Energetic reasoning and mixed-integer linear programming for scheduling with a continuous resource and linear efficiency functions

Margaux Nattaf, Christian Artigues, Pierre Lopez, David Rivreau

Energetic reasoning and mixed-integer linear programming for scheduling with a continuous resource and linear efficiency functions

Margaux Nattaf[1,3], Christian Artigues[2,3], Pierre Lopez[2,3], and David Rivreau [4]

[1]Univ. de Toulouse, UPS, F-31400 Toulouse France, *Email: mnattaf@laas.fr*
[2]Univ. de Toulouse, LAAS, F-31400 Toulouse France, *Email: {artigues,lopez}@laas.fr*
[3]CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
[4]LUNAM Université, Université Catholique de l'Ouest, LISA, 3 Place André Leroy, F-49008 Angers, France,
Email : david.rivreau@uco.fr

Abstract

This paper addresses a scheduling problem with a continuously-divisible, cumulative and renewable resource with limited capacity. During its processing, each task consumes a part of this resource, which lies between a minimum and a maximum requirement. A task is finished when a certain amount of energy is received by it within its time window. This energy is received via the resource and an amount of resource is converted into an amount of energy with a non-decreasing and continuous function. The goal is to find a feasible schedule, which is already NP-complete, and then to minimize the resource consumption. For the case where all functions are linear, we present two new Mixed Integer Linear Programs (MILP), as well as improvements of an existing formulation. We also present a detailed version of the adaptation of the well-known "left-shift/right-shift" satisfiability test for the cumulative constraint and the associated time-window adjustments to our problem. For this test, three ways of computing relevant intervals are described. Finally, a hybrid branch-and-bound using both the satisfiability test and the MILP is presented with a new heuristic for choosing the variable on which the branching is done. Computational experiments on randomly generated instances are reported in order to compare all of these solution methods.

Keywords. continuous scheduling, continuous resources, linear efficiency functions, energy constraints, energetic reasoning, branching scheme, mixed-integer programming

1 Introduction

This paper deals with a scheduling problem involving a set of tasks and a continuously-divisible renewable resource of limited capacity shared by the tasks. Each task must be processed between a release date and a due date. During its time window, each task must receive a given total amount of resource units that we will refer to as a required energy amount. The resource is of the cumulative type[1], in the sense that, at each time, each task requires a certain intensity of the resource and that the sum of the intensities cannot exceed the resource capacity. We consider the case where the resource amount (intensity) that a task requires during its processing is not fixed. More precisely, the resource usage is a continuous function of time that must be determined. Once the task is started this resource usage must lie within an

[1]Note that some authors define a cumulative resource with other meanings, such as storage resource [18]. In this paper a resource is called cumulative in the Constraint Programming literature sense [3, 8], i.e. a synonym of a renewable resource with an availability larger than or equal to one.

interval until the total required energy has been received by the task. Furthermore we consider that the total energy received by the task is not equal to the total amount of the resource used by the task. Instead we have efficiency functions[2] that translate the resource usage into energy. Consequently, the duration of the task is not fixed neither but is determined by the resource usage function as the task is finished once the necessary energy has been received.

As typical examples, we cite energy-consuming production scheduling problems. In [1], a foundry application is presented where a metal is melted in induction furnaces. The electrical power of the furnaces that can be adjusted at any time to avoid exceeding a maximum prescribed power limit, can be seen as a continuous function of time to be determined. However the function must lie within a limit; thus, a minimum and a maximum power level must be satisfied for the melting operation. Moreover, the amount of energy received is not in reality proportional to the power used, as efficiency functions must be considered for the furnaces. Finally the duration of the melting operation can be stopped once the necessary energy has been received, depending of the selected power function. Due to the complexity of the problem, the solution method proposed in [1] considers a time discretization, which can lead to sub-optimal or infeasible solutions by over-constraining the problem, as shown in Section 3.1. Furthermore, efficiency functions were not considered. In a continuous time setting but still without considering the efficiency functions, constraint propagation algorithms based on the energetic reasoning concept were proposed in [2]. Note that when non-identity efficiency functions are considered, the total amount of energetic resource used is no more constant and can be an objective to minimize. In this paper we extend the constraint propagation algorithm to the case of linear efficiency functions. We also perform an analysis of the structural properties of the problem and we propose mixed-integer linear programming (MILP) formulations as well as a hybrid branch-and-bound method.

This paper extends the results presented in [16, 17] with the consideration of the preemptive case, two new mixed-integer programming formulations, as well as improvements of the existing one, full details on the energetic reasoning computations, consideration of the total resource usage minimization objective and extended computational results.

The problem definition, related work and structural properties of the problem are given in Section 2. Section 3 introduces the proposed MILP formulations. The constraint propagation algorithms based on energetic reasoning are presented in Section 4. Section 5 presents the hybrid branch-and-bound procedure. Experimental results are reported in Section 6. Concluding remarks are drawn in Section 7.

2 Problem statement, related work and structural properties

2.1 Problem definition

The considered continuous energy-constrained scheduling problem (CECSP) can be defined as follows. A set of tasks $A = \{1, \ldots, n\}$, consuming a continuous, cumulative and renewable resource of capacity B, has to be scheduled. We suppose there is no precedence constraints between these tasks. At each time t, a task $i \in A$ consumes a variable amount of the resource, $b_i(t)$. The objective is to find, for each task $i \in A$, its start time st_i, its end time et_i and its resource allocation function $b_i(t)$. These quantities have to satisfy the following constraints.

First, each task i has to be executed during its time window $[r_i, d_i]$, $r_i \neq d_i$, i.e.

$$r_i \leq st_i < et_i \leq d_i \tag{1}$$

Tasks of 0-duration are not considered as they do not set any constraints in the problem and can be scheduled anywhere in their time window.

Then, if a task i is in process at time t, then $b_i(t)$ has to lie between a minimum and maximum

[2] Some authors call these functions the power processing rate functions [4, 11, 19], so we may use both terms indifferently.

requirement, b_i^{min} and b_i^{max} respectively, and has to be equal to zero otherwise, i.e.

$$b_i^{min} \leq b_i(t) \leq b_i^{max} \qquad \forall t \in [st_i, et_i] \tag{2}$$

$$b_i(t) = 0 \qquad \forall t \notin [st_i, et_i] \tag{3}$$

Note that the case where $b_i^{min} = 0$ corresponds to the preemptive case. Thus, preemption can be allowed in the considered problem. To ensure feasibility, we suppose $0 \leq b_i^{min} \leq b_i^{max} \leq B,\ \forall i \in A$.

Furthermore, during its processing, a task receives an energy quantity from the resource. Thus, each task consumes a part of the same resource but the energy type received from the resource might be different for each task. In this sense, each task has is own conversion function, also called power processing rate function, f_i and a task is finished when it has received a required amount of energy W_i, i.e.

$$\int_{st_i}^{et_i} f_i(b_i(t))dt = W_i \tag{4}$$

Thus, function f_i has to be integrable. In this paper, we only consider non-decreasing, continuous and linear functions. Indeed, the main goal is to approximate real non-linear efficiency function by linear ones. In [2], an example of such approximation is provided and a variant of this example is described later in this paper. Efficiency functions f_i can be defined as follows:

$$f_i(b) = \begin{cases} 0 & \text{if } b = 0 \\ a_i * b + c_i & \text{if } b_i^{min} = 0 \text{ and } b \in]b_i^{min}, b_i^{max}] \\ a_i * b + c_i & \text{if } b_i^{min} \neq 0 \text{ and } b \in [b_i^{min}, b_i^{max}] \end{cases}$$

with $a_i > 0$ and $-a_i * b_i^{min} \geq c_i$ to ensure that $f_i(b) \leq 0,\ \forall b \in [b_i^{min}, b_i^{max}]$.

The last constraint is the resource capacity constraint. At each time t, the resource consumed by all tasks can not exceed the resource capacity, i.e.

$$\sum_{i \in A} b_i(t) \leq B \quad \forall t \tag{5}$$

In the following, we are mostly interested in finding a feasible solution for the CECSP. However, in some cases, since it might be interesting in some practical cases to minimize the total resource consumption, we consider the following objective function:

$$\text{minimize} \sum_{i \in A} \int_{st_i}^{et_i} b_i(t)dt \tag{6}$$

We will denote by CECSP the problem without objective function and by CECSP$_{obj}$ the problem with objective function (6).

We start by presenting an example of an instance with non-linear efficiency functions and we show how we transform them in order to have only linear efficiency functions.

Example Consider the following instance with $n = 4$ and $B = 2$:

- $r = [0, 2, 0, 5]$
- $d = [6, 10, 9, 13]$
- $b^{min} = [0, 0.5, 2, 1]$
- $b^{max} = [1, 1, 2, 1.5]$
- $W = [1, 5, 7, 8]$
- $f(b) = [b, \sqrt{b}, b, \sqrt{b}]$

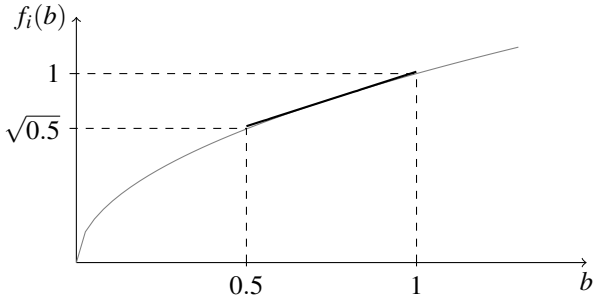


Figure 1: Non-linear efficiency curve: Example of approximation by a linear function.

We have to approximate $f_2(b)$ and $f_4(b)$ with linear functions. For task 2, we calculate the slope of the tangent at the middle point of $[b_i^{min}, b_i^{max}]$, which is 0.75. This slope is equal to $\frac{1}{2\sqrt{0.75}}$, and then, $f_2'(b) = \frac{1}{2\sqrt{0.75}} * b + \frac{\sqrt{0.75}}{2}$ (see Fig. 1).

Similarly, for task 4, we have $f_4'(b) = \frac{1}{2\sqrt{1.25}} * b + \frac{\sqrt{1.25}}{2}$.

With this approximation, it can happen that an efficiency function does not satisfy $b_i^{min} = 0 \Rightarrow f_i(b_i^{min}) = 0$. In this case, we set $f_i(0)$ to 0. Therefore, the function is continuous everywhere except in 0. Hence, we can replace equation (4) by:

$$\int_{st_i}^{et_i} \mathbf{1}_{NZ}(t) f_i(b_i(t)) dt = W_i \tag{4'}$$

where $\mathbf{1}_{NZ}(t) := \begin{cases} 1 & \text{if } t \in NZ := \{t | b_i(t) \neq 0\} \\ 0 & \text{otherwise} \end{cases}$

Now, we present an example of an instance $\mathscr{I}$ of CECSP and one feasible associate solution.

Example Consider an instance with $n = 3$ and $B = 5$. The other data are displayed in Table 1, and a feasible solution is depicted in Fig. 2.

i	r_i	d_i	W_i	b_i^{min}	b_i^{max}	$f_i(b)$
1	0	6	28	1	5	$2b+1$
2	2	6	32	2	5	$b+5$
3	2	5	6	2	2	b

Table 1: An instance of CECSP (continuous energy-constrained scheduling problem).

This solution is feasible since each task lies in its time window, all the constraints of maximum and minimum requirements are satisfied and the total resource usage at each time does not exceed the availability of the resource. Furthermore, the required energy is received by each task. For example, the energy received by task 1 is $(2*5+1)+(2*5+1)+(2*1+1)+(2*1+1) = 11+11+3+3 = 28$, while its total resource consumption is equal to 12.

2.2 Related work

In the literature, several authors considered different elements of the problem addressed in this paper, but generally separately and/or in a discrete time setting. The general category of scheduling problems with variable resource requirement and processing time of the tasks are referred to as scheduling problems with multiple modes. In the state-of-the-art review of multi-mode scheduling problems [19], problems

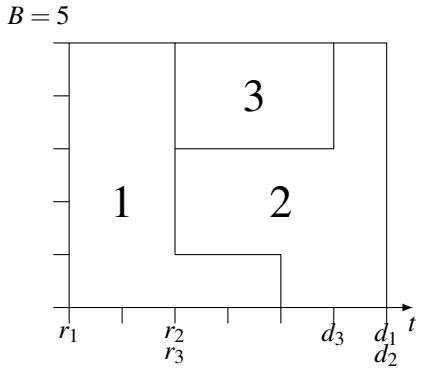


Figure 2: A solution for instance $\mathscr{I}$ of CECSP

with a finite number of modes are distinguished from problem with an infinite (e.g. uncountable) number of modes. In the first category, tasks have generally a rectangular shape in the time×resource space, although each mode defines for each task a particular rectangle with the general requirement that the resource usage increases as the duration decreases. Węglarz *et al.* [19] call this model the processing time vs resource amount model that applies to the multi-mode resource-constrained project scheduling problem and the discrete time/resource tradeoff models.

Our problem belongs to the category of problems with infinite modes where the resource usage may vary continuously and such that the amount of resource required by a task may vary over time. Węglarz *et al.* [19] call this model the processing rate vs resource amount model, as the processing rate of the task is a continuous increasing function of the allotted resource amount at a time, which corresponds to the efficiency function (see also [4]). Providing a general framework for solving mixed discrete/continuous problems with concave processing rate functions, Józefowska *et al.* [11] show that once the sequence of sets of tasks to be scheduled in parallel is determined, the continuous resource allocation can be made by a convex non-linear optimization problem. In the literature on parallel processor scheduling, the malleable task model also considers the possibility of changing the number of processors assigned to a task over time, with non-linear processing rate functions but these problems are generally preemptive [5]. A related work has also been carried out by Kis [12] for a discretized time problem with variable-intensity tasks, who established polyhedral results and proposed a branch-and-cut procedure. Besides time discretization, the problem does not involve efficiency functions. To the best of our knowledge, the existing literature on continuously divisible resources does not consider minimum amount of resource used by a task once it is started.

To complete the presentation of the relevant literature on the subject, the project scheduling problem with work-content constraint is also of interest. The problem considered in [9] involves among other constraints a minimum amount and a maximum amount of resource usage once the task is started. The resource requirement takes discrete values and the model does not involve efficiency functions. There is a single work-content resource and other resources (called dependent resources in [14]) such that the resource requirement of a task on any dependent resource at a given time is a non-decreasing function of the resource requirement on the work-content resource. Naber and Kolisch [14] present several discrete-time MILP models for such problems, considering linear "dependency" functions. A continuous-time formulation based on events is proposed in [15]. The formulation involves events corresponding to task start times, end times and resource usage changes.

In the following section, we present some properties of this problem that show among others that the "change" event is not necessary in our case.

2.3 Problem properties

This problem, as it is a generalization of the Cumulative Scheduling Problem (CuSP) [3], is NP-complete. In CuSP, given a set of n tasks and a discrete, renewable and cumulative resource available in a limited quantity B, the goal is to find a feasible schedule of the tasks where each task consumes a fixed amount of resource b_i, has a duration p_i and has to lie in its time window.

The following theorem states on the NP-completeness of CECSP by proving that CuSP is a particular case of this problem.

Theorem 1 *CECSP is NP-complete.*

Proof Let $\mathscr{I}$ be an instance of CuSP. We construct an instance $\mathscr{I}'$ of CECSP in the following way:

- $b_i^{min} = b_i^{max} = b_i,\ \forall i \in A$
- $f_i(b_i(t)) = b_i(t),\ \forall i \in A$
- $W_i = p_i b_i,\ \forall i \in A$
- all other data being equal.

Instance $\mathscr{I}$ is feasible if and only if instance $\mathscr{I}'$ is feasible. Indeed, if $\mathscr{I}'$ is a feasible instance of CECSP, then it exists a feasible schedule of the tasks satisfying (1)–(5) and it is also a feasible schedule for CuSP and reciprocally. □

Now, we present a property of the CECSP, which will be helpful for solving it. Actually, we prove that if a solution S exists, then another solution S' can be created from S with the property that each function $b_i(t)$ is piecewise constant. This is the statement of the following theorem:

Theorem 2 ([17]) *Let $\mathscr{I}$ be a feasible instance of CECSP, with linear functions f_i, $\forall i \in A$. A solution such that, for all $i \in A$, $b_i(t)$ is piecewise constant, exists. Furthermore, $\forall i \in A$ the only breakpoints of $b_i(t)$ can be restricted to the start and end times of the tasks.*

Proof Let S be a feasible solution of $\mathscr{I}$ and let $(t_q)_{\{q \in Q\}}$ be the increasing series of distinct start time and end time values ($|Q| \leq 2n$). We construct a new solution S' in the following way:

- $st_i' = st_i$ and $et_i' = et_i,\ \forall i \in A$
- $b_i'(t) = b_{iq}' = \frac{\int_{t_q}^{t_{q+1}} b_i(t)dt}{t_{q+1}-t_q},\ \forall q \in \{1,\ldots,Q-1\}$ and $\forall i \in A$.

As S is a feasible solution, S' clearly verifies constraints (1) and (3). So, we only have to prove that S' satisfies constraints (2), (4') and (5).

First, we prove that S' satisfies constraint (2). Indeed, since S verifies:

$$b_i^{min} \leq b_i(t) \leq b_i^{max} \quad \forall i \in A \text{ and } \forall t \in [st_i, et_i]$$

then, $\forall q \in \{1,\ldots,Q-1\}$ with $[t_q, t_{q+1}] \subseteq [st_i, et_i]$, we have:

$$\int_{t_q}^{t_{q+1}} b_i^{min} dt \leq \int_{t_q}^{t_{q+1}} b_i(t)dt \leq \int_{t_q}^{t_{q+1}} b_i^{max} dt$$

$$\Rightarrow (t_{q+1}-t_q) b_i^{min} \leq \int_{t_q}^{t_{q+1}} b_i(t)dt \leq (t_{q+1}-t_q) b_i^{max}$$

$$\Rightarrow b_i^{min} \leq \int_{t_q}^{t_{q+1}} b_i(t)dt/(t_{q+1}-t_q) \leq b_i^{max}$$

and therefore, S' satisfies constraint (2).

Now, in order to prove that S' satisfies (4'), we show that:

$$\forall t_q \in \{1,\ldots,Q-1\} \text{ and } \forall i \in A,\ \int_{t_q}^{t_{q+1}} \mathbf{1}_{NZ}(t) f_i(b_i(t))dt = \int_{t_q}^{t_{q+1}} f_i(b'_{iq})dt$$

We have two cases to consider:

- if $b_i(t) = 0, \forall t \in [t_q, t_{q+1}]$ then $b'_{iq} = 0$ and clearly, the condition holds.
- else we have:

$$\begin{aligned}\int_{t_q}^{t_{q+1}} f_i(b'_{iq})dt &= \int_{t_q}^{t_{q+1}} (a_i b'_{iq} + c_i)dt \\ &= a_i(t_{q+1}-t_q)\frac{\int_{t_q}^{t_{q+1}} b_i(t)dt}{t_{q+1}-t_q} + c_i(t_{q+1}-t_q) \\ &= \int_{t_q}^{t_{q+1}} \mathbf{1}_{NZ}(t) f_i(b_i(t))dt\end{aligned}$$

Therefore, S' satisfies constraint (4').

Finally, for constraint (5), since we have:

$$\sum_{i \in A} b_i(t) \leq B$$

by integrating over $[t_q, t_{q+1}]$, we obtain:

$$\sum_{i \in A} \int_{t_q}^{t_{q+1}} b_i(t)dt \leq B(t_{q+1}-t_q)$$

$$\Rightarrow \sum_{i \in A} \int_{t_q}^{t_{q+1}} b_i(t)dt/(t_{q+1}-t_q) \leq B$$

□

Actually, the same theorem can be established for the case of CECSP$_{obj}$. Indeed, the theorem states that, if a solution for CECSP exists, we can construct a solution in which each task received exactly the same amount of energy and consumes the same amount of resource. So, if the first solution is optimal for objective (6), the modified solution is also optimal for this objective.

An interesting corollary can be derived from Th. 2:

Corollary 3 ([17]) *For fixed $(st_i, et_i)_{i \in A}$ the satisfiability of CECSP can be checked polynomially in function of the input length.*

Indeed, for each interval composed of two consecutive start/end times, i.e. $[st_i, ft_j]$, $[ft_i, st_j]$, $[st_i, st_j]$ or $[ft_i, ft_j]$ (at most $2n$), we have to decide how much resource we give to tasks, i.e. find b_{iq} for each such interval s.t. (1)–(5). This problem can easily be modeled by a linear program.

Let $(t_q)_{q=1..Q}$ be the series defined in the proof of Th. 2, b_{iq} (resp. w_{iq}) the resource usage of (resp.

the energy received by) task i in $[t_q, t_{q+1}]$. The linear program can be written as follows:

$$\min \sum_{i \in A} \sum_{q=1}^{Q-1} b_{iq} \tag{7}$$

$$\sum_{\imath \in A} b_{iq} \leq B \qquad \forall q \in \{1..Q-1\} \tag{8}$$

$$b_{iq} \leq b_i^{max} \qquad \forall i \in A;\ \forall q \in \{1..Q-1\} |\ t_q \in [st_i, et_i[\tag{9}$$

$$b_{iq} \geq b_i^{min} \qquad \forall i \in A;\ \forall q \in \{1..Q-1\} |\ t_q \in [st_i, et_i[\tag{10}$$

$$b_{iq} = 0 \qquad \forall i \in A;\ \forall q \in \{1..Q-1\} |\ t_q \notin [st_i, et_i[\tag{11}$$

$$\sum_{q=1}^{Q-1} w_{iq}(t_{q+1} - t_q) = W_i \qquad \forall i \in A \tag{12}$$

$$w_{iq} \leq a_i b_{iq} + c_i \qquad \forall i \in A;\ \forall q \in \{1..Q-1\} \tag{13}$$

$$w_{iq} \leq M b_{iq} \qquad \forall i \in A;\ \forall q \in \{1..Q-1\} \tag{14}$$

with M some large enough constant. In this program, constraints (8) set the capacity of the resource to B. Constraints (9) and (10) require that the resource usage of task i lies in $[b_i^{min}, b_i^{max}]$ during its execution and constraints (11) set the resource usage to 0 if the task is not in process. Constraints (12) make sure that each task received the required energy. Finally, constraints (13) and (14) ensure energy conversion. Indeed, constraints (14) ensure that $f_i(0) = 0$.

Note that, if $\forall i \in A,\ b_i^{min} = 0$, then the problem is polynomial. Indeed, let $(t_q)_{q=1..Q}$ be the increasing series of distinct release date and deadline values. Then the linear program gives a feasible solution.

Theorem 4 *The preemptive CECSP ($\forall i \in A,\ b_i^{min} = 0$) can be solved in polynomial time.*

In the following, we consider that $\exists i \in A$ such that $b_i^{min} \neq 0$.

Another interesting remark can be made about Th. 2. Actually, in order to find a solution to CECSP, we only have to find, for each task, its start time st_i, its end time et_i and the quantity of resource allocated to i between two consecutive start/end time b_{iq}. This allows us to model this problem with Mixed Integer Linear Programming (MILP). Different such MILPs are presented in the following section.

3 Mixed Integer Linear Programs

In this section, we present three different Mixed Integer Linear Programs to solve CECSP$_{obj}$. The first one is based on a time-indexed formulation and the two other ones on an event-based formulation.

3.1 Time-indexed formulation

The first formulation we propose is a time-indexed formulation. In this formulation, the planning horizon is discretized in intervals of size one. Thus now, $T = \{0, \ldots, |\mathscr{T}|\}$ (by translation, we can assume that $\min_{i \in A} r_i = 0$). For each $t \in T$ and $\forall i \in A$, we define two binary variables x_{it} and y_{it}, which represent the start and end time of task i, respectively. That is, x_{it} (resp. y_{it}) is set to 1 if t is the start (resp. end) time of task i. We also define, for each interval $[t, t+1]$ and $\forall i \in A$, two variables b_{it} and w_{it}, which stand for the quantity of resource and energy received by task i in the corresponding interval. This formulation has $4n|T|$ variables. This yields the following formulation:

$$\min \sum_{i\in A}\sum_{t\in T} b_{it} \tag{15}$$

$$\sum_{t=r_i}^{d_i-1} x_{it} = 1 \qquad \forall i \in A \tag{16}$$

$$\sum_{t=r_i+1}^{d_i} y_{it} = 1 \qquad \forall i \in A \tag{17}$$

$$(\sum_{\tau=r_i}^{t} x_{i\tau} - \sum_{\tau=r_i+1}^{t} y_{i\tau}) b_i^{min} \leq b_{it} \qquad \forall t \in \{0,\ldots,d_i-1\}\ ;\forall i \in A \tag{18}$$

$$(\sum_{\tau=r_i}^{t} x_{i\tau} - \sum_{\tau=r_i+1}^{t} y_{i\tau}) b_i^{max} \geq b_{it} \qquad \forall t \in \{r_i,\ldots,d_i-1\}\ ;\forall i \in A \tag{19}$$

$$\sum_{t=r_i}^{d_i} w_{it} \geq W_i \qquad \forall i \in A \tag{20}$$

$$w_{it} = a_i b_{it} + c_i(\sum_{\tau=r_i}^{t} x_{i\tau} - \sum_{\tau=r_i+1}^{t} y_{i\tau}) \qquad \forall t \in T\ ;\forall i \in A \tag{21}$$

$$\sum_{i\in A} b_{it} \leq B \qquad \forall t \in T \tag{22}$$

$$b_{it} = 0 \qquad \forall t \notin \{r_i,\ldots,d_i-1\}\ ;\forall i \in A \tag{23}$$

$$x_{it} = 0 \qquad \forall t \notin \{r_i,\ldots,d_i-1\}\ ;\forall i \in A \tag{24}$$

$$y_{it} = 0 \qquad \forall t \notin \{r_i+1,\ldots,d_i\}\ ;\forall i \in A \tag{25}$$

$$b_{it} \geq 0 \qquad \forall t \in T;\forall i \in A \tag{26}$$

$$w_{it} \geq 0 \qquad \forall t \in T;\forall i \in A \tag{27}$$

$$x_{it} \in \{0,1\},\ y_{it} \in \{0,1\} \qquad \forall t \in T;\forall i \in A \tag{28}$$

The objective is described by (15). Constraints (16)-(17) ensure that a task starts and ends once and only once. Constraints (18)-(19) enforce that, during its execution, a task satisfies its minimum and maximum requirement. Constraints (19) also ensure that no resource is consumed by a task whenever it is not in process, i.e. $b_{it} = 0$. Note that these constraints also ensure that the start time of i occurs before its end time (otherwise $b_{it} = 0, \forall t$). Constraints (20) make sure that the energy requirement is satisfied. Constraints (21) convert the resource usage in energy. Constraints (22) impose that the resource demand at each time does not exceed the availability of the resource. Constraints (23) set the resource consumption of task i to 0 if $t \notin [r_i, d_i]$. Constraints (24)-(25) ensure that a task is processed during its time window.

Actually, the CECSP can be seen as a relaxation of this formulation. Indeed, in CECSP, we have a continuously-divisible resource and in this program, as the planning horizon is discretized, the resource is discretely divisible. Furthermore, in some case, even if all data are integer and function f_i is the identity function, there is possibly no integer solution, only fractional ones. This is the case of the example of Fig. 3 where the unique solution is fractional.

i	r_i	d_i	W_i	b_i^{min}	b_i^{max}	$f_i(b_i(t))$
1	0	2	3	2	2	$b_i(t)$
2	1	3	3	1	2	$b_i(t)$

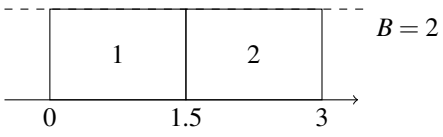


Figure 3: A fractional solution for integer data and identity efficiency function

Based on this observation, we propose two event-based formulations for CECSP$_{obj}$.

3.2 Event-based formulation

In this section, we propose two different event-based formulations. Both formulations are inspired by the event-based formulations for the Resource-Constrained Project Scheduling Problem [13]. In these formulations, an event corresponds either to a task start or a task end time. These events are represented by a set of continuous variables t_e. Let $\mathscr{E} = \{1, \ldots, 2n\}$ be the index set of these events. We also define two continuous variables B_{ie} and W_{ie} which stand for the quantity of resource and energy received by task i between events t_e and t_{e+1}. The main difference between our two formulations lies in the definitions of the binary variables used to assign an event either to a task start time or to a task end time. The first formulation is called a start/end event-based formulation and the second one is called an off/on formulation.

In the start/end formulation, two binary decision variables x_{ie} and y_{ie} are used to represent the start and end time of a task. The variable x_{ie} (resp. y_{ie}) is equal to 1 if task i starts (resp. ends) at event e. Since there are $2n$ events, this model has $8n^2$ variables ($4n^2$ binary). This yields the following formulation:

$$\min \sum_{i \in A} \sum_{e \in \mathscr{E} \setminus \{2n\}} B_{ie} \tag{29}$$

$$t_e \leq t_{e+1} \qquad \forall e \in \mathscr{E} \setminus \{2n\} \tag{30}$$

$$\sum_{e \in \mathscr{E}} x_{ie} = 1 \qquad \forall i \in A \tag{31}$$

$$\sum_{e \in \mathscr{E}} y_{ie} = 1 \qquad \forall i \in A \tag{32}$$

$$x_{ie} r_i \leq t_e \qquad \forall i \in A \,; \forall e \in \mathscr{E} \tag{33}$$

$$t_e \leq x_{ie} s_i^{max} + (1 - x_{ie}) D_{max} \qquad \forall i \in A \,; \forall e \in \mathscr{E} \tag{34}$$

$$t_e \geq y_{ie} e_i^{min} \qquad \forall i \in A \,; \forall e \in \mathscr{E} \tag{35}$$

$$d_i y_{ie} + (1 - y_{ie}) D_{max} \geq t_e \qquad \forall i \in A \,; \forall e \in \mathscr{E} \tag{36}$$

$$\sum_{i \in A} B_{ie} \leq B(t_{e+1} - t_e) \qquad \forall e \in \mathscr{E} \setminus \{2n\} \tag{37}$$

$$t_f \geq t_e + (x_{ie} + y_{if} - 1) W_i / f_i(b_i^{max}) \qquad \forall i \in A \,; \forall e, f \in \mathscr{E} \,; f > e \tag{38}$$

$$W_{ie} \leq a_i B_{ie} + c_i (t_{e+1} - t_e) \qquad \forall i \in A \,; \forall e \in \mathscr{E} \setminus \{2n\} \tag{39}$$

$$W_{ie} \leq W_i \Big(\sum_{f=0}^{e} x_{if} - \sum_{f=0}^{e} y_{if} \Big) \qquad \forall i \in A \,; \forall e \in \mathscr{E} \setminus \{2n\} \tag{40}$$

$$\sum_{e \in \mathscr{E} \setminus \{2n\}} W_{ie} = W_i \qquad \forall i \in A \tag{41}$$

$$B_{ie} \geq b_i^{min} (t_{e+1} - t_e) - b_i^{min} (\max_{j \in A} (d_j - r_j)) \Big(1 - \sum_{f=0}^{e} x_{if} + \sum_{f=0}^{e} y_{if}\Big) \qquad \forall i \in A \,; \forall e \in \mathscr{E} \setminus \{2n\} \tag{42}$$

$$B_{ie} \leq b_i^{max} (t_{e+1} - t_e) \qquad \forall i \in A \,; \forall e \in \mathscr{E} \setminus \{2n\} \tag{43}$$

$$\Big(\sum_{f=0}^{e} x_{if} - \sum_{f=0}^{e} y_{if} \Big) (b_i^{max} (d_i - r_i)) \geq B_{ie} \qquad \forall i \in A \,; \forall e \in \mathscr{E} \setminus \{2n\} \tag{44}$$

$$t_e \geq 0 \qquad \forall e \in \mathscr{E} \tag{45}$$

$$B_{ie} \geq 0 \qquad \forall i \in A \,; \forall e \in \mathscr{E} \setminus \{2n\} \tag{46}$$

$$W_{ie} \geq 0 \qquad \forall i \in A \,; \forall e \in \mathscr{E} \setminus \{2n\} \tag{47}$$

$$x_{ie} \in \{0,1\},\ y_{ie} \in \{0,1\} \qquad \forall i \in A \,; \forall e \in \mathscr{E} \tag{48}$$

where $s_i^{max} = d_i - W_i / f_i(b_i^{max})$ (resp. $e_i^{min} = r_i + W_i / f_i(b_i^{max})$) is the latest start (resp. earliest end) time of task i and $D_{max} = \max_{i \in A} d_i$.

The objective is described by (29). Constraints (30) order the events. Constraints (31) (resp. (32)) require that a task has one and only one start event (resp. end event). Constraints (33)–(36) state that,

$\forall i$, $r_i \leq st_i \leq s_i^{max}$ and $e_i^{min} \leq et_i \leq d_i$. Constraints (37) limit the demand of resource during $[t_e, t_{e+1}]$ to the availability of the resource during this interval. Constraints (38) are valid inequalities stating that the events corresponding to a start and an end time of a task must be separated by, at least, the minimal duration of this task. Constraints (39)–(41) ensure that the required energy is received by the tasks. Indeed, notice that $\sum_{f=0}^{e} x_{if} - \sum_{f=0}^{e} y_{if} = 1$ if and only if task i is in process in $[t_e, t_{e+1}]$. Furthermore, with the objective function (29), the inequality of constraint (39) is always satisfied with equality. Constraints (42) (resp. (43)) impose that, during its execution, a task satisfies its minimum (resp. maximum) resource requirement. Constraints (44) set the resource consumption of task i to 0 if the task is not in process. Note that these constraints also ensure a task cannot end before it has started (otherwise, $B_{ie} = 0$, $\forall e \in \mathscr{E}$).

This formulation is similar to the one in [17]. The main difference lies in the values given to the big-M constants. In constraints (40), M is set to W_i, in (42) to $b_i^{min}(\max_{j \in A}(d_j - r_j))$ and in (44) to $b_i^{max}(d_i - r_i)$.

Notice that this model is also valid for the CECSP without objective function. Indeed, the value of variables W_{ie} can not be set to a value greater than the real quantity of energy received by task i in $[t_e, t_{e+1}]$ with resource usage equals to B_{ie}. So, if a solution for the model is found, it can be used to compute a feasible solution for CECSP in polynomial time (same st_i, et_i set to the time when the required energy is received by the task).

We now describe the on/off event-based formulation. In this formulation, a binary variable z_{ie} is equal to 1 if task i is in process during interval $[t_e, t_{e+1}]$. This model has only $6n^2$ variables and yields to the following formulation:

$$\min \sum_{i \in A} \sum_{e \in \mathscr{E} \setminus \{2n\}} B_{ie} \tag{49}$$

$$t_e \leq t_{e+1} \qquad \forall e \in \mathscr{E} \setminus \{2n\} \tag{50}$$

$$\sum_{e \in \mathscr{E}} z_{ie} \geq 1 \qquad \forall i \in A \tag{51}$$

$$r_i z_{ie} \leq t_e \leq s_i^{max}(z_{ie} - z_{ie-1}) + (1 - (z_{ie} - z_{ie-1}))D_{max} \qquad \forall e \in \mathscr{E} \setminus \{1\};\ \forall i \in A \tag{52}$$

$$e_i^{min}(z_{ie-1} - z_{ie}) \leq t_e \leq d_i(z_{ie-1} - z_{ie}) + (1 - (z_{ie-1} - z_{ie}))D_{max} \qquad \forall e \in \mathscr{E} \setminus \{1\};\ \forall i \in A \tag{53}$$

$$t_f \geq t_e + ((z_{ie} - z_{ie-1}) - (z_{if} - z_{if-1}) - 1)W_i / f_i(b_i^{max}) \qquad \forall e, f \in \mathscr{E} \setminus \{1\};\ f > e;\ \forall i \in A \tag{54}$$

$$\sum_{e'=1}^{e} z_{ie'} \leq e(1 - (z_{ie} - z_{ie-1})) \qquad \forall e \in \mathscr{E} \setminus \{1\};\ \forall i \in A \tag{55}$$

$$\sum_{e'=e}^{2n} z_{ie'} \leq (2n - e)(1 + (z_{ie} - z_{ie-1})) \qquad \forall e \in \mathscr{E} \setminus \{1\};\ \forall i \in A \tag{56}$$

$$\sum_{i \in A} B_{ie} \leq B(t_{e+1} - t_e) \qquad \forall e \in \mathscr{E} \setminus \{2n\} \tag{57}$$

$$W_{ie} \leq a_i B_{ie} + c_i(t_{e+1} - t_e) \qquad \forall e \in \mathscr{E} \setminus \{2n\};\ \forall i \in A \tag{58}$$

$$W_{ie} \leq W_i z_{ie} \qquad \forall e \in \mathscr{E} \setminus \{2n\}\ ; \forall i \in A \tag{59}$$

$$\sum_{e \in \mathscr{E} \setminus \{2n\}} W_{ie} = W_i \qquad \forall i \in A \tag{60}$$

$$B_{ie} \geq b_i^{min}(t_{e+1} - t_e) - (b_i^{min}(\max_{j \in A}(d_j - r_j)))(1 - z_{ie}) \qquad \forall e \in \mathscr{E} \setminus \{2n\}\ ; \forall i \in A \tag{61}$$

$$B_{ie} \leq b_i^{max}(t_{e+1} - t_e) \qquad \forall e \in \mathscr{E} \setminus \{2n\}\ ; \forall i \in A \tag{62}$$

$$z_{ie}(b_i^{max}(d_i - r_i)) \geq B_{ie} \qquad \forall e \in \mathscr{E} \setminus \{2n\}\ ; \forall i \in A \tag{63}$$

$$t_e \geq 0 \qquad \forall e \in \mathscr{E} \tag{64}$$

$$B_{ie} \geq 0 \qquad \forall i \in A\ ; \forall e \in \mathscr{E} \setminus \{2n\} \tag{65}$$

$$W_{ie} \geq 0 \qquad \forall i \in A\ ; \forall e \in \mathscr{E} \setminus \{2n\} \tag{66}$$

$$z_{ie} \in \{0, 1\} \qquad \forall i \in A\ ; \forall e \in \mathscr{E} \tag{67}$$

In this formulation, constraints are similar to constraints of the start/end formulation. The only difference lies in constraints (55) and (56). These constraints ensure that there is no preemption in the processing of tasks.

Proposition 5 ([13]) *Constraints (55) and (56), called contiguity constraints, ensure non-preemption.*

A proof of this proposition for the RCPSP can be found in [13] and is still valid for the CECSP.

Experiments have been done on randomly generated instances to compare these three models (see Section 6.1) with the hybrid branch-and-bound procedure proposed in Section 5.

4 Energetic reasoning

In this section we propose a polynomial satisfiability test for CECSP. This test is based on the so-called energetic reasoning [10] and is an adaptation on the well-known "left-shift/right-shift" test of Baptiste et al. [3] for the CuSP. We also extend to the CECSP the results of Derrien and Petit [8]. A previous version of the test for the CECSP with identity function was proposed in [2].

This part extends the results of [17] by providing a much more detailed explanation, a description of all the results and by considering the case where some tasks can be preempted.

4.1 Mandatory consumption

We start by presenting an elementary necessary condition to check the data consistency and then, we present our energetic reasoning based satisfiability test.

First, we can observe that, since f_i is a non-decreasing function, processing the task at b_i^{max} during an interval $[t_1,t_2]$ gives the most possible energy in this interval. Based on this observation, we can state the following proposition:

Lemma 6 *Let $\mathscr{I}$ be an instance of CECSP. If it exists a task $i \in A$ for which the condition $W_i > f_i(b_i^{max})(d_i - r_i)$ is satisfied, then $\mathscr{I}$ is infeasible.*

Indeed, since task i has to be processed in interval $[r_i,d_i]$, if this interval is not sufficiently large for task i to received its required energy, then CECSP can not have a solution.

Now, in order to present our satisfiability test, we define the minimum resource consumption (resp. energy requirement) of a task i over an interval $[t_1,t_2]$, $\underline{b}(i,t_1,t_2)$ (resp. $\underline{w}(i,t_1,t_2)$). These quantities are expressed by the following equations:

$$\underline{b}(i,t_1,t_2) = \min_{\mathscr{S}} \int_{t_1}^{t_2} b_i(t)dt \qquad \mathscr{S} = \{b_i(t) | b_i(t) \text{ satisfies } (1)-(4')\} \tag{68}$$

$$\underline{w}(i,t_1,t_2) = \min_{\mathscr{S}} \int_{t_1}^{t_2} \mathbf{1}_{NZ} f_i(b_i(t))dt \tag{69}$$

These two quantities are used to compute the slack function of the interval $[t_1,t_2]$ defined by $SL(t_1,t_2) = B(t_2-t_1) - \sum_{i\in A} \underline{b}(i,t_1,t_2)$. These definitions allow us to provide the following necessary condition for CECSP:

Theorem 7 ([2]) *Let $\mathscr{I}$ be an instance of CECSP. If it exists $(t_1,t_2) \in \mathscr{T}^2$ such that $SL(t_1,t_2) < 0$ then $\mathscr{I}$ is infeasible.*

Proof By contradiction, suppose that it exists $(t_1,t_2) \in \mathscr{T}^2$ such that $SL(t_1,t_2) < 0$ and $\mathscr{I}$ is feasible. Then, by definition of $\underline{b}(i,t_1,t_2)$, we have $\int_{t_1}^{t_2} b_i(t) \geq \underline{b}(i,t_1,t_2)$. It implies $\sum_{i\in A} \int_{t_1}^{t_2} b_i(t) \geq \sum_{i\in A} \underline{b}(i,t_1,t_2) \geq B(t_2-t_1)$, which contradicts (5). □

In order to have a complete polynomial satisfiability test, we prove that the slack function can be computed in a polynomial time and that it is sufficient to perform the test on a polynomial number of intervals.

To compute the slack function in polynomial time, we have analyzed the possible configurations of minimum resource consumption. First, since $f_i(b)$ is a non-decreasing function, we can observe that, given an interval $[t_1,t_2]$, the minimum consumption always corresponds to a configuration where task i is either:

- left-shifted (Figs 4(a),(b),(d),(g)): the task starts at r_i and is scheduled at its maximum requirement between r_i and t_1,
- right-shifted (Figs 4(b),(c),(f),(i)): the task ends at d_i and is scheduled at its maximum requirement between t_2 and d_i,
- or both-shifted (Figs 4(e),(h)): when scheduling at minimum requirement inside $[t_1,t_2]$ implies to have a non-zero requirement both in $[r_i,t_1]$ and in $[t_2,d_i]$.

These configurations are displayed in Fig. 4.

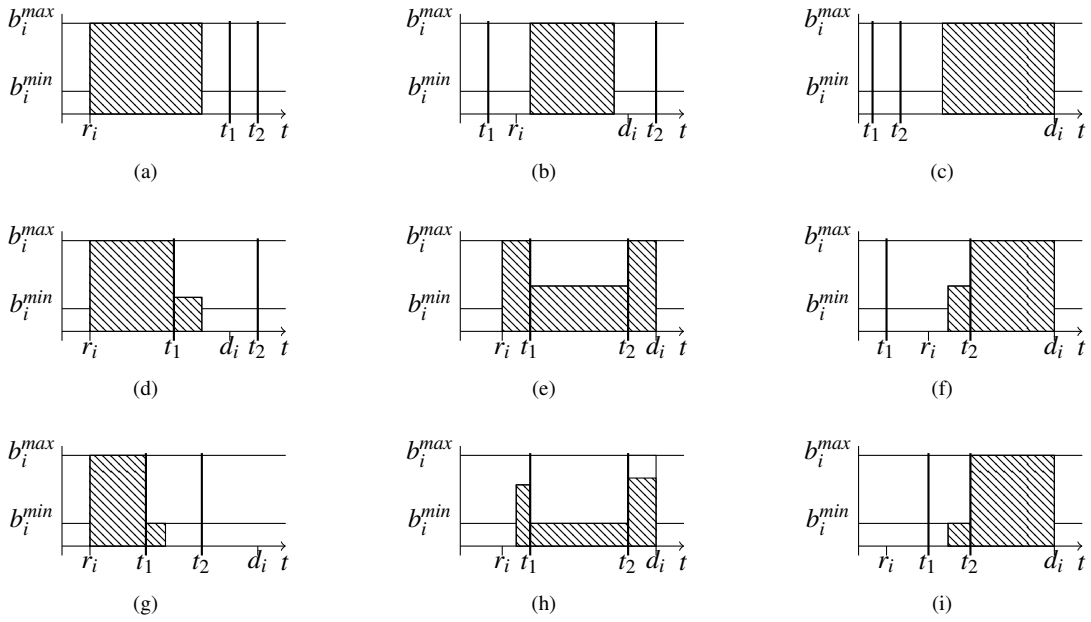


Figure 4: Possible configurations for minimal resource consumptions of task i over interval $[t_1,t_2]$.

We will denote $\omega_i^+(t_1,t_2)$ (resp. $\omega_i^-(t_1,t_2)$ and $\omega_i(t_1,t_2)$) the minimum energy requirement of task i inside $[t_1,t_2]$ if the task is left-shifted (resp. right-shifted or both-shifted). We have:

- $\omega_i^+(t_1,t_2) = \max\left(0\ ,\ W_i - \max(0,t_1-r_i)f_i(b_i^{max})\right)$
- $\omega_i^-(t_1,t_2) = \max\left(0\ ,\ W_i - \max(0,d_i-t_2)f_i(b_i^{max})\right)$
- $\omega_i(t_1,t_2) = \max\left(f_i(b_i^{min})(t_2-t_1)\ ,\ W_i - f_i(b_i^{max})(t_1-r_i+d_i-t_2)\right)$

Therefore, the minimum energy requirement in $[t_1,t_2]$ is:

$$\underline{w}(i,t_1,t_2) = \min\left(\omega_i^+(t_1,t_2), \omega_i^-(t_1,t_2), \omega_i(t_1,t_2)\right) \tag{70}$$

We still have to compute the minimum required resource consumption. For this, let I be the interval over which task i has to received an energy $\underline{w}(i,t_1,t_2)$, i.e. $I = [t_1,t_2] \cap [r_i,d_i]$.

We treat separately the case where $b_i^{min} = 0$ and the case where $b_i^{min} \neq 0$.
First, suppose that $b_i^{min} \neq 0$. Then, we can observe that processing a task i at its minimum requirement, b_i^{min}, has the best efficiency ratio, i.e. $\max_{x \in [b_i^{min}, b_i^{max}]} f(x)/x = f(b_i^{min})/b_i^{min}$. So, we have two cases to consider :

- I is sufficiently large to schedule the task at b_i^{min}, i.e. $|I| \geq \frac{w(i,t_1,t_2)}{f_i(b_i^{min})}$, and then $\underline{b}(i,t_1,t_2) = b_i^{min} \frac{\underline{w}(i,t_1,t_2)}{f_i(b_i^{min})}$
- I is not large enough to schedule the task at b_i^{min} and finding $\underline{b}(i,t_1,t_2)$ is equivalent to solving:

$$\begin{aligned} &\text{minimize } \int_I b_i(t)dt \\ &\text{subject to } \int_I f_i(b_i(t))dt \geq \underline{w}(i,t_1,t_2) \end{aligned}$$

By simplifications, we obtain: $\underline{b}(i,t_1,t_2) = \frac{1}{a_i}(\underline{w}(i,t_1,t_2) - |I|c_i)$.

And, the expression of the minimum resource consumption of i inside $[t_1,t_2]$ is:

$$\underline{b}(i,t_1,t_2) = \begin{cases} 0 & \text{if } \underline{w}(i,t_1,t_2) = 0 \\ \frac{1}{a_i}(\underline{w}(i,t_1,t_2) - |I|c_i)) & \text{otherwise} \end{cases} \quad (71)$$

Example Consider the instance of Example 2.1 with $W_1 = 31$ (instead of 28) and let $[t_1,t_2] = [2,5]$. Then we have:

- $\underline{w}(1,2,5) = \min(31-11*2\ ,\ 3*3\ ,\ 31-11*1) = 9$ and
  $\underline{b}(1,2,5) = \max(1*\frac{9}{3}\ ,\ \frac{1}{2}(9-3*1)) = 3$
- $\underline{w}(2,2,5) = \min(32-10*0\ ,\ 32-(10*(0+1))\ ,\ 32-10*1) = 22$ and
  $\underline{b}(2,2,5) = \max(2*\frac{22}{7}\ ,\ \frac{1}{1}(22-3*5)) = 7$
- $\underline{w}(3,2,5) = \min(6-2*0\ ,\ 2*3\ ,\ 6-0*2) = 6$ and
  $\underline{b}(3,2,5) = \max(2*\frac{6}{2}\ ,\ \frac{1}{1}(6-3*2)) = 6$

Thus, in $[2,5]$, $\sum_{i \in A} \underline{b}(i,2,5) = 3+7+6 = 16 > 5*(5-2) = 15$. Then the instance is infeasible.

4.2 Time-window adjustments

In this section, we describe some time-adjustments that can be deduced from the satisfiability test. These adjustments are an adaptation of the adjustments of Baptiste *et al.* [3].

We start by defining some notations. We denote by $\beta_i^+(t_1,t_2)$ (resp. $\beta_i^-(t_1,t_2)$ and $\beta_i(t_1,t_2)$) the minimal resource consumption corresponding to $\omega_i^+(t_1,t_2)$ (resp. $\omega_i^-(t_1,t_2)$ or $\omega_i(t_1,t_2)$).

We have:

$$\beta_i^+(t_1,t_2) = \begin{cases} 0 & \text{if } \omega_i^+(t_1,t_2) = 0 \\ \frac{1}{a_i}(\omega_i^+(t_1,t_2) - |I|c_i)) & \text{otherwise} \end{cases}$$

and similar expressions for $\beta_i^-(t_1,t_2)$ and $\beta_i(t_1,t_2)$.

Now, we are able to describe our time-window adjustments. Here, we only present the adjustments on s_i^{max} and d_i, the adjustments on r_i and e_i^{min} can be defined in a similar way.

Given a task i and an interval $[t_1,t_2]$ the goal is to decide whether i can start after t_1.

Lemma 8 *If it exists* $[t_1,t_2]$ *such that:*

$$\sum_{\substack{j\in A\\ j\neq i}} \underline{b}(j,t_1,t_2)+\beta_i^-(t_1,t_2) > B(t_2-t_1) \tag{72}$$

then

$$s_i^{max} \leq t_1 - \frac{1}{b_i^{max}}\left(\sum_{\substack{j\in A\\ j\neq i}} \underline{b}(j,t_1,t_2)+\beta_i^-(t_1,t_2)-B(t_2-t_1)\right))$$

Indeed, the only configuration for which task i starts after t_1 and leading to the minimum resource consumption inside $[t_1,t_2]$ is if the task is right-shifted. Therefore, $\sum_{j\in A;\ j\neq i}\underline{b}(j,t_1,t_2)+\beta_i^-(t_1,t_2)$ is the total minimum resource consumption in $[t_1,t_2]$ when task i starts after t_1. Hence, if this quantity is greater than the quantity of available resource in $[t_1,t_2]$, i has to start before t_1 otherwise $\sum_{j\in A;\ j\neq i}\underline{b}(j,t_1,t_2)+\int_{t_1}^{t_2} b_i(t) \geq \sum_{j\in A;\ j\neq i}\underline{b}(j,t_1,t_2)+\beta_i^-(t_1,t_2) \geq B(t_2-t_1)$.

Furthermore, $\sum_{j\in A;\ j\neq i}\underline{b}(j,t_1,t_2)+\beta_i^-(t_1,t_2)-B(t_2-t_1)$ is the minimum amount of resource that has to be allocated to i before t_1. Hence, we can divide this number by b_i^{max} to obtain a valid upper bound of the start time of i.

Similarly, we have the following adjustment on the end time of a task.

Lemma 9 *If* $b_i^{min} \neq 0$ *and it exists* $[t_1,t_2]$ *such that:*

$$\sum_{\substack{j\in A\\ j\neq i}} \underline{b}(j,t_1,t_2)+\min(\beta_i(t_1,t_2),\beta_i^-(t_1,t_2)) > B(t_2-t_1) \tag{73}$$

then

$$d_i \leq t_1 + \frac{1}{b_i^{min}}\left(B(t_2-t_1)-\sum_{\substack{j\in A\\ j\neq i}} \underline{b}(j,t_1,t_2)\right)$$

In this case, we have to divide $B(t_2-t_1)-\sum_{j\in A;\ j\neq i}\underline{b}(j,t_1,t_2)$ by b_i^{min} instead of b_i^{max} because we are looking for a lower bound on the start time of task i. So this bound has to be as far as possible from t_1.

Example Consider the instance described in Example 2.1. Let $i=1$ and $[t_1,t_2]=[2,5]$. We have:

- $\underline{b}(2,2,5)=7$
- $\underline{b}(3,2,5)=6$
- $\beta_1^-(2,5)=7$ and $\beta_1(2,5)=3$

Then, $\sum_{j\in A;\ j\neq i}\underline{b}(j,2,5)+\beta_i^-(2,5)=7+7+6=20>5(5-2)=15$. Therefore s_1^{max} can be set to $2-\frac{1}{5}(20-15)=1$ and d_i to $2+(15-13)=4$. Indeed, the available resource quantity in $[2,5]$ for task 1 is equal to $15-7-6=2$. If task 1 starts after t_1 , we need either $\beta_1^-(2,5)=7$ (the task is right-shifted) or $\beta_1^-(2,5)=7$ (the task is right-shifted) units of resource. So, task 1 cannot start after t_1. Since $b_i^{min}\neq 0$, the task cannot be interrupted so we have to schedule task 1 before t_1. Furthermore, since only 2 units of resource are available in $[2,5]$, s_i^{max} can be set to $2-1=1$.

In our algorithm (see Section 6), we perform these adjustments (on s_i^{max} and d_i) and their symmetric (on r_i and e_i^{min}) over the intervals on which we perform the satisfiability test. So for each interval $[t_1,t_2]$ and for each task i, we check whether condition (73) holds (or its symmetric) and if so, we adjust the time window according to Lemma 9. An interesting question is to know whether these intervals are sufficient to perform a complete satisfiability test, i.e. to make all possible time-window adjustments.

4.3 Relevant intervals

In this section, we start by proving that it is sufficient to perform the satisfiability test on a polynomial number of intervals and then, we present three ways of computing these intervals.

4.3.1 Complexity

Theorem 10 ([2]) *The energetic reasoning (Th. 7) needs only to be applied on $O(n^2)$ intervals (with n the number of tasks).*

Proof Since the slack function is the difference of one linear function $B(t_2 - t_1)$ and a sum of two-dimensional piecewise linear functions, it is a two-dimensional piecewise linear function. Therefore, its minimum is reached on an extreme point of one of the convex polygons on which it is linear. As the breakline segments of the slack function are the same as the ones of the sum of the individual minimum consumption functions, an extreme point of the slack function is the intersection of two breakline segments of an individual task minimum consumption.

Thus, we only have to perform the satisfiability test over the intervals corresponding to these intersection points and, since for each task there is a constant number of breakline segments, there is at most $O(n^2)$ such points and then $O(n^2)$ intervals to consider. □

For each of such intersection points (intervals), we compute the slack function in $O(n)$. Hence, the satisfiability test and the time-window adjustments take $O(n)$ time. Since the test is performed on a quadratic number of intervals, the total time complexity is $O(n^3)$.

Now, we present three ways of computing these intervals. The first and second one are based on an analysis of task breakline segments of the individual task minimum consumption function, as done in [2]. The last one, which is an adaptation of the work of Derrien *et al.*[8], is based on an analysis of the derivative of the slack function.

In our analysis, we have considered the following three cases:

- $b_i^{min} = 0$
- $W_i \leq f_i(b_i^{min})(d_i - r_i)$
- $W_i \geq f_i(b_i^{min})(d_i - r_i)$

Although the results for all of these cases are displayed in Fig. 5 and Table 2, we only explain how we get the results for the case where $W_i \geq f_i(b_i^{min})(d_i - r_i)$. All other cases are treated in a similar way.

4.3.2 Breakline segment analysis

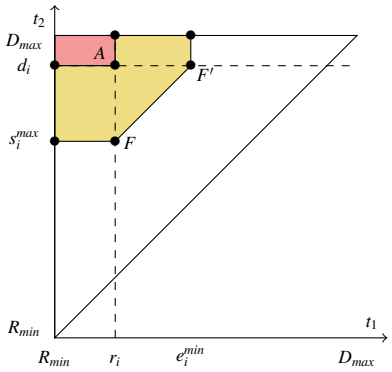
As the relevant intervals for the satisfiability test correspond to intersection points of two task breakline segments, we have analyzed the different breakline segments of a task. In [2], this analysis is provided for function $\underline{w}(i,t_1,t_2)$. Therefore, we briefly present their results before explaining how we get similar results for $\underline{b}(i,t_1,t_2)$.

First, an analysis of expression of $\underline{w}(i,t_1,t_2)$ depending on the value of (t_1,t_2) is performed. These results are displayed in Fig. 5.

For case $W_i \geq f_i(b_i^{min})(d_i - r_i)$, we have:

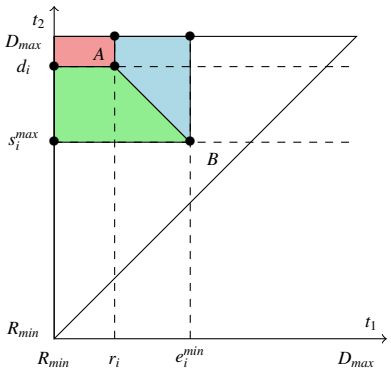
- in the red polygon, $\underline{w}(i,t_1,t_2) = W_i$
- in both green ones $\underline{w}(i,t_1,t_2) = W_i - (d_i - t_2) f_i(b_i^{max})$
- in blue ones $\underline{w}(i,t_1,t_2) = W_i - (t_1 - r_i) f_i(b_i^{max})$
- in the yellow one $\underline{w}(i,t_1,t_2) = W_i - (d_i - t_2 + t_1 - r_i) f_i(b_i^{max}))$
- and, in the white one $\underline{w}(i,t_1,t_2) = (t_2 - t_1) f_i(b_i^{min})$

(1) $b_i^{min} = 0$

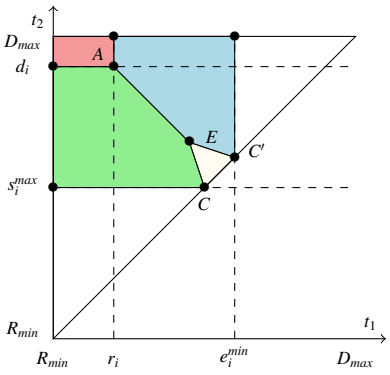


(2) $W_i \leq f_i(b_i^{min})(d_i - r_i)$

(a) $e_i^{min} \leq s_i^{max}$

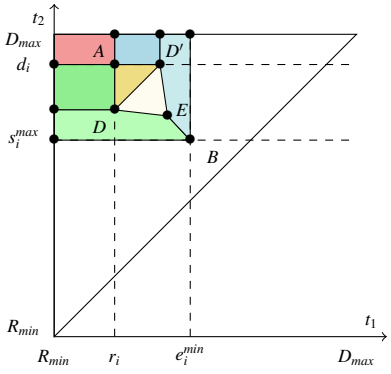


(b) $e_i^{min} \geq s_i^{max}$



(3) $W_i \geq f_i(b_i^{min})(d_i - r_i)$

(a) $e_i^{min} \leq s_i^{max}$



(b) $e_i^{min} \geq s_i^{max}$

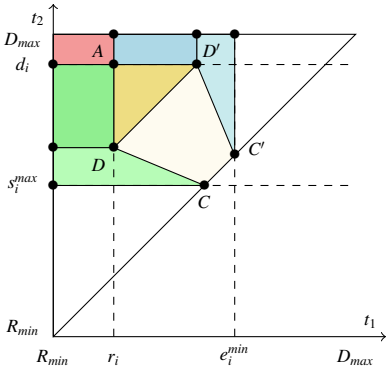


Figure 5: Analysis of a task breakline segments

All the other areas correspond to $\underline{w}(i,t_1,t_2) = 0$.

For ease of notation, we define the following set of points:

- $A = (r_i, d_i)$, $B = (e_i^{min}, s_i^{max})$, $C = (s_i^{max}, s_i^{max})$ and $C' = (e_i^{min}, e_i^{min})$
- $D = (r_i, \frac{d_i f_i(b_i^{max}) - r_i f_i(b_i^{min}) - W_i}{f_i(b_i^{max}) - f_i(b_i^{min})})$, $D' = (\frac{r_i f_i(b_i^{max}) - d_i f_i(b_i^{min}) + W_i}{f_i(b_i^{max}) - f_i(b_i^{min})}, d_i)$

- $E = (\frac{r_i(f_i(b_i^{max})-f_i(b_i^{min}))-d_i f_i(b_i^{min})+W_i}{f_i(b_i^{max})-2f_i(b_i^{min})}, \frac{d_i(f_i(b_i^{max})-f_i(b_i^{min}))-r_i f_i(b_i^{min})-W_i}{f_i(b_i^{max})-2f_i(b_i^{min})})$

In order to perform the same analysis for function $\underline{b}(i,t_1,t_2)$, we have to consider, for each polygon, the following inequality:

$$\underline{w}(i,t_1,t_2) \leq f_i(b_i^{min})|I| \tag{74}$$

i.e. knowing whether the interval $|I| = [r_i,d_i] \cap [t_1,t_2]$ is large enough to execute i at b_i^{min}.

In the red polygon, since $W_i \leq f_i(b_i^{min})(d_i - r_i)$ is never satisfied, $\underline{b}(i,t_1,t_2) = \frac{1}{a_i}(W_i - c_i|I|)$.

For the green polygon, the inequality (74) gives

$$\frac{d_i f_i(b_i^{max}) - t_1 f_i(b_i^{min}) - W_i}{f_i(b_i^{max}) - f_i(b_i^{min})} \leq t_2$$

In case $t_1 \leq r_i$, this inequality is equal to the abscissa of point D and, in the opposite case, the equation of segment DE. So, the green polygon is divided into two parts:

- the light one where $\underline{b}(i,t_1,t_2) = b_i^{min}(\underline{w}(i,t_1,t_2)/f_i(b_i^{min}))$
- the dark one where $\underline{b}(i,t_1,t_2) = \frac{1}{a_i}(\underline{w}(i,t_1,t_2) - c_i|I|)$

The same results holds for the blue polygon.

By applying the same reasoning, we obtain that, in the yellow polygon $\underline{b}(i,t_1,t_2) = \frac{1}{a_i}(W_i - (d_i - t_2 + t_1 - r_i)f_i(b_i^{max}) - c_i(t_2 - t_1))$ and in the white one $\underline{b}(i,t_1,t_2) = (t_2 - t_1)b_i^{min}$. All these results are displayed in Fig. 5.

Hence, in the case where $W_i \geq f_i(b_i^{min})(d_i - r_i)$ the breakline segments to consider are (we denote by D_{t_1} (resp. D_{t_2}) the x-coordinate (resp. y-coordinate) of point D):

- for case $e_i^{min} \leq s_i^{max}$: $(r_i, D_{max})A$, $(R_{min}, d_i)A$, $(R_{min}, s_i^{max})B$, $B(e_i^{min}, D_{max})$, AD, AD', $D(0, D_{t_2})$, $D'(D'_{t_1}, D_{max})$, DD', DE, $D'E$ and EB.
- for case $e_i^{min} \geq s_i^{max}$: $(r_i, D_{max})A$, $(R_{min}, d_i)A$, $(R_{min}, s_i^{max})C$, $C'(e_i^{min}, D_{max})$, AD, AD', $D(0, D_{t_2})$, $D'(D'_{t_1}, D_{max})$, DD', DC and $D'C$.

In order to compute the relevant intervals on which we have to perform the satisfiability test, we have to find, for each pair of breakline segments, their intersection point, if it exists. To achieve this, we can use either a naive enumeration algorithm or we can use the sweep line algorithm of Bentley-Ottmann [6]. In the first case, the total complexity of our test will be $O(n^3)$ and, in the second case, since the complexity of the sweep line algorithm is $O((n+k)\log n)$ (with k the total number of intersection points), the complexity will be $O((n^2+nk)\log n)$. Even if the theoretical complexity is higher with the sweep line algorithm (k may be in $O(n^2)$), in practice, the algorithm may be faster than the naive one (see Section 6.1).

4.3.3 Slack function analysis

The last way of computing relevant intervals is an adaptation of work of Derrien *et al.* [8] and is based on the following theorem:

Theorem 11 *The slack function is locally minimum in interval $[t_1,t_2]$ only if it exists two tasks i and j such that the following conditions are satisfied:*

$$\frac{\delta^+ \underline{b}(i,t_1,t_2)}{\delta t_1} < \frac{\delta^- \underline{b}(i,t_1,t_2)}{\delta t_1} \tag{75}$$

$$\frac{\delta^+ \underline{b}(j,t_1,t_2)}{\delta t_2} < \frac{\delta^- \underline{b}(j,t_1,t_2)}{\delta t_2} \tag{76}$$

with $\frac{\delta^+ \underline{b}(j,t_1,t_2)}{\delta t_2}$ (resp. $\frac{\delta^- \underline{b}(j,t_1,t_2)}{\delta t_2}$) the right (resp. left) derivative of $\underline{b}(j,t_1,t_2)$ w.r.t. t_2.

Proof By contradiction, suppose (t_1,t_2) is a local minimum of the slack function and equation (75) is satisfied for all tasks. Then, $SL(t_1,t_2)$ has its left derivative greater than or equal to its right. Since, by the second derivative test, minimal value of a function can only be reached at a point where its left derivative is lower than its right, (t_1,t_2) can not be a local minimum of the slack function. The proof for condition (76) is similar. □

As for the breakline segment analysis, we only describe our results for the case where $W_i \geq f_i(b_i^{min})(d_i - r_i)$ although the results for the other cases are displayed in Table 2.

Theorem 12 *Let i be a task. Then, for any fixed t_1, at most two intervals $[t_1,t_2]$ satisfying (76) exist and are as displayed in Table 2 with:*

$$\Delta'(t_2) = \frac{t_2(f_i(b_i^{min}) - f_i(b_i^{max})) + d_i f_i(b_i^{max}) - W_i}{f_i(b_i^{min})} \quad ; \quad \Gamma'(t_2) = \frac{W_i - t_2 f_i(b_i^{min}) + r_i f_i(b_i^{max})}{f_i(b_i^{max}) - f_i(b_i^{min})}$$

being breakpoints of function $t_1 \to \underline{b}(i,t_1,t_2)$ and

$$\Gamma(t_1) = \frac{W_i - t_1(f_i(b_i^{min}) - f_i(b_i^{max})) + r_i f_i(b_i^{max})}{f_i(b_i^{min})} \quad ; \quad \Delta(t_1) = \frac{W_i - f_i(b_i^{min}) d_i + t_1 f_i(b_i^{max})}{f_i(b_i^{max}) - f_i(b_i^{min})}$$

of function $t_2 \to \underline{b}(i,t_1,t_2)$.

Proof Since all cases are treated in a similar way, we only prove the result for the case corresponding to the eighth column of the left table of Table 2.

In order to prove the lemma, we analyze the variation of $t_2 \to \underline{b}(i,t_1,t_2)$. Fig. 6 represents these variations. The color corresponds to its expression w.r.t. Fig. 5.

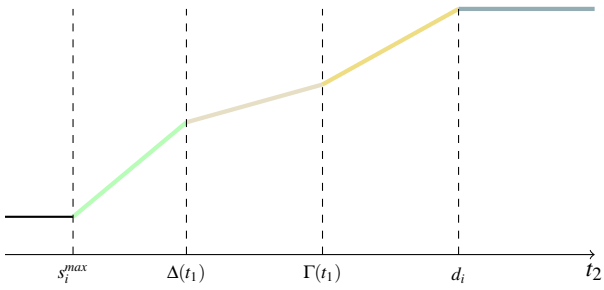


Figure 6: Relevant intervals for a particular case

The only intervals for which condition (76) is satisfied are $[t_1,d_i]$ and $[t_1,\Delta(t_1)]$. □

If we apply the symmetric reasoning for fixed t_2, we obtain a list of relevant intervals, which is described in Lemma 13.

Lemma 13 *Suppose tasks i and j satisfy: $W_l \geq f_l(b_l^{min})(d_l - r_l)$, $l = i, j$. Then the slack function is*

$W_i \geq f_i(b_i^{min})(d_i - r_i)$			
	$t_1 \leq r_i$		$[t_1, d_i]$
	$r_i \leq t_1 \leq e_i^{min}$	$t_1 \geq D'_{t_1}$ and $(t_1 \leq s_i^{max}$ or $t_1 \leq E_{t_1})$	$[t_1, \Gamma(t_1)]$ $[t_1, \Delta(t_1)]$
		$t_1 \leq D'_{t_1}$	$[t_1, d_i]$ $[t_1, \Delta(t_1)]$
		$t_1 \geq s_i^{max}$	$[t_1, \Gamma(t_1)]$
		$t_1 \geq E_{t_1}$	$[t_1, r_i + d_i - t_1]$

$W_i \leq f_i(b_i^{min})(d_i - r_i)$			
	$t_1 \leq r_i$		$[t_1, d_i]$
	$r_i \leq t_1 \leq e_i^{min}$	$e_i^{min} \leq s_i^{max}$ or $t_1 \leq E_{t_1}$	$[t_1, r_i + d_i - t_1]$
		$t_1 \geq s_i^{max}$	$[t_1, \Gamma(t_1)]$
		$s_i^{max} \geq t_1 \geq E_{t_1}$	$[t_1, \Gamma(t_1)]$ $[t_1, \Delta(t_1)]$
$b_i^{min} = 0$ $t_1 \leq e_i^{min}$			$[t_1, d_i]$

$W_i \geq f_i(b_i^{min})(d_i - r_i)$			
	$t_2 \geq d_i$		$[r_i, t_2]$
	$d_i \geq t_2 \geq s_i^{max}$	$t_2 \geq D_{t_2}$ and $(t_2 \geq e_i^{min}$ or $t_2 \geq E_{t_2})$	$[\Delta'(t_2), t_2]$ $[\Gamma'(t_2), t_2]$
		$t_2 \geq D_{t_2}$	$[r_i, t_2]$ $[\Gamma'(t_2), t_2]$
		$t_2 \leq e_i^{min}$	$[\Delta'(t_2), t_2]$
		$t_2 \leq E_{t_2}$	$[r_i + d_i - t_2, t_2]$

$W_i \leq f_i(b_i^{min})(d_i - r_i)$			
	$t_2 \geq d_i$		$[r_i, t_2]$
	$d_i \geq t_2 \geq s_i^{max}$	$e_i^{min} \leq s_i^{max}$ or $t_2 \geq E_{t_2}$	$[r_i + d_i - t_2, t_2]$
		$t_2 \leq e_i^{min}$	$[\Delta'(t_2), t_2]$
		$e_i^{min} \leq t_2 \leq E_{t_2}$	$[\Delta'(t_2), t_2]$ $[\Gamma'(t_2), t_2]$
$b_i^{min} = 0$ $t_2 \geq s_i^{max}$			$[r_i, t_2]$

Table 2: Relevant intervals depending of values of t_1 and t_2

locally minimum in (t_1,t_2) only if it is one of the following intervals:

$$\begin{cases}
[r_j,d_i] & \text{if } (r_j \le r_i \vee (r_j \le e_i^{min} \wedge r_j \le D'_{t_1})) \wedge \\
 & (d_i \ge d_j \vee (d_i \ge s_j^{max} \wedge d_i \ge D_{t_2})) \\
[\Delta'(d_i),d_i] & \text{if } (\Delta'(d_i) \le r_i \vee (\Delta'(d_i) \le e_i^{min} \wedge \Delta'(d_i) \le D'_{t_1})) \wedge \\
 & d_j \ge d_i \ge s_j^{max} \wedge d_i \le D_{t_2} \wedge d_i \ge E_{t_2} \\
[\Gamma'(d_i),d_i] & \text{if } (\Gamma'(d_i) \le r_i \vee (\Gamma'(d_i) \le e_i^{min} \wedge \Gamma'(d_i) \le D'_{t_1})) \\
 & \wedge d_j \ge d_i \ge s_j^{max} \wedge (d_i \ge e_j^{min} \vee d_i \ge E_{t_2}) \\
[d_j+r_j-d_i,d_i] & \text{if } (d_j+r_j-d_i \le r_i \vee (d_j+r_j-d_i \le e_i^{min} \wedge d_j+r_j-d_i \le D'_{t_1})) \wedge \\
 & d_j \ge d_i \ge s_j^{max} \wedge d_i \le E_{t_2} \\
[r_j,\Gamma(r_j)] & \text{if } r_i \le r_j \le e_i^{min} \wedge r_j \ge D'_{t_1} \wedge r_j \le E_{t_1} \wedge \\
 & (\Gamma(r_j) \ge d_j \vee (\Gamma(r_j) \ge s_j^{max} \wedge \Gamma(r_j) \ge D_{t_2})) \\
[r_j,\Delta(r_j)] & \text{if } (\Delta(r_j) \ge d_j \vee (\Delta(r_j) \ge s_j^{max} \wedge \Delta(r_j) \ge D_{t_2})) \wedge \\
 & r_i \le r_j \le e_i^{min} \wedge (s_i^{max} \ge r_j \vee r_j \le E_{t_1}) \\
[r_j,d_i+r_i-r_j] & \text{if } (d_i+r_i-r_j \ge d_j \vee (d_i+r_i-r_j \ge s_j^{max} \wedge d_i+r_i-r_j \ge D_{t_2})) \wedge \\
 & r_i \le r_j \le e_i^{min} \wedge r_j \ge E_{t_1}
\end{cases}$$

Lemma 13 By contradiction, suppose the slack function is locally minimum in (t_1,t_2) and $[t_1,t_2]$ is not one of the intervals defined by the lemma. Then, either condition (75) or (76) is not satisfied. Then, by Th. 11, the slack function can not be locally minimum in (t_1,t_2). □

We have described only the case where i and j are such that $W_l \ge f_l(b_l^{min})(d_l - r_l),\ l = i,j$. The other cases to consider are:

- $b_i^{min} = 0$ and $b_j^{min} = 0$
- $b_i^{min} = 0$ and $W_j \le f_j(b_j^{min})(d_j - r_j)$
- $b_i^{min} = 0$ and $W_j \ge f_j(b_j^{min})(d_j - r_j)$
- $W_l \le f_l(b_l^{min})(d_l - r_l),\ l = i,j$
- $W_i \ge f_i(b_i^{min})(d_i - r_i)$ and $W_j \ge f_j(b_j^{min})(d_j - r_j)$

These cases are not described in this paper since they are similar to the case we have presented.

In terms of complexity, since three cases of Lemma 13 can not happen simultaneously, we have only, for all couples of tasks (i,j), at most two intervals to consider. So, the total complexity of the satisfiability test is still $O(n^3)$ but the hidden constant is much smaller than the one of the naive algorithm.

Experimentations have been done on randomly generated instances to compare these three methods (see Section 6.1).

5 Hybrid Branch and Bound

In this section, we present a hybrid branch-and-bound algorithm to solve the CECSP. First, we use a branch-and-bound algorithm to reduce the size of possible start and end intervals (until their size is less than a given $\varepsilon > 0$) and, then, we use our on/off event-based MILP in order to find exact task start and end times and to determine the quantity of resource allocated to i between two consecutive events, i.e. $B_{ie},\ \forall i \in A;\ \forall e \in \mathscr{E}$.

The branching procedure is inspired by the work of Carlier *et al.* [7]. At the beginning, a task can start (resp. end) at any time $st_i \in [r_i, s_i^{max}]$ (resp $et_i \in [e_i^{min}, d_i]$). The idea is, at each node, to reduce the size of one of these intervals. Suppose that we choose to reduce the start time interval of i, then we create two nodes: one with constraint $st_i \in [r_i, (r_i + s_i^{max})/2]$ and one with constraint $st_i \in [(r_i + s_i^{max})/2, s_i^{max}]$ (see Fig. 7).

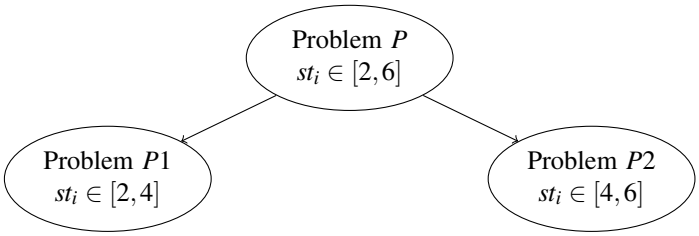


Figure 7: Example of branching procedure

At each node, we apply our satisfiability test and, if the test does not fail, we perform the associated time-window adjustments. We continue this procedure until all intervals are smaller than an ε, i.e. until arriving on a leaf of the search tree. When the algorithm is on a leaf, the remaining solution space is searched via the on/off event-based MILP.

We follow a depth-first strategy in the search tree and we choose the variable on which we will branch with the following heuristic: we choose the smallest interval among all $[r_i, s_i^{max}]$ and $[e_i^{min}, d_i]$. We backtrack when the satisfiability test fails, i.e. the node is infeasible, or when the algorithm is on a leaf and the MILP fails to provide a solution.

Experimental results, with different values of ε, are provided in Section 6.3.

6 Computational results

In this section, we start by presenting the results of the experiments for the complete satisfiability test (including time-window adjustments), the MILP models and the hybrid branch-and-bound algorithm. Then, we compare the results of the branch-and bound algorithm with the ones of the On/Off MILP formulation and we also compare our results with the ones of [17].

The experiments are conducted on an Intel Core i7-4770 processor with 4 cores and 8 gigabytes of RAM under the 64-bit Ubuntu 12.04 operating system. We use IBM CPLEX 12.6 with one thread and a time limit of 7200 seconds for solving the MILP models. The hybrid branch-and-bound algorithm is coded in C++ and uses CPLEX for solving the On/Off event-based model at each leaf. The total time limit of the algorithm is set to 7200 seconds.

We generated 15 instances of 10 and 60 tasks respectively, and 30 instances of 20, 25 and 30 tasks respectively. One third of these instances are instances with identical function f_i and the other instances are generated from these ones by randomly generating a_i and c_i, $\forall i \in A$, within interval $[1, 10]$ and we set W_i to a random number within $[0, f_i(W_i)]$.

6.1 Satisfiability test

We start by presenting the results of the comparison of the three ways for computing relevant intervals (see Section 4.3). These results are displayed in Table 3. The first column correspond to the naive algorithm, the second one to the sweep-line algorithm and the last one to the adaptation of [8]. The sweep line algorithm is the one from the CGAL C++ library [3]. The time is set in milliseconds and corresponds to the time needed to perform the satisfiability test and the time-window adjustments.
As expected, the best way of computing relevant intervals is the third method. The performance of the

[3] CGAL, Computational Geometry Algorithms Library. http://www.cgal.org.

# tasks	naive	sweep-line	adaptation of [8]
10	0.462	1.586	0.39
20	3.898	6.195	1.052
25	7.182	7.504	1.731
30	11.544	11.783	3.056
60	45.968	62.824	14.408

Table 3: Results of experiments for computing relevant intervals

sweep-line algorithm is disappointing compared to the naive one. In the remaining experiments, we use the adaptation of method [8] to perform the energetic reasoning satisfiability test.

6.2 MILP formulation

Table 4 presents the results of the three MILP models. For each model, three columns sum up the results. The first one corresponds to the time needed to solve instances (if solved), the second one corresponds to the percentage of solved instances[4], and the last one corresponds to the percentage of instances proved feasible. This last column is set in order to show that the time-indexed MILP model may conclude that no integer solution exists whereas a fractional solution exists.

	Time-indexed			On/off event-based			Start/End event-based		
#tasks	time(s)	%solved	%feasible	time(s)	%solved	%feasible	time(s)	%solved	%feasible
10	0.03	100	86.67	0.23	100	100	0.77	100	100
20	0.2	100	81.82	69.50	96.97	100	592.88	84.85	100
25	0.46	100	81.48	1080.23	92.59	100	1591.93	50	100
30	1.06	100	83.33	1918.33	63.33	100	2441.39	20	100
60	324.46	100	80	–	0	0	–	0	0

Table 4: Results of experiments for the three MILP models

As we can see, the time-indexed model is faster than the event-based models but it cannot be seen as an exact solution method since, for some instances, only fractional solution exists. Note that among the two event-based formulations, the On/Off formulation performs better than the Start/End one. Therefore, this is the formulation used in the hybrid branch-and-bound algorithm. Table 5 shows how many of the instances proved infeasible by the time-indexed formulation are proved feasible by one of the event-based formulations.

#tasks	%of error of time-indexed model
10	100
20	83.33
25	80
30	80
60	0

Table 5: Comparison of time-indexed and event-based formulations

Table 6 presents the results of both the On/Off and the time-indexed model on CECSP$_{obj}$. For each formulation, the first column shows the time spent on solving the MILP. The second column represents the percentage of instances solved optimally and the third one the percentage of instances for which a feasible solution has been found. Finally, the last column of the table stands for the average difference

[4] such that a feasible solution has been found or a proof that no feasible solution exists was obtained

of the objective value between the two models. For example, the first value of this column shows that, for 20-task instances, the objective value returned by the time-indexed formulation is 10% higher than the one returned by the On/Off formulation.

	On/Off model			Time-indexed model			
#tasks	time (s)	%optimal	%feasible	time(s)	%optimal	%feasible	%obj. dev.
20	7200	0	100	7200	0	100	10.08
25	7200	0	44.44	7200	0	66.67	10.11

Table 6: Comparison of the On/Off and the time-indexed formulation for CECSP$_{obj}$

First, notice that the time-indexed formulation can find feasible solutions for more instances than the On/Off one. However, none of the two models is able to solve the instances to optimality. Finally, the On/Off formulation clearly finds solutions with better objective values, i.e. lower resource consumption.

6.3 Hybrid branch-and-bound

In this section, we present the results of the experiments for the hybrid branch-and-bound algorithm. We present the results of the algorithm with a branching heuristic that chooses the smallest interval $[r_i, s_i^{max}]$ or $[e_i^{min}, d_i]$ and split this interval.

Table 7 presents these results for different values of ε (see Section 5), that is $5, 10, 15$. The first column corresponds to the time needed to solve the instances (if solved), the second one corresponds to the percentage of solved instances, and the third one is the average number of nodes of the search tree.

	$\varepsilon = 5$			$\varepsilon = 10$			$\varepsilon = 15$		
#tasks	time(s)	%solved	#nodes	time(s)	%solved	#nodes	time(s)	%solved	#nodes
10	0.28	100	15.07	0.29	100	6.6	0.32	100	3.4
20	85.21	87.88	37.69	26.41	96.97	19.13	52.81	96,97	12
25	519.26	74.07	53.7	124.75	88.89	29	472.11	77.78	19.33
30	570.54	66.67	70.15	782.16	76.67	39.4	670.12	63.33	27.8
60	0.01	13.33	1	0.01	13.33	1	0.01	13.33	1

Table 7: Results of experiments for the hybrid branch-and-bound algorithm

We can see that among all values of tested ε, the value $\varepsilon = 10$ is the one for which the algorithm performs best. It solves more instances and takes less time on four cases out of five. The only case where it takes more time is for the instance set of 30 tasks but it solves more instances. Note that for the 60-task instances, the results are similar. This is due to the fact that one of the instances has been proved infeasible by the energetic reasoning at the root node.

6.4 Comparison of the MILP formulations with hybrid branch-and-bound

In this section, we compare the performance of the MILP formulations with the hybrid branch-and-bound algorithm. Since we use the On/Off event-based formulation in our algorithm, we start by the comparison with this formulation.

Table 8 shows the results of the On/Off MILP model and the hybrid branch-and-bound algorithm for $\varepsilon = 10$. For each solution method, the first column shows the time spent on solving the instances and the second column presents the percentage of solved instances.

As shown by Table 8 our hybrid branch-and-bound runs faster and solves more instances than the On/Off MILP model except for the 25-task instances where the hybrid branch-and-bound runs faster but solves a little less instances.

	On/Off formulation		branch-and-bound	
#tasks	time(s)	%solved	time(s)	%solved
10	0.23	100	0.29	100
20	69.5	96.97	26.41	96.97
25	1080.23	92.6	124.75	88.89
30	1918.32	63.33	782.16	76.67
60	–	0	0.01	13.33

Table 8: Comparison of the On/Off formulation and the hybrid branch-and-bound

In conclusion, we can see that among all solution methods, the time-indexed MILP provides the best results but it is not an exact method. If we restrict the experiments to exact methods, the hybrid branch-and-bound provides the best results.

In the next section, we tested our methods on the instances of [17] and we compare our results with the ones previously obtained in [17].

6.5 Comparison with existing methods

In this section, we have tested our algorithms on the instances of [17]. Five instances of 10 and 60 tasks and 10 instances of 20, 25 and 30 tasks have been generated with identical power processing rate functions according to the following framework. The resource availability B is set to 10 and all other data are randomly generated in their corresponding interval: $W_i \in [1, 1.25 * B]$, $b_i^{min} \in [0, 0.25 * W_i]$, $b_i^{max} \in [b_i^{min}, 2 * b_i^{min}]$, $r_i \in [0, 0.5 * n]$ and $d_i \in [e_i^{min}, e_i^{min} + n]$. Then, the instances are transformed in order to obtain two families of instances with power processing rate functions in the following way: the parameters of the function, a_i and c_i, $\forall i \in A$, are randomly generated within interval $[1, 10]$ and, for the first family (Family 1), W_i is set to a random number within $[0, f_i(W_i)]$ and, for the second family (Family 2), W_i is set to $f_i(W_i)$. Experiments are conducted on all instances of Family 1, a subset of instances of Family 2 (5 instances with 20, 25 and 30 tasks respectively) and on the family with identity functions (Family 3).

Table 9 presents the results of the comparison of the event-based MILP model proposed in this paper against the previously existing Start/End event-based model [17]. In this table, the first column describes the number of tasks, the second and third columns correspond to the result of the existing model, the fourth and fifth present the result of the new Start/End model and the two last columns set the results of the On/Off model.

For each of these models, the first column corresponds to the time needed to solve the instances. The time, set in seconds, is computed in the following way: when one run of the branch-and-bound reaches the time limit, we set the execution time of this run at 7200 seconds (this execution time is playing the role of a penalty). The second column shows the percentage of solved instances, i.e. the MILP either find a solution or prove no solution exists.

First, we can see that our Start/End model solves more instances and finds a solution faster than the Start/End model of [17]. We can also see that, among all the models, the On/Off one obtains the best results, as already shown by Table 4.

Table 10 presents the results of the hybrid branch-and-bound algorithm of [17] (first sub-table) and of the hybrid branch-and-bound proposed in the paper (second sub-table). Both algorithms have been tested on the three families of instances and with $\varepsilon = 5$.

In this table, the first column describes the number of tasks. The second column represents the average time (arithmetic mean) needed to solve the instances. The time is computed with the penalty of 7200s whenever a run of the algorithm failed to solve the instance or to prove its infeasibility. The third and fourth columns show the comparison of the time spent to solve the MILPs in leaves and the time spent in the tree. The fifth column corresponds to the percentage of solved instances. The sixth column corresponds to the average number of nodes of the branching tree and, finally, the seventh column shows

	Start/End [17]		Start/End		On/Off	
#tasks	time(s)	%solved	time(s)	%solved	time(s)	%solved
Family 1						
10	0.62	100	0.63	100	0.10	100
20	295.44	100	73.87	100	50.4	100
25	2060.64	77	1980.96	77	554.40	100
30	5418.2	40	5325.92	40	1842.03	90
60	7200	0	7200	0	7200	0
Family 2						
20	4788	40	3182.95	60	405.5	100
25	7200	0	7200	0	6020.65	20
30	7200	0	7200	0	7200	0
Family 3						
10	0.94	100	0.69	100	0.16	100
20	2237.76	77	2210.48	77	699.41	90
25	5508.41	33	4986.32	33	2636.74	66
30	6509.03	10	6481.32	33	4806.98	40
60	7200	0	7200	0	7200	0

Table 9: Results of experiments for the event-based MILP models on the three families of instances

the percentage of nodes on which either the instance is proved infeasible by the energetic reasoning test or some time-window adjustments are performed.

From these results, we can see that, due to the use of the On/Off formulation instead of the Start/End one and of another branching heuristic, our hybrid branch-and-bound works better than the one in [17] except for the 10-task instances of the third family. Thus, for the small-size instances, using just the MILP formulation can bring better results than using the branch-and-bound algorithm. Another remark we can make about these results is that a lot of time is spent in solving the MILP model comparing to the time spent in the tree. Therefore, improving the model can be an interesting direction for further research.

7 Conclusions and Perspectives

In this paper, we have presented a polynomial satisfiability test and time-window adjustments for the CECSP. We have recalled the three different ways for computing relevant intervals for this test and confirmed that the most efficient ways of doing it is with the adaptation of the slack function analysis of Derrien *et al.* [8, 17]. In a second time, we have used these results to design an efficient hybrid branch-and-bound for our problem, incorporating an event-based MILP formulation that significantly improved the method proposed in [17]. We also showed that a time-indexed MILP can provide good approximated solutions in reasonable computational times.

In the near future, the improvement of the hybrid branch-and-bound with a better branching heuristic, both for choosing the branching variable and the interval cutting point, appears as an important follow-up of this work.

In a second time, the adaptation of the incremental algorithm of Baptiste *et al.* [3] for energetic reasoning as well as the research of the minimal set of interval necessary to perform all time-window adjustments, as done in [8], seems to be an interesting research direction.

Another improvement of these research works is to include valid inequalities, directly deduced from energetic reasoning or from a polyhedral analysis, to the linear programs.

Finally, in order to provide better applications to actual scheduling problems under energy constraints, it will be interesting to study the case where efficiency function are no longer linear.

References

[1] C. Artigues, P. Lopez, and A. Haït (2013). The energy scheduling problem: Industrial case study and constraint propagation techniques. *International Journal of Production Economics*, 143(1):13-23.

[2] C. Artigues and P. Lopez. Energetic reasoning for energy-constrained scheduling with a continuous resource. *Journal of Scheduling*, In press, DOI:10.1007/s10951-014-0404-y.

[3] P. Baptiste, C. Le Pape and W. Nuijten (2001). *Constraint-based scheduling*, Kluwer Academic Publishers, Boston/Dordrecht/London.

[4] J. Błażewicz, K. H. Ecker, E. Pesch, G. Schmidt and J. Węglarz (2001). *Scheduling computer and manufacturing processes*, Springer-Verlag, Berlin/Heidelberg.

[5] J. Błażewicz, M. Machowiak, J. Węglarz, M. Kovalyov and D. Trystram (2004). Scheduling malleable tasks on parallel processors to minimize the makespan. *Annals of Operations Research*, 129(1-4): 65–80.

[6] J.L. Bentley and T.A. Ottmann (1979). Algorithms for reporting and counting geometric intersections. *IEEE Transactions on Computers*, 28(9): 643–647.

[7] J. Carlier and B. Latapie (1991). Une méthode arborescente pour résoudre les problèmes cumulatifs. *RAIRO. Recherche opérationnelle*, 25(3): 311–340.

[8] A. Derrien and T. Petit (2014). A new characterization of relevant intervals for energetic reasoning. *Principles and Practice of Constraint Programming*, Lecture Notes in Computer Science, 8656, pp. 289–297.

[9] C.-U. Fündeling and N. Trautmann (2010). A priority-rule method for project scheduling with work-content constraints. *European Journal of Operational Research*, 203(3): 568–574.

[10] J. Erschler and P. Lopez (1990). Energy-based approach for task scheduling under time and resources constraints. *2nd International Workshop on Project Management and Scheduling*, pp. 115-121, Compiègne, France.

[11] J. Józefowska, M. Mika, R. Różycki, G. Waligóra and J. Węglarz (1999). Project scheduling under discrete and continuous resources. In: J. Węglarz (Ed.), *Project Scheduling: Recent Models, Algorithms, and Applications*. Kluwer Academic Publishers, Boston, pp. 289–308.

[12] T. Kis (2005). A branch-and-cut algorithm for scheduling of project with variable-intensity activities. *Mathematical Programming*, Series A, 103: 515–539.

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

[14] A. Naber and R. Kolisch (2014). MIP models for resource-constrained project scheduling with flexible resource profiles. *European Journal of Operational Research*, 239: 335–348.

[15] A. Naber and R. Kolisch (2014). A continuous time model for the resource-constrained project scheduling with flexible resource profiles. *Proceedings of the 14th International Conference on Project Management and Scheduling*, pp. 166–168, Munich, Germany.

[16] M. Nattaf, C. Artigues and P. Lopez (2014). A polynomial satisfiability test using energetic reasoning for energy constraint scheduling. *14th International Workshop on Project Management and Scheduling*, pp. 169–172, Munich, Germany.

[17] M. Nattaf, C. Artigues and P. Lopez (2015). A hybrid exact method for a scheduling problem with a continuous resource and energy constraints. *Constraints*, 20(3): 304–324.

[18] C. Schwindt (2005). *Resource allocation in project management*. Springer, Berlin.

[19] J. Węglarz, J. Józefowska, M. Mika and G. Waligóra (2009). Project scheduling with finite or infinite number of activity processing modes - A survey. *European Journal of Operational Research*, 208: 177–205.

#tasks	Hybrid Branch-and-Bound [17] $\varepsilon = 5$					
Family 1						
	Total time(s)	CPLEX time(s)	Tree time(s)	%sol	#nodes	%cons./adj.
10	0.52	0.51	0.01	100	13.1	5.50
20	111.58	11.49	0.08	100	26.35	5.44
25	1434.84	1434.71	0.14	100	43.65	11
30	3684.34	3684.14	0.22	60	58.77	7.08
Family 2						
20	3637.63	3637.59	0.07	40	25.33	61.44
25	5086.14	5086.13	0.01	20	1.5	75
30	7200	7200	–	0	–	–
Family 3						
10	0.48	0.47	0.01	100	16	26.1
20	2079.57	2079.52	0.05	73	21.64	61.38
25	3523.38	3523.32	0.06	56	27.28	77.78
30	5193.32	5193.28	0.06	10	35.5	63.04
60	5760	5760	0.01	20	1	100
#tasks	Hybrid Branch-and-Bound $\varepsilon = 5$					
Family 1						
	Total time(s)	CPLEX time(s)	Tree time(s)	%sol	#nodes	%cons./adj.
10	0.14	0.13	0.01	100	15.8	0
20	10.33	10.27	0.06	100	38.72	11.03
25	30.81	30.67	0.14	100	54.22	9.22
30	330.59	330.32	0.28	100	54.22	5.02
Family 2						
20	1613.49	1613.43	0.07	80	37.25	69.13
25	5060	5060	<0.01	20	1	100
30	5842.3	5842.24	0.3	20	70	68.57
Family 3						
10	860.02	1.34	858.68	100	26.8	32.09
20	2008.62	2008.57	0.06	73	34.36	69.45
25	3345.08	3345.05	0.08	56	27	88.9
30	4315.42	4315.32	0.2	50	43.4	55.3
60	4320	4320	<0.01	40	1	100

Table 10: Results of experiments for the hybrid branch-and-bound on the three families of instances.