Constraints (2015) 20:304–324
DOI 10.1007/s10601-015-9192-z

A hybrid exact method for a scheduling problem with a continuous resource and energy constraints

Margaux Nattaf[1,2] · **Christian Artigues**[1,3] · **Pierre Lopez**[1,3]

Published online: 22 April 2015

Abstract This paper addresses a scheduling problem with a cumulative continuous resource and energy constraints. Given a set of non-preemptive tasks, each task requires a continuously-divisible resource. The instantaneous resource usage of a task is limited by a minimum and maximum resource requirement. Its processing has to lie within a time-window and the total energy received obtained by integrating a function f_i of the instantaneous resource usage over the processing interval must reach a required value (where f_i is a non-decreasing, continuous function). The problem is to find a feasible schedule of the tasks, which satisfies all the constraints. This problem, which is a generalization of the well-known cumulative scheduling problem, is NP-complete. For the case where all functions f_i are linear, we exhibit structural properties of the feasible solutions and we present a Mixed Integer Linear Program (MILP) based on an event-based formulation. We also adapt the famous "left-shift/right-shift" satisfiability test (energetic reasoning) and the associated time-window adjustments to our problem. To achieve this test, we present three different ways for computing the relevant intervals. Finally, we present a hybrid branch-and-bound method to solve the problem, which performs, at each node, the satisfiability test and time-window adjustments and, when the domains of all start and end times are small enough, the remaining solution space is searched via the event-based MILP.

✉ Christian Artigues
artigues@laas.fr

Margaux Nattaf
mnattaf@laas.fr

Pierre Lopez
lopez@laas.fr

[1] CNRS, LAAS, 7 avenue du colonel Roche, 31400 Toulouse, France

[2] Université de Toulouse, UPS, 31400 Toulouse, France

[3] Université de Toulouse, LAAS, 31400 Toulouse, France

Keywords Continuous scheduling · Energy constraints · Energetic reasoning · Branching scheme · Mixed integer programming

1 Introduction

The Continuous Energy-Constrained Scheduling Problem (CECSP) is a generalization of the well-known Cumulative Scheduling Problem (CuSP). In CuSP, given a resource with a limited capacity and a set of tasks each one having a release date, a due date, a duration and a resource requirement, we want to schedule all tasks in their time windows and without exceeding the capacity limit of the resource.

One of the main limitations of CuSP is that task durations and resource requirements do not vary over time. However, in many practical cases these variations are part of the problem.

A practical example where the task duration and resource requirement are not fixed is presented in [1]. In this paper, a foundry application is presented where a metal is melted in induction furnaces. The electrical power of the furnaces, which 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. Additionally, the melting operation can be stopped once the necessary energy has been received, depending of the selected power function, so the duration of this operation is not known in advance. Moreover, if we increase the power of an electrical furnace to accelerate melting operations, the energy received by the operation is not identical to the electrical energy consumed but is linked to it via a function. Efficiency functions should then be considered for the furnaces. However, the paper did not consider them. Finally, due to the complexity of the problem, the solution method proposed in [1] considers a time discretization, which can lead to suboptimal or infeasible solutions by over-constraining the problem, as shown below in Section 3.

In the literature, several scheduling problem involving controllable time-varying resource requirements of the tasks can be found [12, 13, 18] but none of them encompasses all the characteristics of the problem we consider in this paper. For the scheduling problem with malleable tasks [5], the task duration depends on the number of processors allocating to it. Another problem with this property is the scheduling problem with continuous resources [4], although in this case there is no task time-windows. The Resource-Constrained Project Scheduling Problem (RCPSP) with work-content constraint [10] also has this property as a work quantity needs to be received by each task but the resource requirement of a task can only be changed at discrete time periods. Some variants of the CuSP have been proposed to relax the constraint of constant or fixed resource requirement. Baptiste et al. [3] propose two such relaxations: the fully and the partially elastic case. But, in both cases, no fixed energy requirement is set for the tasks.

To tackle this issue, we model the scheduling problem with a continuously-divisible resource thus, now, the resource-usage profile of a task can take any shape bounded by a time-window and a minimum and maximum resource requirement, provided that a fixed amount of energy is received by the task. This problem, called CECSP (Continuous Energy-Constrained Scheduling Problem), has been introduced by Artigues et al. in [2] who considered the particular case in which the resource consumed by a task is equal to the energy received by it (identity function). We consider a more general case where the energy is expressed as a linear function of the resource consumed.

For CuSP and CECSP with identity functions, a polynomial satisfiability test called "left-shift/right-shift" exists [2, 3]. This test is based on the so-called energetic reasoning. One of the goals of this paper is to adapt this satisfiability test and the corresponding time-window adjustments to our problem, extending the work done in [17]. We also provide an adaptation of the new way of computing relevant intervals that was proposed for CuSP in [9].

Another goal of this paper is to present a solution method for our problem. We present two methods, one consisting in solving an event-based MILP inspired by the existing one for RCPSP [14] and the RCPSP with flexible resource profiles [15, 16]. The second method is a hybrid branch-and-bound algorithm, using the branching scheme of Carlier et al. [8], the "left-shift/right-shift" test and the corresponding time-window adjustments and the event-based MILP.

In Section 2, we describe the problem. Section 3 is dedicated to the event-based MILP. Section 4 presents the adaptation of the "left-shift/right-shift" test and Section 5 shows how we compute relevant intervals for this test. In Section 6, we describe the hybrid branch-and-bound algorithm and in Section 7 we present some computational results showing the interest of our approach.

2 Problem statement and properties

In the considered scheduling problem with a continuous resource and energy constraints (CECSP), we are given as input a set $A = \{1, \dots, n\}$ of tasks and a cumulative, continuous and renewable resource with limited capacity B. During its execution, a task uses a variable amount of this resource which has to lie between a minimum and a maximum requirement, b_i^{min} and b_i^{max}, respectively. A task finishes when a required amount of energy W_i has been received by it. To compute this energy we use a non-decreasing power processing rate function f_i which is continuous in $[b_i^{min}, b_i^{max}]$ with a special behavior at point zero, i.e. $f(0) = 0$. This function allows us to convert the resource quantity used by i in an energy quantity. Furthermore, each task needs to be performed in its time-window $[r_i, d_i]$. All the following results can be adapted to the case where tasks are preemptive but, for ease of notation, in this paper, we only consider the non-preemptive case, i.e. $b_i^{min} \neq 0$. These notations, their corresponding domains and the main notations used in this paper are summarized in Appendix (see Table 3).

Finding a feasible solution is equivalent to finding, for each $i \in A$, a start time st_i, a finish time ft_i and for every $t \in [st_i, ft_i]$ the amount $b_i(t)$ of resource allocated to task i at time t. These three quantities have to satisfy the following constraints:

$$r_i \leq st_i \leq ft_i \leq d_i \qquad \forall i \in A \tag{1}$$

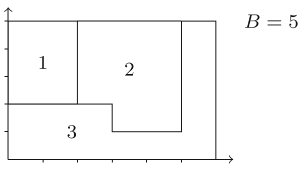
$$b_i^{min} \leq b_i(t) \leq b_i^{max} \qquad \forall i \in A \text{ and } t \in [st_i, ft_i] \tag{2}$$

$$b_i(t) = 0 \qquad \forall i \in A \text{ and } t \notin [st_i, ft_i] \tag{3}$$

$$\int_{st_i}^{ft_i} f_i(b_i(t))dt = W_i \qquad \forall i \in A \tag{4}$$

$$\sum_{i \in A} b_i(t) \leq B \qquad t \in [0, D_{max}] \tag{5}$$

with $D_{max} = \max_{i \in A} d_i$ (by translation, we can always assume that $\min_{i \in A} r_i = 0$).



i	r_i	d_i	W_i	b_i^{min}	b_i^{max}	$f_i(b)$
1	0	2	6	3	3	b
2	1	5	25	3	4	$2b+1$
3	0	6	39	1	5	$3b$

Fig. 1 An example of instance and corresponding solution of CECSP

This problem is NP-complete by simple reduction from the well-known Cumulative Scheduling Problem (CuSP). In this problem, given a set of n tasks and a discrete, cumulative and renewable resource available in 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 $[r'_i, d'_i]$.

The reduction from CuSP to CECSP is as follows. Let $\mathcal{I}'$ be an instance of CuSP. We construct an instance $\mathcal{I}$ of CECSP in the following way: b_i^{min} and b_i^{max} are set to b'_i, $\forall i \in A$, f_i is the identity function, W_i is set to $p'_i b'_i$ and all other data remain unchanged ($r_i = r'_i$, $d_i = d'_i$ and $B = B'$). Trivially, $\mathcal{I}$ is feasible if and only if $\mathcal{I}'$ is feasible. Therefore, the CECSP is NP-complete.

In this paper, we consider the case where all functions f_i are linear, i.e. of the form $a_i b + c_i$ with $a_i > 0$ and $c_i \geq 0$. We restricted the study to positive coefficients a_i and c_i in order to avoid case where $f_i(b) = 0$ when $b \in [b_i^{min}, b_i^{max}]$. We start by presenting an example of an instance of CECSP and one corresponding solution (see Fig. 1).

In the example of Fig. 1, we can see that, the energy received by task 2 is equal to $(2 \times 3 + 1) + (2 \times 4 + 1) + (2 \times 4 + 1) = 25$ which is not equal to the amount of resource consumed (11 in this case).

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 1 *Let $\mathcal{I}$ 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 $\mathcal{I}$ and let $(t_q)_{\{q=1..Q\}}$ 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: $\forall i \in A$, we set st'_i and ft'_i to st_i and ft_i respectively, and $\forall q \in \{1, \dots, Q-1\}$ we set, $\forall t \in [t_q, t_q+1]$, $b'_i(t)$ to $\frac{\int_{t_q}^{t_{q+1}} b_i(t)dt}{t_{q+1}-t_q} = b'_{iq}$.

As S is a feasible solution, S' clearly satisfies constraints (1) and (3). First, we prove that S' satisfies constraint (4). In order to do so, we prove that $\forall q \in \{1, \dots, Q-1\}$ and $\forall i \in A$,z$\int_{t_q}^{t_{q+1}} f_i(b_i(t))dt = \int_{t_q}^{t_{q+1}} f_i(b'_{iq})dt$.

As we have:

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

S' satisfies constraint (4).

We can prove that S' also satisfies constraints (2) and (5) in a similar way. □

Notice that this theorem may be no longer valid when the CECSP constraints are used in a larger model, for example, if allocation functions $b_i(t)$ are involved in other constraints.

An interesting corollary of Theorem 1 is as follows:

Corollary 1 *For fixed $(st_i, ft_i)_{i\in A}$ the satisfiability of CECSP can be checked polynomially in function of n.*

Indeed, for each interval $[st_i, ft_j]$ or $[ft_j, st_i]$ (at most $2n$), we have to decide how much resource we give to tasks w.r.t (1)–(5). This problem can easily be modeled by a linear program.

Another interesting remark can be made about Theorem 1. Indeed, in order to find a solution to CECSP, we only have to find, for each task, its start time st_i, its finish time ft_i and the quantity of resource allocated to i between two consecutive start/end time. This allows us to model this problem with a Mixed Integer Linear Program.

3 Mixed integer program

In this section, we present an event-based MILP for solving CECSP. We choose this representation instead of a time-indexed one for the following reason: we can build an instance, with integer data and identity functions f_i, having only non-integer solutions (see Fig. 2). Note that we could scale the input in order to get back to integer start and end times. However this might lead to an instance with a very large time horizon and then to a prohibitive number of variables for a time-indexed model.

Furthermore, it might not be possible to do the same scaling in order to have integer $b_i(t)$. In this case, any solver/model used to solve correctly this problem would need to be able to handle continuous variables in the context of a cumulative constraint.

Our formulation is inspired by the start/end event-based formulation for the RCPSP [14]. In this formulation, the events correspond to the start and end times of tasks and are represented by continuous variables t_e. Let $\mathcal{E} = \{1, \dots, 2n\}$ be the index set of these events. A decision variable x_{ie} (resp. y_{ie}) is equal to 1 if task i starts (resp. ends) at event e. In addition, we define two new variables B_{ie} and W_{ie}, which stand for the quantity of resource (resp. energy) received by task i during interval $[t_e, t_{e+1}]$. Since there are $2n$ events, this model has $8n^2$ variables. This yields the following formulation:

$$\max \quad \sum_{i\in A}\sum_{e\in\mathcal{E}} W_{ie} \tag{6}$$

$$t_e \le t_{e+1} \forall e \in \mathcal{E} \tag{7}$$

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

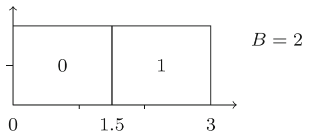


Fig. 2 Counter-example for integer data

$$\sum_{e\in\mathcal{E}} x_{ie} = 1 \quad \forall i \in A \tag{8}$$

$$\sum_{e\in\mathcal{E}} y_{ie} = 1 \quad \forall i \in A \tag{9}$$

$$x_{ie} r_i \leq t_e \quad \forall i \in A\,;\forall e \in \mathcal{E} \tag{10}$$

$$t_e \leq x_{ie} s_i^{max} + (1 - x_{ie}) D_{max} \quad \forall i \in A\,;\forall e \in \mathcal{E} \tag{11}$$

$$t_e \geq y_{ie} e_i^{min} \quad \forall i \in A\,;\forall e \in \mathcal{E} \tag{12}$$

$$d_i y_{ie} + (1 - y_{ie}) D_{max} \geq t_e \quad \forall i \in A\,;\forall e \in \mathcal{E} \tag{13}$$

$$\sum_{i\in A} B_{ie} \leq B(t_{e+1} - t_e) \quad \forall e \in \mathcal{E} \tag{14}$$

$$t_d \geq t_e + (x_{ie} + y_{id} - 1) Wi/f_i(b_i^{max}) \quad \forall i \in A\,;\forall (e,d) \in \mathcal{E}; d > e \tag{15}$$

$$W_{ie} \leq a_i B_{ie} + c_i (t_{e+1} - t_e) \quad \forall i \in A\,;\forall e \in \mathcal{E} \tag{16}$$

$$W_{ie} \leq M\Big(\sum_{e'=0}^{e} x_{ie'} - \sum_{e'=0}^{e} y_{ie'}\Big) \quad \forall i \in A\,;\forall e \in \mathcal{E} \tag{17}$$

$$\sum_{e\in\mathcal{E}} W_{ie} = W_i \quad \forall i \in A \tag{18}$$

$$B_{ie} \geq b_i^{min}(t_{e+1} - t_e) - M\Big(1 - \sum_{e'=0}^{e} x_{ie'} + \sum_{e'=0}^{e} y_{ie'}\Big) \quad \forall i \in A\,;\forall e \in \mathcal{E} \tag{19}$$

$$B_{ie} \leq b_i^{max}(t_{e+1} - t_e) \quad \forall i \in A\,;\forall e \in \mathcal{E} \tag{20}$$

$$\Big(\sum_{e'=0}^{e} x_{ie'} - \sum_{e'=0}^{e} y_{ie'}\Big) W_i - B_{ie} \geq 0 \quad \forall i \in A\,;\forall e \in \mathcal{E} \tag{21}$$

$$t_e \geq 0 \quad \forall e \in \mathcal{E} \tag{22}$$

$$B_{ie} \geq 0 \quad \forall i \in A\,;\forall e \in \mathcal{E} \tag{23}$$

$$W_{ie} \geq 0 \quad \forall i \in A\,;\forall e \in \mathcal{E} \tag{24}$$

$$x_{ie} \in \{0,1\},\ y_{ie} \in \{0,1\} \quad \forall i \in A\,;\forall e \in \mathcal{E} \tag{25}$$

where M is some large enough constant, $s_i^{max} = d_i - W_i/f_i(b_i^{max})$ (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 $D_{max} = \max_{i\in A} d_i$.

In order to provide better understanding of the model, we want to point out the fact that, if task i is in process at event e, then $\sum_{f=0}^{e}(x_{if} - y_{if}) = 1$ and 0 otherwise. We now described the constraints of the model. Constraints (7)–(15) are classical constraints of an event-based MILP model for a cumulative constraint. Constraints (16)–(18) combined with objective function (6) guarantee that the required energy is available for the tasks. Indeed, constraints (17) set W_{ie} to 0 if the task is not in process and constraints (16) combined with the objective function ensure resource conversion. Constraints (19) (resp. (20)) impose that, during its execution, a task satisfies its minimum (resp. maximum) resource requirement. Constraints (21) set the resource consumption of task i to 0 if the task is not in process. Indeed, in this case, constraints become $B_{ie} \leq 0$.

We have presented an event-based MILP solving the CECSP. Experimental results are described in Section 7. The rest of the paper is dedicated to the hybrid branch-and-bound. We start by presenting the checking and filtering algorithms which will be used in the main algorithm.

4 Energetic reasoning based satisfiability test

4.1 Mandatory consumption

In this section, we present a polynomial satisfiability test for CECSP. This test is based on the famous "left-shift/right-shift" test for the Cumulative Scheduling Problem [3] and use the so-called energetic reasoning [11].

Before explaining how this reasoning yields a polynomial satisfiability test for our problem, we exhibit an elementary necessary condition to check whether all task data are consistent. This condition can be expressed as follows: if there exists a task i such that $f_i(b_i^{max})(d_i - r_i) < W_i$ then the instance is infeasible.

Indeed, as $f_i(b)$ is a non-decreasing linear function, execute a task i at its maximum requirement during interval $[r_i, d_i]$ gives the maximum possible energy. Therefore, if W_i is greater than this quantity, then we cannot have a solution for the instance.

In order to present our satisfiability test, we define two quantities: the minimum resource consumption (resp. minimum 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:

$$\underline{b}(i, t_1, t_2) = \min_{\mathcal{S}} \int_{t_1}^{t_2} b_i(t)dt \qquad \mathcal{S} = \{b_i(t) | b_i(t) \text{ satisfies } (1) - (4)\} \tag{26}$$

$$\underline{w}(i, t_1, t_2) = \min_{\mathcal{S}} \int_{t_1}^{t_2} \mathbf{1}_{NZ}(t) f_i(b_i(t))dt \tag{27}$$

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

These two values are used to compute the slack of the interval $[t_1, t_2]$ defined by $SL(t_1, t_2) = B(t_2 - t_1) - \sum_{i \in A} \underline{b}(i, t_1, t_2)$. With these definitions, we are now able to define a necessary condition for CECSP to have a solution:

Theorem 2 *Let $\mathcal{I}$ be an instance of CECSP. If there exists $(t_1, t_2) \in \mathbb{R}^2$, with $t_1 < t_2$, such that $SL(t_1, t_2) < 0$ then $\mathcal{I}$ is an infeasible instance of CECSP.*

Proof By contradiction, suppose the condition is satisfied for some (t_1, t_2). Since, $\underline{b}(i, t_1, t_2)$ is the minimum resource consumption of i over $[t_1, t_2]$, for any feasible solution, we have $\int_{t_1}^{t_2} b_i(t) \geq \underline{b}(i, t_1, t_2)$.

It implies $\sum_{i \in A} \int_{t_1}^{t_2} b_i(t) \geq \sum_{i \in A} \underline{b}(i, t_1, t_2) > B(t_2 - t_1)$, and this is a contradiction with the integration of (5) on $[t_1, t_2]$. □

An example of this theorem is described by Fig. 3 in which the available quantity of resource is equal to $B(t_2 - t_1) = 3(6 - 1) = 15$ and the sum of all resource mandatory

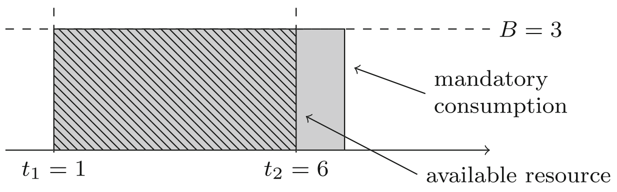


Fig. 3 Illustration of Theorem 2

consumptions is equal to 18. So, $SL(t_1, t_2) = -3 < 0$ and the instance is infeasible. The method used for computing the mandatory consumption of a task is described after the example.

In order to have a complete polynomial satisfiability test, we have to 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 (the task starts at r_i and is scheduled at b_i^{max} between r_i and t_1) or right-shifted (the task ends at d_i and is scheduled at b_i^{max} between t_2 and d_i) or both (both-shifted).

We will denote by $\omega_i^+(t_1, t_2)$ (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:

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

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

$$\underline{w}(i, t_1, t_2) = \min(\omega_i^+(t_1, t_2), \omega_i^-(t_1, t_2), \omega_i(t_1, t_2)) \tag{28}$$

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

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

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

  The constraint can be written as: $a_i \int_J b_i(t)dt + c_i \int_J dt \geq \underline{w}(i, t_1, t_2)$, which is equivalent to $\int_J b_i(t)dt \geq \frac{1}{a_i}(\underline{w}(i, t_1, t_2) - |J|c_i)$.

  Then, since there is only one constraint, the minimum value of $\int_J b_i(t)dt = \underline{b}(i, t_1, t_2) = \frac{1}{a_i}(\underline{w}(i, t_1, t_2) - |J|c_i)$.

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

$$\underline{b}(i, t_1, t_2) = \max(b_i^{min} \frac{\underline{w}(i, t_1, t_2)}{f_i(b_i^{min})}, \frac{1}{a_i}(\underline{w}(i, t_1, t_2) - |J|c_i)) \tag{29}$$

We show that we can compute the slack function in polynomial time. To have a complete polynomial satisfiability test, we have to prove that it is sufficient to perform the test on a polynomial number of intervals and that we can compute them in polynomial time. Since the same intervals are used to perform the time-window adjustments, we start by presenting them and after that, in Section 5, we will describe the interval computation method.

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 $\beta_i^+(t_1, t_2) = \max(b_i^{min} \frac{\omega_i^+(t_1,t_2)}{f_i(b_i^{min})}, \frac{1}{a_i}(\omega_i^+(t_1, t_2) - c_i|J|))$ 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. Given a task i and an interval $[t_1, t_2]$ the goal is to decide whether i can end before t_2.

Lemma 1 *If there exists* $[t_1, t_2]$ *such that:*

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

then, we have:

$$e_i^{min} \geq t_2 + \frac{1}{b_i^{max}}(\sum_{\substack{j \in A \\ i \neq j}} \underline{b}(j, t_1, t_2) + \min(\beta_i^+(t_1, t_2), \beta_i(t_1, t_2)) - B(t_2 - t_1)) \qquad (30)$$

Indeed, $\sum_{j \in A; i \neq j} \underline{b}(j, t_1, t_2) + \min(\beta_i^+(t_1, t_2), \beta_i(t_1, t_2))$ is the total minimum resource consumption in $[t_1, t_2]$ when i is left-shifted. Therefore, if this quantity is greater than the quantity of available resource then a part of i must be scheduled after t_2.

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

Similarly, if $b_i^{min} \neq 0$ then, when:

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

then, we have:

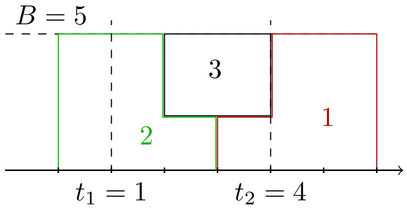
$$r_i \geq t_2 - \frac{1}{b_i^{min}}(B(t_2 - t_1) - \sum_{\substack{j \in A \\ i \neq j}} \underline{b}(j, t_1, t_2)) \qquad (31)$$

We perform these adjustments on the intervals on which we perform the satisfiability test[1] (see Algorithm 1). These intervals are described in Section 5.

Example 1 Consider the following instance:

[1]Note that we did not prove that these intervals are sufficient to perform all the relevant adjustments.

i	r_i	d_i	W_i	b_i^{min}	b_i^{max}	$f_i(b)$
1	0	6	12	1	5	b
2	0	4	12	2	5	b
3	1	4	6	2	2	b



On interval [1, 4], we have:

- $\underline{b}(i, t_1, t_2)[1][1][4] = \min(12 - 5 \times 1, 3, 12 - 5 \times 2) = 2$ (right-shifted)
- $\underline{b}(i, t_1, t_2)[2][1][4] = \min(12 - 5 \times 1, 7, 12) = 7$ (left-shifted)
- $\underline{b}(i, t_1, t_2)[3][1][4] = \min(6, 6, 6) = 6$ (both-shifted)
- $B(t_2 - t_1) = 5(4 - 1) = 15$

Considering task 1, the quantity of resource available for 1 in [1, 4] is $15 - (15 - 2) = 15 - 13 = 2$. If task 1 starts before t_1, we need either $\beta_1^+(1, 4) = 7$ (the task is left-shifted) or $\beta_1(1, 4) = 3$ (the task is both-shifted) units of resource available in [1, 4]. So, since the task is non-preemptive, i.e. $b_i^{min} \neq 0$, task 1 cannot start before t_1. Furthermore, since only 2 units of resource are available in [1, 4], r_i can be set to $t_2 - 2/b_i^{min} = 4 - 2/1 = 2$.

Algorithm 1 Time window adjustments

```
for every relevant interval $[t_1, t_2]$ do
    $W \leftarrow 0$
    $R \leftarrow 0$
    for all $i \in A$ do
        $W \leftarrow W + \min(\omega_i^+(t_1, t_2), \omega_i(t_1, t_2), \omega_i^-(t_1, t_2))$
        $R \leftarrow R + \max(b_i^{min} \frac{W}{f_i(b_i^{min})}, \frac{1}{a_i}(W - c_i|J|))$
    end for
    if $R > B(t_2 - t_1)$ then
        instance infeasible, exit
    else
        for all $i \in A$ do
            $W \leftarrow \min(\omega_i^+(t_1, t_2), \omega_i(t_1, t_2), \omega_i^-(t_1, t_2))$
            $slack \leftarrow B(t_2 - t_1) - R + \max(b_i^{min} \frac{W}{f_i(b_i^{min})}, \frac{1}{a_i}(W - c_i|J|))$
            if $slack < \min(\beta_i^+(t_1, t_2), \beta_i(t_1, t_2))$ then
                $e_i^{min} \leftarrow \max(e_i^{min}, t_2 + \frac{1}{b_i^{max}}(\min(\beta_i^+(t_1, t_2), \beta_i(t_1, t_2)) - slack))$
                $r_i \leftarrow \max(r_i, t_2 - \frac{slack}{b_i^{min}})$
            end if
            if $slack < \min(\beta_i^-(t_1, t_2), \beta_i(t_1, t_2))$ then
                $s_i^{max} \leftarrow \min(s_i^{max}, t_1 - \frac{1}{b_i^{max}}(\min(\beta_i^-(t_1, t_2), \beta_i(t_1, t_2)) - slack))$
                $d_i \leftarrow \min(d_i, t_1 + \frac{slack}{b_i^{min}})$
            end if
        end for
    end if
end for
```

We have presented the time adjustments we performed in the hybrid branch-and-bound procedure. Now, we prove that the time needed to apply the satisfiability test on an instance is polynomial by proving it is sufficient to do the test only on a quadratic number of intervals $[t_1, t_2]$.

4.3 Complexity

The following theorem establishes the polynomiality of the test by proving that the number of relevant intervals is quadratic.

Theorem 3 ([2]) *The energetic reasoning (Theorem 2) needs only to be applied on a quadratic number of intervals.*

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 break line 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 break line segments of an individual task minimum consumption.

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

For each intersection point, the slack function is computed in $O(n)$. So, the satisfiability test needs also $O(n)$ time and, since the test is performed on a quadratic number of intervals, the total time complexity is $O(n^3)$ with a naive enumeration algorithm. In the following section, we described three different methods for computing these intervals.

5 Computing relevant intervals

We now present three ways for computing these intervals. The first and second ones are based on an analysis of the break line segments of the individual task minimum consumption functions, as done in [2]. The first one computes the intersection points in a naive way and the second uses a sweep line algorithm to compute them. The last one is an adaptation of the work of Derrien et al. [9]. It is based on an analysis of the partial derivatives of the slack function.

In our work, we have considered the following cases:

1. $b_i^{min} = b_i^{max}$
2. $0 < b_i^{min} < b_i^{max}$
   (a) $W_i \leq (d_i - r_i) f_i(b_i^{min})$
   (b) $W_i \geq (d_i - r_i) f_i(b_i^{min})$

Since all cases are treated in a similar way, we only describe our results for case (22b).

5.1 Task break line segment analysis

In this section, we perform an analysis of the task break line segments. Indeed, we know that an extreme point of the slack function, i.e. a point for which it can be minimal, is at the intersection of two break line segments of an individual task minimum consumption. So, we are interested in finding, for each task, a list of these break line segments.

Once these lists are computed, we only have to test intersection of each pair of break line segments.

First, we have to analyse the expression of $\underline{w}(i, t_1, t_2)$ depending on the value of (t_1, t_2). This analysis has already been done in [2]. So, we just summarize these results in Fig. 4 before doing the analysis for function $\underline{b}(i, t_1, t_2)$. The left part of the figure corresponds to the case where $e_i^{min} \leq s_i^{max}$ and the right part to the case where $e_i^{min} \geq s_i^{max}$.

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 white 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 yellow one $\underline{w}(i, t_1, t_2) = (t_2 - t_1) f_i(b_i^{min})$. 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:

- $C = (r_i, d_i)$, $F = (e_i^{min}, s_i^{max})$, $G = (s_i^{max}, s_i^{max})$ and $G' = (e_i^{min}, e_i^{min})$
- $I = (r_i, \frac{d_i f_i(b_i^{max}) - r_i f_i(b_i^{min}) - W_i}{f_i(b_i^{max}) - f_i(b_i^{min})})$, $I' = (\frac{r_i f_i(b_i^{max}) - d_i f_i(b_i^{min}) + W_i}{f_i(b_i^{max}) - f_i(b_i^{min})}, d_i)$
- $H = (\frac{r_i(f_i(b_i^{max}) - f_i(b_i^{min})) - d_i f_i(b_i^{min}) + W_i}{f_i(b_i^{max}) - 2f_i(b_i^{min})}, \frac{d_i(f_i(b_i^{max}) - f_i(b_i^{min})) - r_i f_i(b_i^{min}) - W_i}{f_i(b_i^{max}) - 2f_i(b_i^{min})})$

These points correspond to the intersection of two segments delimiting two areas with different expressions of $\underline{w}(i, t_1, t_2)$. For example, H is the intersection point of line $W_i - (d_i - t_2) f_i(b_i^{max}) = (t_2 - t_1) f_i(b_i^{min})$ and line $W_i - (t_1 - r_i) f_i(b_i^{max}) = (t_2 - t_1) f_i(b_i^{min})$.

To perform the same analysis to function $\underline{b}(i, t_1, t_2)$, we have to consider, for each polygon, the following inequality: $\underline{w}(i, t_1, t_2) \leq f_i(b_i^{min})|J|$, i.e. knowing whether the interval $J = [r_i, d_i] \cap [t_1, t_2]$ is large enough to execute i at b_i^{min}.

So, in the blue polygon, we consider this inequality:

$$W_i - (t_1 - r_i) f_i(b_i^{max}) \leq f_i(b_i^{min})|J|$$

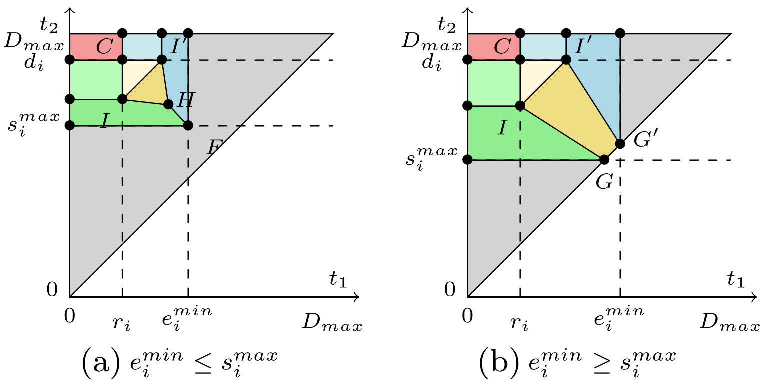


Fig. 4 Break line analysis

Since, the blue area is delimited by equation $t_1 = r_i$, we only have to consider two cases: $t_2 \geq d_i$ and $t_2 \leq d_i$. In both cases, the inequality becomes

$$t_1 \geq \frac{W_i + r_i f_i(b_i^{max}) - t_2 f_i(b_i^{min})}{f_i(b_i^{max}) - f_i(b_i^{min})}$$

In the case where $t_2 \geq d_i$, replacing t_2 by d_i gives us the ordinate of point I' and, in the other case, the inequality corresponds to the equation of either the segment $I'H$ ($e_i^{min} \leq s_i^{max}$) or the segment $I'G'$ ($e_i^{min} \geq s_i^{max}$). So, the blue polygon is separated into two parts:

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

and so does for the green polygon.

By applying the same reasoning, we find that:

– in the red polygon $\underline{b}(i, t_1, t_2) = \frac{1}{a_i}(W_i - c_i(d_i - r_i))$
– in the yellow one $\underline{b}(i, t_1, t_2) = \frac{1}{a_i}(W_i - (d_i - t_2 + t_1 - r_i) f_i(b_i^{max}) - c_i(t_2 - t_1))$
– and, in the white one $\underline{b}(i, t_1, t_2) = (t_2 - t_1) b_i^{min}$

All the other areas correspond to $\underline{b}(i, t_1, t_2) = 0$. These results are displayed in Fig. 4.

We have analyzed the expression of $\underline{b}(i, t_1, t_2)$ depending of the value of (t_1, t_2). The break line segments to consider correspond to segments delimiting two areas with different expressions of $\underline{b}(i, t_1, t_2)$. Thus, the break line segments to consider are (we denote by I_{t_1} (resp. I_{t_2}) the x-coordinate (resp. y-coordinate) of point I):

– in both cases: $(r_i, D_{max})C$, $(0, d_i)C$, $(0, s_i^{max})F$, $F(e_i^{min}, D_{max})$, CI, CI', $I(0, I_{t_2})$, $I'(I'_{t_1}, D_{max})$, II'
– for case $e_i^{min} \leq s_i^{max}$: IH, $I'H$ and HF.
– for case $e_i^{min} \geq s_i^{max}$: IG and $I'G$.

To identify the relevant intervals, we need to compute, for all pairs of break line segments, their intersection (t_1, t_2). To achieve this, either we use a naive algorithm, i.e. we test intersection of all couples of break line segments, or we use the sweep line algorithm of Bentley-Ottmann [6].

The main idea of the sweep line algorithm is that two segments cannot have an intersection point if they do not share x-coordinates and y-coordinates. A fictive horizontal line is used to sweep the x-axis and, at some "events", we test the intersection of two segments if they both cross this line and if they follow each other in vertical order. So, the number of tested intersections may decrease in comparison with a naive algorithm.

In the first case, we obtain a total complexity for the satisfiability test of $O(n^3)$ and, in the second case, the complexity is $O((n^2 + nk)\log n)$ with k the number of intersection points. Even if the theoretical complexity is higher with the sweep line algorithm (k may be in $O(n^2)$), in practice, the algorithm can be faster than the naive one (see Section 7).

5.2 Slack function analysis

The last way of computing relevant intervals is an adaptation of work of Derrien et al. [9] and is based on the following theorem:

Theorem 4 *The slack function is locally minimum in interval $[t_1, t_2]$ only if there exists two tasks i and j such that the following conditions are satisfied:*

$$\frac{\delta^+ \underline{b}(i, t_1, t_2)}{\delta t_1} < \frac{\delta^- \underline{b}(i, t_1, t_2)}{\delta t_1} \tag{32}$$

$$\frac{\delta^+ \underline{b}(j, t_1, t_2)}{\delta t_2} < \frac{\delta^- \underline{b}(j, t_1, t_2)}{\delta t_2} \tag{33}$$

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 (32) 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 (33) is similar. □

In the following lemma, we characterize, for a task i and a fixed t_1, the value of function $t_2 \rightarrow \underline{b}(i, t_1, t_2)$ for which its left derivative is greater than its right.

Lemma 2 *Suppose task i satisfies the following condition: $W_i \geq f_i(b_i^{min})(d_i - r_i)$. Then, for any fixed t_1, only one interval $[t_1, t_2]$ satisfying (33) exists:*

1. *if $t_1 \leq r_i$ then only interval $[t_1, d_i]$ has to be considered*
2. *if $t_1 \geq e_i^{min}$ then no interval has to be considered*
3. *if $r_i \leq t_1 \leq e_i^{min} \wedge t_1 \geq I'_{t_1} \wedge (t_1 \leq s_i^{max} \vee t_1 \leq H_{t_1})$ then intervals $[t_1, U(t_1)]$ and $[t_1, D(t_1)]$ have to be considered*
4. *if $r_i \leq t_1 \leq e_i^{min} \wedge t_1 \leq I'_{t_1}$ then intervals $[t_1, d_i]$ and $[t_1, D(t_1)]$ have to be considered*
5. *if $r_i \leq t_1 \leq e_i^{min} \wedge t_1 \geq H_{t_1}$ then only interval $[t_1, d_i + r_i - t_1]$ has to be considered*
6. *if $r_i \leq t_1 \leq e_i^{min} \wedge t_1 \geq s_i^{max}$ then only interval $[t_1, U(t_1)]$ has to be considered*

with $U(t_1) = \frac{W_i - t_1(f_i(b_i^{min}) - f_i(b_i^{max})) + r_i f_i(b_i^{max})}{f_i(b_i^{min})}$ and

$$D(t_1) = \frac{W_i - f_i(b_i^{min})d_i + t_1 f_i(b_i^{max})}{f_i(b_i^{max}) - f_i(b_i^{min})}.$$

Proof We only present the third case, the other ones are similar.

In order to prove the lemma, we analyse the variation of $t_2 \rightarrow \underline{b}(i, t_1, t_2)$. Figure. 5 represents these variations. The color corresponds to its expression w.r.t Fig. 4.

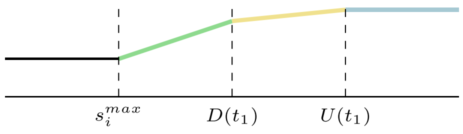


Fig. 5 Relevant intervals for case (3)

The two intervals for which condition (33) is satisfied are $[t_1, U(t_1)]$ and $[t_1, D(t_1)]$. □

We can apply the symmetric reasoning on t_2 in order to obtain a list of relevant intervals. This list is described in Lemma 3:

Lemma 3 *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 locally minimum in* (t_1, t_2) *only if it is one of the following intervals:*

$$\begin{array}{ll}
[r_j, d_i] & \textit{if}\ (r_j \leq r_i \vee (r_j \leq e_i^{min} \wedge r_j \leq I'_{t_1}))\wedge \\
 & (d_i \geq d_j \vee (d_i \geq s_j^{max} \wedge d_i \geq I_{t_2})) \\
[D'(d_i), d_i] & \textit{if}\ (D'(d_i) \leq r_i \vee (D'(d_i) \leq e_i^{min} \wedge D'(d_i) \leq I'_{t_1}))\wedge \\
 & d_j \geq d_i \geq s_j^{max} \wedge d_i \leq I_{t_2} \wedge d_i \geq H_{t_2} \\
[U'(d_i), d_i] & \textit{if}\ (U'(d_i) \leq r_i \vee (U'(d_i) \leq e_i^{min} \wedge U'(d_i) \leq I'_{t_1})) \\
 & \wedge d_j \geq d_i \geq s_j^{max} \wedge (d_i \geq e_j^{min} \vee d_i \geq H_{t_2}) \\
[d_j + r_j - d_i, d_i] & \textit{if}\ (d_j + r_j - d_i \leq r_i \vee (d_j + r_j - d_i \leq e_i^{min} \wedge d_j + r_j - d_i \leq I'_{t_1}))\wedge \\
 & d_j \geq d_i \geq s_j^{max} \wedge d_i \leq H_{t_2} \\
[r_j, U(r_j)] & \textit{if}\ r_i \leq r_j \leq e_i^{min} \wedge r_j \geq I'_{t_1} \wedge r_j \leq H_{t_1} \wedge \\
 & (U(r_j) \geq d_j \vee (U(r_j) \geq s_j^{max} \wedge U(r_j) \geq I_{t_2})) \\
[r_j, D(r_j)] & \textit{if}\ (D(r_j) \geq d_j \vee (D(r_j) \geq s_j^{max} \wedge D(r_j) \geq I_{t_2}))\wedge \\
 & r_i \leq r_j \leq e_i^{min} \wedge (s_i^{max} \geq r_j \vee r_j \leq H_{t_1}) \\
[r_j, d_i + r_i - r_j] & \textit{if}\ (d_i + r_i - r_j \geq d_j \vee (d_i + r_i - r_j \geq s_j^{max} \wedge d_i + r_i - r_j \geq I_{t_2}))\wedge \\
 & r_i \leq r_j \leq e_i^{min} \wedge r_j \geq H_{t_1}
\end{array}$$

with $D'(t_2) = \frac{t_2(f_i(b_i^{min}) - f_i(b_i^{max})) + d_i f_i(b_i^{max}) - W_i}{f_i(b_i^{min})}$ *and*

$$U'(t_2) = \frac{W_i - t_2 f_i(b_i^{min}) + r_i f_i(b_i^{max})}{f_i(b_i^{max}) - f_i(b_i^{min})}.$$

Proof Lemma 3 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 (32) or (33) is not satisfied. Then, by Theorem 4, the slack function can not be locally minimum in (t_1, t_2). □

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

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

There is no need to consider case where $b_i^{min} = b_i^{max}$ since it is included in case $W_i \leq f_i(b_i^{min})(d_i - r_i)$. Cases not described in this paper can be found in a similar way to the case we have presented.

In terms of complexity, since three cases of Lemma 3 can not happen simultaneously, we have only, for all couples of tasks (i, j), at most two intervals to consider. 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.

Table 1 Results of experiments for computing relevant intervals

# tasks	time (ms)		
	naive	sweep-line	adaptation [9]
10	0.51	1.56	0.22
20	3.67	5.39	0.92
25	7.79	6.45	1.59
30	14.59	15.26	4.08
60	43.03	54.76	13.7

Experiments have been done on randomly generated instances to compare these three methods (see Section 7).

6 Hybrid branch and bound

In this section, we define a hybrid branch and bound algorithm to solve 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 $\epsilon > 0$) and, then, we use our 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}, $\forall i \in A$; $\forall e \in \mathcal{E}$.

We start by describing our branching procedure. This procedure is inspired by the work of Carlier et al. [8]. At the beginning, a task can start (resp. end) at any time $st_i \in [r_i, s_i^{max}]$ (resp $ft_i \in [e_i^{min}, d_i]$). 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 $st_i \in [(r_i + s_i^{max})/2, s_i^{max}]$. We choose the interval to reduce randomly.

At each node, we apply first, the data consistency check and, if the data are consistent, our satisfiability test. 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 of the tree, the remaining solution space is searched via the event-based MILP.

We follow a depth-first strategy in the search tree. 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. In the case where the MILP finds a solution, then, since the goal is only to find a feasible solution, the algorithm stops.

7 Computational results

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

Table 2 Comparison between the MILP model and the hybrid branch-and-bound

# task	MILP model			hybrid branch-and-bound					
				$\epsilon = 5$					
	time(s)	%sol.	#nodes	Total time(s)	CPLEX time(s)	Tree time(s)	%sol.(3)	#nodes	%cons./adj.
Family 1									
10	0.62	100	0	0.52	0.51	0.01	100	13.1	5.50
20	295.44	100	2544.97	111.58	111.49	0.08	100	26.35	5.44
25	2060.64	77	1979.79	1434.84	1434.71	0.14	100	43.65	11
30	5418.2	40	4614	3684.34	3684.14	0.22	60	58.77	7.08
60	7200	0	X	6968.01	6967.91	0.53	0	78	0
Family 2									
20	4788	40	5789.5	3637.63	3637.59	0.07	40	25.33	61.44
25	7200	0	X	5086.14	5086.13	0.01	20	1.5	75
30	7200	0	X	7200	7200	X	0	X	X
Family 3									
10	0.94	100	17.4	0.48	0.47	0.01	100	16	26.1
20	2237.76	77	10406.38	2079.57	2079.52	0.05	73	21.64	61.38
25	5508.41	33	9619.46	3523.38	3523.32	0.06	56	27.28	77.78
30	6509.03	10	4146.5	5193.32	5193.28	0.06	10	35.5	63.04
60	7200	0	X	5760	5760	0.01	20	1	100

First, the instances have been randomly generated with identity power processing rate functions, i.e. $f_i(b) = b,\ \forall i \in A$. We generated 5 instances of 10 and 60 tasks and 10 instances of 20, 25 and 30 tasks 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, we transform them in order to obtain two families of instances with power processing rate functions in the following way: we randomly generated the parameters of the function a_i and c_i, $\forall i \in A$, within interval [1, 10] and, for the first family (Family 1), we set W_i to a random number within $[0, f_i(W_i)]$ and, for the second family (Family 2), we set W_i to $f_i(W_i)$. To illustrate the algorithm performance on different configurations, 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).

On these sets of instances, at least 76 % of them are feasible and 6 % are infeasible. For the other 18 %, we were not able to know whether there feasible or not.

Table 1 presents the results of the comparison of the three ways for computing relevant intervals for the energetic satisfiability test (see Section 5). The first column corresponds to the naive algorithm, the second one to the sweep-line algorithm, and the last one to the adaptation of the algorithm presented in [9]. The sweep line algorithm is the one from the CGAL C++ library.[2] The time is set in milliseconds and corresponds to the arithmetic mean time needed to perform the satisfiability test and the time-window adjustments on one node.

As expected, the best way of computing relevant intervals is the third method. Moreover, we can see that the sweep-line algorithm does not provide better results than the naive algorithm, except for the 25-task instances. The main reason of this result is the great number of intersection points.

Table 2 presents the results of the MILP model and of the hybrid branch-and-bound algorithm. Both algorithms have been tested on the three families of instances. Since the average size of interval $[r_i, s_i^{max}]$ and $[e_i^{min}, d_i]$ being 32, we tested our branch-and-bound procedure for parameter $\epsilon \in \{2.5, 5, 10, 15\}$. However, we only present our results for $\epsilon = 5$ since it is the parameter value which gives the best results.

The first three columns correspond to the results of the MILP model. The first column represents the average time (arithmetic mean) needed to solve the instances. If the MILP reaches the time limit, we set the execution time at 7200s. The second column corresponds to the percentage of solved instances and the last one shows the number of nodes consumed by CPLEX.

The other six columns correspond to the results of the hybrid branch-and-bound. The first column represents the average time (arithmetic mean) needed to solve the instances. The time, set in seconds, is the average of four runs of the algorithm. Furthermore, when one run of the branch-and-bound reaches the time limit, we set the execution time of this run to 7200 seconds (this execution time is playing the role of a penalty especially if only one run of the branch-and-bound solves the instance). The second and third columns show the comparison of the time spent to solve the MILPs in leaves and the time spent in the tree. The fourth column corresponds to the percentage of solved instances. We consider that an instance is solved by the algorithm if it is solved on at

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

least three runs. The fifth column corresponds to the average number of nodes of the branching tree. Finally, the last column shows the percentage of nodes on which either the checker fails (the instance is proved infeasible) or the algorithm performs some time adjustments.

The hybrid branch-and-bound solves generally more instances than the event-based MILP alone and takes less time to solve these instances. We can also see that, for the first family of instances, the "left-shift/right-shift" test is not really efficient. This comes from the fact that instances from this set are not very constrained, due to the required energy random generation, i.e. there exist many feasible solutions for them. However, the test is crucial for the relative good performance of the hybrid method on Families 2 and 3. We also see that on constrained instances (Families 2 and 3), some small-sized instances are still out of reach of all the tested methods.

We also compute the average deviation of all runs, i.e. if we denote by x_i, $i = 1, \ldots, 4$ the CPU time of each run of the branch-and-bound and by $\tilde{x}$ the average time (in column 4), then the average deviation is $\frac{1}{4}\sum_{i=1}^{4} |x_i - \tilde{x}|$. We obtain an average deviation of 0.13 for 10-task instances, 959.15 for 20-task instances, 1491.32 for 25-task instances, 1161.79 for 30-task instances and 115.99 for 60-task instances. Thus, we can see that, most of the time, the difference between the random runs may be large. Therefore, the development of better branching heuristics is an important continuation of this work.

8 Conclusions and perspectives

We have adapted the famous "left-shift/right-shift" satisfiability test for CuSP to our problem and we present three ways for computing relevant intervals for this test. These methods have been compared experimentally and we show that the adaptation of the methods of Derrien et al. [9] is the most efficient.

We also presented a new hybrid branch-and-bound algorithm for CECSP as well as an event-based MILP. We have compared these two methods and we have thereby shown the interest of integrating MILP and energetic reasoning to solve the problem.

Further research on this subject is necessary, especially to obtain an efficient method for large instances. For example, knowing whether an adaptation of the incremental algorithm for energetic reasoning [3] exists will be an interesting problem. Another open question is whether the intervals on which we perform the time-window adjustments are sufficient to perform all the possible adjustments and whether we can adapt the algorithm of [7] to our problem. The adaptation of the MILP (event-based and time-indexed), heuristics and priority rules of Resource-Constrained Project Scheduling Problem with flexible resource profiles [15] or of scheduling problems with work-content resources [10] might bring interesting results. Furthermore, these linear programs might include valid inequalities deduced from energetic reasoning.

Finally, in order to provide better applications to actual scheduling problems under energy constraints, it will be interesting to study the case where function $f_i(b)$ is no longer linear.

Acknowledgments The authors thank the referees for making relevant comments that improve the presentation of the paper.

Appendix

Table 3 Main notations

Name	Description	Domain/Expression
Problem parameters		
A	set of tasks	$\{1, \ldots, n\}$
B	resource capacity	$\mathbb{R}^+$
r_i	release date of task i	$\mathbb{R}^+$
d_i	deadline of task i	$\mathbb{R}^+$
b_i^{min}	minimum resource requirement of i	$]0, B]$
b_i^{max}	maximum resource requirement of i	$[b_i^{min}, B]$
W_i	required energy of i	$]0, f_i(b_i^{max})(d_i - r_i)]$
$f_i(b)$	power processing rate function of i	$a_i b + c_i$
e_i^{min}	earliest end time of i	$r_i + W_i/f_i(b_i^{max})$
s_i^{max}	latest start time of i	$d_i - W_i/f_i(b_i^{max})$
Recurrent notations		
st_i	starting time of i	$[r_i, d_i[$
ft_i	finishing time of i	$]r_i, d_i]$
$b_i(t)$	resource allocation function of i	$[0, b_i^{max}]$
D_{max}	deadline of the project	$\max_{i \in A} d_i$
$\underline{w}(i, t_1, t_2)$	minimum energy requirement of i inside	$\min \int_{t_1}^{t_2} f_i(b_i(t))dt$
$\underline{b}(i, t_1, t_2)$	minimum resource consumption of i inside	$\min \int_{t_1}^{t_2} b_i(t)dt$
$SL(t_1, t_2)$	slack function of	cf. page 8
Main notations of Section 4		
	minimum energy requirement of task i inside $[t_1, t_2]$ if i is:	cf. page 9
$\omega_i^+(t_1, t_2)$	left-shifted	
$\omega_i^-(t_1, t_2)$	right-shifted	
$\omega_i(t_1, t_2)$	both-shifted	
	minimum resource consumption of task i inside $[t_1, t_2]$ if i is:	cf. page 10
$\beta_i^+(t_1, t_2)$	left-shifted	
$\beta_i^-(t_1, t_2)$	right-shifted	
$\beta_i(t_1, t_2)$	both-shifted	
Main notations of Section 5		
I, I', H	intersection point of Fig. 4	cf. page 13
I_x	x-coordinate of point I	
I_y	y-coordinate of point I	
$U(t_1), D(t_1)$	variation point of $t_2 \rightarrow \underline{b}(i, t_1, t_2)$	cf. page 15
$U'(t_2), D'(t_2)$	variation point of $t_1 \rightarrow \underline{b}(i, t_1, t_2)$	cf. page 16

References

1. Artigues, C., Lopez, P., & Haït, A. (2013). The energy scheduling problem: industrial case-study and constraint propagation techniques. *International Journal of Production Economics*, *143*(1), 13–23.
2. Artigues, C., & Lopez, P. Energetic reasoning for energy-constrained scheduling with a continuous resource. *Journal of Scheduling.* doi:10.1007/s10951-014-0404-y.
3. Baptiste, P., Le Pape, C., & Nuijten, W. (2001). *Constraint-based scheduling*. Boston/Dordrecht/London: Kluwer Academic Publishers.
4. Błażewicz, J., Ecker, K.H., Pesch, E., Schmidt, G., & Węglarz, J. (2001). *Scheduling computer and manufacturing processes*. Berlin/Heidelberg: Springer-Verlag.
5. Błażewicz, J., Machowiak, M., Węglarz, J., Kovalyov, M., & Trystram, D. (2004). Scheduling malleable tasks on parallel processors to minimize the makespan. *Annals of Operations Research*, *129*(1-4), 65–80.
6. Bentley, J.L., & Ottmann, T.A. (1979). Algorithms for reporting and counting geometric intersections. *IEEE Transactions on Computers*, *28*(9), 643–647.
7. Bonifas, N. (2014). Fast propagation for the energy reasoning. In *Doctoral program of the 20th international conference on principles and practice of constraint programming (CP)* (pp. 16–22).
8. Carlier, J., & Latapie, B. (1991). Une méthode arborescente pour résoudre les problèmes cumulatifs. *RAIRO Recherche opérationnelle*, *25*(3), 311–340.
9. Derrien, A., & Petit, T. (2014). A new characterization of relevant intervals for energetic reasoning. Principles and practice of constraint programming. *Lecture Notes in Computer Science*, *8656*, 289–297.
10. Fündeling, C.-U., & Trautmann, N. (2010). A priority-rule method for project scheduling with work-content constraints. *European Journal of Operational Research*, *203*(3), 568–574.
11. Erschler, J., & Lopez, P. (1990). Energy-based approach for task scheduling under time and resources constraints. In *2nd international workshop on project management and scheduling* (pp. 115–121). Compiègne, France.
12. Józefowska, J., Mika, M., Różycki, R., Waligóra, G., & Węglarz, J. (1999). Project scheduling under discrete and continuous resources. In Węglarz, J. (Ed.) *Project scheduling: recent models, algorithms, and applications* (pp. 289–308). Boston: Kluwer Academic Publishers.
13. Kis, T. (2005). A branch-and-cut algorithm for scheduling of project with variable-intensity activities. *Mathematical Programming Series A*, *103*, 515–539.
14. Koné, O., Artigues, C., Lopez, P., & Mongeau, M. (2011). Event-based MILP models for resource-constrained project scheduling problems. *Computers & Operations Research*, *38*(1), 3–13.
15. Naber, A., & Kolisch, R. (2014). MIP models for resource-constrained project scheduling with flexible resource profiles. *European Journal of Operational Research*, *239*, 335–348.
16. Naber, A., & Kolisch, R. (2014). A continuous time model for the resource-constrained project scheduling with flexible resource profiles. In *Proceedings of the 14th international conference on project management and scheduling* (pp. 166–168). Munich, Germany.
17. Nattaf, M., Artigues, C., & Lopez, P. (2014). A polynomial satisfiability test using energetic reasoning for energy constraint scheduling. In *14th international workshop on project management and scheduling* (pp. 169–172). Munich, Germany.
18. Węglarz, J., Józefowska, J., Mika, M., & Waligóra, G. (2009). Project scheduling with finite or infinite number of activity processing modes - A survey. *European Journal of Operational Research*, *208*, 177–205.