Horizontally Elastic Edge Finder Rule for Cumulative Constraint Based on Slack and Density

Roger Kameugne
Faculty of Sciences, Department of Mathematics and Computer Science, University of Maroua, Cameroon

Sévérine Fetgo Betmbe
Faculty of Sciences, Department of Mathematics and Computer Science, University of Dschang, Cameroon

Thierry Noulamo
UIT Fotso Victor of Bandjoun, Department of Computer Engineering, University of Dschang, Cameroon

Clémentin Tayou Djamegni
Faculty of Sciences, Department of Mathematics and Computer Science, University of Dschang, Cameroon
UIT Fotso Victor of Bandjoun, Department of Computer Engineering, University of Dschang, Cameroon

Abstract

In this paper, we propose an enhancement of the filtering power of the edge finding rule, based on the *Profile* and the *TimeTable* data structures. The minimal slack and the maximum density criteria are used to select potential task intervals for the edge finding rule. The strong detection rule of the horizontally elastic edge finder of Fetgo and Tayou is then applied on those intervals, which results in a new filtering rule, named *Slack-Density Horizontally Elastic Edge Finder*. The new rule subsumes the edge finding rule and it is not comparable to the Gingras and Quimper horizontally elastic edge finder rule and the *TimeTable* edge finder rule. A two-phase filtering algorithm of complexity $\mathcal{O}(n^2)$ (where n is the number of tasks sharing the resource) is proposed for the new rule. Improvements based on the *TimeTable* are obtained by considering fix part of external tasks which overlap with the potential task intervals. The detection and the adjustment of the improve algorithm are further increased, while the algorithm remains quadratic. Experimental results, on a well-known suite of benchmark instances of Resource-Constrained Project Scheduling Problems, show that the propounded algorithms are competitive with the state-of-the-art algorithms, in terms of running time and tree search reduction.

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

Keywords and phrases Horizontally Elastic Scheduling, Edge Finder Rule, Profile, TimeTable, Resource-Constrained Project Scheduling Problem

Digital Object Identifier 10.4230/LIPIcs.CP.2023.20

Acknowledgements The authors thank Dr. Dany Nantchouang and Prof. Emmanuel Hebrard for their proofreading.

1 Introduction

Scheduling is the allocation of scarce resources to tasks or activities over time. The economic impact of the scheduling in industries and organizations [6] makes it an important combinatorial optimization problem. Industrial resources are workers, machines, electricity power and raw materials, etc. In computer science, the resources are processors while tasks are processes to be proceeded. The success of constraint programming on scheduling problems is due to the existence of specific global constraints like UNARY [28] and CUMULATIVE [1] combined with special search strategies [2, 3, 28]. The global constraint is repeatedly called at each node of the tree search, to remove values from the variables domains when there are inconsistent by resource constraint. It is NP-hard [12] to remove all such inconsistent values from the variables domain.

The filtering algorithms used to prune the domain of variables are generally based on the relaxation of the problem. It can be called many times at each node of the tree search. Therefore, each filtering algorithm has to be fast, sound, and powerful. The global constraint CUMULATIVE [1] generally embeds timetabling [14, 21, 4] and edge finding [29, 19, 22] as basis. Some extensions and enhancement of the edge-finding filtering power have been proposed for more pruning of the domains of variables [18, 23, 30, 15, 11]. Energetic reasoning [24, 2, 7] and not-first/not-last rules [9, 16, 17, 27] are two other rules generally embedded in the CUMULATIVE constraint when the problem is highly cumulative [2]. The energetic reasoning rule subsumes all other rules except the not-first / not-last rule [2, 30, 11].

In this paper, we propose a new enhancement of the edge finder rule with the *Profile* [15] and to the *TimeTable* [30] data structures. The new rule called *Slack-Density Horizontally Elastic Edge Finder* uses the minimum slack and the maximum density criteria of [19] to select the potential edge finding task intervals on which the detection rule of [11] is going to be applied. This new rule subsumes the edge finding rule, is not comparable to the Gingras and Quimper rule [15], and the TimeTable edge finding rule [30]. A quadratic algorithm of two-phase (Detection and Adjustment) is proposed for the new rule. To further enhance the algorithm, we improve the new algorithm by considering the fix part of external tasks, which overlap with the intervals during the horizontally elastic scheduling of the task intervals. The final algorithm remains with a quadratic complexity. Experimental results on a well-known suite of benchmark instances of Resource-Constrained Project Scheduling Problems (RCPSPs) from the BL set [2] and the PSPLib set [20] show that, the propounded algorithms are competitive to the state-of-the-art algorithms, in terms of running time and tree search reduction.

The remaining part of the paper is organized as follows: Section 2 is devoted to preliminary notions and related works on previous enhancement of the edge finder rule with data structures; Section 3 presents the new rule based on the global minimum slack and maximum density criteria; Section 4 proposes an $\mathcal{O}(n^2)$ algorithm (where n is the number of tasks sharing the resource) for the new rule; Section 5 uses the *TimeTable* data structure to improve the detection and the adjustment of the previous algorithm; Section 6 consists of an empirical evaluation of the propounded algorithms with the state-of-the-art algorithms on RCPSP instances; and Section 7 concludes the paper.

2 Preliminaries

This section specifies the Cumulative Scheduling Problem (CuSP) and reviews the edge finder rule and its enhancements based on data structures.

2.1 A Cumulative Scheduling Problem (CuSP)

In a Cumulative Scheduling Problem (CuSP), a set of tasks T has to be executed on a resource of capacity C. Each task $i \in T$ is executed without interruption during p_i time units and uses $c_i \leq C$ units of the resource. For a task $i \in T$, the earliest starting time est_i and the latest completion time lct_i are specified. A solution of a CuSP instance is an assignment of a valid starting time s_i to each task $i \in T$ such that the resource constraint is satisfied, i.e.,

$$\forall i \in T, \qquad \mathrm{est}_i \leq s_i \leq s_i + p_i \leq \mathrm{lct}_i \tag{1}$$

$$\forall \tau, \qquad \sum_{i \in T,\ s_i \leq \tau < s_i + p_i} c_i \leq C \tag{2}$$

Inequalities in (1) ensure that each task is assigned a feasible starting and ending time, while (2) enforces the resource constraint. We use the notation e_i to denote the energy of a task $i \in T$ and it is computed with $e_i = c_i \cdot p_i$. Each task $i \in T$ has an earliest completion time $\mathrm{ect}_i = \mathrm{est}_i + p_i$ and a latest starting time $\mathrm{lst}_i = \mathrm{lct}_i - p_i$. These notations can be extended to nonempty sets of tasks as follows:

$$e_\Omega = \sum_{j \in \Omega} e_j, \quad \mathrm{est}_\Omega = \min_{j \in \Omega} \mathrm{est}_j, \quad \mathrm{lct}_\Omega = \max_{j \in \Omega} \mathrm{lct}_j. \tag{3}$$

By convention, for empty sets, we have: $e_\emptyset = 0$, $\mathrm{est}_\emptyset = +\infty$ and $\mathrm{lct}_\emptyset = -\infty$. Throughout the paper, we assume that for any task $i \in T$, $\mathrm{ect}_i \leq \mathrm{lct}_i$ and $c_i \leq C$, otherwise the problem has no solution. We let $n = |T|$ denote the number of tasks and $k = |\{c_i, i \in T\}|$ denote the number of distinct capacity demands of the tasks. The global constraint CUMULATIVE [1] is used to solve the CuSP problem, which is a NP-Hard problem [12]. The constraint removes inconsistent values from the domain of starting time variable $s_i \in [\mathrm{est}_i, \mathrm{lst}_i]$ and it is NP-Hard to remove all such values.

2.2 Task Interval, Minimum Slack and Maximum Density

Given two tasks l and u with $\mathrm{est}_l < \mathrm{lct}_u$. A task interval denoted Ω_u^l is a set of tasks that must run entirely within the interval $[\mathrm{est}_l, \mathrm{lct}_u]$. It is formally specified in the following definition.

▶ **Definition 1.** *[8] Let u and l be two tasks that satisfy $est_l < lct_u$. The task interval Ω_u^l is a set of tasks specified by $\Omega_u^l = \{j \in T \mid est_l \leq est_j \wedge lct_j \leq lct_u\}$.*

When the condition $\mathrm{est}_l \leq \mathrm{est}_j$ is released in the definition of the task interval, it is called in [29] the left cut of T by task u denoted $\mathrm{LCut}(T, u)$, i.e., $\mathrm{LCut}(T, u) = \{j \in T \mid \mathrm{lct}_j \leq \mathrm{lct}_u\}$. The slack and the density of a task interval are concepts that are useful when designing an edge-finding algorithm [19].

▶ **Definition 2.** *Let l and u be two tasks that satisfy $est_l < lct_u$.*

1. *The slack of the task interval Ω_u^l is the integer denoted $sl(\Omega_u^l)$ and defined by*

$$sl(\Omega_u^l) = C \cdot (lct_u - est_l) - e_{\Omega_u^l}. \tag{4}$$

2. *The density of the task interval Ω_u^l is the real number denoted $ds(\Omega_u^l)$ and defined by*

$$ds(\Omega_u^l) = \frac{e_{\Omega_u^l}}{lct_u - est_l}. \tag{5}$$

In [19], the minimum slack and the maximum density criteria are successfully used to design a fast edge-finding algorithm.

▶ **Definition 3.** *[19] Let $u \in T$ be a task.*

1. *The task interval of minimum slack $\Omega_u^{\tau(u)}$ (where $\tau(u)$ is a task depending on task u) is the task interval satisfying the condition:*

$$sl(\Omega_u^{\tau(u)}) \leq sl(\Omega_u^l), \quad \forall l \in T \text{ with } est_l < lct_u. \tag{6}$$

2. *The task interval of maximum density* $\Omega_u^{\rho(u)}$ *(where* $\rho(u)$ *is a task depending on task* u*) is the task interval satisfying the condition:*

$$ds(\Omega_u^{\rho(u)}) \geq ds(\Omega_u^l), \quad \forall l \in T \text{ with } est_l < lct_u. \tag{7}$$

These two concepts are used in [19, 18] to select the task intervals that are of interest for the (extended) edge-finder rule.

2.3 (Extended) Edge Finding Rule

Let $\Omega \subset T$ be a set of tasks, and $i \notin \Omega$ be a task. If the scheduling of the set of tasks Ω and the contribution of task i in the interval $[\text{est}_\Omega, \text{lct}_\Omega)$ when task i starts at est_i cause an overload of the interval $[\text{est}_\Omega, \text{lct}_\Omega)$, then it is deduced that all tasks in Ω end before the end of i and is denoted $\Omega \lessdot i$. The detection rules are specified by the formulas: $\forall \Omega \subset T$, $\forall i \in T \setminus \Omega$

$$e_\Omega + e_i > C \cdot (\text{lct}_\Omega - \text{est}_{\Omega \cup \{i\}}) \Rightarrow \Omega \lessdot i \tag{EF}$$

$$\begin{cases} \text{est}_i < \text{est}_\Omega \\ \wedge \\ e_\Omega + c_i \cdot (\text{ect}_i - \text{est}_\Omega) > C \cdot (\text{lct}_\Omega - \text{est}_\Omega) \end{cases} \Rightarrow \Omega \lessdot i \tag{EEF}$$

The rule (EF) is called edge-finder detection rule, while (EEF) is known as its extension. After each detection, the adjustment follows, using this rule:

$$\Omega \lessdot i \Rightarrow \text{est}_i \geq \max_{\Theta \subseteq \Omega \wedge \text{rest}(\Theta, c_i) > 0} \text{est}_\Theta + \left\lceil \frac{\text{rest}(\Theta, c_i)}{c_i} \right\rceil \tag{Adj}$$

where $\text{rest}(\Theta, c_i) = e_\Theta - (C - c_i) \cdot (\text{lct}_\Theta - \text{est}_\Theta)$ is the energy of e_Θ that disables the starting time of task i when scheduled on a resource of capacity $C - c_i$ in the interval $[\text{est}_\Theta, \text{lct}_\Theta)$. A two-phase algorithm of complexity $\mathcal{O}(kn \log(n))$ (where n is the number of tasks and k the different number of resource demands of tasks) based on Θ-Λ-tree data structure was proposed in [29]. A quadratic algorithm based on the minimum slack and the maximum density of task intervals is proposed in [19].

2.4 *TimeTable* Edge Finding Rule

Research revealed that, the first enhancement of the filtering power of the edge finder rule was made with the *TimeTable* data structure in [30]. The fix part of the task is taken into account in the computation of the energy of the set of tasks, resulting in a strengthened rule. If $i \in T$ is a task satisfying $\text{lst}_i < \text{ect}_i$, then the interval $[\text{lst}_i, \text{ect}_i)$ determines the mandatory spanning interval of i. We denote by $f(\Omega, t)$ the aggregate of the fix parts over time t by tasks in Ω, and $f(\Omega, [a, b))$ the fix parts aggregation over the time interval $[a, b)$ by the tasks in Ω.

$$f(\Omega, t) = \sum_{i \in \Omega | \text{lst}_i \leq t < \text{ect}_i} c_i; \tag{8}$$

$$f(\Omega, [a, b)) = \sum_{t \in [a, b)} f(\Omega, t). \tag{9}$$

Let e_Ω^{TT} be the *TimeTable* energy of tasks in Ω. This energy is equal to the energy of Ω plus the fix energy of the tasks in $T \setminus \Omega$ spent within the interval $[\text{est}_\Omega, \text{lct}_\Omega)$, i.e., $e_\Omega^{TT} = e_\Omega + f(T \setminus \Omega, [\text{est}_\Omega, \text{lct}_\Omega))$. The *TimeTable* edge finder rule, denoted (TT-EF), is obtained from (EF), (EEF) and (Adj) by substituting e_Ω and e_Θ by e_Ω^{TT} and e_Θ^{TT} respectively.

A quadratic algorithm was proposed in [30] and later refined in [23]. This algorithm was used in [26] with the lazy clause generation approach, where the explanation of its propagation is incorporated as nogoods.

2.5 Horizontally Elastic Earliest Completion Time of Tasks Set

The computation of the earliest completion time of a set of tasks for scheduling problems with resource constraints is known to be NP-hard [12]. Depending on the way tasks are scheduled, a lower bound of the earliest completion time can be computed. According to [2], the set of tasks Ω is said to be fully elastic scheduled, if each task $i \in \Omega$ starts at est_Ω and occupies a total area of $e_i = c_i \cdot p_i$. At any time $t \in [\text{est}_\Omega, \text{lct}_\Omega)$, a task i can occupy more or less c_i units of height and when time t reaches the height C, time $t+1$ starts being occupied. The fully elastic earliest completion time of Ω (denoted ect^F_Ω) occurs when all tasks are completed. It is computed in [29] with the formula.

$$\text{ect}^F_\Omega = \left\lceil \frac{\max\{C\text{est}_{\Omega'} + e_{\Omega'} | \Omega' \subseteq \Omega\}}{C} \right\rceil \tag{10}$$

In [15], a new way of scheduling a set of tasks called horizontally elastic scheduling is introduced. A set of tasks Ω is said to be horizontally elastic scheduled, when each task $i \in \Omega$ starts at its earliest starting time est_i and cannot consume more than its required capacity at any time during the time interval $[\text{est}_i, \text{lct}_i)$. At any time $t \in [\text{est}_\Omega, \text{lct}_\Omega)$, 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 Ω denoted ect^H_Ω occurs when all tasks are completed. The computation of the horizontally elastic earliest completion time of tasks set is done with a data structure called *Profile* that stores the resource utilization over time. The tuples $\langle time, cap, \Delta_{\max}, \Delta_{req} \rangle$ (where time is the starting time, cap is the remaining capacity of the resource at the starting time, $\Delta_{\max}$ is the maximum resource available at starting time, and Δ_{req} is the maximum resource required by tasks at starting time) are stored in a sorted linked-list whose nodes are called time points. The *Profile* is initialized with a time point of capacity C for every distinct value of est, ect and lct. A sufficiently large time point is added to act as a sentinel. Finally, while initializing the data structure, the pointers $t.\text{est}_i$, $t.\text{ect}_i$ and $t.\text{lct}_i$ are used to return the time point associated with est_i, ect_i, and lct_i for each task $i \in T$. The horizontally elastic earliest completion time of a set of tasks Ω denoted ect^H_Ω is computed using the functions $c_{req}(t)$, $c_{max}(t)$, $c_{cons}(t)$ and $ov(t)$ on the *Profile* P.

- $c_{max}(t) = \min\left(\sum_{i \in \Omega | \text{est}_i \le t < \text{lct}_i} c_i, C\right)$ is the amount of resource that can be allocated to the tasks in Ω at time t;
- $c_{req}(t) = \sum_{i \in \Omega | \text{est}_i \le t < \text{ect}_i} c_i$ is the amount of resource required at time t by the tasks in Ω if they were all starting at their earliest starting times;
- $ov(t)$ is the overflow of energy from $c_{req}(t)$ that cannot be executed at time t due to the limited capacity $c_{max}(t)$, and
- $c_{cons}(t)$ is the amount of resource that is actually consumed at time t with $c_{cons}(t) = \min(c_{req}(t) + ov(t-1), c_{max}(t))$; $ov(t) = ov(t-1) + c_{req}(t) - c_{cons}(t)$ and $ov(-1) = 0$.

The horizontally elastic earliest completion time occurs when all tasks are completed. Given a set of tasks Ω, the horizontally elastic schedule of Ω uses in [15] the function $ScheduleTasks(\Omega, C)$ of Algorithm 1 to compute the horizontally elastic earliest completion time of Ω. The properties of this data structure is the linearity of the function *ScheduleTasks*, since the *Profile* contains at most $4n+1$ time points. For a set of tasks Ω, it is proved in [15] that $\mathrm{ect}^F_\Omega \leq \mathrm{ect}^H_\Omega \leq \mathrm{ect}_\Omega$ where ect_Ω is the earliest completion time of Ω.

Algorithm 1 ScheduleTasks(Ω, C) algorithm in $\mathcal{O}(n)$ time from [15].

```
Input: All time point $t$ of the profile $P$, $\Omega$ a set of tasks, $C$ the resource capacity.
Output: the earliest completion time $ect^H$ of the set $\Omega$
forall  $t \in P$ do
    $t.\Delta_{\max} \leftarrow 0$ and $t.\Delta_{req} \leftarrow 0$
for $i \in \Omega$ do
    Increase $t.\mathrm{est}_i.\Delta_{\max}$ and $t.\mathrm{est}_i.\Delta_{req}$ by $c_i$
    Decrease $t.\mathrm{lct}_i.\Delta_{\max}$ and $t.\mathrm{ect}_i.\Delta_{req}$ by $c_i$
$ect \leftarrow -\infty$; $ov \leftarrow 0$; $c_{req} \leftarrow 0$; $S \leftarrow 0$
$t \leftarrow P.first$
while $t.time < lct_\Omega$ do
    $l \leftarrow t.next.time - t.time$; $S \leftarrow S + t.\Delta_{\max}$; $c_{max} \leftarrow \min(S, C)$;
     $c_{req} \leftarrow c_{req} + t.\Delta_{req}$; $c_{cons} \leftarrow \min(c_{req} + ov, c_{max})$
    if $ov > 0 \wedge ov < (c_{cons} - c_{req}) * l$ then
        $l \leftarrow \lceil \frac{ov}{c_{cons} - c_{req}} \rceil$
        t.InsertAfter($t.time + l$, t.cap)
    $ov \leftarrow ov + (c_{req} - c_{cons}) * l$
    $t.cap = C - c_{cons}$
    if $t.cap < C$ then
        $ect \leftarrow t.next.time$;
    $t \leftarrow t.next$
if $ov > 0$ then
    return $+\infty$;
return ect
```

2.6 Existing Horizontally Elastic Edge Finder

The *Profile* data structure is used in [15] to enhance the edge finding rule with the detection rule: for all $i, j \in T$ with $\mathrm{lct}_i > \mathrm{lct}_j$,

$$\mathrm{ect}^H_{\mathrm{LCut}(T,j)\cup\{i\}} > \mathrm{lct}_j \Rightarrow \mathrm{LCut}(T,j) \lessdot i. \tag{GQHE-EF}$$

where $\mathrm{LCut}(T,j) = \{k \in T \mid \mathrm{lct}_k \leq \mathrm{lct}_j\}$. The detection proceeds by batching of tasks of the same height (capacity demand) and all precedences are made in $\mathcal{O}(kn^2)$ where $k \leq n$ is the number of distinct capacities required by the tasks, and n the number of tasks that share the resource [15]. If the free energy of height c_i of the profile of $\mathrm{LCut}(T,j)$ from lct_j to est_i is less than the contribution of task i in the interval $[\mathrm{est}_i, \mathrm{lct}_j)$ when i starts at est_i, then it is deducted that $\mathrm{LCut}(T,j) \lessdot i$. When the relation $\mathrm{LCut}(T,j) \lessdot i$ is detected, the adjustment phase schedules the tasks of $\mathrm{LCut}(T,j)$ on the lower part of the resource of capacity $C - c_i$. Because of the capacity restriction, it results in an overflow, scheduled on the upper part of the resource of capacity c_i. The ending time of the scheduling is where the earliest starting time of task i (est_i) is adjusted. A quadratic algorithm is presented in [15] for the adjustment phase.

In the horizontally elastic edge finder of [11], task i is more constrained than in the previous one [15] (preemption is allowed to task i). The new rule is based on the formulation: for all $i, j \in T$ with $\text{lct}_i > \text{lct}_j$,

$$\text{ect}^H_{\text{LCut}(T,j)\cup\{i'\}} > \text{lct}_j \Rightarrow \text{LCut}(T,j) \lessdot i \qquad \text{(FTHE-EF)}$$

where i' is a task derived from tasks i and j whose parameters $\langle \text{est}_{i'}, \text{lct}_{i'}, p_{i'}, c_{i'}\rangle$ are

$$\langle \text{est}_i, \min(\text{ect}_i, \text{lct}_j), \min(\text{ect}_i, \text{lct}_j) - \text{est}_i, c_i\rangle.$$

It is proved in [11] that this rule is stronger than rule (GQHE-EF). Its authors propose an algorithm of complexity $\mathcal{O}(kn^2)$ (where $k \le n$ is the number of distinct capacities required by the tasks and n the number of tasks sharing the resource) for the detection. A quadratic algorithm was proposed for the adjustment of this rule for an overall complexity of $\mathcal{O}(kn^2)$ [11].

3 Slack-Density Horizontally Elastic Edge Finder Detection Rule

According to [15, 10, 11], the most challenging part of the strengthening of edge finding rule with *Profile* is the detection phase. The rule (GQHE-EF) proposed in [15] is a relaxation of the rule (FTHE-EF) proposed in [11]. In this section, we propose another relaxation of this rule based on the notions of minimum slack and maximum density. In fact, in [19], it is proved that a complete edge finder can be designed using the minimum slack and the maximum density of task intervals.

For a given task $i \in T$ and for a given task $u \in T$ with $\text{lct}_i > \text{lct}_u$, there exists a task $\tau(u,i)$ such that $\text{est}_{\tau(u,i)} \le \text{est}_i$ and for all $l \in T$ with $\text{est}_l \le \text{est}_i$ we have $sl(\Omega_u^{\tau(u,i)}) \le sl(\Omega_u^l)$ (See Definition 6 of [19]). We denote by $\Omega^{\alpha(i)}_{\beta(i)}$ the task interval of minimum slack among the task interval $\Omega_u^{\tau(u,i)}$ for all $u \in T$ with $\text{lct}_i > \text{lct}_u$, i.e.,

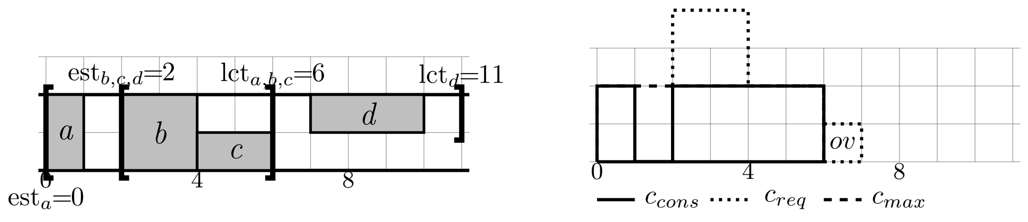
$$\Omega^{\alpha(i)}_{\beta(i)} = \text{argmin}\{sl(\Omega_u^{\tau(u,i)}) \mid \forall u \in T \text{ with } \text{lct}_i > \text{lct}_u\}. \qquad (11)$$

In the following lemma, we are going to prove that each detection of the rule (EF) with pair (Ω, i) is realized by the rule (FTHE-EF) with pair $(\text{LCut}(T,\beta(i)), i)$.

▶ **Lemma 4.** *Let i be a task and Ω be a set of tasks such that $i \notin \Omega$. If the edge finding rule* (EF) *holds with the set Ω and task i, then the rule* (FTHE-EF) *holds also with $LCut(T,\beta(i))$ and task i.*

Proof. Let u be the task such that $\text{lct}_u = \text{lct}_\Omega$. We have $sl(\Omega^{\alpha(i)}_{\beta(i)}) \le sl(\Omega_u^{\tau(u,i)}) \le sl(\Omega)$ by definition of $\Omega^{\alpha(i)}_{\beta(i)}$ and $\Omega_u^{\tau(u,i)}$. The rule (EF) applies to Ω and i (i.e., $e_\Omega + e_i > C(\text{lct}_\Omega - \text{est}_\Omega)$) is equivalent to $sl(\Omega) < e_i$ which implies $sl(\Omega^{\alpha(i)}_{\beta(i)}) < e_i$. According to [11, 15], we have the following dominance properties (FTHE-EF) $\succeq$ (GQHE-EF) $\succeq$ (EF) + (EEF) where $(A) \succeq (B)$ means that rule (A) subsumes rule (B) and $(A) + (B)$ means the conjunction of rules (A) and (B). Therefore, from the inequalities $\text{ect}^H_{\text{LCut}(T,\beta(i))\cup\{i'\}} \ge \text{ect}^H_{\text{LCut}(T,\beta(i))\cup\{i\}} \ge \text{ect}^F_{\text{LCut}(T,\beta(i))\cup\{i\}} > \text{lct}_{\beta(i)}$, it follows that the rule (FTHE-EF) holds for $\text{LCut}(T,\beta(i))$ and task i. ◀

▶ **Example 5.** Consider the CuSP instance of Figure 1a where four tasks share a resource of capacity 2. The rule (EF) holds for $\Omega = \{b, c\}$ and $i = d$. The tasks interval of minimum slack $\Omega^{\alpha(d)}_{\beta(d)}$ is the set $\Omega^{\alpha(d)}_{\beta(d)} = \{b, c\}$ and $\text{LCut}(T, \beta(d)) = \{a, b, c\}$. The profile of the set $\text{LCut}(T, \beta(d)) \cup \{d'\}$ is depicted in Figure 1b. The overflow remaining after time 6 allows to detect that $\text{LCut}(T, \beta(d)) \lessdot d$.



(a) CuSP instance of four tasks sharing a resource of capacity 2

(b) The profile of the set $\mathrm{LCut}(T, \beta(d)) \cup \{d'\} = \{a, b, c, d'\}$

Figure 1 1a: CuSP instance of four tasks sharing a resource of capacity 2 where (EF) detects $\{b, c\} \lessdot d$; 1b: The profile of the set $\mathrm{LCut}(T, \beta(d)) \cup \{d'\} = \{a, b, c, d'\}$ and it is detected that $\mathrm{LCut}(T, \beta(d)) \lessdot d$.

For a given task $i \in T$ and for a given task $u \in T$ with $\mathrm{lct}_i > \mathrm{lct}_u$, there exists a task $\rho(u, i)$ such that $\mathrm{est}_i < \mathrm{est}_\rho(u, i)$ and for all $l \in T$ with $\mathrm{est}_i < \mathrm{est}_l$ we have $ds(\Omega_u^{\rho(u,i)}) \geq ds(\Omega_u^l)$ (See Definition 8 of [19]). We denote by $\Omega_{\delta(i)}^{\gamma(i)}$ the task interval of maximum density among the task interval $\Omega_u^{\rho(u,i)}$ for all $u \in T$ with $\mathrm{lct}_i > \mathrm{lct}_u$, i.e.,

$$\Omega_{\delta(i)}^{\gamma(i)} = \mathrm{argmax}\{ds(\Omega_u^{\rho(u,i)}) \mid \forall u \in T \text{ with } \mathrm{lct}_i > \mathrm{lct}_u\}. \tag{12}$$

If the extended edge finding rule (EEF) holds with a set Ω and a task i, then the rule (FTHE-EF) holds also with $\mathrm{LCut}(T, \delta(i))$ and task i as it is proved in the following lemma.

▶ **Lemma 6.** *Let i be a task and Ω be a set of tasks such that $i \notin \Omega$. If the extended edge finding rule* (EEF) *holds with the set Ω and task i, then the rule* (FTHE-EF) *holds also with $LCut(T, \delta(i))$ and task i.*

Proof. Let i be a task and Ω be a set of tasks such that $i \notin \Omega$. We assume that the extended edge finding rule (EEF) holds with the set Ω and task i. Let Θ be the set of tasks used to update the earliest starting time of task i by rule (Adj). The proof of this lemma will distinguish the case $\mathrm{ect}_i \geq \mathrm{lct}_{\delta(i)}$ from the case $\mathrm{ect}_i < \mathrm{lct}_{\delta(i)}$.

- If $\mathrm{ect}_i \geq \mathrm{lct}_{\delta(i)}$ then $ds(\Omega_{\delta(i)}^{\gamma(i)}) \geq ds(\Theta) > C - c_i$ since $rest(\Theta, c_i) > 0$. The rule (EEF) is detected by $\Omega_{\delta(i)}^{\gamma(i)}$ and task i since the contribution of task i in the interval $[\mathrm{est}_{\gamma(i)}, \mathrm{lct}_{\delta(i)})$ is $c_i(\mathrm{lct}_\delta - \mathrm{est}_{\gamma(i)})$. Thus, $\mathrm{ect}^H_{\mathrm{LCut}(T,\delta(i)) \cup \{i'\}} \geq \mathrm{ect}^H_{\mathrm{LCut}(T,\delta(i)) \cup \{i\}} \geq \mathrm{ect}^F_{\mathrm{LCut}(T,\delta(i)) \cup \{i\}} > \mathrm{lct}_{\delta(i)}$, and the rule (FTHE-EF) holds for $\mathrm{LCut}(T, \delta(i))$ and task i.
- We assume that $\mathrm{ect}_i < \mathrm{lct}_{\delta(i)}$. Let u be a task such that $\mathrm{lct}_u = \mathrm{lct}_\Omega$. The task interval $\Omega_{\delta(i)}^{\gamma(i)}$ has the highest resource consumption spike. Therefore, the set of tasks $\mathrm{LCut}(T, \delta(i))$ is the most indicated to disable the start time of task i. The rest of the proof is going to distinguish two cases: $\mathrm{lct}_\delta < \mathrm{lct}_u$ and $\mathrm{lct}_\delta \geq \mathrm{lct}_u$.
  - If $\mathrm{lct}_{\delta(i)} < \mathrm{lct}_u$ then, $\mathrm{LCut}(T, \delta(i)) \subseteq \mathrm{LCut}(T, u)$ and the horizontally elastic scheduling of $\omega = \mathrm{LCut}(T, u) \setminus \mathrm{LCut}(T, \delta(i))$ is not enough to fill the profile from lct_u to $\mathrm{lct}_{\delta(i)}$. Therefore, the slack of $\mathrm{LCut}(T, u)$ is greater than the one of $\mathrm{LCut}(T, \delta(i))$. Task i has the same contribution in both intervals $\mathrm{LCut}(T, u)$ and $\mathrm{LCut}(T, \delta(i))$. When this contribution is considered during the horizontally elastic scheduling of $\mathrm{LCut}(T, u)$, it results to in an overload. The same overload happens when the contribution of task i is considered during the scheduling of the interval $\mathrm{LCut}(T, \delta(i))$. Thus, the rule (FTHE-EF) holds for $\mathrm{LCut}(T, \delta(i))$ and task i.
  - If $\mathrm{lct}_{\delta(i)} \geq \mathrm{lct}_u$ then, $\mathrm{LCut}(T, u) \subseteq \mathrm{LCut}(T, \delta(i))$ and the horizontally elastic scheduling of $\omega = \mathrm{LCut}(T, \delta(i)) \setminus \mathrm{LCut}(T, u)$ is not enough to fill the profile from $\mathrm{lct}_{\delta(i)}$ to lct_u).

Therefore, the slack of $\mathrm{LCut}(T,u)$ is greater than the one of $\mathrm{LCut}(T,\delta(i))$. The contribution of task i in $\mathrm{LCut}(T,\delta(i))$ is greater than its contribution in $\mathrm{LCut}(T,u)$. When this contribution is considered during the scheduling of $\mathrm{LCut}(T,u)$, it results in an overload. Therefore, the same overload happens when the contribution of task i is considered during the scheduling of the interval $\mathrm{LCut}(T,\delta(i))$. Thus, the rule (FTHE-EF) holds for $\mathrm{LCut}(T,\delta(i))$ and task i. ◀

According to Lemmas 4 and 6, the application of rule (FTHE-EF) to the intervals $\mathrm{LCut}(T,\beta(i))$ and $\mathrm{LCut}(T,\delta(i))$ is enough to subsume the (extended) edge finding detection rule. The new detection rule is based on the application of the rule (FTHE-EF) on the intervals $\mathrm{LCut}(T,\beta(i))$ and $\mathrm{LCut}(T,\delta(i))$ for all $i \in T$. This rule, denoted *Slack-Density Horizontally Elastic Edge Finder* detection, is specified by the formula: for all $i \in T$,

$$\begin{cases} \mathrm{ect}^H_{\mathrm{LCut}(T,\beta(i))\cup\{i'\}} > \mathrm{lct}_{\beta(i)} \Rightarrow \mathrm{LCut}(T,\beta(i)) \lessdot i \\ \mathrm{ect}^H_{\mathrm{LCut}(T,\delta(i))\cup\{i'\}} > \mathrm{lct}_{\delta(i)} \Rightarrow \mathrm{LCut}(T,\delta(i)) \lessdot i \end{cases} \tag{SDHE-EF}$$

We denote by (SDHE-EF) the filtering rule resulting from the combination of the detection rule (SDHE-EF) with the adjustment of [15]. This rule subsumes the edge finding rule and its extension combined with the rule (Adj).

▶ **Theorem 7.** *When the detection rule* (SDHE-EF) *is combined with the adjustment of [15], the resulting filtering rule subsumes the conjunction of rules* (EF) *and* (EEF) *combined with the adjustment rule* (Adj)*.*

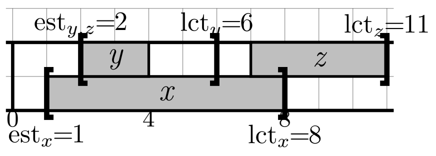
Proof. According to Lemmas 4 and 6, any detection made by (EF) and (EEF) is also done by (SDHE-EF). It is proved in (Theorem 4 of [15]) that the adjustment of [15] is better than the one performed by rule (Adj) after detection made by rules (EF) and (EEF). Therefore, the propagation of (SDHE-EF) combined with the adjustment of [15] subsumes the conjunction of rules (EF) and (EEF) combined with the adjustment rule (Adj). ◀

Back to the CuSP of Figure 1a, after filling the lower part of the resource of capacity 1, the overflow of three units of energy remains. This overflow is scheduled on the upper part of the resource of capacity 1. No energy is consumed in the interval $[1,2)$ since no task is scheduled in this interval. The scheduling of the upper part of the profile ends at time 4 which corresponds to the same adjustment value made by rule (Adj) with $\Theta = \{b,c\}$. Indeed, $\mathit{rest}(\Theta,c_d) = 2 > 0$ and $\mathrm{est}_\Theta + \mathit{rest}(\Theta,c_d)/c_d = 4$. It is known from [11] that (GQHE-EF) is a relaxation of (FTHE-EF). Therefore, (GQHE-EF) and (SDHE-EF) are both relaxations of (FTHE-EF) which subsumes the (extended) edge finding rule. In the following theorem, we prove that the rules (GQHE-EF) and (SDHE-EF) are not comparable, i.e., there exists propagation only performed by (GQHE-EF) and propagation only performed by (SDHE-EF).

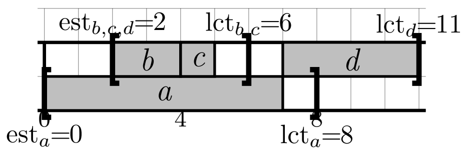
▶ **Theorem 8.** *The rules* (SDHE-EF) *and* (GQHE-EF) *are not comparable.*

Proof. Consider the CuSP instance of Figure 2a, where three tasks $\{x,y,z\}$ share a resource of capacity 2. We have $\Omega^{\alpha(z)}_{\beta(z)} = \{x,y\}$ and the rule (SDHE-EF) detects $\{x,y\} \lessdot z$. This precedence is missed by the rule (GQHE-EF) since, in the profile of $\{x,y\}$, from time 8 back to time 2, we have enough free energy to schedule task z.

Converse, we consider the CuSP instance of figure 2b, where four tasks $\{a,b,c,d\}$ share a resource of capacity 2. In the profile of $\{a,b,c\}$, it remains three units of free energy of height 1 when we move from time 8 back to time 2. This free energy is not enough to schedule task d and the rule (GQHE-EF) detects the relation $\{a,b,c\} \lessdot d$. We have $\Omega^{\alpha(d)}_{\beta(d)} = \{b,c\}$ and the rule (SDHE-EF) misses the relation $\{a,b,c\} \lessdot d$. ◀



(a) CuSP instance of three tasks sharing a resource of capacity 2



(b) CuSP instance of four tasks sharing a resource of capacity 2

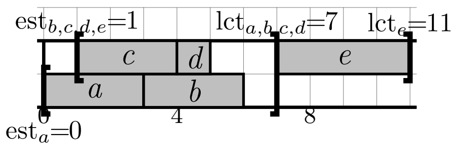
Figure 2 2a: (SDHE-EF) detects $\{x, y\} \lessdot z$ while (GQHE-EF) misses the detection; (2b): (GQHE-EF) detects $\{a, b, c\} \lessdot d$ while (SDHE-EF) misses the detection.

The rule (SDHE-EF) is a relaxation of the rule (FTHE-EF), since it restricts the detection to two intervals instead of n. It is proved in [11] that the rule (FTHE-EF) is not comparable to the rule (TT-EF). This result remains between (SDHE-EF) and (TT-EF) as it is proved in the following theorem.

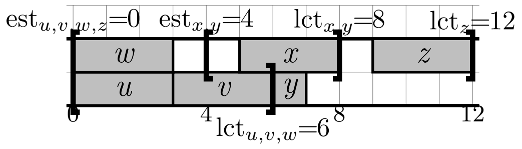
▶ **Theorem 9.** *The rules* (SDHE-EF) *and (TTEF) are not comparable.*

Proof. Consider the CuSP instance of Figure 3a where five tasks $\{a, b, c, d, e\}$ share a resource of capacity 2. We have $\Omega^{\alpha(d)}_{\beta(d)} = \{a, b, c, d\}$ and the rule (SDHE-EF) detects $\{a, b, c, d\} \lessdot e$. This precedence is missed by the rule (TTEF). Indeed, none of the tasks has a fix part and the edge finding rule (EF) fails to detect $\{a, b, c, d\} \lessdot e$.

Conversely, we consider the CuSP instance of figure 3b where six tasks $\{u, v, w, x, y, z\}$ share a resource of capacity 2. When the fix part of task x which intersects the interval $[0, 6]$ is considered, it is detected that $\{u, v, w\} \lessdot z$ and the rule (TTEF) holds. This detection is missed by the rule (SDHE-EF), since the contribution of task x in the interval $\mathrm{LCut}(T, u) = \{u, v, w\}$ is not considered. ◀



(a) CuSP instance of five tasks sharing a resource of capacity 2



(b) CuSP instance of six tasks sharing a resource of capacity 2

Figure 3 3a: (SDHE-EF) detects $\{a, b, c, d\} \lessdot e$ while (TTEF) misses the detection; 3b: (TTEF) detects $\{u, v, w\} \lessdot z$ while (SDHE-EF) misses the detection.

4 Slack-Density Horizontally Elastic Edge Finder Algorithm

The detection algorithm identifies the right bound of the task intervals $\Omega^{\alpha(i)}_{\beta(i)}$ and $\Omega^{\gamma(i)}_{\delta(i)}$ for any task $i \in T$. To do so, the function *ComputesBound()* receives as input the set T_{est} (resp. T_{lct}) of tasks sorted in increasing order of est (resp. lct). The global maximum density and minimum slack are initialized at line 2, while the local maximum density and minimum slack are initialized at line 4 of the loop of line 3. The loop of line 5 updates the value of the local maximum density at line 9 and the global value at line 12. Similarly, the loop of line 14 updates the value of the local minimum slack at line 17 and the global value at line 19.

The function *ComputesBound* has a quadratic complexity as it is shown in Proposition 10.

▶ **Proposition 10.** *ComputesBound runs in* $\mathcal{O}(n^2)$ *in time.*

Algorithm 2 *ComputesBound*$(T_{\text{est}}, T_{\text{lct}})$ in $\mathcal{O}(n^2)$ time.

```
Input: $T_{\text{est}}$ and $T_{\text{lct}}$ the sets of tasks sorted in increasing order of est and lct
       respectively .
Output: The bounds $\beta(i)$ and $\delta(i)$ for all task $i \in T$.
forall $i \in T_{est}$ do
    $\beta(i) \leftarrow -1$, $\delta(i) \leftarrow -1$, $maxDensity(i) \leftarrow 0$, $minSlack(i) \leftarrow \infty$
forall $j \in T_{lct}$ with $lct_j < lct_{j+1}$ do
    $E \leftarrow 0$, $maxD \leftarrow 0$, $minS \leftarrow \infty$
    forall $k \in T_{est}$ in reverse order of est do
        if $lct_k \leq lct_j$ then
            $E \leftarrow E + e_k$, $density \leftarrow E/(\text{lct}_j - \text{est}_k)$
            if $density \geq maxD$ then
                $maxD \leftarrow density$
        else
            if $maxD \geq maxDensity(k)$ then
                $maxDensity(k) \leftarrow maxD$, $\delta(k) \leftarrow j$
        $Energy(k) \leftarrow E$
    forall $k \in T_{est}$ do
        $slack \leftarrow C(\text{lct}_j - \text{est}_k) - Energy(k)$
        if $slack < minS$ then
            $minS \leftarrow slack$
        if $lct_k > lct_j \wedge minS < minSlack(k)$ then
            $minSlack(k) \leftarrow minS$, $\beta(k) \leftarrow j$
```

Proof. The loop of line 3 of complexity $\mathcal{O}(n)$ calls sequentially the loops of lines 5 and 14 of complexity $\mathcal{O}(n)$ each. Therefore, the overall complexity of *ComputesBound* is $\mathcal{O}(n(n+n)) = \mathcal{O}(n^2)$. ◀

In Algorithm 3, the bounds of the task interval of minimum slack and maximum density are computed at line 1. In the loop of line 2, for any unscheduled task (line 3), it is checked at line 6 whether the task interval of minimum slack precedes the task and the relation is recorded at line 7. When no detection is made by the task interval of minimum slack (line 8), the task interval of maximum density is used to check the relation at line 10 and the relation is recorded at line 11.

The complexity of *Slack-Density Horizontally Elastic Edge Finder* is analyzed in Proposition 11.

▶ **Proposition 11.** *Slack-Density Horizontally Elastic Edge Finder runs in $\mathcal{O}(n^2)$ in time.*

Proof. The linear function *ScheduleTasks* is sequentially called twice in the linear loop of line 2. Combined with the quadratic complexity of the function *ComputesBound*, the overall complexity of *Slack-Density Horizontally Elastic Edge Finder* is $\mathcal{O}(n^2+n(n+n)) = \mathcal{O}(n^2)$. ◀

We combine this detection algorithm with the quadratic adjustment algorithm of [15] proposed for the rule (GQHE-EF). Therefore, the complexity of the two-phase algorithm (Detection and Adjustment) is $\mathcal{O}(n^2)$. To our knowledge, this is the first quadratic horizontally elastic edge finder algorithm.

Algorithm 3 *Slack-Density Horizontally Elastic Edge Finder*$(T_{\text{est}}, T_{\text{lct}})$ in $\mathcal{O}(n^2)$ time.

```
Input: $T_{\text{est}}$ and $T_{\text{lct}}$ the sets of tasks sorted in increasing order of est and lct
       respectively.
Output: Prec: the precedence relation.
1  $\beta, \delta \leftarrow ComputesBound(T_{\text{est}}, T_{\text{lct}})$
2  forall $i \in T_{lct}$ do
3      if $ect_i < lct_i$ then
4          if $\beta(i) \neq -1$ then
5              $\text{ect} \leftarrow ScheduleTask(\text{LCut}(T, \beta(i)) \cup \{i'\})$
6              if $ect > lct_{\beta(i)}$ then
7                  $\text{Prec}(i) \leftarrow \beta(i)$
8          if $Prec(i) \neq -1 \wedge \delta(i) \neq -1$ then
9              $\text{ect} \leftarrow ScheduleTask(\text{LCut}(T, \delta(i)) \cup \{i'\})$
10             if $ect > lct_{\delta(i)}$ then
11                 $\text{Prec}(i) \leftarrow \delta(i)$
```

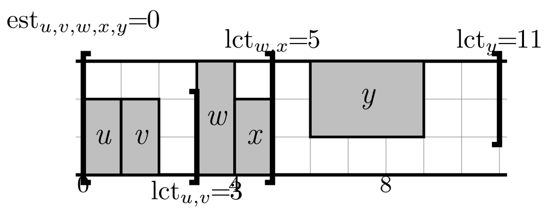


Figure 4 A CuSP instance of five tasks sharing a resource of capacity 3.

▶ **Example 12.** Consider the CuSP instance of Figure 4 where five tasks share a resource of capacity 3.

In this example, $\Omega^u_u = \{u, v\}$, $sl(\Omega^u_u) = 5$ while $\Omega^u_w = \{u, v, w, x\}$, $sl(\Omega^u_w) = 6$. Therefore, $\Omega^{\alpha(y)}_{\beta(y)} = \{u, v\}$ and the rules (EF) and (SDHE-EF) detect that $\Omega^{\alpha(y)}_{\beta(y)} \lessdot y$.

5 Improvements

To improve our algorithm, we have taken into account the fix part of tasks $T \setminus \text{LCut}(T, \beta(i)) \cup \{i\}$ (resp. $T \setminus \text{LCut}(T, \delta(i)) \cup \{i\}$) which overlap with $\text{LCut}(T, \beta(i))$ (resp. $\text{LCut}(T, \delta(i))$) during the computation of $\text{ect}^H_{\text{LCut}(T,\beta(i))\cup\{i'\}}$ (resp. $\text{ect}^H_{\text{LCut}(T,\delta(i))\cup\{i'\}}$) and the adjustment. For a given task $j \in T \setminus \text{LCut}(T, \beta(i)) \cup \{i\}$ with a fix part (i.e., $\text{lst}_j < \text{ect}_j$), which overlap with $\text{LCut}(T, \beta(i))$ (i.e., $\text{lst}_j < \text{lct}_{\beta(i)}$), a new task denoted $f(j, \beta(i))$ is deduced with the following attributes $\langle \text{lst}_j, \min(\text{ect}_j, \text{lct}_{\beta(i)}), \min(\text{ect}_j, \text{lct}_{\beta(i)}) - \text{lst}_j, c_j\rangle$. We denote by $f(T \setminus \text{LCut}(T, \beta(i)) \cup \{i\}, \beta(i))$ the set of deduced fix parts of tasks which overlap with $\text{LCut}(T, \beta(i))$. Analogously, the set of fix part of tasks $f(T \setminus \text{LCut}(T, \delta(i)) \cup \{i\}, \delta(i))$ which overlap with $\text{LCut}(T, \delta(i))$ is considered during the computation of $\text{ect}^H_{\text{LCut}(T,\delta(i))\cup\{i'\}}$.

To do so, we first extend the initial time points, by adding those corresponding to the lst_i for all $i \in T$. The number of time points moves from $4n + 1$ to $5n + 1$ and the function *ScheduleTasks* remains linear. During the initialization of increments, for any task $j \in T \setminus \text{LCut}(T, \beta(i)) \cup \{i\}$ (resp. $j \in T \setminus \text{LCut}(T, \delta(i)) \cup \{i\}$), if tasks j has a fix part which overlap with $\text{LCut}(T, \beta(i))$ (resp. $\text{LCut}(T, \delta(i))$), Δ_{max} and Δ_{req} are increased by c_j at $t.\text{lst}_j$. If $\text{ect}_j < \text{lct}_{\beta(i)}$ (resp. $\text{ect}_j < \text{lct}_{\delta(i)}$) then Δ_{max} and Δ_{req} are decreased by c_j

at $t.\text{ect}_j$. If $\text{ect}_j \geq \text{lct}_{\beta(i)}$ (resp. $\text{ect}_j \geq \text{lct}_{\delta(i)}$) then Δ_{max} and Δ_{req} are decreased by c_j at $t.\text{lct}_{\beta(i)}$ (resp. $t.\text{lct}_{\delta(i)}$). For the adjustment, the fix part tasks $f(T \setminus \text{LCut}(T, \beta(i)) \cup \{i\}, \beta(i))$ (resp. $f(T \setminus \text{LCut}(T, \delta(i)) \cup \{i\}, \delta(i))$) is considered during the computation of the maximum overflow obtained when the set $\text{LCut}(T, \beta(i))$ (resp. $\text{LCut}(T, \delta(i))$) is scheduled on a resource of restricted capacity $C - c_i$. The additional fix part of tasks can increase both the detection and the adjustment of the *Slack-Density Horizontally Elastic Edge Finder* algorithm. For example, in the CuSP of Figure 2b, when the fix part of task a is considered, the task interval $\Omega^{\alpha(d)}_{\beta(d)} = \{b, c\}$ can detect that $\Omega^{\alpha(d)}_{\beta(d)} \lessdot d$ and the adjustment follows. It is also the case for the CuSP instance of Figure 3b, when the fix part of task x is considered.

6 Experimental Results

The new algorithm with improvements was compared to the state-of-the-art horizontally elastic edge finder algorithms ([15, 11]) on Resource-Constrained Project Scheduling Problems (RCPSPs). Comparisons are done on instances of benchmark suites of RCPSP of libraries BL [2] and PSPLib [20] (on sets j30, j60 and j90). Starting time of tasks and makespan was used as variables of the model. They were constrained by the precedence graph and resource limitations. Each resource was modeled with a single CUMULATIVE constraint [1]. The lower bound of the makespan variable was updated with the horizontally elastic earliest completion time of the whole set of tasks T. The TimeTabling algorithm of [21] was added to the common core model. The optimization is based on depth-first search and restart. Anytime a solution is found, the resolution restarts with an additional constraint, which states that the next makespan must be (strictly) better than the current one. The optimum solution is the last solution found within the time limit.

Three configurations of the global constraint CUMULATIVE was considered. The first one denoted GQHE-EF uses the horizontally elastic edge-finder from [15], while the second one denoted FTHE-EF considers the horizontally elastic edge-finder algorithm of [11]. The last configuration denoted SDHE-EF is based on the Algorithm 3 for detection, combined with the adjustment algorithm of [15], all with improvements of Section 5.

Three strategies of selection of variables and values were used to speed up the solving process. Static heuristic were unscheduled tasks are selected in the chronological order of the indices and assigned to their lower bound value. COS + DomOvWDeg where the Conflict Ordering Search heuristic (COS) [13] is combined with the default search strategy of Choco solver *domOverWDeg* [5]. COS + Smallest where the COS is combined to the smallest heuristic (a variable of smallest lower bound among those not yet instantiated is selected and assigned to its lower bound). The implementation was done in Java using Choco solver 4.10.8 [25]. Any search taking more than 10 minutes was counted as a failure.

In Table 1, the column "solve" indicates the number of instances solved to optimality by each configuration, "common solve" indicates the numbers of instances each configuration solves commonly with the baseline configuration SDHE-EF. The column "back$_<$" (resp. "back$_>$") indicates the number of instances were each configuration reduces (resp. need) more backtracks than the baseline configuration. The column "back" (resp. "time") indicates the average number of backtracks (resp. runtime in sec) on common solved instances for each configuration with the baseline configuration.

SDHE-EF always solves more instances than the other configurations, whatever the heuristic selection considers. On common instances solves with static heuristic by SDHE-EF and GQHE-EF (resp. SDHE-EF and FTHE-EF) 100 (resp. 67) instances were solved by SDHE-EF with fewer backtracks (see Table 1). Figures 5a, 5c and 5e compare the number of

Table 1 Number of instances solved, commonly solved with the baseline configuration SDHE-EF, where the number of backtracks is less (resp. great) than the one of the baseline. The average number of backtracks and runtime (in sec) are also provided for commonly solved instances. The one of the baseline configuration are in brackets.

	solve	common solve	$\text{back}_<$	$\text{back}_>$	back	time
Static						
GQHE-EF	1037	1030	72	100	**5569** (5939)	8.294 (**4.666**)
FTHE-EF	1031	1024	93	67	**6997** (7810)	6.552 (**3.695**)
SDHE-EF	**1044**	-	-	-	-	-
COS + DomOvWDeg						
GQHE-EF	1097	1093	213	238	**4675** (4912)	9.377 (**5.207**)
FTHE-EF	1086	1085	209	219	3304 (**3209**)	11.788 (**3.479**
SDHE-EF	**1121**	-	-	-	-	-
COS + Smallest						
GQHE-EF	1053	1053	32	87	6219 (**5859**)	7.976 (**4.489**)
FTHE-EF	1049	1049	42	69	2417 (**2360**)	10.003 (**3.132**)
SDHE-EF	**1060**	-	-	-	-	-

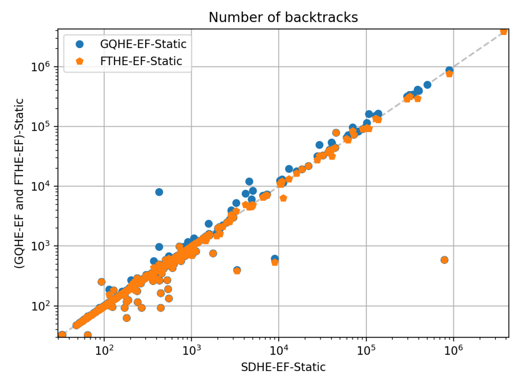
backtracks of GQHE-EF and FTHE-EF with the baseline configuration SDHE-EF on each instances commonly solved. The points above the line $y = x$ shown that SDHE-EF records less backtracks than the other configurations. Figures 5b, 5d and 5f represent the number of solved instances as a function of time for each configuration. The running time gains between SDHE-EF and the two other configurations for static heuristic, is not significant enough as it is illustrated in Figure 5b.

When the heuristic COS + DomOvWDeg is considered, the average number of backtracks of SDHE-EF is 3209, which is less than the average number of backtracks of FTHE-EF (3304), on commonly solved instances with baseline configuration (see Table 1). The average number of backtracks of GQHE-EF (4675) is less than the one of SDHE-EF (4912). The average runtime of SDHE-EF is half of the other configurations whatever the heuristic selection considered (see Table 1). The running time gains between SDHE-EF and the two other configurations is more prominent as illustrated in Figure 5d and 5f.

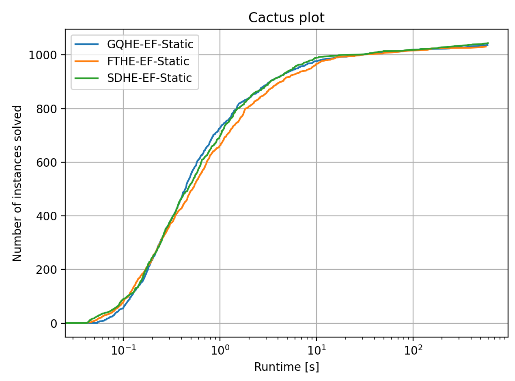
7 Conclusion

In this paper, we have proposed a new strengthening of the edge finder rule based on the *Profile* data structure. The minimum slack and the maximum density are used to select the potential task interval for the edge finder rule. The application of the *Profile* on those task intervals results in a new rule named *Slack-Density Horizontally Elastic Edge Finder*. This new rule subsumes the classic (extended) edge finder rule, but is not comparable to Gingras and Quimper's horizontally elastic edge finder rule [15], and the TimeTable edge finding rule [30]. A quadratic detection algorithm for the new rule is combined with the quadratic adjustment algorithm of [15], which results in an overall complexity of $\mathcal{O}(n^2)$ in time. Improvements based on the *TimeTable* data structure are obtained by considering fix part of external tasks during the horizontally elastic scheduling of task intervals. Experimental results showed that our new algorithm is competitive with start-of-the-art strengthening of edge finding algorithms, with *Profile* for time and tree search reduction. It is a good trade-off between the speed and the filtering power for the rule (FTHE-EF) of [11].

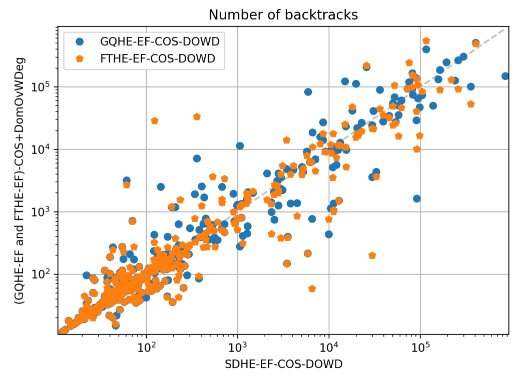
Future works will be devoted to the utilization of the *Profile* data structure in the energetic reasoning to reduce its complexity and increase its filtering power.



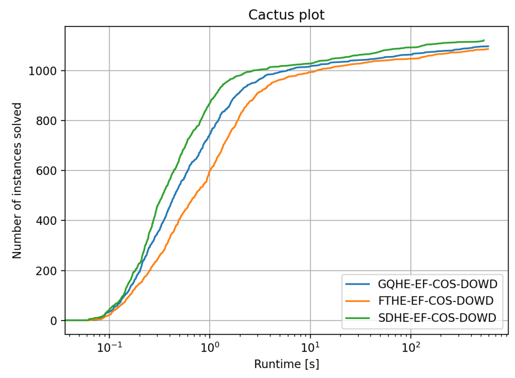
(a) # Backtrack Comparison



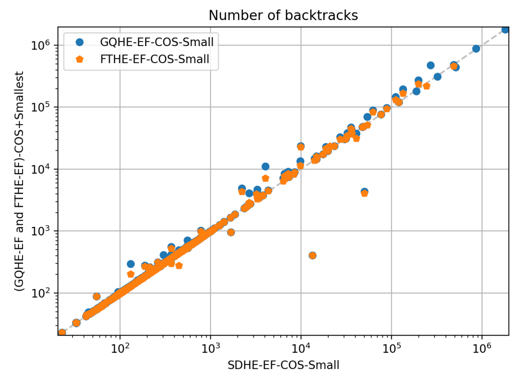
(b) Runtime(sec) Comparison



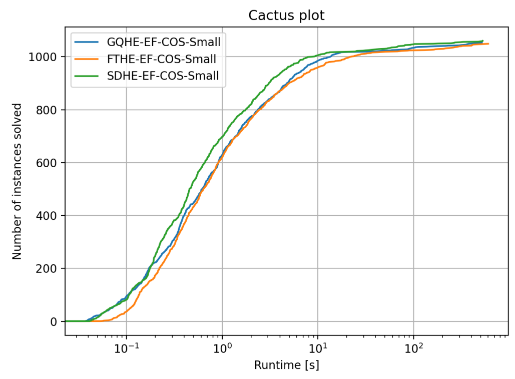
(c) # Backtrack Comparison



(d) Runtime(sec) Comparison



(e) # Backtrack Comparison



(f) Runtime(sec) Comparison

Figure 5 5a, 5c and 5e: Comparison of the number of backtracks of different configurations of the CUMULATIVE constraint. SDHE-EF is used as the baseline model. 5b, 5d and 5f : Number of solved instances as a function of time for the different configurations of the CUMULATIVE constraint.

References

1 Abderrahmane Aggoun and Nicolas Beldiceanu. Extending CHIP in order to Solve Complex Scheduling and Placement Problems. *Mathl. Comput. Modelling*, 17(7):57–73, 1993. URL: `https://hal.archives-ouvertes.fr/hal-00442821`.

2 P. Baptiste, C.L. Pape, and W. Nuijten. *Constraint-Based Scheduling: Applying Constraint Programming to Scheduling Problems*. International Series in Operations Research & Manage-

ment Science. Springer US, 2012. URL: `https://books.google.cm/books?id=qUzhBwAAQBAJ`.

3 Philippe Baptiste, Philippe Laborie, Claude Le Pape, and Wim Nuijten. Constraint-based scheduling and planning. In *Handbook of Constraint Programming*, volume 2 of *Foundations of Artificial Intelligence*, pages 761–799. Elsevier, 2006.

4 Nicolas Beldiceanu and Mats Carlsson. A new multi-resource cumulatives constraint with negative heights. In *CP*, volume 2470 of *Lecture Notes in Computer Science*, pages 63–79. Springer, 2002.

5 Frédéric Boussemart, Fred Hemery, Christophe Lecoutre, and Lakhdar Sais. Boosting systematic search by weighting constraints. In *ECAI*, pages 146–150. IOS Press, 2004.

6 Rajkumar Buyya, M. Manzur Murshed, David Abramson, and Srikumar Venugopal. Scheduling parameter sweep applications on global grids: a deadline and budget constrained cost-time optimization algorithm. *Softw. Pract. Exp.*, 35(5):491–512, 2005.

7 Jacques Carlier, Eric Pinson, Abderrahim Sahli, and Antoine Jouglet. An $O(n^2)$ algorithm for time-bound adjustments for the cumulative scheduling problem. *Eur. J. Oper. Res.*, 286(2):468–476, 2020.

8 Yves Caseau and François Laburthe. Improved CLP scheduling with task intervals. In *ICLP*, pages 369–383. MIT Press, 1994.

9 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.

10 Sévérine Fetgo Betmbe and Clémentin Tayou Djamegni. Horizontally Elastic Edge-Finder Algorithm for Cumulative Resource Constraint Revisited. In *CARI 2020*, THIES, Senegal, October 2020. URL: `https://hal.archives-ouvertes.fr/hal-02931383`.

11 Sévérine Fetgo Betmbe and Clémentin Tayou Djamegni. Horizontally elastic edge-finder algorithm for cumulative resource constraint revisited. *Oper. Res. Forum*, 3(65), 2022. `doi:10.1007/s43069-022-00172-6`.

12 M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.

13 Steven Gay, Renaud Hartert, Christophe Lecoutre, and Pierre Schaus. Conflict ordering search for scheduling problems. In *CP*, volume 9255 of *Lecture Notes in Computer Science*, pages 140–148. Springer, 2015.

14 Steven Gay, Renaud Hartert, and Pierre Schaus. Simple and scalable time-table filtering for the cumulative constraint. In *CP*, volume 9255 of *Lecture Notes in Computer Science*, pages 149–157. Springer, 2015.

15 Vincent Gingras and Claude-Guy Quimper. Generalizing the edge-finder rule for the cumulative constraint. In *IJCAI*, pages 3103–3109. IJCAI/AAAI Press, 2016.

16 Roger Kameugne, Sévérine Betmbe Fetgo, Vincent Gingras, Yanick Ouellet, and Claude-Guy Quimper. Horizontally elastic not-first/not-last filtering algorithm for cumulative resource constraint. In *CPAIOR*, volume 10848 of *Lecture Notes in Computer Science*, pages 316–332. Springer, 2018.

17 Roger Kameugne and Laure Pauline Fotso. A cumulative not-first/not-last filtering algorithm in o(n2log(n)). *Indian Journal of Pure and Applied Mathematics*, 44:95–115, 2013.

18 Roger Kameugne, Laure Pauline Fotso, and Joseph D. Scott. A quadratic extended edge-finding filtering algorithm for cumulative resource constraints. *International Journal of Planning and Scheduling (IJPS)*, 1(4), 2013.

19 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.

20 Rainer Kolisch and Arno Sprecher. Psplib – a project scheduling problem library. *EUROPEAN JOURNAL OF OPERATIONAL RESEARCH*, 96:205–216, 1996.

21 Arnaud Letort, Nicolas Beldiceanu, and Mats Carlsson. A scalable sweep algorithm for the cumulative constraint. In *CP*, volume 7514 of *Lecture Notes in Computer Science*, pages 439–454. Springer, 2012.

22 Luc Mercier and Pascal Van Hentenryck. Edge finding for cumulative scheduling. *INFORMS J. Comput.*, 20(1):143–153, 2008.

23 Pierre Ouellet and Claude-Guy Quimper. Time-table extended-edge-finding for the cumulative constraint. In *CP*, volume 8124 of *Lecture Notes in Computer Science*, pages 562–577. Springer, 2013.

24 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.

25 Charles Prud'homme, Jean-Guillaume Fages, and Xavier Lorca. *Choco Solver Documentation*. TASC, INRIA Rennes, LINA CNRS UMR 6241, COSLING S.A.S., 2016. URL: `http://www.choco-solver.org`.

26 Andreas Schutt, Thibaut Feydy, and Peter J. Stuckey. Explaining time-table-edge-finding propagation for the cumulative resource constraint. In *CPAIOR*, volume 7874 of *Lecture Notes in Computer Science*, pages 234–250. Springer, 2013.

27 Andreas Schutt and Armin Wolf. A new $O(n^2 \log n)$ not-first/not-last pruning algorithm for cumulative resource constraints. In *CP*, volume 6308 of *Lecture Notes in Computer Science*, pages 445–459. Springer, 2010.

28 Petr Vilím. *Global Constraints in Scheduling*. PhD thesis, Charles University in Prague, Faculty of Mathematics and Physics, Department of Theoretical Computer Science and Mathematical Logic, KTIML MFF, Universita Karlova, Malostranské náměstí 2/25, 118 00 Praha 1, Czech Republic, August 2007. URL: `http://vilim.eu/petr/disertace.pdf`.

29 Petr Vilím. Edge finding filtering algorithm for discrete cumulative resources in $O(kn \log n)$. In *CP'09: Proceedings of the 15th international conference on Principles and practice of constraint programming*, pages 802–816, Berlin, Heidelberg, 2009. Springer-Verlag. URL: `http://vilim.eu/petr/cp2009.pdf`.

30 Petr Vilím. Timetable edge finding filtering algorithm for discrete cumulative resources. In *CPAIOR*, volume 6697 of *Lecture Notes in Computer Science*, pages 230–245. Springer, 2011.