


Improved Energetic Reasoning Checker for Cumulative Constraint with Profile

Rosaly Zoller Bodo Ngono ✉ 

University of Calabar, Faculty of Physical Sciences, Department of Computer Science, Calabar, Nigeria.

Yves Pascal Ndjopnang Wantiep ✉ 

University of Maroua, Faculty of Science, Department of Mathematics and Computer Science, P.O.Box: 814 Maroua, Cameroon.

Roger Kameugne ✉ 

University of Maroua, Faculty of Science, Department of Mathematics and Computer Science, P.O.Box: 814 Maroua, Cameroon.

Abstract

This paper applies for the first time the Profile data structure to the energetic reasoning checker rule. Tasks are decomposed following an upper (resp. a lower) bound, and the Profile is applied to the resulting tasks. The new checker rule, named *Horizontally Elastic Energetic Reasoning Checker* rule subsumes the classic energetic reasoning checker rule and only requires a linear number of intervals of interest.

2012 ACM Subject Classification Computing methodologies → Planning and scheduling; Theory of computation → Constraint and logic programming

Keywords and phrases Energetic Reasoning Checker, Profile Data Structure, Cumulative Scheduling, Constraint Programming, Horizontally Elastic Scheduling

Digital Object Identifier 10.4230/LIPIcs.CP.2025.23

1 Introduction

Constraint Programming (CP) is widely used in artificial intelligence and operations research to solve combinatorial problems like scheduling, relying on propagators to eliminate inconsistent values in search trees. Applications span economics [3], computer science [4], and production [19]. Techniques such as edge-finding [13], timetabling [9], and not-first/not-last [12] support constraint filtering, along with energetic reasoning [15] and overload checks [11].

Overload checking detects when required energy exceeds availability, prompting backtracking. It began with task intervals [6], evolved into a quadratic algorithm [16], and was optimized to quasi-linear time using the Θ -tree [20, 21]. Later, a linear-time version based on the union-find *TimeLine* data structures was introduced in [8], improving on [17]. *Profile* [10] and *TimeTable* [14] further enhanced the verification [11]. Baptiste [2] proposed a quadratic energetic reasoning checker based on relevant intervals. Derrien and Petit [7] reduced their count without increasing complexity. Ouellet and Quimper [18] introduced a checker using Monge Matrix and range trees in $\mathcal{O}(n \log^2(n))$. Carlier et al. [5] later improved this to $\mathcal{O}(n\alpha(n) \log(n))$ (where $\alpha(n)$ is Ackermann's inverse function, and n the number of tasks that share the resource) using critical intervals; To our knowledge, this is the state-of-the-art checker based on the rule of energetic reasoning.

In the following, we apply the *Profile* data structure to the energetic reasoning checker. The tasks are decomposed following an upper bound, allowing the *Profile* data structure to be applied to the decomposed tasks. We get a new checker rule that is more powerful than the classic energetic reasoning checker. The remainder of the paper is organized as follows. Section 2 is devoted to preliminary concepts, which are very useful. In Section 3,



© Rosaly Zoller Bodo Ngono, Yves Pascal Ndjopnang Wantiep and Roger Kameugne; licensed under Creative Commons License CC-BY 4.0

Thirty First Conference on Principles and Practice of Constraint Programming (CP 2025).

Editors: Maria Garcia de la Banda; Article No. 23; pp. 23:1–23:10

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

task decomposition is presented, and Section 4 is devoted to the presentation of the new energetic reasoning checker rule. In contrast, Section 5 focuses on the supremacy of the new rule and Section 6 concludes the paper.

2 Preliminaries

2.1 The Cumulative Scheduling Problems (CuSP)

The Cumulative Scheduling Problem (CuSP) is specified with the given finite set of tasks T such that $|T| = n$ (n denotes the number of tasks) to be executed on a resource of capacity $C \in \mathbb{N}$. We let $k = |\{h_i, i \in T\}|$ be the number of distinct capacity demands for the tasks. Each task $i \in T$ is proceeded without interruption during $p_i \in \mathbb{N}$ time units and requires $h_i \in \mathbb{N}$ units of resource between its earliest starting time est_i and its latest completion time lct_i . The earliest (resp. latest) starting (resp. completion) time, the duration, and the resource demand of a task are specified. The attributes of a task $i \in T$ are denoted $i = \langle \text{est}_i, \text{lct}_i, p_i, h_i \rangle$. A solution of a CuSP is an assignment of starting time s_i to each task $i \in T$ such that the resource constraint is satisfied, i.e.,

$$\text{est}_i \leq s_i \leq s_i + p_i \leq \text{lct}_i \quad (1)$$

$$\sum_{i \in T | s_i \leq t < s_i + p_i} h_i \leq C \quad \text{for all } t \in [0, \max_{k \in T} \text{lct}_k] \quad (2)$$

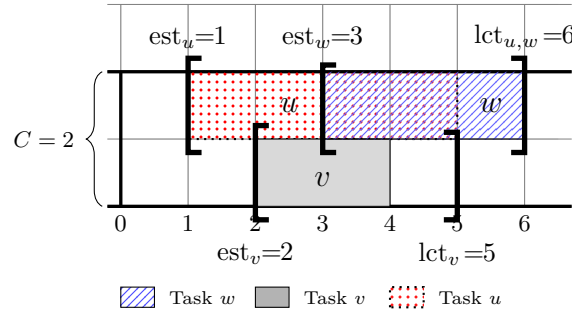
Inequalities in (1) ensure that each task start and end time are feasible, whereas (2) guarantees the non-violation of the resource capacity. From the attributes est_i , lct_i and p_i of the task $i \in T$, we can deduce its earliest completion time $\text{ect}_i = \text{est}_i + p_i$ and its latest starting time $\text{lst}_i = \text{lct}_i - p_i$. The energy of a task $i \in T$ denoted e_i is computed with $e_i = h_i \times p_i$. Let $\Omega \subseteq T$ be a non-empty set of tasks. In that case, we define the energy e_Ω , the earliest starting time est_Ω , and the latest completion time lct_Ω of Ω as follows:

$$e_\Omega = \sum_{k \in \Omega} e_k, \quad \text{est}_\Omega = \min_{k \in \Omega} \text{est}_k, \quad \text{lct}_\Omega = \max_{k \in \Omega} \text{lct}_k \quad (3)$$

By convention, for empty sets, we have: $e_\emptyset = 0$, $\text{est}_\emptyset = \infty$ and $\text{lct}_\emptyset = -\infty$.

Throughout this paper, we assume that for any task $i \in T$, $\text{est}_i \leq \text{lct}_i$ and $h_i \leq C$ otherwise, the problem has no solution.

► **Example 1.** Figure 1 illustrates a CuSP instance with three tasks $\{u, v, w\}$ sharing a resource of capacity 2. Each task is defined by attributes $\langle \text{est}, \text{lct}, p, h \rangle$, with est and lct shown as bracketed points. The processing time, the resource demand, and the energy represent, respectively, the length, the height, and the area of each rectangle. Interval $[2, 5]$ is the segment on which the energetic reasoning rule and the task decomposition will be applied later.



■ **Figure 1** CuSP instance of 3 tasks on a resource of capacity 2.

77 2.2 The Profile Data Structure

78 Introduced early in [10], the *Resource Utilization Profile* (or simply *Profile* data structure)
 79 is an aggregation of stacked rectangles of different lengths and heights that stores the use
 80 of resources by tasks over time. Rectangles are expressed with tuples $\langle time, \Delta_{\max}, \Delta_{req} \rangle$,
 81 where time is the start time, Δ_{\max} is the maximum resource available at starting time, and
 82 Δ_{req} is the maximum resource required by tasks at starting time. The end of a rectangle is
 83 the starting time of the next one. These tuples are stored in a sorted linked list whose nodes
 84 are called *time points*, referring to the starting times of the rectangles. The time points are
 85 sorted in increasing order of time and are kept in a linked list structure called *Profile*.

■ **Algorithm 1** ScheduleTasks(Ω, C) in $\mathcal{O}(n)$ time from [10]

Input: All-time point $t \in P$, $\Omega \subseteq T$ a set of tasks, C the capacity.
Output: The earliest completion time ect_{Ω}^H of Ω

```

1: for all  $t \in P$  do
2:    $t.\Delta_{\max} \leftarrow 0$  and  $t.\Delta_{req} \leftarrow 0$ 
3: for all  $i \in \Omega$  do
4:   Increase  $t.est_i.\Delta_{\max}$  and  $t.est_i.\Delta_{req}$  by  $h_i$ 
5:   Decrease  $t.lct_i.\Delta_{\max}$  and  $t.ect_i.\Delta_{req}$  by  $h_i$ 
6:  $ect^H \leftarrow -\infty$ ;  $overf \leftarrow 0$ ;  $hreq \leftarrow 0$ ;  $S \leftarrow 0$   $t \leftarrow P.first$ 
7: while  $t.time < lct_{\Omega}$  do
8:    $l \leftarrow t.next.time - t.time$ 
9:    $S \leftarrow S + t.\Delta_{\max}$ ,  $hmax \leftarrow \min(S, C)$ 
10:   $hreq \leftarrow hreq + t.\Delta_{req}$ ,  $hcons \leftarrow \min(hreq + overf, hmax)$ 
11:  if  $overf > 0 \wedge overf < (hcons - hreq) \cdot l$  then
12:     $l \leftarrow 1 + \lfloor \frac{overf}{hcons - hreq} \rfloor$ 
13:     $t.InsertAfter(t.time + l)$ 
14:   $overf \leftarrow overf + (hreq - hcons) \cdot l$ 
15:  if  $hcons > 0$  then
16:     $ect^H \leftarrow t.next.time$ 
17:   $t \leftarrow t.next$ 
18: if  $overf > 0$  then
19:   return  $+\infty$ 
20: return  $ect^H$ 

```

86 Δ_{\max} and Δ_{req} contain information relative to the resource consumption for the duration

of the rectangle and are initialized to zero for every time point. The *Profile* is initialized for every distinct value of est , ect , and lct . The *Profile* data structure is used in [10] to efficiently compute the horizontally elastic earliest completion time of a set of tasks Ω denoted ect_{Ω}^H . A set of tasks $\Omega \subseteq T$ is said to be horizontally elastic scheduled when each task $i \in \Omega \subseteq T$ starts at its earliest starting time est_i and cannot consume more than its required capacity at any time during the time interval $[est_i, lct_i)$. At any time $t \in [est_{\Omega}, lct_{\Omega})$, the energy that cannot be executed due to the limited capacity is accumulated as an overflow and released when the resource is no longer saturated. The horizontally elastic earliest completion time of $\Omega \subseteq T$ denoted ect_{Ω}^H occurs when all tasks are completed. It is computed using the functions $hreq(t)$, $hmax(t)$, $hcons(t)$ and $overf(t)$ on the *Profile* P .

- $hmax(t) = \min \left(\sum_{i \in \Omega | est_i \leq t < lct_i} h_i, C \right)$ is the amount of resource that can be allocated to the tasks in Ω at time t ;
 - $hreq(t) = \sum_{i \in \Omega | est_i \leq t < ect_i} h_i$ is the amount of resource required at time t by the tasks in Ω if they were all starting at their earliest starting times;
 - $overf(t)$ is the overflow of energy from $hreq(t)$ that cannot be executed at time t due to the limited capacity $hmax(t)$, and
 - $hcons(t)$ is the amount of resource that is consumed at time t with $hcons(t) = \min(hreq(t) + overf(t-1), hmax(t))$; $overf(t) = overf(t-1) + hreq(t) - hcons(t)$ and $overf(-1) = 0$.
- The function $ScheduleTasks(\Omega, C)$ of Algorithm 1 from [10] computes the horizontally elastic earliest completion time of $\Omega \subseteq T$.

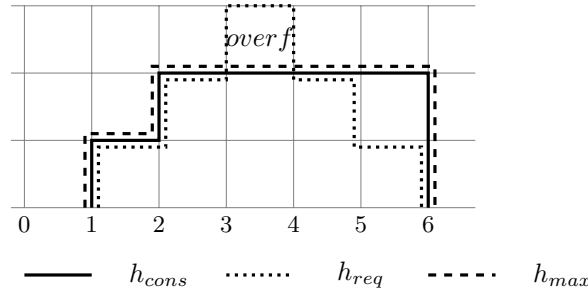
The loop of Line 1 initializes the attributes $t.\Delta_{max}$ and $t.\Delta_{req}$ to zero for any time point t . In the loop of Line 3, the increments are updated following the events est , ect and lct of tasks in Ω . The While loop of Line 7 updates the values of $hmax(t)$, $hreq(t)$, $hcons(t)$ and $overf(t)$ for every time point t starting from the first time point denoted $P.first$. When the remaining overflow is not enough to end at the next time point (Line 11), a new time point is inserted using the function $InsertAfter$ (Line 13). If it remains an overflow at the end of the profile, the earliest completion time is set to ∞ (see Line 18). The property of this data structure is the linearity of the function $ScheduleTasks$ since the *Profile* contains at most $4n$ time points. In [20], the earliest completion time of a set of tasks $\Omega \subseteq T$ is computed with the data structure Θ -tree based on the formula

$$ect_{\Omega}^F = \left\lceil \frac{\max\{Cest_{\Omega'} + e_{\Omega'} | \Omega' \subseteq \Omega\}}{C} \right\rceil. \quad (4)$$

This value called fully elastic earliest completion time of Ω and denoted ect_{Ω}^F is obtained when all tasks of Ω are fully elastic scheduled. A set of tasks $\Omega \subseteq T$ is said to be fully elastic scheduled [1] if each task $i \in \Omega$ starts at est_{Ω} and occupies a total area of $e_i = h_i \times p_i$. When the maximum height is reached at time t , time $t+1$ starts being occupied. It has been shown in [10] that the horizontally elastic relaxation is stronger than the fully elastic one. i.e, for all set of tasks $\Omega \subseteq T$

$$ect_{\Omega}^F \leq ect_{\Omega}^H \leq ect_{\Omega}. \quad (5)$$

► **Example 2.** The profile of the CuSP instance of Figure 1 is depicted in Figure 2. The functions h_{max} , h_{cons} , h_{req} and $overf$ are illustrated. One unit of overflow is stored at a time point 3. It is scheduled at a time 5. The horizontally elastic earliest completion time of the set of tasks $\{u, v, w\}$ is 6. On the other hand, the fully elastic earliest completion time of $\{u, v, w\}$ obtained with formula (4) is 6.



■ **Figure 2** The profile of the tasks $\{u, v, w\}$.

2.3 Energetic Reasoning Checker

For any task $i \in T$, energetic reasoning (ER) compares the energy available in a time interval $[a, b)$ with the minimum energy required by tasks during that interval. For each task, ER evaluates its minimum contribution by considering three configurations—left-shifted (starting earliest at est_i), right-shifted (ending later at lct_i), or centered (occupies the entire interval $[a, b)$)—and selects the one with the least overlap, thus minimizing energy consumption in the interval. Of these three configurations, the one leading to minimum resource consumption is the one whose intersection with the interval $[a, b)$ is minimal.

The Left-Shifted energy of task i in interval $[a, b)$ is the integer denoted $\text{LS}(i, a, b)$ and defined by

$$\text{LS}(i, a, b) = h_i \times \max(0, \min(\text{ect}_i - a, p_i, b - a)) \quad (6)$$

The Right-Shifted energy of task i in interval $[a, b)$ is the integer denoted $\text{RS}(i, a, b)$ and defined by

$$\text{RS}(i, a, b) = h_i \times \max(0, \min(b - \text{lst}_i, p_i, b - a)) \quad (7)$$

The minimum intersection energy of task i in the interval $[a, b)$ is the integer denoted $\text{MI}(i, a, b)$ and defined by

$$\text{MI}(i, a, b) = \min(\text{LS}(i, a, b), \text{RS}(i, a, b)) \quad (8)$$

Let Ω be a set of tasks. The minimum intersection energy of Ω in the interval $[a, b)$ is denoted $\text{MI}(\Omega, a, b)$ and defined by

$$\text{MI}(\Omega, a, b) = \sum_{i \in \Omega} \text{MI}(i, a, b) \quad (9)$$

For a cumulative resource of capacity C , the margin (or slack) function of Ω over $[a, b)$ noted $\text{SL}(\Omega, a, b)$ is the integer defined by

$$\text{SL}(\Omega, a, b) = C \times (b - a) - \text{MI}(\Omega, a, b) \quad (10)$$

The energetic reasoning checker tests for every interval $[a, b)$, if the slack of an instance T of the cumulative constraint over $[a, b)$ is non-negative, i.e., $\text{SL}(T, a, b) \geq 0$. When the test fails, an inconsistency is detected as specified in the following rule

$$\exists a, b \in \mathbb{N} \text{ with } a < b, \quad \text{SL}(T, a, b) < 0 \Rightarrow \text{fail} \quad (\text{ERC})$$

► **Example 3.** Back to the CuSP instance of Figure 1 where three tasks share a resource of capacity 2, we consider the interval $[2, 5)$. We have $MI(\{u, v, w\}, 2, 5) = MI(u, 2, 5) + MI(v, 2, 5) + MI(w, 2, 5) = 3 + 2 + 2 = 7$ and $SL(\{u, v, w\}, 2, 5) = 2 \times (5 - 2) - MI(\{u, v, w\}, 2, 5) = 6 - 7 = -1 < 0$. Therefore, a failure is detected, and the CuSP is infeasible (this means that the amount of resources available in the interval $[2, 5)$ is insufficient to schedule a minimum consumption for all activities).

3 Tasks Decomposition

Let b be an integer and $i \in T$ be a task. There is a whole number $a(i, b) \in [est_i, lst_i]$ whose left-shift energy of i is equal to its right-shift energy in the interval $[a(i, b), b]$ i.e., $LS(i, a(i, b), b) = RS(i, a(i, b), b)$.

► **Lemma 4.** Given an integer b and a task $i \in T$, the integer $a(i, b)$ is defined as

$$a(i, b) = \begin{cases} est_i & \text{if } lct_i \leq b \\ est_i + lct_i - b & \text{if } lct_i > b \wedge lst_i < b \wedge ect_i < b \\ lst_i & \text{if } lct_i > b \wedge lst_i < b \wedge ect_i \geq b \end{cases} \quad (11)$$

Proof. The relation is obtained when replacing $a(i, b)$ in $LS(i, a(i, b), b)$. ◀

Let b be an integer and i be a task. The minimum intersection span time of task i in the interval (∞, b) denoted $span(i, b)$ is defined by

$$span(i, b) = \begin{cases} p_i & \text{if } lct_i \leq b \\ b - lst_i & \text{if } lct_i > b \wedge lst_i < b \\ 0 & \text{otherwise} \end{cases} \quad (12)$$

We consider the task $dec(i, b)$ derives from task i and integer b with the following attributes:

$$est_{dec(i, b)} = a(i, b), \quad lct_{dec(i, b)} = \begin{cases} lct_i & \text{if } lct_i \leq b \\ b & \text{if } lct_i > b \wedge lst_i < b \\ -1 & \text{otherwise} \end{cases}, \quad p_{dec(i, b)} = span(i, b), \text{ and}$$

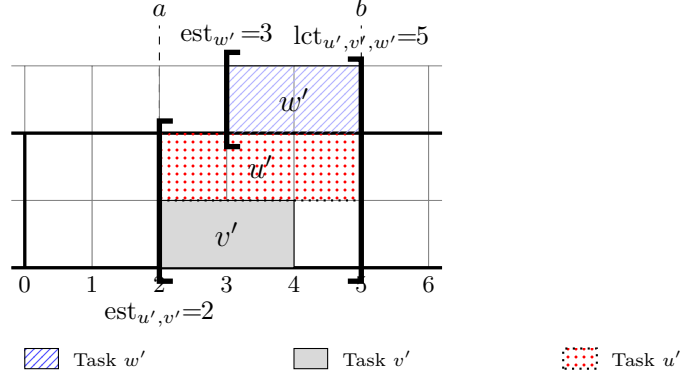
$h_{dec(i, b)} = h_i$. The task $dec(i, b)$ is the decomposition of task i following the upper bound b . For all integer a with $a < b$ such that $MI(i, a, b) > 0$. When task $dec(i, b)$ is scheduled horizontally and elastically, its span time in $[a, b)$ corresponds to the minimum intersection span time of task i in $[a, b)$. This result is formally proved in Lemma 5.

► **Lemma 5.** Let b be an integer and $i \in T$ be a task. For all integer a with $a < b$ with $MI(i, a, b) > 0$, the task $dec(i, b)$ decomposition of task i following the upper bound b span time in $[a, b)$, when it is horizontally elastically scheduled corresponds to the minimum intersection span time of task i in $[a, b)$.

Proof. (See the appendix section A) ◀

We denote by $Dec(T, b)$ the set of decomposed tasks following the upper bound b i.e., $Dec(T, b) = \{dec(i, b) \mid i \in T\}$. Symmetrically, given an integer a and a task i . It is possible to decompose task i following the lower bound a . The task $dec(i, a)$ deduced from task i and the integer a denotes the decomposition of task i following the lower bound a and $Dec(T, a)$ denotes the set of decomposed tasks following the lower bound a i.e., $Dec(T, a) = \{dec(i, a) \mid i \in T\}$.

► **Example 6.** For the upper bound $b = 5$, we decompose the tasks of the CuSP instance of Figure 1. We denote by $\{u', v', w'\}$ the decomposed tasks where $u' = \text{dec}(u, 5)$, $v' = \text{dec}(v, 5)$, and $w' = \text{dec}(w, 5)$ respectively. After application of the decomposition rule, we have $u' = \langle 2, 5, 3, 1 \rangle$, $v' = \langle 2, 5, 2, 1 \rangle$ and $w' = \langle 3, 5, 2, 1 \rangle$. The decomposed tasks are depicted in Figure 3.



■ **Figure 3** The CuSP instance of the decomposed tasks of Figure 1 following the upper bound 6.

4 Horizontally Elastic Energetic Reasoning Checker Rule

The new checker compares the horizontally elastic earliest completion (resp. latest starting) time of the set of decomposed tasks following an upper (resp. lower) bound to the bound. It is formally stated as follows:

$$\exists b \in \{\text{ect}_i, \text{lct}_i \mid i \in T\}, \text{ect}_{\text{Dec}(T,b)}^H > b \Rightarrow \text{fail} \quad (\text{HE-ERCb})$$

$$\exists a \in \{\text{est}_i, \text{lst}_i \mid i \in T\}, \text{lst}_{\text{Dec}(T,a)}^H < a \Rightarrow \text{fail} \quad (\text{HE-ERCa})$$

In the rest of the paper, we only deal with the rule (HE-ERCb) since the rule (HE-ERCa) can be made symmetrically. Indeed, to perform the rule (HE-ERCa), a symmetric problem is obtained by replacing any task i into a task i_1 such that: $\text{est}_{i_1} = C_{\max} - \text{lct}_i$, $\text{lct}_{i_1} = C_{\max} - \text{est}_i$, $p_{i_1} = p_i$, and $h_{i_1} = h_i$ where $C_{\max} = \max_{i \in T} \text{lct}_i$. The decomposition and the earliest completion time of the symmetric problem correspond to the decomposition and the latest starting time of the original problem. The correctness of the rule uses the following lemma.

► **Lemma 7.** Let b be an integer and a^* be the smallest integer such that the rule (ERC) holds with the interval $[a^*, b)$. For all $i \in T$ such that $MI(i, a^*, b) > 0$, the energy of the horizontally elastic scheduling of $\text{dec}(i, b)$ in $[a^*, b)$ is equal to $MI(i, a^*, b)$.

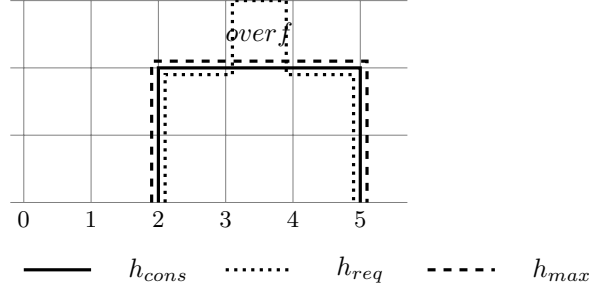
Proof. (See the appendix section A) ◀

► **Theorem 8.** If the rule (ERC) holds with the interval $[a, b]$, then the rule (HE-ERCb) or (HE-ERCa) holds with b or a respectively.

Proof. (See the appendix section A) ◀

The number of useful intervals is the main drawback of the energetic reasoning approach. The new formulation reduces this number from a factor of $\mathcal{O}(n)$. This is the first formulation with a linear number of relevant intervals.

218 ► **Example 9.** The profile of the decomposed tasks of Figure 3 is depicted in Figure 4. An
 219 overflow of one unit of energy is recorded at time 3, and from the condition of Line 18 of
 220 Algorithm 1, the horizontally elastic earliest completion time of $\{u', v', w'\}$ is ∞ . Therefore,
 221 the failure is detected.



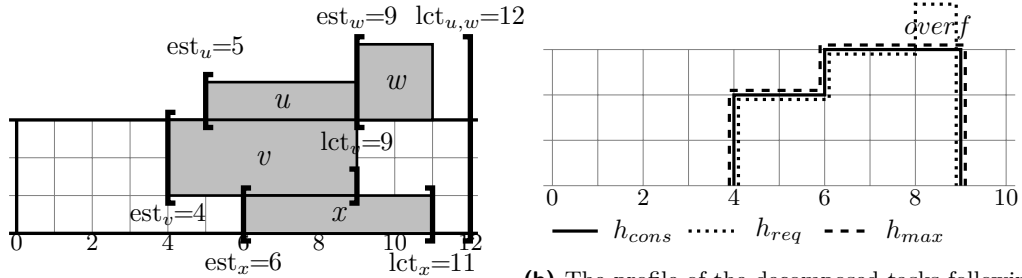
■ **Figure 4** The profile of the decomposed tasks of Figure 3.

222 The profile data structure also increases the filtering power of the new rule, which can
 223 detect more superset inconsistencies than the classic energetic reasoning checker.

224 **5 Dominance property**

225 ► **Theorem 10.** *The combination of rules (HE-ERCa) and (HE-ERCb) detects more supersets*
 226 *of inconsistencies than the rule (ERC).*

227 **Proof.** Consider the CuSP instance of Figure 5a where four tasks share a resource of capacity
 228 3. The tasks x and v are fixed while tasks u and w remain free. There is no way to insert tasks
 229 u and w before time 12 without preemption. Therefore, the problem is inconsistent. The
 230 classic energetic reasoning checker does not detect this inconsistency. After the decomposition
 231 of tasks following the upper bound $b = 9$, the decomposed tasks are $dec(x, 9) = \langle 6, 9, 3, 1 \rangle$,
 232 $dec(u, 9) = \langle 8, 9, 1, 1 \rangle$, and $dec(v, 9) = \langle 4, 9, 5, 2 \rangle$.



(a) Four tasks on a resource of capacity 3

(b) The profile of the decomposed tasks following the upper bound 9

■ **Figure 5** (5a) CuSP instance of 4 tasks on a resource of capacity 3 and (5b) the profile of the decomposed tasks following the upper bound 9.

233 The horizontally elastic earliest completion time of $Dec(T, 9)$ is ∞ , and rule (HE-ERCb)
 234 detects the inconsistency. On the Profile of decomposed tasks, only one unit of overflow is
 235 recorded at time 8. This overflow is not released at the end. ◀

236 The new rule subsumes the classic energetic reasoning checker and only requires a linear
 237 number of relevant intervals.

6 Conclusion

In this paper, we have applied the Profile data structure to the energetic reasoning checker. To do so, tasks are decomposed following an upper (resp. a lower) bound, and the profile is applied to decomposed tasks. The new rule subsumes the classic energetic reasoning checker and only considers a linear number of useful intervals. For future work, we plan to propose an algorithm with low time complexity for this new rule.

References

- 1 P. Baptiste, C.L. Pape, and W. Nuijten. *Constraint-Based Scheduling: Applying Constraint Programming to Scheduling Problems*. International Series in Operations Research & Management Science. Springer US, 2012. URL: <https://books.google.cm/books?id=qUzhBwAAQBAJ>.
- 2 Philippe Baptiste, Claude Le Pape, and Wim Nuijten. Satisfiability tests and time-bound adjustments for cumulative scheduling problems. *Ann. Oper. Res.*, 92:305–333, 1999.
- 3 Rajkumar Buyya, Manzur Murshed, David Abramson, and Srikumar Venugopal. Scheduling parameter sweep applications on global grids: a deadline and budget constrained cost–time optimization algorithm. *Software: Practice and Experience*, 35(5):491–512, 2005.
- 4 Jacques Carlier and Christian Prins. Optimisation des plans de trame dans le système amrt/cnc d’eutelsat. *Annales Des Télécommunications*, 43:506–521, 1988. URL: <https://api.semanticscholar.org/CorpusID:107386270>.
- 5 Jacques Carlier, Abderrahim Sahli, Antoine Jouglet, and Eric Pinson. A faster checker of the energetic reasoning for the cumulative scheduling problem. *Int. J. Prod. Res.*, 60(11):3419–3434, 2022.
- 6 Yves Caseau and François Laburthe. Improved CLP scheduling with task intervals. In *ICLP*, pages 369–383. MIT Press, 1994.
- 7 Alban Derrien and Thierry Petit. A new characterization of relevant intervals for energetic reasoning. In *CP*, volume 8656 of *Lecture Notes in Computer Science*, pages 289–297. Springer, 2014.
- 8 Hamed Fahimi, Yanick Ouellet, and Claude-Guy Quimper. Linear-time filtering algorithms for the disjunctive constraint and a quadratic filtering algorithm for the cumulative not-first not-last. *Constraints An Int. J.*, 23(3):272–293, 2018.
- 9 Steven Gay, Renaud Hartert, and Pierre Schaus. Simple and scalable time-table filtering for the cumulative constraint. In *CP’15: Proceedings of the 15th international conference on Principles and practice of constraint programming*, volume 9255 of *Lecture Notes in Computer Science*, pages 149–157. Springer, 2015.
- 10 Vincent Gingras and Claude-Guy Quimper. Generalizing the edge-finder rule for the cumulative constraint. In *IJCAI*, pages 3103–3109. IJCAI/AAAI Press, 2016.
- 11 Roger Kameugne, Séverine Betmbe Fetgo, Thierry Noulamo, and Clémentin Tayou Djamégni. Improved timetable edge finder rule for cumulative constraint with profile. *Comput. Oper. Res.*, 172:106795, 2024.
- 12 Roger Kameugne and Laure Pauline Fotso. A cumulative not-first/not-last filtering algorithm in $O(n^2 \log(n))$. *Indian Journal of Pure and Applied Mathematics*, 44:95–115, 2013.
- 13 Roger Kameugne, Laure Pauline Fotso, Joseph D. Scott, and Youcheu Ngo-Kateu. A quadratic edge-finding filtering algorithm for cumulative resource constraints. *Constraints An Int. J.*, 19(3):243–269, 2014.
- 14 Abdelkader Lahrichi. Ordonnancements: La notion de "parties obligatoires" et son application aux problèmes cumulatifs. *RAIRO - Operations Research - Recherche Opérationnelle*, 16(3):241–262, 1982.
- 15 Pierre Lopez. *Approche énergétique pour l’ordonnancement de tâches sous contraintes de temps et de ressources. (Energy-based approach for task scheduling under time and resource constraints)*. PhD thesis, Paul Sabatier University, Toulouse, France, 1991.

- 287 **16** Luc Mercier and Pascal Van Hentenryck. Edge finding for cumulative scheduling. *INFORMS J. Comput.*, 20(1):143–153, 2008.
- 288 **17** W.P.M. Nuijten. *Time and resource constrained scheduling : a constraint satisfaction approach*. Phd thesis 1 (research tu/e / graduation tu/e), Mathematics and Computer Science, 1994. doi:10.6100/IR431902.
- 289 **18** Yanick Ouellet and Claude-Guy Quimper. A $O(n \log^2 n)$ checker and $O(n^2 \log n)$ filtering algorithm for the energetic reasoning. In *CPAIOR*, volume 10848 of *Lecture Notes in Computer Science*, pages 477–494. Springer, 2018.
- 290 **19** Charles Thomas and Pierre Schaus. A constraint programming approach for aircraft disassembly scheduling. In *CPAIOR (2)*, volume 14743 of *Lecture Notes in Computer Science*, pages 211–220. Springer, 2024.
- 291 **20** Petr Vilím. Max energy filtering algorithm for discrete cumulative resources. In *CPAIOR*, volume 5547 of *Lecture Notes in Computer Science*, pages 294–308. Springer, 2009a.
- 292 **21** Armin Wolf and Gunnar Schrader. $O(n \log n)$ overload checking for the cumulative constraint and its application. In *INAP*, volume 4369 of *Lecture Notes in Computer Science*, pages 88–101. Springer, 2005.

303 **A** Proofs of lemmas and theorems

304 **Proof. of lemma 5**

305 The task $dec(i, b)$ is horizontally elastic scheduled if it starts at its earliest starting time
306 $a(i, b)$, consumes between 0 and h_i units of energy at any time. When the task is alone, it
307 consumes exactly h_i units of energy at any time during its execution. Let a be an integer
308 with $a < b$ and $MI(i, a, b) > 0$. The rest of the proof will distinguish the case $a \leq a(i, b)$ from
309 the case $a > a(i, b)$.

310 ■ if $a \leq a(i, b)$ then the span of task $dec(i, b)$ in $[a, b]$ corresponds to $span(i, b)$ when it
311 is horizontally elastic scheduled. From the definition of $a(i, b)$, we have $LS(i, a, b) \geq$
312 $RS(i, a, b) = span(i, b) \times h_i$ and the result follows.

313 ■ if $a > a(i, b)$ then span of task $dec(i, b)$ is $ect_{dec(i, b)} - a$ where $ect_{dec(i, b)} = \begin{cases} ect_i & \text{if } ect_i < b \\ b & \text{if } ect_i \geq b \end{cases}$.

314 Therefore, $(ect_{dec(i, b)} - a) \times h_i = LS(i, a, b) \leq RS(i, a, b)$ and the result follows.

315 ◀

316 **Proof. of lemma 7** If the energy of the horizontally elastic scheduling of $dec(i, b)$ in $[a^*, b)$ is
317 different from $MI(i, a^*, b)$, then there exists an integer $a < a^*$ such that less than h_i units of
318 task i is used at time a . The last peak starts at a and the remaining energy not consumed at
319 time a is postponed until the future time point when the resource will be available. Therefore,
320 rule (ERC) holds with the interval $[a, b]$ which contradicts the minimality of a^* . ◀

321 **Proof. of Theorem 8**

322 Assume that the rule (ERC) holds with the interval $[a, b]$. According to the charac-
323 terization of relevant intervals for energetic reasoning of [7], if $a = est_i + lct_i - b$ (resp.
324 $b = est_i + lct_i - a$) with $i \in T$, then $b \in \{ect_k, lct_k \mid k \in T\}$ (resp. $a \in \{est_k, lst_k \mid k \in T\}$).
325 Therefore, if $b \in \{ect_k, lct_k \mid k \in T\}$, then the rule (HE-ERCb) is considered; otherwise, the
326 rule (HE-ERCa) is used. We assume that $b \in \{ect_k, lct_k \mid k \in T\}$ and no detection is made
327 by the rule (HE-ERCa). The last peak of the profile of $Dec(T, b)$ contains the large interval
328 $[a^*, b)$ on which the rule (ERC) holds. According to Lemma 7, the horizontally elastic energy
329 of the interval $[a^*, b]$ is its minimal intersection energy. Since horizontally elastic scheduling
330 is more constrained than fully elastic scheduling (see formula (5)), we can deduce that rule
331 (HE-ERCb) holds. ◀