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[4,5]

[1] CIRRELT, Université de Montréal,
C.P. 6128, succ. Centre-Ville, Montreal, QC, Canada H3C 3J7.
[2] CNRS ; LAAS ; 7 avenue du colonel Roche, F-31077 Toulouse, France.
[3] Université de Toulouse ; UPS, INSA, INP, ISAE ; UT1, UTM, LAAS ; F-31077 Toulouse, France.
[4] Université de Toulouse ; UPS, INSA, UT1, UTM ; Institut de Mathématiques de Toulouse, F-31062 Toulouse, France.
[5] CNRS ; Institut de Mathématiques de Toulouse UMR 5219 ; F-31062 Toulouse, France.
e-mails: mr.okone@gmail.com, {artigues,lopez}@laas.fr, mongeau@math.univ-toulouse.fr

Abstract

This paper addresses an extension of the resource-constrained project scheduling problem, which takes into account non-renewable resources. 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.

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 *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 [22] who formalize the problem of project scheduling with inventory constraints and generalized temporal constraints, including both renewable and non-renewable resources. The authors propose a branch-and-bound approach associated with a beam-search heuristic to solve the problem. Carlier *et al.* [11] include generalized time-lag constraints and propose a new list algorithm. Laborie [21] defines the concept of Resource Temporal Network, offering a powerful modeling and providing a support for other authors such as Bouley *et al.* [8]. For more detail on the state of the art in RCPSP/CPR, we refer the reader to [8, 21, 11].

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 inside 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 [19], 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 in industrial applications. For instance, for scheduling problems occurring in the process industry, such as the RCPSP/CPR, we show in the current paper that MILP models are well suited for modeling consumption and production of specific utilities such as energy resources. Numerous research papers aim at proposing MILP formulations for batch scheduling in the process industry, see [15, 27].

There exist roughly three categories of MILP formulations for the standard resource-constrained project scheduling problem. Note that we do not consider here the formulations involving an exponential number of variables and/or constraints. They can only be solved through branch-and price and/or branch-and-cut techniques, which fall out of the scope of the paper, considering only direct solving with an MILP solver.

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. Assuming integer activity durations and possible start times, 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.* [24] and refined in Christofides *et al.* [12]. They generally have a good LP relaxation but as a counterpart they involve a pseudo-polynomial number of variables, since T depends on activity durations.

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 [5, 2] 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 [3]). 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}, where either activity i starts, ends or is in process at event e. Each event corresponds to the start or the end time of an activity. Continuous variables t_e are used for event start times. As there is a polynomial number of events, these formulations are also compact. Originally proposed for single machine scheduling in [20], they were extended to more complex problems including flow-shop [14], batch sequencing [15], and variants of the RCPSP [27, 19]. Although these formulations do not involve big-M constraints, the experiments carried out in [19] show that they yield poor LP relaxations.

For integer solving, it is shown in [19] that flow-based and event-based formulations can be the only alternative for RCPSP instances with high duration ranges. 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.

We describe in detail in Section 2 the RCPSP/CPR. Section 3 is dedicated to our proposals.

In Sections 3.1 and 3.2, we propose two MILP models generalized to model the RCPSP/CPR. We then present one model based on the notion of event (Section 3.3). 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 consumption. 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 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 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 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 [7, 26].

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 resources. These resources, described sometimes as *cumulative* by some authors [23], 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_{ip}^- units of resource p at the beginning of its processing, and produces c_{ip}^+ units at the end of its processing, where both c_{ip}^- and c_{ip}^+ are (given) input data of the problem. Furthermore, the total amount of each resource must remain non-negative throughout the scheduling horizon.

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

1. If $c_{ip}^- = c_{ip}^+$, then p is simply a renewable resource, as in the basic RCPSP.
2. If $c_{ip}^- \neq c_{ip}^+$:
   (a) if $c_{ip}^- = 0$ and $c_{ip}^+ > 0$ (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 $c_{ip}^- < c_{ip}^+$, then we first consume a quantity of a given product before producing more.
   (c) if $c_{ip}^- > c_{ip}^+$, then p is a non-renewable resource such as a 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 adaptation formulation of the RCPSP to the RCPSP/CPR problem. Finally, we introduce an MILP formulation based on the concept of event [15, 19, 27].

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 (more detail in [13, 16]). Hence the time interval $[ES_i, LS_i]$ represents the time window during which activity i can start.

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 [24], and the disaggregated discrete-time formulation (DDT) proposed by Christofides in 1987 [12]. 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:

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

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

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

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

$$x_{00} = 1$$

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

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

The values of the above-mentioned starting-time variables 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 (8) impose non-preemption of the project activities. Constraints (7) mean that the start time of every activity occurs between its earliest start time and its latest start time. They are actually used in a preprocessing phase. 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:

$$s_{0p} = C_p - \sum_{i=1}^{n} x_{i0} c_{ip}^{-} \qquad \forall p \in P \tag{9}$$

$$s_{tp} = s_{t-1,p} - \sum_{i=1}^{n} x_{it} c_{ip}^{-} + \sum_{i=1}^{n} x_{i,t-p_i} c_{ip}^{+} \qquad \forall (t,p) \in H \times P, t > 0 \tag{10}$$

$$s_{tp} \geq 0 \qquad \forall (t,p) \in H \times P. \tag{11}$$

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

Constraints (11) enforce non negativity of each resource level.

Note that the intermediate decision variables s_{tp} can be substituted out by their value as a function of x_{it}.

As previously announced, the DDT model proposed by Christofides [12] 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 requires one constraint for each pair of activities, for each precedence relation, *and* for every time of the scheduling horizon:

$$\sum_{\tau=t}^{LS_i} x_{i\tau} + \sum_{\tau=ES_j}^{\min(LS_j,t+p_i-1)} x_{j\tau} \leq 1, \qquad \forall (i,j) \in E, \forall t \in \{ES_i, LS_i\}. \tag{12}$$

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 (9) to (11).

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 bounds by linear relaxation on the classical instances of the RCPSP (see [4]).

3.2 Flow-based continuous-time formulation

Inspired by the work of Balas *et al.* [5], and on the basis of the formulation of Alvarez-Valdes and Tamarit [1], Artigues *et al.* [3] 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. 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-variables 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 considered problem.

$$\min S_{n+1} \tag{13}$$

$$x_{ij} + x_{ji} \leq 1, \qquad \forall (i,j) \in (A \cup \{0, n+1\})^2, i < j \tag{14}$$

$$x_{ik} \geq x_{ij} + x_{jk} - 1 \qquad \forall (i,j,k) \in (A \cup \{0, n+1\})^3 \tag{15}$$

$$S_j - S_i \geq -M_{ij} + (p_i + M_{ij}) x_{ij} \qquad \forall (i,j) \in (A \cup \{0, n+1\})^2 \tag{16}$$

$$f_{ijk} \leq \min(\tilde{b}_{ik}, \tilde{b}_{jk}) x_{ij} \qquad \forall (i,j) \in (A \cup \{0\} \times A \cup \{n+1\}), \forall k \in R \tag{17}$$

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

$$\sum_{i \in A \cup \{0, n+1\}} f_{ijk} = \tilde{b}_{jk} \qquad \forall i \in A \cup \{0, n+1\}, \forall k \in R \tag{19}$$

$$f_{n+1,0,k} = B_k \qquad \forall k \in R \tag{20}$$

$$x_{ij} = 1 \qquad \forall (i,j) \in TE \tag{21}$$

$$x_{ji} = 0 \qquad \forall (i,j) \neq TE \tag{22}$$

$$f_{ijk} \geq 0 \qquad \forall (i,j) \in (A \cup \{0, n+1\})^2, \forall k \in R \tag{23}$$

$$S_0 = 0 \tag{24}$$

$$ES_i \leq S_i \leq LS_i \qquad \forall i \in A \cup \{n+1\} \tag{25}$$

$$x_{ij} \in \{0, 1\} \qquad \forall (i,j) \in (A \cup \{0, n+1\})^2, \tag{26}$$

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 (14) and (15) are redundant constraints (valid inequalities expressing logical conditions on sequencing variables). Constraints (16) link the start time variables and the sequential binary variables. Constraints (17) link flow variables and x_{ij} variables. Constraints (18-20) are resource flow-conservation constraints. Preprocessing constraints (21) and (22) set the preexisting precedence constraints. Constraints (25) 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] show that this formulation, ranked among the *compact* models, produces bad relaxations due to the use of big-M constants in constraints (16). 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, hence 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 (27) 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 non-renewable resources are

also managed through flows, as follows:

$$D_{ijp} \leq \min\left(c^+_{ip}, c^-_{jp}\right) x_{i,j} \qquad \forall (i,j) \in A^2, p \in P \tag{27}$$

$$\sum_{i \in A \cup \{0\}} D_{ijp} = c^-_{jp} \qquad \forall j \in A \cup \{n+1\}, p \in P \tag{28}$$

$$\sum_{j \in A \cup \{n+1\}} D_{ijp} = c^+_{ip} \qquad \forall i \in A \cup \{0\}, p \in P. \tag{29}$$

Constraints (27) link our new flow variables D_{ijp} with sequencing variables x_{ij} by enforcing the resource p 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 p 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 (28) require that the total amount of resource $p \in P$ received by activity j from previous activities, is equal to the amount this resource consumes during its processing. Conversely, constraints (29) 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 such that $c^+_{0p} = C_p, \forall p \in P$, to model the resource initial levels. Since unused produced (or initially available) resource amount can be stocked, we set the resource consumption of activity $n+1$ to a sufficiently large constant level $c^-_{(n+1)p} = m, \forall p \in P$. Last, resource production of activity $n+1$ and resource consumption of activity 0 are set to $c^-_{n+1,p} = c^+_{n+1,p} = 0, \forall p \in P$.

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 [15] or on flow-shop problems [14], an extension of event-based formulation was proposed in [27, 19] 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.* [27] 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 [19] 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 [9] 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**). Compared to the formulation of [27], the number of binary variables is divided by 3, which drastically reduces the search space for integer solving. The OOE formulation involves fewer variables compared to the models indexed by time and it does *not* require as many big-M constants as the flow-based formulation. Furthermore, the OOE model involves a number of events that is lower than or equal to the number of activities: $|\mathcal{E}| \leq n$, where $\mathcal{E}$ is the set of events. Figure 1 displays a schedule of four activities (g, h, i, and j) with a single resource. It requires only three events (0,1, and 2) with the OOE model. Since no activity starts at the end of the processing of activities i and j, it is not necessary to associate events to these times.

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 straightfor-

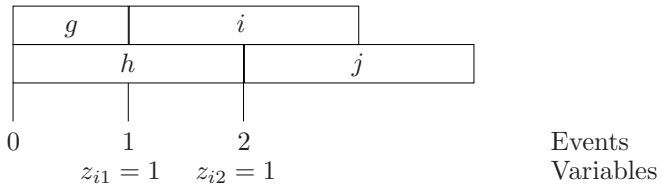

Figure 1: Example of variables associated to activity i and events in the *on/off* event-based formulation

ward manner. The management of non-renewable resources, still ensured by the material-balance constraints, requires the introduction of the following continuous variables:

- s_{ep}: the level of stock for resource $p \in P$ at event e;
- p_{iep}: the quantity of resources $p \in P$ *produced* by activity i at event e;
- u_{iep}: the amount of resources $p \in P$ *consumed* by activity i at event e.

Here is the OOE formulation we are proposing:

$$\min C_{\max} \tag{30}$$

$$\sum_{e\in\mathcal{E}} z_{ie} \geq 1 \qquad \forall i \in A \tag{31}$$

$$C_{\max} \geq t_e + (z_{ie} - z_{i(e-1)})p_i \qquad \forall e \in \mathcal{E}, \forall i \in A \tag{32}$$

$$t_0 = 0 \tag{33}$$

$$t_{e+1} \geq t_e \qquad \forall e \neq n-1 \in \mathcal{E} \tag{34}$$

$$t_f \geq t_e + ((z_{ie} - z_{i,e-1}) - (z_{if} - z_{i,f-1}) - 1)p_i \qquad \forall (e,f,i) \in \mathcal{E}^2 \times A, f > e \tag{35}$$

$$\sum_{e'=0}^{e-1} z_{ie'} \leq e(1 - (z_{ie} - z_{i(e-1)})) \qquad \forall i \neq A, \quad \forall e \neq 0 \in \mathcal{E} \tag{36}$$

$$\sum_{e'=e}^{n-1} z_{ie'} \leq (n-e)(1 + (z_{ie} - z_{i(e-1)})) \qquad \forall i \neq A, \quad \forall e \neq 0 \in \mathcal{E} \tag{37}$$

$$z_{ie} + \sum_{e'=0}^{e} z_{je'} \leq 1 + (1 - z_{ie})e \qquad \forall e \in \mathcal{E}, \forall (i,j) \in E \tag{38}$$

$$\sum_{i=0}^{n-1} b_{ik} z_{ie} \leq B_k \qquad \forall e \in \mathcal{E}, \forall k \in R \tag{39}$$

$$p_{iep} \geq 0 \qquad \forall (e,i,p) \in \mathcal{E} \times A \times P \tag{40}$$

$$p_{iep} \geq c^+_{ip}(z_{i,e-1} - z_{i,e}) \qquad \forall (e,i,p) \in \mathcal{E} \times A \times P \tag{41}$$

$$p_{iep} \leq c^+_{ip} z_{i,e-1} \qquad \forall (e,i,p) \in \mathcal{E} \times A \times P \tag{42}$$

$$p_{iep} \leq c^+_{ip}(1 - z_{i,e}) \qquad \forall (e,i,p) \in \mathcal{E} \times A \times P \tag{43}$$

$$u_{iep} \geq 0 \qquad \forall (e,i,p) \in \mathcal{E} \times A \times P \tag{44}$$

$$u_{iep} \geq c^-_{ip}(z_{i,e} - z_{i,e-1}) \qquad \forall (e,i,p) \in \mathcal{E} \times A \times P \tag{45}$$

$$u_{iep} \leq c^-_{ip} z_{ie} \qquad \forall (e,i,p) \in \mathcal{E} \times A \times P \tag{46}$$

$$u_{iep} \leq c^-_{ip}(1 - z_{i,e-1}) \qquad \forall (e,i,p) \in \mathcal{E} \times A \times P \tag{47}$$

$$s_{ep} = s_{e-1,p} + \sum_{i\in A} p_{iep} - \sum_{i\in A} u_{iep} \qquad \forall (e,p) \in \mathcal{E} \times P, e > 0 \tag{48}$$

$$s_{0p} = C_p - \sum_{i\in A} u_{i0p} \qquad \forall p \in P \tag{49}$$

$$s_{ep} \geq 0 \qquad \forall (e,p) \in \mathcal{E} \times P \tag{50}$$

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

First, remark $z_{ie} - z_{i,e-1}$ is equal to 1 if and only if activity i *starts* at event e , and -1 if and only if i *ends* at e. Constraints (31) are used to ensure that each activity is processed at least once during the project. Constraints (32) link the makespan with the event dates. Constraints (33) and (34) impose the event sequencing. The duration constraints (35) are used to link the binary optimization variables z_{ie} to the continuous optimization variables t_e, and to 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 (36) and (37) are called contiguity constraints and ensure non-preemption since they force the events after which an activity is being processed to be adjacent. We refer to [19] for a detailed explanation of the

contiguity constraints. Constraints (38) 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 (39) are the renewable resource constraints limiting the total demand of activities in process at each event.

Constraints (40) to (43) amount to require that the value of variable p_{iep} must be equal to c_{ip}^{+}, if activity i finishes its process at event e, for any resource $p \in P$ and is zero otherwise. Similarly, the resource consumption variable u_{iep} is determined by constraints (44) to (47). Finally, balance constraints (48) determine the level of $p \in P$ at event e, taking into account productions and consumptions at event e. Constraints (49) set the level of stock at event 0 to the initial level minus the consumed amount for each resource $p \in P$. Constraints (50) 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 proposed in [19] 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.

Let $A(i)$ be the set of predecessors of activity i in the precedence graph G, and $D(i)$ be the set of its successors.

Proposition 1 (extended from [19]) *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, 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 G, then the event f assigned to j will be strictly prior to the event e assigned to i: $t_e > t_f$. Proposition 1 is a consequence of these two observations. Thus, we can set the following variables to 0:

$$z_{ie} = 0, \quad i \in A, e \in \{0, \ldots, |A(i)|\} \cup \{n - |D(i)| + 1, \ldots, n\}. \tag{51}$$

□

Thus, equations (51) 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 event formulations OOE and OOE_Prec with those obtained by the time-indexed formulations DT and DDT, and the flow-based continuous-time formulation FCT on a set of randomly generated problem instances.

The most popular instances used to test classical RCPSP propositions are KSD [17], BL [6], and PACK [10]. Among these instances, the most used are the KSD instances, available on the PSPLIB web site [18]. Among them, we only focus on the KSD30 ($n = 30$ activities) and we select the first 100 out of the 480 instances involved in this class. However, the KSD30 instances are 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. Highly cumulative instances (where many activities can be processed in parallel) are known to be easy to solve [6]. The BL instances are 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, PACK instances are a set of 55 instances with a small number of precedence relations, from 17 to 35 activities, and 3 resources. These are considered as difficult-to-solve highly cumulative instances [19].

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 [19]) obtained from the PACK and KSD30 instances by increasing the range of processing times, as large scheduling horizon and disparate processing times are common in the process industry [15]. The authors of [19] 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. We randomly select 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, and so are the increased values of durations.

The resulting instance sets KSD15_d and PACK_d are publicly available (see [25]).

Since benchmark instances for the RCPSP/CPR are not available, we used the classical RCPSP instances (KSD30, PACK, BL, and KSD15_d, PACK_d), to which we made some further modifications. Typically, for each of these instances, changes consist mainly in generating for each activity, three new resources (consumed and produced). Each of these new resources (so-called non-renewable) is characterized by the quantity consumed, c_{ip}^{-} (respectively the quantity produced, c_{ip}^{+}) at the beginning (resp. 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_p. These different characteristics of the new (non-renewable) resources are randomly generated so that :

- No consumption should be greater than the initial stock. This not only ensures that any activity can start the project, but especially that the project can start.
- The total sum of quantities produced is greater than the total sum of the quantities consumed to avoid increasing disjunctions.

- All instances are feasible (when an instance is found infeasible, it is eliminated).

The RCPSP/CPR instance sets that we just described have a CPR suffix in Table 1 and are available online [25]. Table 1 displays the results we obtained with ILOG-CPLEX (version 11) on a Xeon 5110 biprocessor Dell PC clocked at 1.6 GHz with 4 GB RAM, running Linux Fedora as operating system.

Table 1: Comparative results RCPSP/CPR instances

Instance set	Model	%integer	%Optimal	BestSol gap	ΔCPM gap	Time
KSD30-CPR	DT	84	63	10	25	53
	DDT	82	71	0.13	8	84
	OOE_Prec	78	1	53	76	416
	FCT	69	20	47	70	289
	OOE	1	0	66	66	
PACK-CPR	OOE	95	2	21	278	111
	OOE_Prec	93	4	13	258	449
	DT	91	18	48	365	127
	DDT	47	33	1	246	168
	FCT	9	0	6	96	
BL-CPR	OOE_Prec	100	0	18	73	
	DDT	95	49	1	48	126
	DT	87	38	50	120	109
	OOE	74	0	28	88	
	FCT	21	0	27	73	
KSD15_d-CPR	FCT	100	94	0.12	10	18
	OOE_Prec	100	81	0.05	10	31
	OOE	100	80	0.10	10	62
	DT	0	0			
	DDT	0	0			
PACK_d-CPR	OOE_Prec	96	5	2	250	252
	OOE	96	5	6	264	321
	FCT	5	2	0	44	100
	DT	0	0			
	DDT	0	0			

In Table 1, *%integer* is the percentage of instances for which a (non-necessarily optimal) integer solution was found within 500 seconds of CPU time. We call these instances the "solved" (not necessarily to optimality) instances. The percentage of instances for which an optimal solution was found is noted *%Optimal*. *BestSol gap* represents the average deviation percentage for solved instances from the value of the best known solution. Δ*CPM gap* is the average deviation percentage (for solved instances) from the critical-path-method lower bound; and finally, *Time* is the average time (in seconds) to find an optimal solution (average over the cases where optimal were found).

We observe that, in terms of number of (not necessarily optimal) integer solutions found, the OOE model and its variant OOE_Prec achieve the best performance on three instance sets (PACK-CPR, BL-CPR, and PACK_d-CPR), the second best performance on KSD15_d-CPR, and the third best performance on KSD30-CPR. Overall, these results allow us to conclude that OOE and its variant OOE_Prec are the best at finding integer solutions. Moreover, thanks to the preprocessing, OOE_Prec obtains almost always better results than OOE.

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 relaxations of the OOE models [19]. However, OOE models are incomparably better than DT and DDT for the instance KSD15_d-CPR and PACK_d-CPR sets involving high duration ranges.

The comparison between the time-indexed and the event-based formulations yields in fact expected results (also in line with the ones obtained for the standard RCPSP [19]). Indeed, on the one hand, the LP-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 [19] 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-PC instances while the reverse applies to the PACK and PACK_d-PC instances. For the KSD instances, the two formulations cannot be compared, as OOE_Prec is slightly 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 [19]) :

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

Table 2 below now shows the impact of taking into account consumption and production of resource on the performance of different formulations compared to data from the classical RCPSP. This can also be seen as a preliminary analysis of the sensibility of the formulations studied, since it is the variation in performance due to the disruption of input data. More precisely, if $*\in \{KSD30, PACK, BL, KSD15\_d, PACK\_d\}$, we denote

- Δ_Integer% := %integer on the *-CPR instance set − % integer on the * instance set,
- Δ_Optimal% := %Optimal on the *-CPR instance set − % Optimal on the *-CPR instance set.

It appears that adding non-renewable resources has a strong negative impact on the performance of the models DDT and DT in terms of integer solutions found. For example, DT decreases from 100 % of integer solutions found for KSD15 to 45 % for KSD15-CPR. On the other hand, OOE and OOE_Prec improve strongly their performance when applied to RCPSP/CPR instance sets. For example, the percentage of integer solutions for OOE_Prec goes from 54 % for BL to 100 %, for BL-CPR. Regarding the performance in terms of number of optimal solutions found, although all formulations result in declines, DT and DDT remain superior.

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

Table 2: Impact of resource consumption and production

		Δ_Integer%	Δ_Optimal%
KSD30-CPR	**DT**	-2	-15
	DDT	-9	-11
	OOE_Prec	32	-29
	FCT	2	-42
	OOE	-32	-24
PACK-CPR	**DT**	6	-37
	DDT	-48	-43
	OOE_Prec	38	-1
	FCT	7	0
	OOE	46	-7
BL-CPR	**DT**	-13	-62
	DDT	-5	-51
	OOE_Prec	46	0
	FCT	0	-3
	OOE	25	0
KSD15_d-CPR	**DT**	-55	-54
	DDT	-1	-1
	OOE_Prec	0	-5
	FCT	1	0
	OOE	1	-3
PACK_d-CPR	**DT**	0	0
	DDT	0	0
	OOE_Prec	36	-9
	FCT	-2	-5
	OOE	36	-13

MILP formulation for the RCPSP/CPR by extending the on/off event-based model introduced in [19] for the RCPSP. Computational tests on both benchmark and application-oriented instances provide encouraging results. These tests confirm those on the standard RCPSP, and show that, in addition to being more appropriate than conventional models on instances involving large and disparate durations, our on/off event-based model solves the largest number of instances, whatever their features. It is also clear from this study that our proposals are even more effective in solving the RCPSP/CPR in terms of number of integer solutions found, better than those of the standard RCPSP.

We believe that future work should consider designing MILP approaches to the RCPSP/CPR that combine the advantages of time-indexed and event-based formulations.

Acknowledgment

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).

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

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

[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] 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.

[12] 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.

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

[14] 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.

[15] 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.

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

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

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

[19] 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.

[20] 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.

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

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

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

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

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

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

[27] 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.