Constraints (2017) 22:530–547
DOI 10.1007/s10601-017-9271-4

Cumulative scheduling with variable task profiles and concave piecewise linear processing rate functions

Margaux Nattaf[1] · **Christian Artigues**[1] · **Pierre Lopez**[1]

Published online: 23 May 2017

Abstract We consider a cumulative scheduling problem where a task duration and resource consumption are not fixed. The consumption profile of the task, which can vary continuously over time, is a decision variable of the problem to be determined and a task is completed as soon as the integration over its time window of a non-decreasing and continuous processing rate function of the consumption profile has reached a predefined amount of energy. The goal is to find a feasible schedule, which is an NP-hard problem. For the case where functions are concave and piecewise linear, we present two propagation algorithms. The first one is the adaptation to concave functions of the variant of the energetic reasoning previously established for linear functions. Furthermore, a full characterization of the relevant intervals for time-window adjustments is provided. The second algorithm combines a flow-based checker with time-bound adjustments derived from the time-table disjunctive reasoning for the cumulative constraint. Complementarity of the algorithms is assessed via their integration in a hybrid branch-and-bound and computational experiments on small-size instances.

Keywords Continuous scheduling · Continuous resources · Concave piecewise linear functions · Energy constraints · Energetic reasoning

This article belongs to the Topical Collection: *Integration of Artificial Intelligence and Operations Research Techniques in Constraint Programming*
Guest Editors: Michele Lombardi and Domenico Salvagnin

✉ Margaux Nattaf
nattaf@laas.fr

Christian Artigues
artigues@laas.fr

Pierre Lopez
lopez@laas.fr

[1] LAAS-CNRS, Université de Toulouse, CNRS, INSA, Toulouse, France

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. We consider the case where tasks resource requirement is not fixed but is part of the problem and has to be determined. Thus, the duration of a task is not fixed either but is determined by the resource requirement function as the task is finished once it has received a necessary amount of energy. Furthermore, we consider that the total energy received by a task is not equal to the total amount of the resource used by it. Instead we have processing rate (or efficiency) functions, which translates the required resource amounts into energy.

We perform an analysis of the structural properties of the problem for concave piecewise linear processing rate functions. To our knowledge, this variant of the cumulative constraint has never been considered in the literature. We show first that the resource demand profile of a task can be restricted to a piecewise constant function with break points at the starts and ends of tasks. From these theoretical properties, we are able to compute the minimal resource consumption of a task inside an interval in $O(1)$ and we prove that the set of the relevant intervals of polynomial size that was shown sufficient for energetic reasoning with linear functions is also sufficient in our case. Furthermore, a full characterization of relevant intervals for time-window adjustments is provided, completing the work for linear function. We also define a new propagation algorithm together with a satisfiability test, which relies on the time-table disjunctive reasoning for the cumulative constraint [6] for the first one and on a flow-based linear program for the latter one. Finally, complementarity of the algorithms is assessed via their integration in a hybrid branch-and-bound and computational experiments on small-size instances.

1 Problem statement, properties and context

In this section, we formally define the Continuous Energy-Constrained Scheduling Problem (CECSP). Then, we present a foundry application in details and finally, we exhibit some properties of the CECSP, which we will use throughout the paper.

1.1 Problem definition

In the CECSP, a set of tasks $\mathcal{A} = \{1, \ldots, n\}$ has to be scheduled using a continuous, cumulative and renewable resource of capacity B. The resource amount that a task requires during its processing time is not fixed but instead the resource usage of a task $i \in \mathcal{A}$ is a function of time, $b_i(t)$, that must be determined in a continuous time setting ($t \in \mathbb{R}^+$). Once the task is started and until its finishing time, its resource usage is constrained to lie between a maximum and a minimum requirement, b_i^{max} and $b_i^{min} \neq 0$, respectively.[1] In addition, when a task uses a part of the resource, it receives a certain amount of energy and we say that a task is finished when it has received an energy W_i. This energy is computed using a continuous, non-decreasing, concave and piecewise linear power processing rate function [3]:

$$\begin{array}{rccc} f_i : & [b_i^{min}, b_i^{max}] & \longrightarrow & \mathbb{R} \\ & b_i(t) & \longrightarrow & f_i(b_i(t)) \end{array}$$

Furthermore, each task has a release date r_i and a deadline d_i, and has to be fully executed in $[r_i, d_i]$.

[1] Although all results presented in this paper can be adapted when $b_i^{min} = 0$, for ease of notation, we assume $b_i^{min} \neq 0$.

The CECSP consists of finding, for each task, a start time $st_i \in [est_i, lst_i]$, an end time $et_i \in [eet_i, let_i]$, and its resource usage $b_i(t) \in \mathbb{R}^+, \forall t \in \mathbb{R}^+$ such that:

$$r_i \leq st_i < et_i \leq d_i \quad \forall i \in \mathcal{A} \tag{1}$$

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

$$\int_{st_i}^{et_i} f_i(b_i(t))dt = W_i \quad \forall i \in \mathcal{A} \tag{3}$$

$$\sum_{i \in \mathcal{A}} b_i(t) \leq B \quad \forall t \tag{4}$$

Example 1 In the example of Fig. 1, the energy received by task 2 is equal to $(2 \times 3 + 1) + (2 \times 4 + 1) + (2 \times 4 + 1) = 25$; the amount of resource consumed is equal to $3 + 4 + 4 = 11$.

Throughout this paper, we will use the following expression for function f_i (see Fig. 2):

$$f_i(b_i(t)) = \begin{cases} a_{i1} \cdot b_i(t) + c_{i1} & \text{if } b_i(t) \in [b_i^{min}, x_1^i[\\ a_{i2} \cdot b_i(t) + c_{i2} & \text{if } b_i(t) \in [x_1^i, x_2^i[\\ \vdots & \\ a_{iP_i} \cdot b_i(t) + c_{iP_i} & \text{if } b_i(t) \in [x_{P_i-1}^i, b_i^{max}] \end{cases}$$

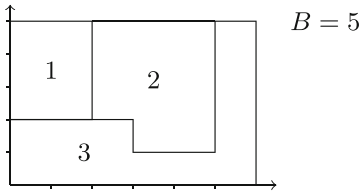
with P_i being the number of pieces of f_i and x_p^i, $p \in \{1, \ldots, P_i - 1\}$ its breakpoints.

1.2 Context

The CECSP comes from an industrial problem. This problem, presented in [1], arises in a pipe-manufacturing plant and more precisely, the foundry where metal is melted in induction furnaces. In this department, melting and heating use a huge amount of energy especially electricity. The expenses of the plant for electricity represent more than half of the annual energy costs. The cost of electricity depends on the total energy consumed and

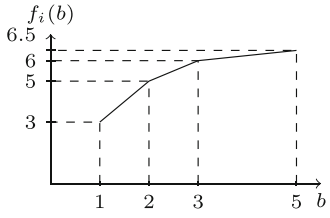
i	r_i	d_i	W_i	b_i^{min}	b_i^{max}	$f_i(b)$
1	0	2	6	3	3	b
2	1	5	25	2	4	$2b + 1$
3	0	6	21.5	1	5	Fig. 1c

a: Instance data.



b: A solution.

$$f_3(b) = \begin{cases} 2 \cdot b + 1 & \text{if } b \in [1, 2[\\ b + 3 & \text{if } b \in [2, 3[\\ 1/4 \cdot b + 21/4 & \text{if } b \in [3, 5] \end{cases}$$



c: Function $f_3(b)$.

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

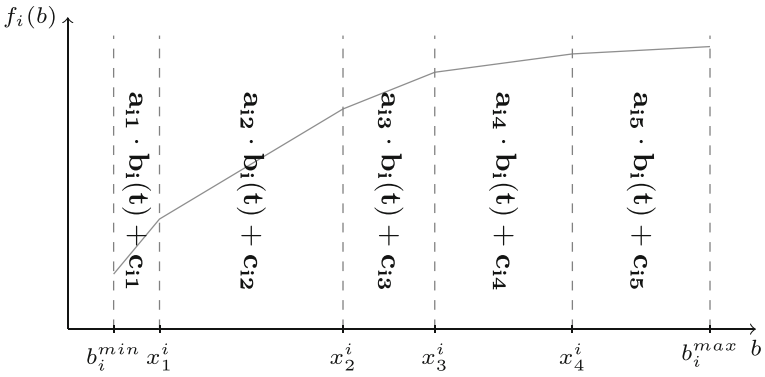


Fig. 2 Illustration of the notations for function f_i with $P_i = 5$

on penalties for power overrun in reference to a subscribed maximal power. The goal is to minimize the energy bill.

The foundry has several production lines (furnaces) and each metal operation has to be assigned to a furnace and scheduled within a time window. Furthermore, an operation has a variable duration that depends on the power given to the furnace. Thus, the electrical power of the furnaces, which can be adjusted at any time to avoid exceeding a maximum limit, can be seen as a continuous function of time to be determined. However, the function must lie within a limit (due to physical and operational considerations); thus, a minimum and a maximum power level must be satisfied for the melting operation. A melting job is composed of three sequential parts: loading, heating and unloading. The duration of loading and unloading are known but heating duration has to be determined. The heating operation can be stopped once the necessary energy has been received.

To solve this problem, the authors suppose that jobs have a piecewise constant consumption profile and define a two-step method. In the first step, scheduling of jobs on the furnaces is performed, using a constraint programming model, with fixed job durations. The task sequence and assignment are then used as data in the second step and a MILP model is used to determine their consumption profile and duration. The algorithm then returns to the first step with new jobs durations and it stops when the objective function is no more improved.

Unfortunately, the initially considered energy model was not sufficient to achieve a good energy consumption. Therefore, a new problem, which can be used to determine task starting and finishing times as well as their consumption profile was introduced in [1], the Energy Scheduling Problem (EnSP). Due to the complexity of the problem, the proposed model considers a time discretization, which can lead to suboptimal or infeasible solutions by over-constraining the problem. 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] for the CECSP, which is the continuous version of the EnSP.

Despite the fact that processing rate functions were not considered in these papers, actual processing rate functions in scheduling problems that involve energy-consuming tasks are intrinsically continuous and non-linear. As a typical example, the non-linear function given in Fig. 3 gives the shape of the processing rate function of a fuel cell [7]. An extension of the work of [2] to linear processing rate functions as well as several solution methods was proposed in [12, 13]. A preliminary version of the work presented in this paper can be found in [14].

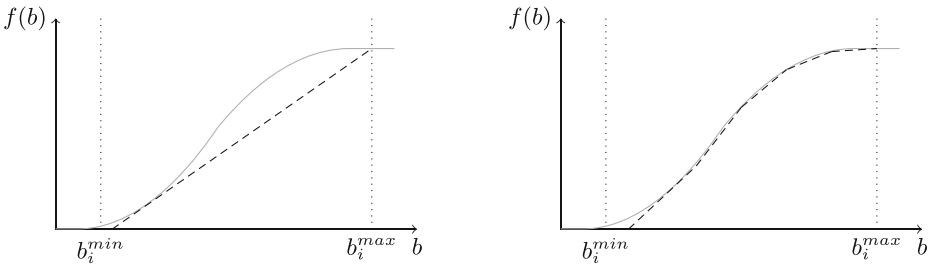


Fig. 3 Approximation from below of a processing rate function by a linear one (*left*) and by a concave piecewise linear one (*right*)

1.3 Properties and remarks

In [12], the authors use linear functions to approximate real-world ones. Despite the fact that there exist many real-world processing functions, which are concave [8, 11], approximating a function by a concave piecewise linear one is always at least as good as an approximation by a linear function (see Example 2 below). Furthermore, the authors of [12] prove the NP-hardness of their problem using a reduction from the cumulative problem. This proof is still valid in our case and then the CECSP with concave functions is NP-hard.

Example 2 Consider the neither concave nor convex function described in Fig. 3. We approximate the function from below (to obtain a relaxation) with a linear function (left side of the figure) or with a concave piecewise linear one (right side of the figure).

Clearly, the approximation is tighter with the concave piecewise linear function. Note also that this result is also valid for approximation from above or at the median point.

In order to define an efficient solution method for our problem, we prove that if there exists a solution for the CECSP, then a solution where all resource usage functions, $b_i(t)$, are piecewise constant exists. This is the statement of the following theorem:

Theorem 1 *Let $\mathcal{I}$ be a feasible instance of CECSP, with non-decreasing, concave piecewise linear functions f_i. A solution such that, for all $i \in A$, $b_i(t)$ is piecewise constant, exists. Furthermore, $\forall i \in A$ the breakpoints of $b_i(t)$ can be restricted to the start and end times of the tasks.*

Let us assume a solution S with non-piecewise constant $b_i(t)$ exists. Then, S can be transform into a new solution S' with piecewise constant $b'_i(t)$, $\forall i \in \mathcal{A}$.

Before proving Theorem 1, we prove that, if $\exists[t_1, t_2]$ where $b_i(t)$ is not constant, then $b_i(t)$ can be set to its mean value over $[t_1, t_2]$. Doing so, i will consume the same resource quantity in $[t_1, t_2]$ and will received more energy.

Lemma 1 *Let $b_{iq} = \frac{\int_{t_1}^{t_2} b_i(t)dt}{t_2 - t_1}$. Then, we have:*

$$\int_{t_1}^{t_2} b_{iq}dt = \int_{t_1}^{t_2} b_i(t)dt \tag{5}$$

$$\int_{t_1}^{t_2} f_i(b_{iq})dt \geq \int_{t_1}^{t_2} f_i(b_i(t))dt \tag{6}$$

Proof Equation (5) is trivially verified by replacing b_{iq} by its value. To prove that (6) is satisfied, we can use the following theorem, due to Jensen [9].

Theorem 2 (**Jensen**) *Let $\alpha(t)$ and $g(t)$ be two integrable functions on $[t_1, t_2]$ such that $\alpha(t) \geq 0,\ \forall t \in [t_1, t_2]$. We have:*

$$\phi\left(\frac{\int_{t_1}^{t_2} \alpha(t)g(t)dt}{\int_{t_1}^{t_2} \alpha(t)dt}\right) \geq \frac{\int_{t_1}^{t_2} \alpha(t)\phi(g(t))dt}{\int_{t_1}^{t_2} \alpha(t)dt} \tag{7}$$

where ϕ is a continuous concave function in $[\min_{t\in[t_1,t_2]} g(t)][\max_{t\in[t_1,t_2]} g(t)]$.

Replacing $\phi(t)$ by $f_i(t),\ g(t)$ by $b_i(t)$ and $\alpha(t)$ by the constant function equal to 1 gives the desired result. □

Example 3 Consider a task i and the concave piecewise linear efficiency function described in Example 1, $f_3(b)$.

Consider an interval $[t_1, t_2 = t_1 + 6]$ and functions (Fig. 4):

$$b_i(t) = \begin{cases} 3 & \text{if } t \in [t_1, t_1 + 3[\\ 1 & \text{if } t \in [t_1 + 3, t_2] \end{cases} \text{ which yields } f_i(b_i(t)) = \begin{cases} 6 & \text{if } t \in [t_1, t_1 + 3[\\ 3 & \text{if } t \in [t_1 + 3, t_2] \end{cases}$$

Thus, $\int_{t_1}^{t_2} b_i(t)dt = 12$ and $\int_{t_1}^{t_2} f_i(b_i(t))dt = 27$. Applying Lemma 1 we can replace $b_i(t)$ by $b_{iq} = 12/6 = 2$ between t_1 and t_2, which yields $f_i(b_{iq}) = 5$, $\int_{t_1}^{t_2} b_{iq}dt = 12$ and $\int_{t_1}^{t_2} f_i(b_{iq})dt = 30 \geq 27$.

Proof (Theorem 1) Let S be a feasible solution of $\mathcal{I}$ and let $(t_q)_{\{q=1..Q\}}$ be the increasing series of distinct start and end time values ($Q \leq 2n$). S' can be defined as follows:

- $b'_i(t) = \begin{cases} b_{i0} & \text{if } t \in [t_0, t_1] \\ \vdots & \\ b_{i(Q-1)} & \text{if } t \in [t_{Q-1}, t_Q] \end{cases}$ with $b_{iq} = \frac{\int_{t_q}^{t_{q+1}} b_i(t)dt}{t_{q+1}-t_q}$
- $st'_i = st_i$
- $et'_i = \min(\tau | \int_{st_i}^{\tau} f_i(b'_i(t))dt = W_i)$ and then $b'_i(t) = 0,\ \forall t > et'_i$

S' clearly verifies the energy constraints (3) since it is defined in this way. S' also satisfies the time window constraints (1) since $st_i \leq st'_i$ and $et_i \geq et'_i$ (Fig. 5).

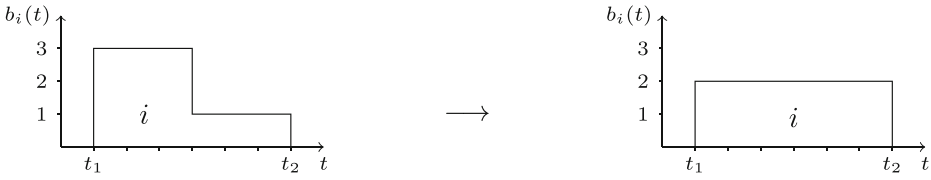
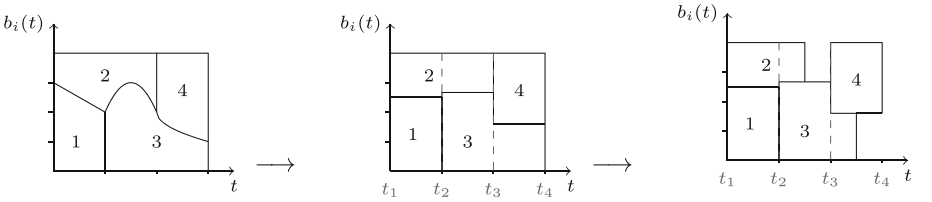


Fig. 4 Illustration of Lemma 1



a: $b_i(t)$ is not piecewise constant.

b: $b_i(t)$ is set to its mean value over $[t_q, t_{q+1}]$.

c: et_i is adjusted, so each task receives exactly W_i units of energy.

Fig. 5 Construction of S' from S

In addition, as S is a feasible solution, we have $\forall q \in \{1, \ldots, Q\}$ and $\forall t \in [t_q, t_{q+1}]$:

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

$$\Rightarrow \sum_{i \in A} b'_i(t) \leq \sum_{i \in A} b_{iq} = \sum_{i \in A} \frac{\int_{t_q}^{t_{q+1}} b_i(t)dt}{t_{q+1} - t_q} \leq B$$

and S' also verifies the resource capacity constraints (4).

Finally, we can show that S' satisfies the resource requirement constraints (2) in a similar way. □

An interesting remark can be made about Theorem 1. 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 times b_{iq}.

2 Time-table based reasoning

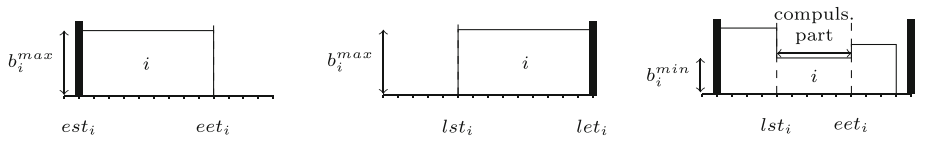
In this section, we first describe an algorithm which allows us to detect some infeasibilities and then, another algorithm performing time-bound adjustments is presented. Both are based on the well-known Time-Table reasoning [10] but the first one combines it with a flow-based algorithm, whereas the second one is an adaptation of the Time-Table Disjunctive reasoning for the cumulative constraints [6].

The Time-Table reasoning is based on the following observation: if the earliest end time eet_i of a task i is higher than its latest start time lst_i then i must be in process during interval $[lst_i, eet_i]$; this interval is called the compulsory part of i. For the CECSP, as the resource usage of i is not fixed, we can only deduce that the task will consume at least b_i^{min} units of resource (see Fig. 6).

Aggregating all compulsory parts, we can compute in $O(n^2)$ the minimum profile of the resource, denoted by $TT(t), \forall t$ and use it to detect infeasibility and to perform time-bound adjustments.

2.1 Time-table flow based reasoning

The first algorithm described in this paper embeds the concept of compulsory part into a flow-based linear program. This linear program allows us to detect some infeasibility. Indeed, if the program has no solution, then the considered CECSP instance is infeasible.



a: i starts as early as possible. b: i starts as late as possible. c: i has to be in process within $[lst_i, eet_i]$

Fig. 6 Compulsory part of a task i

To describe this program, let $(t_q)_{q\in\mathcal{Q}}$ be the increasing series of distinct variable domain bounds, i.e. t_q gather all distinct latest/earliest start/end times of all tasks. Clearly, $|\mathcal{Q}| \leq 4 \cdot n$. Then, $\forall [t_q, t_{q+1}]$, let β_{iq} (resp. w_{iq}) represent the quantity of consumed resource (resp. received energy) in this interval. Thus, the following linear program can be used as a checker:

$$\sum_{i\in A} \beta_{iq} \leq B(t_{q+1} - t_q) \quad \forall q \in \mathcal{Q} \tag{8}$$

$$\beta_{iq} \geq b_i^{min}(t_{q+1} - t_q) \quad \forall i \in A\,; \forall q \in \mathcal{Q} \mid s_i^{max} \leq t_q \leq e_i^{min} \tag{9}$$

$$\beta_{iq} \leq b_i^{max}(t_{q+1} - t_q) \quad \forall i \in A\,; \forall q \in \mathcal{Q} \tag{10}$$

$$\beta_{iq} = 0 \quad \forall i \in A\,; \forall q \in \mathcal{Q} \mid t_q \notin [r_i, d_i] \tag{11}$$

$$w_{iq} \leq a_{ip}\beta_{iq} + c_{ip}(t_{q+1} - t_q) \quad \forall i \in A\,; \forall q \in \mathcal{Q}\,; \forall p \in \{1, \ldots, P_i\} \tag{12}$$

$$w_{iq} \leq M\beta_{iq} \quad \forall i \in A\,; \forall q \in \mathcal{Q} \tag{13}$$

$$\sum_{q\in\mathcal{Q}} w_{iq} = W_i \quad \forall i \in A \tag{14}$$

$$\beta_{iq} \geq 0,\ w_{iq} \geq 0 \quad \forall i \in A\,; \forall q \in \mathcal{Q} \tag{15}$$

where M is some large enough constant.

The compulsory part constraints are expressed by constraints (9). Constraints (8) model the resource capacity limitations. Constraints (10) impose that the maximum resource requirements are satisfied. Constraints (11) set the resource consumption of task i in $[t_q, t_{q+1}]$ to be equal to 0, if $[t_q, t_{q+1}] \not\subseteq [r_i, d_i]$. Constraints (12) together with constraints (13) ensure a correct resource conversion. Indeed, as f_i is concave and piecewise linear, the first constraints ensure that w_{iq} can take the value $f_i(\frac{\beta_{iq}}{t_{q+1}-t_q}) \cdot (t_{q+1} - t_q)$ whereas the second ones set w_{iq} to zero if $\beta_{iq} = 0$. Finally, constraints (14) state that the tasks received the required energy.

Note that this test ensures the same level of consistency than the Time-Table Overload Check [16]. Indeed, it does the Overload Check at the same time as the Time-Table. This is the same consistency check that performs Time-Table Edge-Finding [15].

We now describe the adaptation of the Time-Table Disjunctive reasoning for the cumulative constraint [6] to our problem. This reasoning will be used to perform time-bound adjustments on the start and end time variables.

2.2 Time-table disjunctive reasoning

The idea of the Time-Table Disjunctive reasoning is to take advantage of the minimum resource profile to detect disjunctive pairs of tasks, i.e. tasks that cannot be processed in parallel, dynamically.

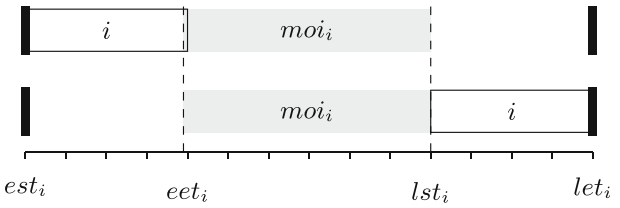


Fig. 7 Minimum overlapping interval

Indeed, the disjunctive reasoning only considers pairs of tasks that exceed the capacity of the resource if they are scheduled in parallel, i.e. $b_i^{min} + b_j^{min} > B$. However, tasks in $\mathcal{A} \setminus \{i, j\}$ may not leave B units of resource available during the overlap of i and j.

Furthermore, if task i has no compulsory part then an additional filtering can be done when starting a task j at est_j would make it overlap i in every schedule. This is due to the fact that j cannot contain a time interval that i must overlap.

Therefore, the authors in [6] defined the smallest interval overlapping i in every solution, called the minimum overlapping interval and denoted by moi_i. For the CECSP, this interval is exactly $[est_i, lst_i] \setminus [est_i, eet_i[$, i.e the smallest interval containing $[eet_i, lst_i]$ (cf. Fig. 7). Note that, if a task has a compulsory part, then $moi_i = \emptyset$.

Let $et_i^{max} = r_i + W_i / f_i(b_i^{min})$. Then, using the minimum resource profile to compute the available resource and to search for disjunctive pairs of tasks, the time-table disjunctive reasoning can be stated as follows:

Proposition 1 *Let $i \neq j$ be two tasks having no compulsory consumption and such that $b_i^{min} + b_j^{min} + min_{t \in moi_i} TT(t) > B$. If $moi_i \subseteq [est_j, et_j^{max}]$ we must have $eet_i \leq st_j$ and so est_j can be adjusted to eet_i.*

Example 4 Consider the example of Fig. 8 adapted from [6]. Due to the minimum profile, i and j cannot overlap and, since j cannot be scheduled before i ($moi_i \subseteq [est_j, et_j^{max}]$), j has to be scheduled after and then we can set est_j to eet_i.

A symmetrical reasoning allows to make adjustment of let_j. Furthermore, Proposition 1 can easily be extended to the case where tasks i and j has no compulsory consumption following the method presented in [6] for the cumulative constraint.

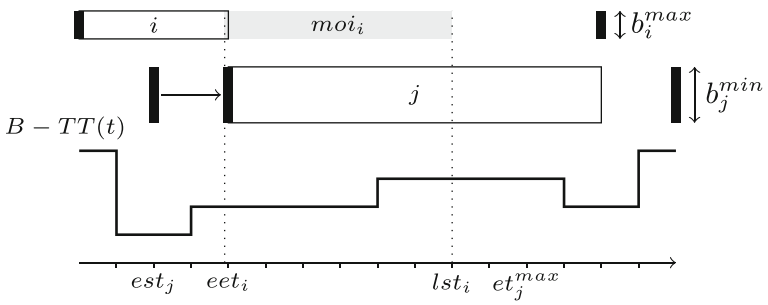


Fig. 8 Example of time-table disjunctive adjustments

3 Energetic reasoning

In this section, we present the extension of the well-known energetic reasoning to the CECSP with concave and piecewise linear processing rate functions. This work extends the work of [13] for linear processing rate functions. The full characterization of relevant intervals for the time-bound adjustments is also provided, extending the result for the cumulative constraint [4]. This characterization was not presented in [13].

First, we present the central idea of the reasoning and we use it to provide a satisfiability test as well as time-bound adjustments. The second part of this section will be dedicated to the characterization of relevant intervals.

3.1 Satisfiability test and time-bound adjustments

The idea of the energetic reasoning is to test whether the available resource within an interval $[t_1, t_2]$ is sufficient to provide the minimum resource quantity needed by each task i in this interval, denoted by $\underline{\beta}(i, t_1, t_2)$. If not, then the corresponding instance of the CECSP is infeasible.

Theorem 3 [5] *Let $\mathcal{I}$ be an instance of CECSP. If there exists (t_1, t_2) such that $B(t_2 - t_1) - \sum_{i \in A} \underline{\beta}(i, t_1, t_2) < 0$ then $\mathcal{I}$ is infeasible.*

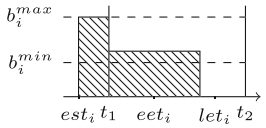
In order to compute the minimum resource consumption of task i in $[t_1, t_2]$, we first have to compute the minimum energy requirement of i in this interval, denoted by $\underline{w}(i, t_1, t_2)$. This minimum requirement is obtained by scheduling as much energy as possible outside the interval $[t_1, t_2]$ while satisfying constraints (1)–(4). This always corresponds to a case where the activity is:

- left-shifted: the task starts at est_i and is scheduled at its maximum requirement between est_i and t_1,
- right-shifted: the task ends at let_i and is scheduled at its maximum requirement between t_2 and let_i,
- or both: when scheduling at minimum requirement inside $[t_1, t_2]$ implies to have a non-zero requirement both in $[est_i, t_1]$ and in $[t_2, let_i]$.

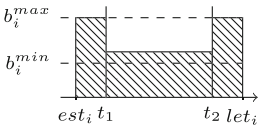
We denote by $\omega_{LS}(i, t_1, t_2)$ (resp. $\omega_{RS}(i, t_1, t_2)$) the minimum energy requirement of task i if it is shifted as early as possible in $[t_1, t_2]$ (resp. as late as possible). Since both cases are symmetric, we only describe the first one. Thus, $\omega_{LS}(i, t_1, t_2)$ is equal to:

Expression	Condition	Figure
0	$t_1 \geq eet_i \vee t_2 \leq est_i$	–
W_i	$t_1 \leq est_i \wedge let_i \leq t_2$	–
$W_i - f_i(b_i^{max})(t_1 - est_i)$	$est_i \leq t_1 \leq eet_i \wedge let_i \leq t_2$	Fig. 9a
$\min\left(\begin{array}{c} W_i - f_i(b_i^{max})(t_1 - est_i), \\ \max\left\{\begin{array}{c} W_i - f_i(b_i^{max})(t_1 - est_i + let_i - t_2)), \\ f_i(b_i^{min})(t_2 - t_1) \end{array}\right\} \end{array}\right)$	$est_i \leq t_1 < t_2 \leq let_i$	Fig. 9b
$\max\left\{\begin{array}{c} W_i - f_i(b_i^{max})(let_i - t_2)), \\ f_i(b_i^{min})(t_2 - est_i) \end{array}\right\}$	$t_1 \leq est_i \wedge t_2 < let_i$	Fig. 9c

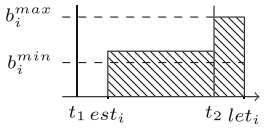
The three last cases are illustrated in Fig. 9.



a: as much energy as possible is scheduled before t_1.



b: as much energy as possible is scheduled before t_1 and after t_2.



c: as much energy as possible is scheduled after t_2.

Fig. 9 Computation of $\omega_{LS}(i, t_1, t_2)$

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

$$\underline{w}(i, t_1, t_2) = \min(\omega_{LS}(i, t_1, t_2), \omega_{RS}(i, t_1, t_2)) \tag{16}$$

The minimum energy requirement is then used to compute the minimum resource requirement. First, let $I = [est_i, let_i] \cap [t_1, t_2]$, then $\underline{\beta}(i, t_1, t_2)$ can be computed by solving the following program:

$$\begin{aligned} \text{minimize} \quad & \int_I b_i(t)dt && (17)\\ \text{subject to} \quad & \int_I f_i(b_i(t))dt \geq \underline{w}(i, t_1, t_2) && (18)\\ & b_i(t) \geq b_i^{min} && (19) \end{aligned}$$

Indeed, the goal is to find the minimum resource quantity (17) i has to consume in I to receive an energy $\underline{w}(i, t_1, t_2)$ (18). In addition, we have to make sure that the minimum requirement constraints are satisfied by the solution of the program (19).

Then, Lemma 1 can be used to simplify the program. Actually, the lemma applied to the problem involving only task i implies a solution with constant $b_i(t)$, say $b_i(t) = \tilde{b_i}$ exists and then the program can be rewritten as follows:

$$\begin{aligned} \text{minimize} \quad & \tilde{b_i}|I| \\ \text{subject to} \quad & f_i(\tilde{b_i})|I| \geq \underline{w}(i, t_1, t_2) \\ & \tilde{b_i} \geq b_i^{min} \end{aligned}$$

Thus, we have two cases to consider: either the task can be scheduled at b_i^{min} during the whole interval I or not. In the first case, we can remove the second constraint and we obtain:

$$\begin{aligned} & \tilde{b_i} = \min(b \in [0, b_i^{max}] \mid f_i(b) \geq \underline{w}(i, t_1, t_2)/|I|) \\ \Rightarrow \quad & \tilde{b_i} = f_i^{-1}(\underline{w}(i, t_1, t_2)/|I|) \end{aligned}$$

Since f_i is concave and piecewise linear, f_i^{-1} can easily be computed[2] and $\tilde{b_i}|I| = f_i^{-1}(\underline{w}(i,t_1,t_2)/|I|) \cdot |I| = \underline{\beta}(i,t_1,t_2)$ is equal to:

$$f_i^{-1}\left(\frac{\underline{w}(i,t_1,t_2)}{|I|}\right) = \begin{cases} \frac{\underline{w}(i,t_1,t_2)-c_{i0}|I|}{a_{i0}|I|} & \text{if } \frac{\underline{w}(i,t_1,t_2)}{|I|} \in [f_i(b_i^{min}),\ f_i(\gamma_1^i)] \\ \frac{\underline{w}(i,t_1,t_2)-c_{i1}|I|}{a_{i1}|I|} & \text{if } \frac{\underline{w}(i,t_1,t_2)}{|I|} \in [f_i(\gamma_1^i),\ f_i(\gamma_2^i)] \\ \vdots & \\ \frac{\underline{w}(i,t_1,t_2)-c_{iP_i}|I|}{a_{iP_i}|I|} & \text{if } \frac{\underline{w}(i,t_1,t_2)}{|I|} \in [f_i(\gamma_{P_i}^i),\ f_i(b_i^{max})] \end{cases}$$

The second case corresponds to the case where executing the task at b_i^{min} during the whole interval I give too much energy to the task, i.e. $\underline{w}(i,t_1,t_2) < |I| \cdot f_i(b_i^{min})$. In this case, $\underline{\beta}(i,t_1,t_2)$ is equal to $b_i^{min} \cdot \underline{w}(i,t_1,t_2)/f_i(b_i^{min})$.

Finally, the expression of the minimum resource consumption is:

$$\underline{\beta}(i,t_1,t_2) = \max\left\{ b_i^{min}\frac{\underline{w}(i,t_1,t_2)}{f_i(b_i^{min})},\ \max_{p\in\{1,\ldots,P_i\}}\left(\frac{1}{a_{ip}}(\underline{w}(i,t_1,t_2) - |I|c_{ip})\right)\right\} \tag{20}$$

Example 5 Consider the instance described in Example 1. In particular, we compute $\underline{w}(i,t_1,t_2)$ and $\underline{\beta}(i,t_1,t_2)$ for task 3 and interval $[0,4]$ and $[0,6]$.

For interval $[0,6]$, we have $\underline{w}(3,0,6) = W_3 = 21.5$ since $[est_3, let_3] \subseteq [t_1,t_2]$. Then, the computation of $\underline{\beta}(3,0,6)$ falls into the case where $\underline{w}(3,0,6) \geq f_3(b_i^{min}[3]) \cdots |I| = 18$. Thus, $\underline{w}(3,0,6)/|I| = 21.5/6 \simeq 3.583$ belonging to interval $[f_3(1),\ f_3(2)[= [2,5[$, we have (see Fig. 10a):

$$\underline{\beta}(3,0,6) = \frac{\underline{w}(3,0,6) - 1\cdot 6}{2\cdot 6}\cdot 6 = \frac{21.5-6}{12}\cdot 6 = \frac{31}{4}$$

For interval $[0,4]$, we have $\underline{w}(3,0,4) = W_3 - 2\cdot f_3(b_i^{max}[3]) = 8.5$. Here, the computation of $\underline{\beta}(3,0,4)$ falls into the case where $\underline{w}(3,0,4) < f_3(b_i^{min}[3]) \cdots |I| = 12$. Thus, we have (see Fig. 10b):

$$\underline{\beta}(3,0,4) = b_i^{min}[3] \cdot \frac{\underline{w}(3,0,4)}{f_3(b_i^{min}[3])} = 1 \cdot \frac{8.5}{3} = \frac{17}{6}$$

We now describe how this quantity is used to perform time-bound adjustments. These adjustments are similar to the ones for linear functions described in [12], so we just briefly present them.

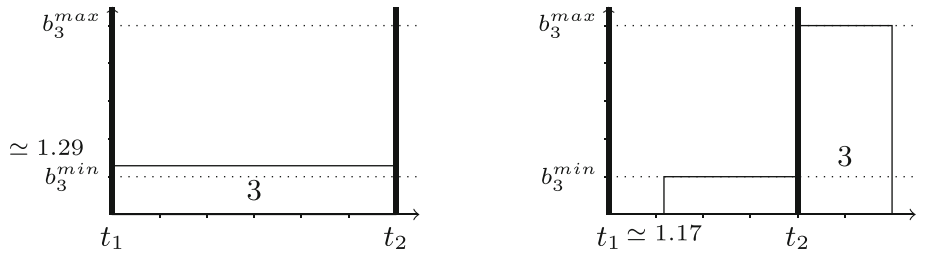
We start by defining some notation. We denote by $\beta_{LS}(i,t_1,t_2)$ (respectively $\beta_{RS}(i,t_1,t_2)$) the minimal resource consumption corresponding to $\omega_{LS}(i,t_1,t_2)$ (resp. $\omega_{RS}(i,t_1,t_2)$).

We have:

$$\beta_{LS}(i,t_1,t_2) = \max\left\{ b_i^{min}\frac{\omega_{LS}(i,t_1,t_2)}{f_i(b_i^{min})},\ \max_{p\in\{1,\ldots,P_i\}}\left(\frac{1}{a_{ip}}(\omega_{LS}(i,t_1,t_2) - |I|c_{ip})\right)\right\}$$

and a similar expression for $\beta_{RS}(i,t_1,t_2)$.

[2]Indeed, suppose that f_i is constant in some pieces the first one being $p \in \{1,\ldots,P_i\}$, which would cause problem to compute f_i^{-1}. In that case, f_i must be constant on all pieces $\{p,\ldots,P_i\}$ due to concavity. It follows that b_i^{max} can be reduced to the start of piece p.



a: Computation of $\underline{\beta}(3, 0, 6)$. b: Computation of $\underline{\beta}(3, 0, 4)$.

Fig. 10 Example of computation of $\underline{\beta}(i, t_1, t_2)$

Lemma 2 *If $t_1 > r_i$ and there exists $[t_1, t_2]$ such that:*

$$\sum_{\substack{j \in A \\ j \neq i}} \underline{\beta}(j, t_1, t_2) + \beta_{RS}(i, t_1, t_2)) > B(t_2 - t_1)$$

then,

$$lst_i \leq t_1 - \frac{1}{b_i^{max}}\Big(\sum_{\substack{j \in A \\ j \neq i}} \underline{\beta}(j, t_1, t_2) + \beta_{RS}(i, t_1, t_2) - B(t_2 - t_1))\Big)$$

and, if $b_i^{min} \neq 0$,

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

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, $\sum_{j \in A;\ j \neq i} \underline{\beta}(j, t_1, t_2) + \beta_{RS}(i, t_1, t_2)$ 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 $\sum_{j \in A;\ j \neq i} \underline{\beta}(j, t_1, t_2) + \int_{t_1}^{t_2} b_i(t) \geq \sum_{j \in A;\ j \neq i} \underline{\beta}(j, t_1, t_2) + \beta_{RS}(i, t_1, t_2) \geq B(t_2 - t_1)$.

Furthermore, $\sum_{j \in A;\ j \neq i} \underline{\beta}(t_1, t_2, j) + \beta_{RS}(i, t_1, t_2) - B(t_2 - t_1)$ 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. Similar arguments lead to the adjustment on let_i. An example of such adjustments can be found in [12].

3.2 Relevant intervals for the energetic reasoning

In [12], the authors present a full characterization of the relevant intervals for the satisfiability test in the case where processing rate functions are linear. These intervals are exactly the ones that are relevant for the case of concave piecewise linear functions. Furthermore, as

these intervals are included in the set of relevant intervals for the time-bound adjustments, we only present the second results.

Recall that an adjustment can be performed on task i if $B(t_2 - t_1) - \sum_{i \neq j} \underline{\beta}(j, t_1, t_2)] - \beta_{RS}(i, t_1, t_2) < 0$. Thus, we are looking for all intervals $[t_1, t_2]$ such that the function $B(t_2 - t_1) - \sum_{i \neq j} \underline{\beta}(j, t_1, t_2) - \beta_{RS}(i, t_1, t_2)$ is negative.

Theorem 4 [4] *$B(t_2 - t_1) - \sum_{i \neq j} \underline{\beta}(j, t_1, t_2) - \beta_{RS}(i, t_1, t_2)$ is locally minimum in interval $[t_1, t_2]$ only if one of the four following conditions is satisfied:*

$$\exists (k, \ell), \quad \frac{\delta^+ \underline{\beta}(k, t_1, t_2)}{\delta t_1} < \frac{\delta^- \underline{\beta}(k, t_1, t_2)}{\delta t_1} \wedge \frac{\delta^+ \underline{\beta}(\ell, t_1, t_2)}{\delta t_2} < \frac{\delta^- \underline{\beta}(\ell, t_1, t_2)}{\delta t_2} \tag{21}$$

$$\exists k, \quad \frac{\delta^+ \underline{\beta}(k, t_1, t_2)}{\delta t_1} < \frac{\delta^- \underline{\beta}(k, t_1, t_2)}{\delta t_1} \wedge \frac{\delta^+ \beta_{RS}(i, t_1, t_2)}{\delta t_2} < \frac{\delta^- \beta_{RS}(i, t_1, t_2)}{\delta t_2} \tag{22}$$

$$\exists \ell, \quad \frac{\delta^+ \beta_{RS}(i, t_1, t_2)}{\delta t_1} < \frac{\delta^- \beta_{RS}(i, t_1, t_2)}{\delta t_1} \wedge \frac{\delta^+ \underline{\ell}(i, t_1, t_2)}{\delta t_2} < \frac{\delta^- \underline{\beta}(\ell, t_1, t_2)}{\delta t_2} \tag{23}$$

$$\frac{\delta^+ \beta_{RS}(i, t_1, t_2)}{\delta t_1} < \frac{\delta^- \beta_{RS}(i, t_1, t_2)}{\delta t_1} \wedge \frac{\delta^+ \beta_{RS}(i, t_1, t_2)}{\delta t_2} < \frac{\delta^- \beta_{RS}(i, t_1, t_2)}{\delta t_2} \tag{24}$$

with $\frac{\delta^+ f}{\delta t_2}$ (resp. $\frac{\delta^- f}{\delta t_2}$) the right (resp. left) derivative of f w.r.t. t_2.

This theorem is then used to characterize, for a task i and a fixed t_1, the value of function $t_2 \rightarrow \underline{\beta}(i, t_1, t_2)$ or $t_2 \rightarrow \beta_{RS}(i, t_1, t_2)$ for which its left derivative is greater than its right, and reciprocally for fixed t_2. Then, the two results are combined to obtain the list of relevant intervals $[t_1, t_2]$.

Since there are many cases to consider, we only present how we obtain relevant t_2 for the case where $W_i > f_i(b_i^{min})(let_i - est_i)$. The other cases to consider are $b_i^{min} = b_i^{max}$ (see [4]) and $W_i \leq f_i(b_i^{min})(let_i - est_i)$.

First, let H (resp. I') be the intersection point of line $f_i(b_i^{min})(t_2 - t_1) = W_i - (let_i - t_2) f_i(b_i^{max})$ and $f_i(b_i^{min})(t_2 - t_1) = W_i - (t_1 - est_i) f_i(b_i^{max})$ (resp. $W_i - (t_1 - est_i + let_i - t_2) f_i(b_i^{max}) = f_i(b_i^{min})(t_2 - t_1)$ and $t_2 = d_i$).

Also, let $U(t_1)$ (resp. $D(t_1)$) be the point t_2 such that $f_i(b_i^{min})(t_2 - t_1) = W_i - (t_1 - est_i) f_i(b_i^{max})$ (resp. $f_i(b_i^{min})(t_2 - t_1) = W_i - (let_i - t_2) f_i(b_i^{max})$

Lemma 3 *Let i be a task s.t. $W_i > f_i(b_i^{min})(let_i - est_i)$. Then, for any fixed t_1, at most two intervals $[t_1, t_2]$ satisfying the second condition of (21) exist and at most two intervals $[t_1, t_2]$ satisfying the second condition of (22) exist. These intervals are described in* Table 1.

With H_{t_1} (resp. I'_{t_1}) being the projection on the x-axis of point H (resp. I').

Proof We only present how to obtain relevant t_2 for the second line of Table 1. The other cases can be obtained in a similar way. In order to prove the lemma, we analyze the variation of $t_2 \rightarrow \underline{\beta}(i, t_1, t_2)$. Figure 11 represents these variations.

Where $expr_{ip} = \frac{1}{a_{ip}} \left(W_i - f_i(b_i^{max})(let_i - t_2 + t_1 - est_i) - c_{ip}(t_2 - t_1) \right)$ The intervals for which the left derivative is smaller than the right are $[t_1, let_i]$ and $[t_1, D(t_1)]$. Indeed, since f_i is a concave piecewise linear function, we have $a_{ip} > a_{ip+1}$ and $c_{ip} < c_{ip+1}$, and we have $\frac{\delta^- expr_{ip}}{\delta t_2} < \frac{\delta^+ expr_{ip+1}}{\delta t_2}$ □

Table 1 Relevant t_2 for case $W_i > f_i(b_i^{min})(let_i - est_i)$

Function	Relevant intervals	Condition
$\underline{\beta}(i, t_1, t_2)$	$[t_1, let_i]$	if $t_1 \leq est_i$
	$[t_1, let_i]$ and $[t_1, D(t_1)]$	else if $t_1 \leq I'_{t_1}$
	$[t_1, U(t_1)]$ and $[t_1, D(t_1)]$	else if $t_1 < lst_i \vee t_1 < H_{t_1}$
	$[t_1, let_i + est_i - t_1]$	else if $eet_i < lst_i \wedge t_1 \geq H_{t_1}$
	$[t_1, U(t_1)]$	else if $lst_i \leq eet_i \wedge t_1 \geq lst_i$
	none	else if $t_1 \geq eet_i$
$\beta_{RS}(i, t_1, t_2)$	$[t_1, let_i]$	if $t_1 \leq est_i \wedge let_i > t_1 \geq lst_i$
	$[t_1, let_i]$ and $[t_1, D(t_1)]$	if $t_1 > est_i \wedge t_1 < lst_i$
	none	otherwise

4 Computational results

In this section, we start by presenting how we have conducted our experiments on the algorithms stated in this paper. Then, we describe the results of these experiments.

Our propagation algorithms and satisfiability tests were embedded in a hybrid branch-and-bound combining branching scheme and mixed-integer linear programming (MILP). This procedure is an adaptation of the one in [12] for linear functions.

Hybrid branch-and-bound At first, a branch-and-bound algorithm is used to reduce the size of possible start and end intervals (until their size is less than a given $\epsilon > 0$) and, then, an event-based MILP is used in order to find exact task start and end times and to determine the quantity of resource allocated to i between two consecutive events.

The branching procedure is as follows. At the beginning, a task can start (resp. end) at any time $st_i \in [est_i, lst_i]$ (resp $et_i \in [eet_i, let_i]$). The idea is, at each node, to reduce the size of one of these intervals by creating two nodes splitting the interval into two parts of equal size.

At each node, we apply one or both of the satisfiability tests described above and, if the test does not fail, we perform the corresponding time-window adjustments. We continue this procedure using a depth-first strategy until all intervals are smaller than an ϵ. When it happens, the remaining solution space is searched via the event-based MILP.

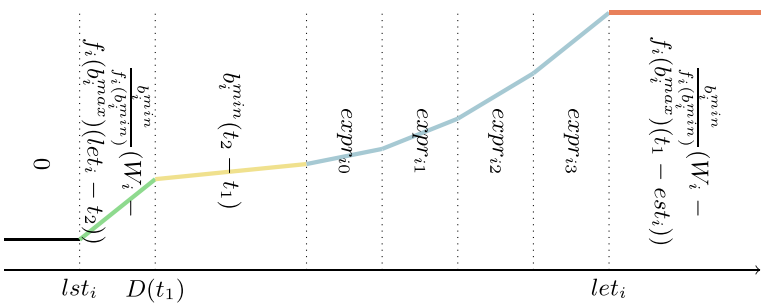


Fig. 11 Relevant intervals for case $est_i < t_1 \leq I'_{t_1}$

The MILP used in our algorithm is based on the on/off event-based formulation for the CECSP with linear processing rate functions [12]. In this formulation, 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 and $\mathcal{E} = \{1, \ldots, 2n\}$ represents the index set of these events. We use a binary variable z_{ie} to assign the different event dates to the start and end times of the tasks. Indeed, z_{ie} is equal to 1 if and only if task i is in process during interval $[t_e, t_{e+1}]$. Finally, two continuous variables b_{ie} and w_{ie} are also defined. These variables stand for the quantity of resource used by task i and for the energy received by i between events t_e and t_{e+1}. Since the event-based MILP used for our case is almost the same than the one used for linear functions, we are not describing the model here. The main difference lies in the constraints converting resource into energy. These constraints were as follows in the previous model:

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

and are replaced by the following ones in the new model:

$$W_{ie} \leq a_{ip} B_{ie} + c_{ip}(t_{e+1} - t_e) \quad \forall i \in A\ ;\ \forall p \in P_i;\ \forall e \in \mathcal{E}$$

Experiments 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. The hybrid branch-and-bound algorithm is coded in C++ and uses CPLEX 12.6 with 2 threads at each leaf.

The heuristic tested for choosing the variable on which the algorithm will branch is the following one: we choose the variable corresponding to the smallest size interval among all $[est_i, lst_i]$ and $[eet_i, let_i]$. The parameter ϵ used in the experiments is equal to 5 since this parameter value provides the best results for the case of linear efficiency functions [12].

To generate instances with concave piecewise linear processing rate functions, we use instances of [12] with identical functions. First, the instances were solved using the time-indexed mixed integer linear program described in [12]. In this formulation, the planning horizon is discretized in T time periods of size 1 and a variable b_{it} is used to represent the resource consumption of task i in period t.

The efficiency functions f_i are generated by randomly selecting a number of pieces P_i. The interval $[b_i^{min}, b_i^{max}]$ is then divided into P_i parts. For each piece p, a random coefficient a_{ip} such that $a_{ip} < a_{ip-1}$ is generated and c_{ip} is computed to ensure the continuity of the function. Finally, W_i is set to $W_i = \sum_{t=1}^{T} f_i(b_{it})$. We repeat this process until we obtain 80 instances with 10 tasks and 140 instances with 20 tasks.

Table 2 presents the results of the hybrid branch-and-bound with a time limit of 7200 seconds. The first row describes the results of the hybrid branch-and-bound with the time-table flow based reasoning (TTFlow), the second row with the energetic reasoning (ER), the third row with the time-table disjunctive reasoning (TTDR) and the last row with both ER and TTFlow tests. For each row, the first column presents the time needed to solve the instances, the second column shows the time spent in the tree, the third column the percentage of solved instances, the fourth column the number of solved MILPs (if the instance is solved), the fifth column the number of explored nodes (if the instance is solved), and the last column describes the percentage of "out of memory".

We can see that the time-table flow and the time-table disjunctive reasoning provide the best results. For the 10-task instances, the time-table flow solves the instances faster but the time spent in the tree is higher than for the time-table disjunctive reasoning while

Table 2 Results of the hybrid branch-and-bound with $\epsilon = 5$

#tasks	Reasoning	Total time(s)	Tree time(s)	%solved	#MILP	#nodes	%OOM
10	TTFlow	124.4	28.3	96.6	1	14.9	3.4
	ER	1176	8.26	84.7	3346.6	6732.4	1.7
	TTDR	244.3	4.80	96.6	1	14.9	3.4
	ER + TTFlow	542.1	25.4	86.4	1	16.8	3.4
20	TTFlow	83.8	27.3	99.2	1.01	37.5	0.8
	ER	4961.9	9.71	31.1	1.03	36.9	3.8
	TTDR	79.1	22.6	99.2	1	37.0	0.8
	ER + TTFlow	834.4	38.5	70.1	1.01	100.8	8.4

the number of nodes and MILP solved is equivalent for both algorithms. For the 20-task instances, both algorithms achieved similar performances both in terms of solving time and solved instances.

The energetic reasoning has the worst performances and the combination of the time-table flow and the energetic reasoning improves significantly the performances of the solution method in comparison with the use of the energetic reasoning alone. However, the time-table flow and time-table disjunctive reasoning is better than these two algorithms.

Another remark we can do concerns the huge difference in terms of number of MILPs and nodes between the resolution of the 10-task and 20-task instances with the energetic reasoning. This can be explained by the small number of solved instances for the 20-task instances. Indeed, the number of nodes/MILPs in the table is associated only with solved instances.

In addition, we can see that most of the time is spent in the MILP resolution. Therefore, improving the formulation is one of the main perspectives of this work.

5 Conclusions

This paper presents the extension of the energetic reasoning and the time-table disjunctive reasoning to the CECSP with concave and piecewise linear processing rate functions. A full characterization of the relevant intervals on which the time-bound adjustments of the energetic reasoning has to be applied is provided. Both methods are then embedded in a hybrid branch-and-bound and tested on small-size instances.

The interest of considering concave piecewise processing rate functions is shown through examples and experiments. Furthermore, all the new results described in this paper are still valid for linear functions.

For the perspectives, time has to be spent on the instance generation and on the resolution of larger instances. One way of doing this will be by improving the MILP. Also, the method can be improved by the use of dedicated branching heuristics. Finally, the consideration of objective functions is an important perspective of this work.

Acknowledgments The authors thank José Verschae for enlightening discussions. This study was partially supported by project "Energy-Efficient and Robust approaches for the Scheduling of Production, Services and Urban Transport", ECOS/CONICYT, N° C13E04.

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. (2015). Energetic reasoning for energy-constrained scheduling with a continuous resource. *Journal of Scheduling*, *18*(3), 225–241.
3. 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.
4. Derrien, A., & Petit, T. (2014). A new characterization of relevant intervals for energetic reasoning. In *International conference on principles and practice of constraint programming, CP 2014 vol. 8656 of lecture notes in computer science* (289–297). Springer International Publishing.
5. 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.
6. Gay, S., Hartert, R., & Schaus, P. (2015). Time-table disjunctive reasoning for the cumulative constraint. In *International conference on AI and OR techniques in constraint programming for combinatorial optimization problems, CPAIOR 2015, vol. 9075 of lecture notes in computer science* (pp. 157–172). Springer International Publishing.
7. Ngueveu, S.U., Artigues, C., & Lopez, P. (2016). Scheduling under a non-reversible energy source: an application of piecewise linear bounding of non-linear demand/cost functions. *Discrete Applied Mathematics*, *208*, 98–113.
8. Hung, M.N., Le Van, C., & Michel, P. (2005). Non-convex aggregative technology and optimal economic growth. Cahiers de la Maison des Sciences Économiques 2005.95 - ISSN : 1624-0340.
9. Jensen, J.L.W.V. (1906). Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta Mathematica*, *30*(1), 175–193.
10. Lahrichi, A. (1982). Ordonnancements. La notion de parties obligatoires et son application aux problèmes cumulatifs. *RAIRO - Operations Research*, *16*(3), 241–262.
11. Lewis, J. Algebra symposium: optimizing fuel consumption. http://homepages.math.uic.edu/~jlewis/math165/asavgcost.pdf.
12. Nattaf, M., Artigues, C., Lopez, P., & Rivreau, D. (2015). Energetic reasoning and mixed-integer linear programming for scheduling with a continuous resource and linear efficiency functions. *OR Spectrum*, 1–34.
13. Nattaf, M., Artigues, C., & Lopez, P. (2015). A hybrid exact method for a scheduling problem with a continuous resource and energy constraints. *Constraints*, *20*(3), 304–324.
14. Nattaf, M., Artigues, C., & Lopez, P. (2015). Flow and energy based satisfiability tests for the continuous energy-constrained scheduling problem with concave piecewise linear functions. In *CP* Doctoral Program 2015 (pp. 70–81). Cork.
15. Vilím, P. (2011). Timetable edge finding filtering algorithm for discrete cumulative resources. In *Integration of AI and OR techniques in constraint programming for combinatorial optimization problems. CPAIOR 2011 vol 6697 of lecture notes in computer science*. Berlin: Springer.
16. Wolf, A., & Schrader, G. (2006). $O(nn)$ overload checking for the cumulative constraint and its application. In Umeda, M., Wolf, A., Bartenstein, O., Geske, U., Seipel, D., & Takata, O. (Eds.) *Declarative programming for knowledge management. INAP 2005 lecture notes in computer science*, Vol. 4369. Berlin: Springer.