Table 2: Results of MILP formulations and CP method on feasible RCPSP/CPR instances

Inst. set	Model	%Integer	%Optimal	%Dev Best	%Gap	#Nodes	Time (s)
KSD30-CPR	DDT	64	49	2.3	4.2	1086	178.7
(456/480)	DT	70	41	25.8	33.3	5701	254.8
	FCT	76	39	6.5	14.0	2554	314.6
	OOE	69	0	33.6	110	544	500.0
	OOE_Prec	92	1	18.5	64.4	3103	496.8
	CP	100	68	6.9	28.3	–	183.2
BL-CPR	DDT	100	95	0	0.003	893	60.9
(38/39)	DT	100	95	0.001	0.004	4403	84.2
	FCT	100	3	0.12	0.48	8100	489.7
	OOE	100	0	0.17	1.89	8097	500.0
	OOE_Prec	100	0	0.14	1.47	12479	500.0
	CP	100	39	0.08	0.3	–	348
PACK-CPR	DDT	93	64	3.2	11.0	2276	258.0
(55/55)	DT	100	24	5.0	37.0	16178	405.6
	FCT	13	0	1.6	59.3	12554	500.0
	OOE	89	0	11.6	2259.5	7750	500.0
	OOE_Prec	91	0	13.2	2699.2	5405	500.0
	CP	100	0	55.8	385.4	–	500.0
KSD15_d-CPR	DDT	22	15	22.9	26.4	238	363.1
(454/480)	DT	28	22	29.5	32.6	350	303.7
	FCT	100	91	0.1	3.5	2911	72.8
	OOE	99.8	33	1.3	31.5	8891	395.8
	OOE_Prec	99.8	41	0.5	20.4	13256	345.2
	CP	100	94	0.4	1.7	–	43.2
PACK_d-CPR	DDT	0	0	–	–	–	mem
(54/55)	DT	0	0	–	–	–	mem
	FCT	20	7.4	7.7	35.1	3338	325.3
	OOE	81	5.6	9.1	1758.2	1825	484.4
	OOE_Prec	80	5.6	6.1	1353.5	2771	480.3
	CP	100	3.7	22.8	234.5	–	486.0

find an integer solution and outperform the DDT and DT formulations. On the BL_CPR sets, all formulations find integer solutions. On the PACK-CPR set, DT and DDT obtain the largest number of integer solutions but are closely followed by the OOE and OOE_Prec formulations. On the other side, the FCT formulation fails in obtaining integer solutions in a majority of cases. Moreover, thanks to the preprocessing, OOE_Prec obtains (almost) always better results than OOE. Overall, these results allow us to conclude that OOE_Prec is, on average, the best among all methods at finding integer solutions. On instances with a high duration range, DDT and DT are obviously disqualified but FCT is only competitive with OOE_Prec on the KSD15_d-CPR instance set and collapses on the highly cumulative PACK-CPR and PACK_d-CPR instances.

In terms of optimal solutions found, OOE models are outperformed by DT and DDT for the instances with small duration ranges. This is due to the weak linear relaxations induced by the OOE models [22]. However, OOE models, as expected, are much better than DT and DDT for the instance sets KSD15_d-CPR and PACK_d-CPR involving high duration ranges. In fact, the

comparison between the time-indexed and the event-based formulations yields expected results (also in line with the ones obtained for the standard RCPSP [22]). Indeed, on the one hand, the linear relaxation of the time-indexed formulations is far better than the other ones. On the other hand, the number of variables of the time-indexed formulations explodes for instances with a high duration range. Consequently, as for the standard RCPSP, it is more meaningful to compare the two compact formulations OOE(_Prec) and FCT.

Again, the conclusions drawn for the standard RCPSP in [22] apply in the case of consumption and production of resources. It appears that OOE(_Prec) and FCT are complementary, considering the type of instances they are able to solve. The flow-based formulation FCT is superior to OOE(_Prec) for the KSD15_d-CPR instances, while the reverse applies to the PACK and PACK_d-CPR instances. For the KSD30-CPR instances, the two formulations cannot be compared, as OOE_Prec is better in terms of number of integer solutions found, while FCT is much better in terms of number of optimal solutions found. Nevertheless, we may conclude with the following suggestions for practitioners (similar to the ones obtained for the standard RCPSP [22]):

- For instances involving a small scheduling horizon, use time-indexed formulations.
- For instances with a high scheduling horizon and a high level of disjunctions, use the flow-based formulation.
- For instances with a high scheduling horizon and a high level of parallelism (highly cumulative instances), use the event-based formulation.

The CP method is clearly better than the MILP formulations on the KSD30-CPR and KSD15_d-CPR (although the FCT formulation obtains close results). CP is outperformed by the time-indexed formulations DDT and DT on the BL-CPR and the PACK-CPR instances. On the PACK_d-CPR instances, CP is the only method to find always a feasible solution but it is slightly worse than the OOE(_Prec) and FCT formulations for the number of optimum found. In terms of quality of the solution found on this set, the OOE formulations find in average better solutions than CP when both methods find a solution. Although the parameters of the CP method could be tuned to obtain better results, this underlines the competitiveness of the proposed ILP formulations.

Table 3 presents the results of the compared method on the infeasible instances. We first remark that for both sets BL-CPR and PACK_d-CPR, only 1 instance is infeasible and that, for PACK-CPR, all instances are feasible. For KSD30-CPR (KSD15_d-CPR) instances, 24 (26) instances are infeasible, respectively. For the same storage resource strength distribution, infeasibility appears almost only for the highly disjunctive instances. This could indicate that a higher level of parallelism allows a better combination of storage resource consumptions and productions. Because of the small number of infeasible instances, it is difficult to draw conclusions on the BL-CPR and PACK_d-CPR. However, we remark that the OOE formulations are the only ones to prove infeasibility of the PACK_d-CPR instance in a very short time. On the KSD30-CPR and KSD15_d-CPR, infeasibility is always proved at the root node for all formulations. It follows that the solving times are much smaller than for finding a feasible solution. This questions the hardness of the feasibility problem for the RCPSP/CPR (when infinite maximal stock is considered). On these sets, FCT is always the best formulation to prove infeasibility. As for the feasibility problem, discrete-time formulations are slightly better than event-based formulation for the KSD30-CPR set but the OOE_Prec formulation proves infeasibility faster. On KSD15_d-CPR, the OOE_Prec formulation is better than the time-indexed formulations with respect to both number of proved infeasibilities and CPU time. OOE_Prec is also faster than FCT. CP shows very good performance for proving infeasibility, which is consistent with the principle of constraint propagation techniques used within

Table 3: Results of MILP formulations and CP method on infeasible RCPSP/CPR instances

Inst. set	Model	%Proved inf	#Nodes	Time (s)
KSD30-CPR	DDT	63	0	61.2
(24/480)	DT	63	0	20.2
	FCT	96	0	41.8
	OOE	50	0	3.9
	OOE_Prec	58	0	0.5
	CP	100	–	0.3
BL-CPR	DDT	100	0	0.1
(1/39)	DT	100	0	0.02
	FCT	100	39	48.2
	OOE	100	0	0.05
	OOE_Prec	100	0	0.05
	CP	100	–	0.12
KSD15_d-CPR	DDT	8	0	20.1
(26/480)	DT	54	0	56.9
	FCT	100	0	4.9
	OOE	58	0	14.9
	OOE_Prec	65	0	0.05
	CP	100	–	0.2
PACK_d-CPR	DDT	0	–	–
(1/55)	DT	0	–	–
	FCT	0	–	–
	OOE	100	0	0.02
	OOE_Prec	100	0	0.03
	CP	100	–	24.3

the CP approach.

We now provide more insight on the performance of FCT and OOE_Prec, the best two formulations for the high duration range instances with production and consumption of resources. We select the smallest set (KSD15_d-CPR) to present the performance of these formulations in function of the instance characteristics.

We first evaluate the impact of the standard RCPSP hardness indicators. As explained in [20], the KSD instances were generated according to a full factorial design involving three indicators. Roughly, network complexity (NC) measures the density of the precedence constraints. The indicator NC has three different values indicating an increasing density: 1.5, 1.8 and 2.1. The resource factor (RF) gives the average number of required renewable resources. As there are four resources in the KSD set, RF has four possible values (0.25, 0.5, 0.75 and 1). Last, the resource strength indicator measures resource scarceness. We distinguish the renewable resource strength RS which takes four values (0.2, 0.5, 0.7 and 1) from the already-mentioned storage resource strength indicator RS^{CPR} whose values were generated according to the distribution detailed in Table 1. The smaller the value, the smaller the average resource availability is when compared to activity requirements. Tables 4–7 present for each formulation and parameter value: the percentage of optimal solutions found (*%Optimal*), the average number of branch-and-bound nodes (*#Nodes*) and the average CPU time (*Time*) needed for finding an optimal solution (and proving its optimality),

the average gap from the best solution (%Dev Best) when an integer solution is found, and the percentage of instances that are proved to be infeasible (%Inf).

Table 4: Impact of network complexity (KSD15_d-CPR)

NC	Model	%Optimal	#Nodes	Time (s)	%Dev Best	%Inf
1.5	FCT	90	1809	33.5	0	23
	OOE_Prec	49	3886	153.1	0.5	15
1.8	FCT	91	1593	27.9	0.1	35
	OOE_Prec	42	3498	214.1	0.5	19
2.1	FCT	93	2201	33.2	0.2	42
	OOE_Prec	32	5242	96.3	0.4	31

Table 5: Impact of resource factor (KSD15_d-CPR)

RF	Model	%Optimal	#Nodes	Time (s)	%Dev Best	%Inf
0.25	FCT	100	734	11.0	0	42
	OOE_Prec	45	4570	117.5	0.5	27
0.5	FCT	99	2302	32.9	0	19
	OOE_Prec	38	3262	105.8	0.31	12
0.75	FCT	87	2938	48.1	0.03	27
	OOE_Prec	37	3742	139.6	0.6	19
1	FCT	79	1542	36.5	0.3	12
	OOE_Prec	44	4661	149.5	0.5	8

Table 6: Impact of renewable resource strength (KSD15_d-CPR)

RS	Model	%Optimal	#Nodes	Time (s)	%Dev Best	%Inf
0.2	FCT	99	1748	31.2	0.2	19
	OOE_Prec	42	5360	139.3	0.4	12
0.5	FCT	91	2114	33.7	0	38
	OOE_Prec	36	4775	148.3	0.6	23
0.7	FCT	94	2098	37.4	0	0
	OOE_Prec	46	3674	126	0.5	0
1	FCT	91	1513	23.6	0.15	23
	OOE_Prec	40	2694	103.5	0.4	23

Table 7: Impact of storage resource strength (KSD15_d-CPR)

RS^{CPR}	Model	%Optimal	#Nodes	Time (s)	%Dev Best	%Inf
$[0.7, 0.8]$	FCT	90	1657	30.5	0.04	62
	OOE_Prec	36	4068	115.6	0.8	50
$(0.8, 0.9]$	FCT	91	1936	31.7	0.1	35
	OOE_Prec	43	4518	139.9	0.4	12
$(0.9, 1]$	FCT	93	2115	33.3	0.1	4
	OOE_Prec	47	2437	108.5	0.05	4

We can compare the results with the ones obtained for the standard RCPSP [18]. For the

standard RCPSP , the difficulty of the problem decreases as NC increases, since the increasing number of precedence constraints reduces the search space. In our case, formulation FCT obtains slightly more optimal solutions when NC increases, while the reverse occurs for OOE_Prec. Also, the deviation from the best solution increases as NC increases for FCT and is almost constant for OOE_Prec. We may conclude that the addition of storage resource reduces the impact of the precedence constraint density on the problem difficulty. However, increasing this density clearly increases the number of infeasible instances. For the standard RCPSP, increasing RF globally increases the problem difficulty. We see from Table 4 that this is also the case for the RCPSP/CPR. For the standard RCPSP, increasing RS also increases the problem difficulty. However, no clear impact of RS on the problem difficulty nor on the the problem infeasibility can be derived from Table 4.

More interestingly, Table 7 shows the impact of the tightness of the storage resource constraints. For both formulations (but to a lesser extent for the FCT formulation), the percentage of optimal solutions found increases as RS^{CPR} increases, which could be expected. The number of infeasible solutions is also clearly higher for small RS^{CPR}, which is consistent with the experiments carried out in [25].

5 Conclusions

In this paper, we extended three famous MILP formulations for the classical RCPSP to model the RCPSP with consumption and production of resources (RCPSP/CPR). We also proposed a new MILP formulation for the RCPSP/CPR by extending the on/off event-based model introduced in [22] for the RCPSP. Computational tests on both benchmark and new instances (sets involving increased ranges of processing times) provide encouraging results. These tests confirm those performed on the standard RCPSP, and show that the on/off event-based model is more appropriate than conventional models on instances involving large and disparate durations and allowing a high level of parallelism. On the other hand, the on/off event-based formulation has a poor relaxation which generally penalizes optimality proofs. In terms of solution quality, the proposed MILP formulations are also competitive when compared with a dedicated method from the literature. This motivates their use in an industrial environment. We have also illustrated through our experiments that decreasing storage resource tightness increases the problem difficulty except when the problem becomes infeasible, in which case infeasibility is proved rather quickly.

As in [24, 25, 27], a further extension of the RCPSP/CPR to problems involving maximum stock constraints could be considered. Another interesting extension would be to consider the case in which one has to follow a predefined stock profile. This is not a trivial issue for the on/off event-based model, as illustrated in Figure 1 where the resource produced at the end of the process of an activity is implicitly assumed to be delayed to the next event, thereby restricting the stock profile adaptability.

Finally, we believe that future work should concentrate on designing MILP approaches to the RCPSP/CPR that combine the advantages of both time-indexed and event-based formulations.

Acknowledgements

This project was partially funded by the CNRS Energy Interdisciplinary Program (PIE), GIMEP project 2008–2010, and partially supported by French National Research Agency (ANR) through COSINUS program (project ID4CS n°ANR-09-COSI-005).

The authors are grateful to Philippe Laborie who kindly provided his code for our computational comparisons and to Emmanuel Hébrard for help in conducting some experiments. We also wish to thank the anonymous referees for their numerous constructive remarks.

This research was initiated while the first author was with CIRRELT, Université de Montréal, Canada.

References

[1] M. H. Agha, R. Thery, G. Hetreux, A. Haït and J.-M. Le Lann, "Integrated production and utility system approach for optimizing industrial unit operations", *Energy*, 35(2):611–627, 2010.

[2] 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.

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

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

[5] 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.

[6] E. Balas, "Project scheduling with resource constraints", in E.M.L. Beale, (Ed.), *Applications of Mathematical Programming Techniques*, pages 187-200, American Elsevier, 1970.

[7] 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.

[8] 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.

[9] H. Bouly, J. Carlier, A. Moukrim, and M. Russo, "Solving RCPSP with resources production possibility by tasks", In: *MHOSI'2005, 24-26 April*, 2005.

[10] E.H. Bowman, "The schedule-sequencing problem", *Operations Research*, 7:621– 624, 1959.

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

[12] J. Carlier, A. Moukrim, and H. Xu, "The project scheduling problem with production and consumption of resources: A list-scheduling based algorithm", *Discrete Applied Mathematics*, 157(17):3631–3642, 2009.

[13] P. M. Castro and . E. Grossmann, "An efficient MILP model for the short-term scheduling of single stage batch plants", *Computers and Chemical Engineering* 30:1003–1018, 2006.

[14] 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.

[15] 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.

[16] 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.

[17] C. A. Floudas and X. Lin, "Mixed integer linear programming in process scheduling: Modeling, algorithms, and applications", *Annals of Operations Research*, 139:131–162, 2005.

[18] W. Herroelen, B. De Reyck, and E. Demeulemeester, "Resource-constrained project scheduling: A survey of recent developments", *Computers and Operations Research*, 25:279–302, 1998.

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

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

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

[22] O. Koné, C. Artigues, P. Lopez, and M. Mongeau, "Event-based MILP models for resource-constrained project scheduling problems", *Computers & Operations Research*, 38(1):3–13, 2011.

[23] J.B. Lasserre and M. Queyranne, "Generic scheduling polyhedra and a new mixed-integer formulation for single-machine scheduling", *Proceedings of the 2nd Integer Programming and Combinatorial Optimization Conference, IPCO* , pages 136–149, 1992.

[24] P. Laborie, "Algorithms for propagating resource constraints in a planning and scheduling: Existing approaches and new results", *Artificial Intelligence*, 143:151–188, 2003.

[25] K. Neumann and C. Schwindt, "Project scheduling with inventory constraints", *Mathematical Methods of Operations Research*, 56:513–533, 2002.

[26] K. Neumann and C. Schwindt, N. Trautmann, "Advanced production scheduling for batch plants in process industries", *OR Spectrum* 24:251–279, 2002.

[27] K. Neumann, C. Schwindt, and J. Zimmermann, *Project Scheduling with Time Windows and Scarce Resources*, Springer, 2003.

[28] J. M. Pinto and I. E. Grossmann, "A continuous time mixed integer linear programming model for short term scheduling of multistage batch plants", *Industrial & Engineering Chemistry Research*, 34(9):3037–3051, 1995.

[29] M. E. Pfund, S. J. Mason and J. W. Fowler, "Semiconductor manufacturing scheduling and dispatching. State of the art and survey of needs", in *Handbook of Production Scheduling*, International Series in Operations Research & Management Science 89, 213–241, Springer, 2006.

[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] High-duration RCPSP instances with consumption and production of resources. `http://www2.laas.fr/laas/files/MOGISA/RCPSP/instances/high_duration_range_with_production.zip`

[32] C. Schwindt, and N. Trautmann, "Batch scheduling in process industries: An application of resource-constrained project scheduling", *OR Spektrum* 22:501–524, 2000.

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

[34] 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.