

A Continuous Multi-Resources *cumulative* Constraint with Positive - Negative Resource Consumption - Production

Nicolas Beldiceanu and Emmanuel Poder

École des Mines de Nantes, LINA FRE CNRS 2729,
4 rue Alfred Kastler, La Chantrerie, BP 20722, 44307 Nantes Cedex 3, France.
{Nicolas.Beldiceanu, Emmanuel.Poder}@emn.fr

Abstract. This article first introduces an extension of the classical *cumulative* constraint: each task is no more a rectangle but rather a sequence of contiguous trapezoid sub-tasks with variable duration and heights. The resource function is no more constant but is a positive or negative piecewise linear function of time. Finally, a task is no more pre-assigned to one resource, but to a task corresponds a set of possible resource assignments. In this context, this article provides an $O(p \cdot (\log p + q))$ for computing all the cumulated resource profiles where q is the number of resources and p is the total number of trapezoid sub-tasks of all the tasks.

1 Introduction

Most of the work in resource-constrained scheduling is dedicated to problems dealing with tasks that use a constant amount of resource during their execution [3]. However, application fields handling continuously divisible resources like water, oil or electric power, prompt at considering more complex resource consumption or production profiles [8], [6], [11] and [7].

A first contribution of this article is a task model that unifies the two tasks models introduced in [1] and [9]. Each task is no more a rectangle but rather a sequence of contiguous trapezoid sub-tasks. The duration and heights of a trapezoid sub-task are variables over intervals and express a positive or a negative linear resource function of time. Hence, a trapezoid sub-task has positive or negative variable heights and a positive variable duration. A task is non-preemptive, needs for its execution to be assigned to exactly one resource, and to each task corresponds a set of possible resource assignments. Finally, a task must start during a given time interval (due, for instance, to a release date, or to the availability of the resource). Symmetrically, it must also finish during a given time interval. Finally, The duration of each sub-task belongs to a specified intervals.

The main contribution is, for a given resource, an effective algorithm in $O(p \cdot (\log p + q))$ for computing a lower and upper estimation of its final resource profile where q is the number of resources and p is the total number of trapezoid

sub-tasks of all the tasks. These two estimations will be respectively called the *minimum* and *maximum cumulated resource's profiles*. The minimum resource utilisation profile provides an easy way for checking that a given partial schedule (i.e., a schedule where the tasks and sub-tasks variables are not yet all fixed) is not feasible according to the fact that one should not exceed an overall given resource limit. Similarly, the maximum resource utilisation profile allows to check that one can effectively reach, at specific time intervals, a given level of minimum resource utilisation. Moreover, these two profiles coincide when all the tasks are completely fixed. The minimum resource utilisation profile is made up from the sum of all minimum task's profiles, where the *minimum task's profile* represents, for each time point, the smallest possible contribution of the task to a resource profile.¹ For a given task T including possibly positive and negative heights, the computation of its minimum task profile is more complex than just determining a compulsory part [5]. In fact, we first fix the heights of each sub-tasks of T to their minimum value and then consider separately its positive sub-tasks T^+ and its negative sub-tasks T^- . For the former, we compute its *compulsory part* [9] (i.e., the intersection of all feasible schedules of T^+) and for the latter its *envelope* (i.e., the union of all feasible schedules of T^-) and recombine them. Note that the method relies on the fact property the compulsory part of T^+ and envelope of T^- don't overlap each other over time.²

The article is organised as follows: Section 2 introduces the piecewise linear *cumulatives_pwl* constraint and the associated new task model. Section 3 presents different possible contributions of a task to the utilisation of a resource: the compulsory part, the envelope, and finally, the minimum and maximum task's profiles. Section 4 first defines, for each resource, its minimum and maximum cumulated resource's profiles. Then, it describes a sweep algorithm for computing them from all minimum and maximum task's profiles. All derived algorithms are polynomial in the total number of trapezoid sub-tasks.

2 The piecewise linear *cumulatives_pwl* constraint

We consider a set of q renewable resources where the k^{th} resource has a maximum capacity $C_k \geq 0$ and a set of n non-preemptive tasks. Each task needs for its execution to be assigned to exactly one resource within a given subset (that depends of the considered task) of the q resources. Finally, each task is composed of a sequence of contiguous trapezoid sub-tasks which expresses a piecewise linear function of resource requirement. Within this context, this section introduces the task model as well as the corresponding constraint.

2.1 Task model

After introducing the task model used throughout this article, this section presents the notion of feasible instance of a task as well as the notion of resource function.

¹ The maximum resource utilisation profile and the maximum task's profile are defined in a similar way.

² The computation of the maximum task's profile is similar.

Definition 1. A task T_i is defined by a quintuple $(s_{T_i}, td_{T_i}, e_{T_i}, Seq_{T_i}, a_{T_i})$ where:

- The variables³ s_{T_i} , td_{T_i} and e_{T_i} represent respectively the start, the duration and the end of task T_i .
- Seq_{T_i} is a sequence of contiguous trapezoid of p_i sub-tasks $\langle T_i^1, T_i^2, \dots, T_i^{p_i} \rangle$ where the trapezoid sub-task T_i^j ($j = 1..p_i$) is defined by a triple $(sh_{T_i^j}, d_{T_i^j}, eh_{T_i^j})$ where the variables $sh_{T_i^j}$, $d_{T_i^j}$ and $eh_{T_i^j}$ respectively represent the start height of T_i^j (i.e., the resource requirement at its start), the duration of T_i^j and the end height of T_i^j (i.e., the resource requirement at its end). Moreover, we assume that $(sh_{T_i^j} \geq 0 \vee \overline{sh_{T_i^j}} \leq 0)$ and $(eh_{T_i^j} \geq 0 \vee \overline{eh_{T_i^j}} \leq 0)$ ⁴ and finally that $\underline{d_{T_i^j}} \geq 0$.
- The variable a_{T_i} represents the resource assignment of task T_i and takes its value in a finite set of integer values $dom(a)$ ⁵.

$\underline{s_{T_i}}$ is called the *release date* (earliest starting time) while $\overline{e_{T_i}}$ is the *due date* (latest ending time) of T_i . $\overline{s_{T_i}}$ and $\underline{e_{T_i}}$ are respectively the *latest starting time* and the *earliest finishing time* of T_i .

Example 1. The fixed task T , defined by the quintuple

$$(s, td, e, Seq, a) = (1, 7, 8, \langle (2, 3, 3), (3, 1, 1), (0, 1, -1), (0, 1, 0), (3, 1, 0) \rangle, 1)$$

starts at instant 1, has a total duration of 7 and ends at instant 8 and consists of 5 contiguous trapezoid sub-tasks. Moreover, T is assigned to resource 1 and its piecewise linear resource function (resource requirement) h_T is defined by: $h_T(t) = 1/3 \cdot (t-1) + 2$ for $t \in [1, 4[$, $h_T(t) = -2 \cdot (t-4) + 3$ for $t \in [4, 5[$, $h_T(t) = -(t-5)$ for $t \in [5, 6[$, $h_T(t) = 0$ for $t \in [6, 7[$ and $h_T(t) = -2 \cdot (t-7) + 2$ for $t \in [7, 8[$.

Definition 3 introduces the notion of fixed feasible task which is a fixed task's instance verifying the constraint $\sum_{j=1}^{p_i} d_{T_i^j} = td \in [\underline{td_{T_i}} .. \overline{td_{T_i}}]$. For this purpose we first introduce the notion of fixed feasible trapezoid sub-task.

Definition 2. Given a task T_i defined by $(s_{T_i}, td_{T_i}, e_{T_i}, Seq_{T_i}, a_{T_i})$ and a given total duration $td \in [\underline{td_{T_i}} .. \overline{td_{T_i}}]$, a fixed feasible trapezoid sub-tasks sequence of the task T_i with total duration td is such that all the variables $sh_{T_i^j}$, $d_{T_i^j}$, $eh_{T_i^j}$ ($j = 1..p$) are fixed within their respective range and satisfy $\sum_{j=1}^{p_i} d_{T_i^j} = td$.

Definition 3. Given a task T_i defined by $(s_{T_i}, td_{T_i}, e_{T_i}, Seq_{T_i}, a_{T_i})$ and given Seq , a fixed feasible trapezoid sub-tasks sequence of T_i , a fixed feasible instance of T_i is such that s_{T_i} and e_{T_i} are fixed within their respective possible values and Seq_{T_i} is fixed to Seq . We note Φ_{T_i} the set of all the fixed feasible tasks of T_i .

³ A variable v ranges over the interval of consecutive integer values $[\underline{v}, \overline{v}]$.

⁴ The signs of $sh_{T_i^j}$ and $eh_{T_i^j}$ are initially known.

⁵ Without loss of generality all variables, except the resource assignment, could be continuous variables.

Example 2. Throughout this article, we will consider the following example where *Tasks* is the collection of the four tasks T_1 , T_2 , T_3 and T_4 defined as follows:

$$\begin{aligned} T_1 &= (1..2, 4..5, 5..6, \langle(1..2, 2..3, 2), (-1, 2, -1)\rangle, \{1, 2\}) \\ T_2 &= (1..2, 6, 7..8, \langle(3, 2, 2), (-2, 2, -1), (1, 2, 1)\rangle, 1) \\ T_3 &= (0..3, 6, 6..9, \langle(1, 2, 2), (1, 2, 1), (1, 2, 0)\rangle, 1) \\ T_4 &= (1..6, 2, 3..8, \langle(-1, 2, -1)\rangle, \{1, 2\}) \end{aligned}$$

All the starts and ends of tasks T_1 to T_4 are not fixed as well as the first trapezoid sub-task of T_1 . T_1 and T_4 are not yet assigned. Figure 1 provides all the fixed feasible trapezoid sub-task sequences associated to tasks T_1 to T_4 . Observe that task T_1 has four feasible sequences (two of duration 4 ($Seq_{1,1}$ and $Seq_{1,3}$) and two of duration 5 ($Seq_{1,2}$ and $Seq_{1,4}$)) while the other tasks have each only one feasible sequence. Finally, Figure 2 provides the six fixed feasible task's instances ($I_{1,k}$ ($k = 1..6$)) of T_1 .

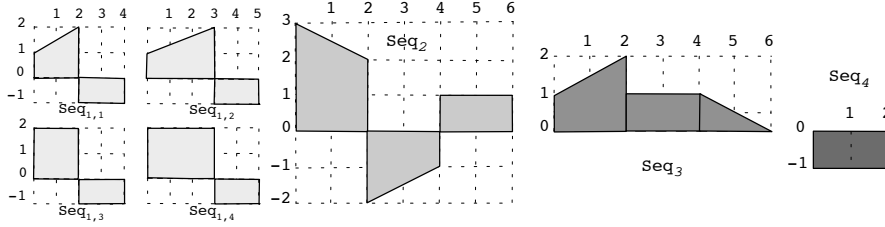


Fig. 1. Feasible trapezoid sub-tasks sequences of the tasks used throughout the article.

Definition 4. We note $st_{T_i^j}$ the start of T_i^j ($st_{T_1^1} = st_{T_1}$). To any $I \in \Phi_{T_i}$ is associated a resource function $h_I(t)$ defined on $[st_{T_i}, e_{T_i}]$ by: $\forall t \in [st_{T_i}, e_{T_i}, \exists! j \in \{1, 2, \dots, p_i\}$ such that $t \in [st_{T_i^j}, st_{T_i^j} + d_{T_i^j}[$. Then $h_I(t) = \frac{eh_{T_i^j} - sh_{T_i^j}}{d_{T_i^j}} (t - st_{T_i^j}) + sh_{T_i^j}$ and $sl_{T_i^j} = \frac{eh_{T_i^j} - sh_{T_i^j}}{d_{T_i^j}}$ is the slope value of the trapezoid sub-task T_i^j . If Φ_{T_i} is reduced to one instance (i.e., T_i is fixed) then we note h_{T_i} the resource function of T_i .

Example 3. (continuation of Example 2) The feasible task's instance $I_{1,1}$ (see Figure 2) associated with T_1 is $(1, 4, 5, \langle(1, 2, 2), (-1, 2, -1)\rangle, \{1, 2\}) = (1, 4, 5, Seq_{1,1}, \{1, 2\})$. Its resource function is defined by $h_I(t) = 1/2 \cdot (t - 1) + 1$ for $t \in [1, 3[$, $h_I(t) = -1$ for $t \in [3, 5[$.

The next definition introduces the notion of non-negative and non-positive task as well as the notion of positive, negative and null trapezoid sub-tasks.

Definition 5. A task T_i is non-negative if the heights of each trapezoid sub-task are non-negative (i.e., $\forall j = 1..p_i, \underline{sh}_{T_i^j} \geq 0$ and $\underline{eh}_{T_i^j} \geq 0$) and is non-positive if the heights of each trapezoid sub-tasks are non-positive (i.e., $\forall j = 1..p_i, \overline{sh}_{T_i^j} \leq 0$ and $\overline{eh}_{T_i^j} \leq 0$). A trapezoid sub-task is positive if its two heights are non-negative and not both null and is negative if its two heights are non-positive and not both null. Otherwise the trapezoid is null ($\underline{sh}_{T_i^j} = \overline{sh}_{T_i^j} = \underline{eh}_{T_i^j} = \overline{eh}_{T_i^j} = 0$).

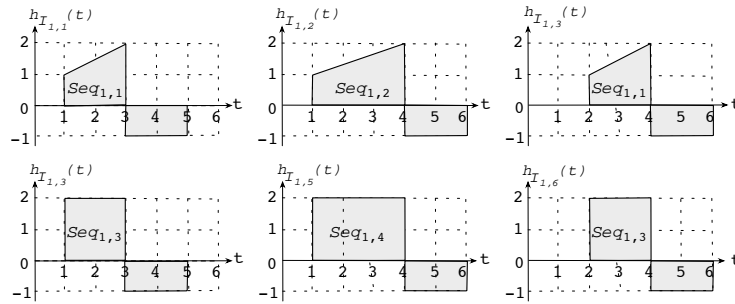


Fig. 2. Fixed feasible instances of T_1 .

Example 4. (continuation of Example 3) T_3 is a non-negative task while T_4 is a non-positive task. T_1^1 is a positive trapezoid sub-task while T_1^2 is a negative trapezoid sub-task.

2.2 The piecewise linear cumulative *cumulatives_pwl*

The *piecewise linear cumulative* constraint has the form *cumulatives_pwl*(*Tasks*, *Resources*, *Constraint*) where:

- *Tasks* is a collection of tasks $\langle T_1, T_2, \dots, T_n \rangle$ where the task T_i ($i = 1..n$) is defined by a quintuple $(s_{T_i}, td_{T_i}, e_{T_i}, Seq_{T_i}, a_{T_i})$.
- *Resources* is a set of q integers C_1, C_2, \dots, C_q where $C_k \geq 0$ ($k = 1..q$) is the resource capacity of the k^{th} resource if *Constraint* = ' \leq ' or the minimum level of the k^{th} resource if *Constraint* = ' \geq '.
- *Constraint*⁶ is the less or equal (i.e., \leq) or the greater or equal (i.e., \geq) constraint.

The constraint *cumulatives_pwl* holds if the following conditions are true:

1. $\forall i = 1..n, s_{T_i} + td_{T_i} = e_{T_i}$ (i.e., the end of a task is equal to the sum of its start and its duration),⁷
2. $\forall i = 1..n, \sum_{j=1}^{p_i} d_j = td_{T_i}$ (i.e., the total duration of a task is equal to the sum of the durations of its trapezoid sub-tasks),
3. Case *Constraint* = ' \leq ': $(\forall k = 1..q), (\forall t \in \mathbb{R}), \sum_{i/a_{T_i}=\{k\}} h_{T_i}(t) \leq C_k$ (i.e., for each resource k and for each instant t , the sum of the values taken at t by the resource functions that are assigned to the resource k , is less than or equal to the capacity C_k of the resource k).
Case *Constraint* = ' \geq ': $(\forall k = 1..q), (\forall t \in \mathbb{R} \text{ such that } \exists T_i/t \in [s_{T_i}, e_{T_i}]), \sum_{i/a_{T_i}=\{k\}} h_{T_i}(t) \leq C_k$ (i.e., for each resource k and for each instant t such that at least one task is executed at t , the sum of the values taken at t by

⁶ Observe that, currently, multiple execution modes for a task cannot be directly defined within the *cumulatives_pwl* constraint.

⁷ When s_{T_i} , td_{T_i} and e_{T_i} belong to specific intervals, none of them is redundant.

the resource functions of the tasks that are assigned to the resource k , is greater than or equal to the minimum level C_k of the resource k).

Example 5. (continuation of Example 2) The constraint *cumulatives-pwl*
 $\langle (2, 4, 6, \langle(1, 2, 2), (-1, 2, -1)\rangle, 2), (1, 6, 7, \langle(3, 2, 2), (-2, 2, -1), (1, 2, 1)\rangle, 1)$
 $(3, 6, 9, \langle(1, 2, 2), (1, 2, 1), (1, 2, 0)\rangle, 1), (1, 2, 3, \langle(-1, 2, -1)\rangle, 1) \rangle$
 $\langle 2, 2 \rangle, \leq$ holds.

3 Minimum and maximum cumulated resource's profiles

Given a resource r , this section shows how to compute its minimum (respectively maximum) cumulated resource's profiles for r . They are in fact determined by cumulating all minimum (maximum) task's profiles. For this purpose, we first remind (3.1 and 3.2) some results, proved by [9], for computing the compulsory part of a task and then we extend them to the computation of the envelope (3.3).⁸ Then, in 3.4, we explain how to use simultaneously the notions of compulsory part and envelope to compute the minimum and maximum task's profiles. Finally, in 4.1, we use the previous notions in order to compute the minimum and maximum resource's profiles for a given resource.

To compute the minimum task's profiles of a given task T , we first need to introduce two specific instances of T that are its earliest and its latest schedules.

3.1 Earliest and latest schedules of a task T

Let T^{min} and T^{max} denote the earliest and the latest schedules of T and st_{T^j} and \bar{st}_{T^j} ($j = 1..p$) respectively denote the starts of the trapezoid sub-task T^j in the schedules T^{min} and T^{max} . Poder *et al.* have shown in [9] how to compute st_{T^j} and \bar{st}_{T^j} with a complexity of $O(p)$. To obtain T^{min} the task T is started at its earliest date \underline{s}_T ; then, $st_{T^1}, st_{T^2}, \dots, st_{T^p}$ are successively fixed as small as possible with respect to the feasibility of the end of the task. The following recursive formulae computes st_{T^j} ($j = 1..p + 1$):

$$\underline{st}_{T^1} = \underline{s}_T \text{ and } \forall j = 1..p, \underline{st}_{T^{j+1}} = \max \left(\underline{st}_{T^j} + \underline{d}_{T^j}, \underline{e}_T - \sum_{k=j+1}^p \bar{d}_{T^k} \right).$$

T^{max} is obtained in a similar way i.e.,:

$$\bar{st}_{T^1} = \bar{s}_T \text{ and } \forall j = 1..p, \bar{st}_{T^{j+1}} = \min \left(\bar{st}_{T^j} + \bar{d}_{T^j}, \bar{e}_T - \sum_{k=j+1}^p \underline{d}_{T^k} \right).$$

Observe that $\underline{st}_{T^{p+1}} = \underline{e}_T$ and $\bar{st}_{T^{p+1}} = \bar{e}_T$.

3.2 Compulsory part of a non-negative task

The *compulsory part* was initially introduced by Lahrichi [5], for a rectangle task as the intersection of all feasible schedules of the task. As the domains of the variables of the task get more and more restricted, the compulsory part will

⁸ Within the context of a non-negative task, its compulsory part and its envelope can be respectively interpreted as the lower and upper bounds of the task (i.e., at each instant they provide the minimum and maximum heights of the task).

increase until becoming a schedule of the task. Within rectangle tasks, [4] use the notion of compulsory part to tighten lower bounds for resource-constrained scheduling problems, while [2] use the compulsory part in a set of propagation rules to solve cumulative constraints.

Here, we consider a non-negative task T . Nevertheless, observe that computing the compulsory part of a non-positive task is symmetric: in fact, if T is defined, for any t , by $h_T(t) = -h_{T'}(t)$ then, for any t , $h_{CP(T')}(t) = -h_{CP(T)}(t)$.

Note that it is sufficient, in the computation of the compulsory part of T , to consider only feasible instances of T where heights are minimum. So in the following, a task T has its heights fixed at their minimum.

Definition 6. *The compulsory part $CP(T)$ of a task T is not empty if and only if $\bar{s}_T < \underline{e}_T$. Its resource function $h_{CP(T)}$ satisfies for any $t \in [\bar{s}_T, \underline{e}_T[$ $h_{CP(T)}(t) = \inf_{I \in \Phi_T} (h_I(t))$ and $h_{CP(T)}(t) = 0$ elsewhere.*

In order to compute the compulsory part of a task, we first need to introduce the notion of valley in a sequence of trapezoid sub-tasks and the associated task named the Level Valley Task.

Definition 7. *Let $H_T^{min} = h_1 h_2 \cdots h_{2p} = \underline{sh}_{T^1} \underline{eh}_{T^1} \underline{sh}_{T^2} \underline{eh}_{T^2} \cdots \underline{sh}_{T^p} \underline{eh}_{T^p}$ be the sequence of all start and end minimum heights of trapezoid sub-tasks of T . An height $h_k \in H_T^{min}$ ($1 < k < 2p$) defines an end of valley if and only if $\exists j$ ($1 < j \leq k$) such that $h_{j-1} > h_j$ and $h_j = h_{j+1} = \cdots = h_k$ and $h_k < h_{k+1}$ (a strict decrease in the resource function is followed by a strict increase).*

Result 1 *Let $h_{j_1} h_{j_2} \cdots h_{j_v}$ be the possibly empty sub-sequence of H_T^{min} of all heights of trapezoid sub-tasks of T that correspond to the ends of the v valleys. Let $LVT(T)$, called the Level Valley Task of T , be the task defined for any $t \in [\bar{s}_T, \underline{e}_T[$ by $h_{LVT(T)}(t) = \min\left(+\infty, \left\{ h_{j_k} \text{ such that } t \in \left[\underline{st}_{T^{\lfloor \frac{j_k}{2} \rfloor + 1}}, \bar{st}_{T^{\lfloor \frac{j_k}{2} \rfloor + 1}} \right] \right\} \right)$*

Then, $CP(T) = T^{min} \cap T^{max} \cap LVT(T)$ i.e., for any $t \in [\bar{s}_T, \underline{e}_T[$, $h_{CP(T)}(t) = \min(h_{T^{min}}(t), h_{T^{max}}(t), h_{LVT(T)}(t))$ and is null elsewhere. If T has no valley ($v = 0$) then, for any $t \in [\bar{s}_T, \underline{e}_T[$ $h_{CP(T)}(t) = \min(h_{T^{min}}(t), h_{T^{max}}(t))$.

Example 6. The left part of Figure 3 illustrates the computation of $CP(T)$ where T has two valleys (see Part (A)). They correspond to the start st_{T^2} of T^2 ($h_1 = \underline{sh}_{T^2}$) and the end st_{T^4} of T^3 ($h_2 = \underline{eh}_{T^3}$). Parts (A) and (B) give respectively T^{min} and T^{max} . Then, Part (C) shows $LVT(T)$ which is defined by $h_{LVT(T)}(t) = +\infty$ for $[\bar{s}_T, \underline{st}_{T^2}[$, $h_{LVT(T)}(t) = h_1$ for $t \in [\underline{st}_{T^2}, \underline{st}_{T^4}[$, and $h_{LVT(T)}(t) = h_2$ for $t \in [\underline{st}_{T^4}, \bar{st}_{T^2}[\cup [\bar{st}_{T^2}, \underline{e}_T[$ (as $h_1 > h_2$). Finally, Part (D) shows the computation of $CP(T)$ as the intersection of T^{min} , T^{max} and $LVT(T)$.

Complexity for computing the compulsory part. Within [9], Poder *et al.* provide an algorithm for computing the compulsory part of a task, made of p trapezoid sub-tasks and $v > 0$ valleys, in $O(p + v \cdot \log v)$. In fact, T^{min} and T^{max}

are computed in $O(p)$ (i.e., the complexity for computing the two sequences st^j and \bar{st}^j ($j = 1..p+1$)), $LVT(T)$ is computed using a heap structure in $O(v \cdot \log v)$. Then, the intersection of T^{min} , T^{max} and $LTT(T)$ is obtained, by scanning them in parallel, in $O(p + v)$. Hence, a complexity of $O(p + v \cdot \log v)$ that is $O(p \log p)$. If $v = 0$ the complexity is $O(p)$.

3.3 Envelope of a non-negative task

The *envelope* of a task is the union of all feasible schedules of the task. As the domains of the variables of the task get more and more restricted, the envelope will decrease until it becomes a schedule of the task. In the context of multiple resources, a task has the same envelope on each resource where it may be potentially assigned and an empty envelope elsewhere. Here, we consider a non-negative task T and we compute its envelope $Env(T)$. Nevertheless, observe that computing the envelope of a task where the heights of each trapezoid sub-task are non-positive is symmetric. In fact, if T is defined, for any t , by $h_T(t) = -h_{T'}(t)$ then, for any t , $h_{Env(T')}(t) = -h_{Env(T)}(t)$.

Definition 8. *The envelope $Env(T)$ of a task T is such that its resource function $h_{Env(T)}$ satisfies $h_{Env(T)} = \sup_{I \in \Phi_T}(h_I(t))$ for any $t \in [\underline{s}_T, \bar{e}_T[$ and $h_{Env(T)}(t) = 0$ elsewhere.*

Note that it is sufficient, in the computation of the envelope of T , to consider only feasible schedules of T where the heights of each trapezoid sub-task are fixed at their maximum. So, in this section, a task T has its heights fixed at their maximum.

As we have introduced, for computing the compulsory part of a task, the notion of valley in a sequence of trapezoid, we introduce now, for computing the envelope, the notion of top and the associated task named the Level Top Task.

Definition 9. *Let $H_T^{max} = h_1 h_2 \cdots h_{2p} = \bar{s}h_{T^1} \bar{e}h_{T^1} \bar{s}h_{T^2} \bar{e}h_{T^2} \cdots \bar{s}h_{T^p} \bar{e}h_{T^p}$ be the sequence of all start and end maximum heights of trapezoid sub-tasks of T . A height $h_k \in H_T^{max}$ ($1 \leq k \leq 2p$) defines an end of a top if and only if $\exists j$ ($1 \leq j \leq k$) such that $h_{j-1} < h_j$ and $h_j = h_{j+1} = \cdots = h_k$ and $h_k > h_{k+1}$ with the convention that $h_0 = h_{2p+1} = 0$.*

Result 2 *Let $h_{j_1} h_{j_2} \cdots h_{j_w}$ be the sub-sequence of HT^{max} of all heights of T that define a top. The Level Top Task $LTT(T)$ of T is the task defined by:*

$$\forall t \in [\underline{s}_T, \bar{e}_T[, h_{LTT(T)}(t) = \max \left(0, \left\{ h_{j_k} \text{ such that } t \in \left[\underline{st}_{T^{\lfloor \frac{j_k}{2} \rfloor + 1}}, \bar{st}_{T^{\lfloor \frac{j_k}{2} \rfloor + 1}} \right] \right\} \right)$$

(with the convention that $s_{T_{p+1}} = e$). The envelope of T is obtained by computing the union of T^{min} , T^{max} and $LTT(T)$ i.e., $h_{Env(T)}$ satisfies $h_{Env(T)}(t) = \sup(h_{T^{min}}(t), h_{T^{max}}(t), h_{LTT(T)}(t))$ for any $t \in [\underline{s}_T, \bar{e}_T[$ and 0 elsewhere.

The demonstration of Result 2 is similar to the proof of Result 1 so is omitted.

Example 7. The right part of Figure 3 illustrates the computation of the envelope of a task T that has three tops with heights $h_1 = \bar{s}h_{T1}$, $h_2 = \bar{e}h_{T2}$ and $h_3 = \bar{e}h_{T4}$ (see part (E)). They correspond to the start $s_T = st_{T1}$ of the task, the end st_{T3} of the second sub-task and the end $e_T = st_{T4}$ of the task T . Parts (E) and (F) give respectively T^{min} and T^{max} . Then, Part (G) shows $LTT(T)$ that is defined by $h_{LTT(T)}(t) = h_1$ for $t \in [s_T, \bar{s}_T]$, $h_{LVT(T)}(t) = h_2$ for $t \in [st_{T3}, e_T]$ (as $h_2 < h_3$ on $[e_T, \bar{s}st_{T3}]$), and $h_{LVT(T)}(t) = h_3$ for $t \in [e_T, \bar{e}_T]$. Finally, part (H) shows the computation of $Env(T)$ as the union of T^{min} , T^{max} and $LTT(T)$.

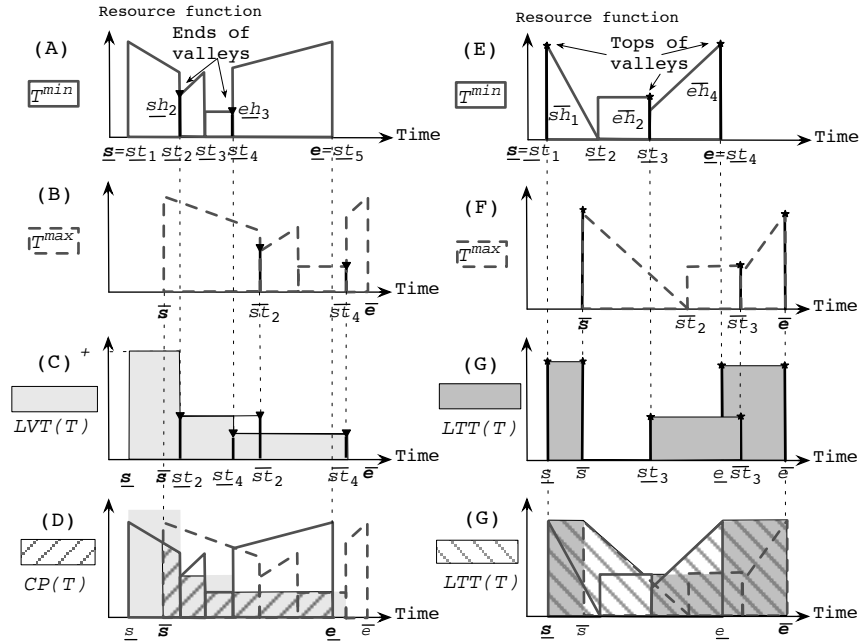


Fig. 3. Computation of the compulsory part and the envelope of a task.

Complexity for computing the envelope of a task. A task T has always at least one top except if T satisfies $\forall t, h_T(t) = 0$. The algorithm to compute the envelope is similar to the one given for the computation of the compulsory part. So it has a complexity of $O(p + w \cdot \log w)$ that is $O(p \log p)$.

3.4 Minimum and maximum task's profiles of any task T

We now explain how to compute the minimum and maximum task's profiles of a task T . Let us first introduce this two notions formally.

Definition 10. Let $mP(T)$ (resp. $MP(T)$) denote the minimum (resp. maximum) task's profile of the task T . Its resource function $h_{mP(T)}$ (resp. $h_{MP(T)}$)

satisfies, for any $t \in [\underline{s}_T, \bar{e}_T]$, $h_{mP(T)}(t) = \inf_{I \in \Phi_T} (h_I(t))$ (resp. $h_{MP(T)}(t) = \sup_{I \in \Phi_T} (h_I(t))$).

Note that it is enough, in the computation of $mP(T)$ (resp. $MP(T)$) to consider only feasible schedules where all trapezoid's heights of T are fixed at their minimum (resp. maximum).

Assume now that heights of T are fixed. Then, let T^+ (resp. T^-) denote the task made from T by replacing each negative (resp. positive) trapezoid sub-task T^j of T by a null trapezoid sub-task and same duration as the replaced trapezoid sub-task (we merge consecutive null sub-tasks in a single sub-task).

The next result expresses the resource function of the minimum (resp. maximum) task's profile of a task T according to the resource functions of the compulsory part of T^+ (resp. T^-) and of the envelope of T^- (resp. T^+).

Result 3 *Let t be a given time point.*

If T is assigned to a resource then, on that resource

$$\begin{cases} h_{mP(T)}(t) = h_{CP(T^+)}(t) + h_{Env(T^-)}(t), \\ h_{MP(T)}(t) = h_{CP(T^-)}(t) + h_{Env(T^+)}(t). \end{cases}$$

else, on each resource where T may be assigned

$$\begin{cases} h_{mP(T)}(t) = h_{Env(T^-)}(t), \\ h_{MP(T)}(t) = h_{Env(T^+)}(t). \end{cases}$$

Moreover, for any t , we can't have neither both $h_{CP(T^+)}(t) \neq 0$ and $h_{Env(T^-)}(t) \neq 0$ nor both $h_{Env(T^+)}(t) \neq 0$ and $h_{CP(T^-)}(t) \neq 0$.

Proof. of Result 3

As the definitions of $mP(T)$ and $MP(T)$ are symmetric, we only prove the result for $mP(T)$. Note that, by definition of T^+ and T^- , for any instance $I \in \Phi_T$, the associated instances I^+ and I^- are such that $h_{I^+}(t) = \max(0, h_I(t))$ and $h_{I^-}(t) = \min(0, h_I(t))$. Therefore $h_{CP(T^+)}(t) = \inf_{I \in \Phi_T} (h_{I^+}(t))$ and $h_{Env(T^-)}(t) = \inf_{I \in \Phi_T} (h_{I^-}(t))$.

We distinguish two cases: $\forall I \in \Phi_T, h_I(t) > 0$ (1) and its contrary $\exists I \in \Phi_T, h_I(t) \leq 0$ (2). In the former case, $\forall I \in \Phi_T, h_I(t) = h_{I^+}(t)$. Then $\inf_{I \in \Phi_T} (h_I(t)) = \inf_{I \in \Phi_T} (h_{I^+}(t)) > 0$. i.e., $h_{mP(T)}(t) = h_{CP(T^+)}(t) > 0$. Moreover, $\forall I \in \Phi_T, h_{I^-}(t) = 0$ so $h_{Env(T^-)}(t) = 0$. In the latter case $\inf_{I \in \Phi_T} (h_I(t)) = \inf_{I \in \Phi_T} (h_{I^-}(t)) > 0$. i.e., $h_{mP(T)}(t) = h_{Env(T^-)}(t)$. Moreover, $\inf_{I \in \Phi_T} (h_{I^+}(t)) = 0$ i.e. $h_{CP(T^+)}(t) = 0$.

We have proved that we can't have both $h_{CP(T^+)}(t) \neq 0$ and $h_{Env(T^-)}(t) \neq 0$ (by definition $h_{CP(T^+)}(t) \geq 0$ and $h_{Env(T^-)}(t) \leq 0$) and that if $h_{CP(T^+)}(t) > 0$ then $h_{mP(T)}(t) = h_{CP(T^+)}(t)$ else $h_{mP(T)}(t) = h_{Env(T^-)}(t)$. \square

Complexity for computing the minimum and maximum profiles.

From Result 3 and as the complexity for computing a compulsory part or an envelope is $O(p \log p)$ then the complexity for computing the minimum or the maximum profile is also $O(p \log p)$.

Example 8. (continuation of Example 2) As T_1 and T_4 may be assigned to resources 1 and 2 ($a_{T_1} = a_{T_4} = \{1, 2\}$), only envelopes are taken into account so $h_{mP(T_1)} = h_{Env(T_1^-)}$ and $h_{mP(T_4)} = h_{Env(T_4)}$ ($T_4^- = T_4$ as T_4 is non-positive). As T_2 and T_3 are assigned to resource 1 ($a_{T_2} = a_{T_3} = \{1\}$), compulsory parts and envelopes are taken into account so $h_{mP(T_2)} = h_{CP(T_2^+)} + h_{Env(T_2^-)}$ and $h_{mP(T_3)} = h_{CP(T_3)}$ ($T_3^+ = T_3$ and $T_3^- = \emptyset$ as T_3 is non-negative). Parts (A) to (D) of Figure 4 respectively give $mP(T_1)$, $mP(T_2)$, $mP(T_3)$ and $mP(T_4)$. Finally, the following table summarises the different profiles (a trapezoid sub-task is encoded by $\langle start, startheight, end, endheight \rangle$).

$T_i; a_{T_i}$	$CP(T_i^+)$	$Env(T_i^-)$	$mP(T_i)$
$T_1; a_{T_1} = \{1, 2\}$	$\langle 2, 1, 3, 3/2 \rangle$	$\langle 3, -1, 6, -1 \rangle$	$Env(T_1^-)$
$T_2; a_{T_2} = 1$	$\langle 2, 5/2, 3, 2 \rangle, \langle 6, 1, 7, 1 \rangle$	$\langle 3, -2, 4, -2 \rangle, \langle 4, -2, 6, -1 \rangle$	$CP(T_2^+) \cup Env(T_2^-)$
$T_3; a_{T_3} = 1$	$\langle 3, 1, 4, 1 \rangle, \langle 4, 1, 2, 6, 0 \rangle$	\emptyset	$CP(T_3)$
$T_4; a_{T_4} = \{1, 2\}$	\emptyset	$\langle 1, -1, 8, -1 \rangle$	$Env(T_4)$

4 Computing all the minimum and maximum cumulated resource's profiles using a sweep algorithm

This section first defines the notions of minimum and maximum cumulated resource's profiles. Then it describes a polynomial (according to the number of sub-tasks and of resources) sweep algorithm to compute, for each resource r its minimum cumulated resource's profile $mcrP(r)$. These profile is computed from all minimum task's profiles $mP(T_i)$ $i = 1..n$ of the *cumulatives_pwl* constraints.

4.1 Minimum and maximum cumulated resource's profiles

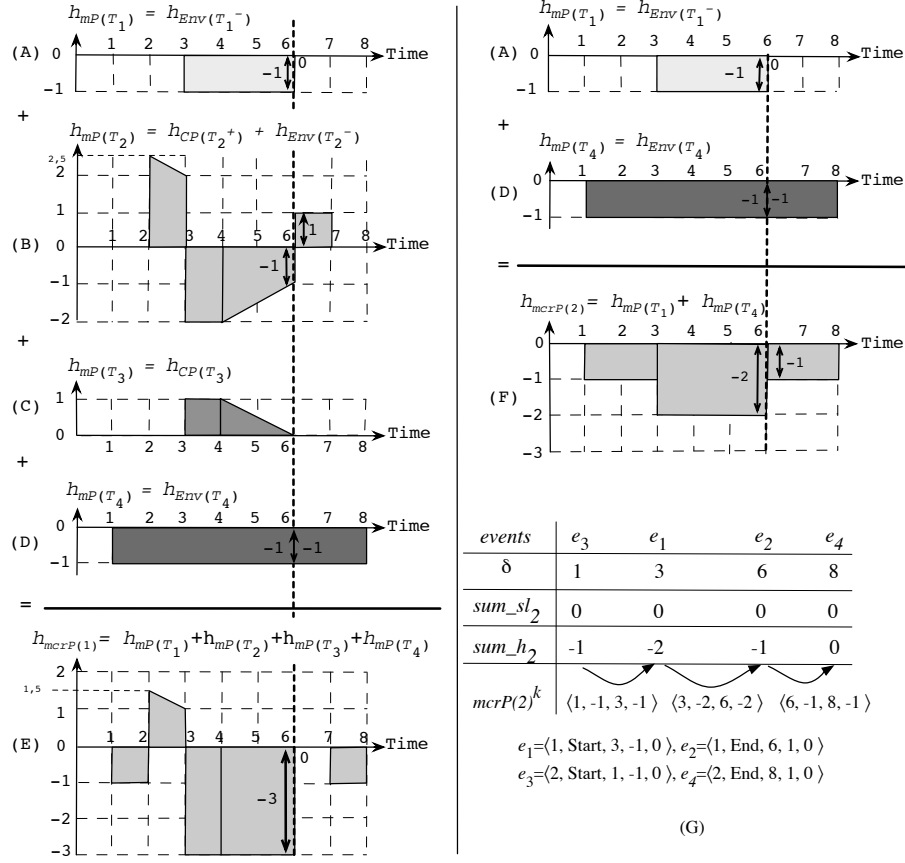
We now show how to compute the minimum and maximum cumulated resource's profiles from the minimum and the maximum task's profiles.

Definition 11. *Let r be a resource and $mcrP(r)$ (resp. $McrP(r)$) denotes the minimum (resp. maximum) cumulated profile of resource r by all the tasks. Then, for any t , $h_{mcrP(r)}(t) = \sum_{T_i/r \in a_{T_i}} h_{mP(T)}(t)$ (resp. $h_{McrP(r)}(t) = \sum_{T_i, r \in a_{T_i}} h_{MP(T)}(t)$).*

As the domains of the variables of all tasks are progressively reduced (until becoming completely fixed), $mcrP(r)$ and $McrP(r)$ respectively increases and decreases (until becoming a same single profile). From now on, as $mcrP(r)$ and $McrP(r)$ have similar definitions, we focus on the computation of $mcrP(r)$

Example 9. (continuation of Example 8) Parts (E) and (F) of Figure 4 and the following table provide $mcrP(1)$ and $mcrP(2)$ ($h_{mcrP(1)} = \sum_{i=1}^4 h_{mP(T_i)}$ and $h_{mcrP(2)} = h_{mP(T_1)} + h_{mP(T_4)}$)

r	Trapezoid sub-tasks of $mcrP(r)$
1	$\langle 1, -1, 2, -1 \rangle, \langle 2, 3/2, 3, 1 \rangle, \langle 3, -3, 4, -3 \rangle, \langle 4, -3, 6, -3 \rangle, \langle 6, 0, 7, 0 \rangle, \langle 7, -1, 8, -1 \rangle$
2	$\langle 1, -1, 3, -1 \rangle, \langle 3, -2, 6, -2 \rangle, \langle 6, -1, 8, -1 \rangle$

Fig. 4. Computation of $mcrP(1)$ and $mcrP(2)$

4.2 Sweep algorithm

To compute $mcrP(r)$, for a given resource r , we have extended the sweep algorithm presented in [1] in order to handle trapezoid sub-tasks. A general introduction on sweep is given in [10] and an example of use of sweep, within the context of disjunctive scheduling, is provided in [12].

The extended sweep algorithm moves a vertical line Δ from one event to the next event - in our context, an event corresponds to a start or an end of a trapezoid sub-task of $mP(T_i)$ ($i = 1..n; r \in a_{T_i}$) - on the time axis and builds $mcrP(r)$ incrementally. It uses two data structures:

- The first one, called *the sweep line status*, contains for the resource r some information related to the current position δ of the vertical line Δ : sum_h_r and sum_sl_r , respectively the height and the slope of $mcrP(r)$ at position δ .
- The second data structure, called *the event points series*, contains the events associated with the trapezoid sub-tasks used for building $mcrP(r)$ (namely

the trapezoid sub-tasks of $mP(T_i)$ such that T_i may be assigned to r (see Result 3 and Definition 11). We encode these events by using the following fields: $\langle task, kind, date, height, slope \rangle$, where:

- *task* indicates which task generates the event.
- *kind* expresses whether the event corresponds to the start ($kind = \text{Start}$) or to the end ($kind = \text{End}$) of trapezoid that has generated this event.
- *date* specifies the location in time of the event i.e., contains, if $kind = \text{Start}$, the start of the trapezoid otherwise its end.
- *height* gives the quantity to add to sum_h_r . It contains, if $kind = \text{Start}$, the height of the trapezoid at its start otherwise it contains the opposite of the height at its ends.
- *slope* gives the quantity to add to sum_sl_r . It contains, if $kind = \text{Start}$, the slope of the trapezoid otherwise it contains the opposite of its slope.

Each non-null sub-task $mP(T_i)^j$ ⁹ of $mP(T_i)$ generates the pair of events:

$$\begin{aligned} &\langle i, \text{Start}, st_{mP(T_i)^j}, sh_{mP(T_i)^j}, sl_{mP(T_i)^j} \rangle \\ &\langle i, \text{End}, et_{mP(T_i)^j}, -eh_{mP(T_i)^j}, -sl_{mP(T_i)^j} \rangle \end{aligned}$$

where $sl_{mP(T_i)^j} = \frac{eh_{mP(T_i)^j} - sh_{mP(T_i)^j}}{d_{mP(T_i)^j}}$.

All the events are initially computed and sorted (by Main_Algorithm), in the list L_{events} , in increasing lexicographically order according to the pair $\langle kind, date \rangle$ where $kind = \text{end}$ is considered to be less than $kind = \text{start}$. Observe that one event participates to the construction of all $mcrP(r)$ such that $r \in a_{T_i}$.

sum_h_r and sum_sl_r are initially set to 0 (line 1 of SA) and δ to the date of the first event, associated to a task T_i such that $r \in a_{T_i}$, on the time axis (line 3 of SA). When the current position of Δ changes (line 5) from δ_k to δ_{k+1} , the sweep algorithm:

- First computes the k^{th} trapezoid sub-task $mcrP(r)^k$ of $mcrP(r)$ (line 6).
- Then (line 7) verifies that $mcrP(r)^k$ do not exceed the resource capacity C_r , otherwise the constraint has no solution.
- Then sum_h_r takes the value of the end height of the previous returned trapezoid sub-task ($mcrP(r)^k$) and δ is updated to δ_{k+1} (line 9).

In any case all the contributions (heights and slopes) of sub-tasks that start or end at δ_{k+1} are taken into account in sum_h_r and sum_sl_r (line 10).

Example 10. (continuation of Example 8) The sub-task $\langle 3, -1, 6, -1 \rangle$ of $mP(T_1)$ generates the pair of events ($e_1 = \langle 1, \text{Start}, 3, -1, 0 \rangle$, $e_2 = \langle 1, \text{End}, 6, 1, 0 \rangle$) while the trapezoid sub-task $\langle 1, -1, 8, -1 \rangle$ of $mP(T_4)$ generates the pair of events ($e_3 = \langle 4, \text{Start}, 1, -1, 0 \rangle$, $e_4 = \langle 4, \text{End}, 8, 1, 0 \rangle$). The sorted relevant events of L_{events} for computing $mcrP(2)$ from $mP(T_1)$ and $mP(T_4)$ are $\langle e_3, e_1, e_2, e_4 \rangle$. When e_1 becomes the current event then, the sweep line moves from position 1 to 3: the algorithm first computes the 1st trapezoid sub-task $\langle 1, -1, 3, -1 \rangle$ of $mcrP(2)$ (see Figure 4 Parts (F) and (G)). Then the sweep line status is updated: $sum_h_r \leftarrow -1$ and $\delta \leftarrow 3$. Finally, the contribution of e_1 is added to sum_h_r which decreases from -1 to -2 and to sum_sl_r which doesn't change.

⁹ $mP(T_i)^j = (st_{mP(T_i)^j}, sh_{mP(T_i)^j}, et_{mP(T_i)^j}, eh_{mP(T_i)^j})$ (start, startheight, end, end-height) denote the j^{th} trapezoid sub-task of the profiles $mP(T_i)$.

Sweep_Algorithm (SA) - Compute $mcrP(r)$ for a given resource r
In: The list L_{events} of all sorted events $\langle task, kind, date, height, slope \rangle$ and a resource r .
Out: Fail if $mcrP(r)$ exceed the resource capacity C_r ; Otherwise Delay.
<pre> 1: $k \leftarrow 1$; $sum_h_r \leftarrow 0$; $sum_sl_r \leftarrow 0$; 2: Extract, if it exists, the first event e from L_{events} such that $r \in a_{T_{e.task}}$; 3: $\delta \leftarrow e.date$; 4: while e is defined do 5: if $e.date \neq \delta$ then /* Δ has just move: Compute the k^{th} trapezoid of $mcrP(r)$ */ 6: $st \leftarrow \delta$; $sh \leftarrow sum_h_r$; $d \leftarrow e.date - \delta$; $eh \leftarrow sum_sl_r * (e.date - \delta) + sum_h_r$; 7: if $sh > C_r \vee eh > C_r$ then return Fail; /* $mcrP(r)^k$ exceeds C_r */ 8: $mcrP(r)^k \leftarrow (st, sh, e.date, eh)$; $k \leftarrow k + 1$; 9: $sum_h_r \leftarrow eh$; $\delta \leftarrow e.date$; /* Update sum_h_r and δ */ 10: $sum_h_r \leftarrow sum_h_r + e.height$; $sum_sl_r \leftarrow sum_sl_r + e.slope$; 11: Extract, if it exists, the next event e from L_{events} such that $r \in a_{T_{e.task}}$; 12: return Delay; </pre>

Main_Algorithm (MA) - Compute all $mcrP$
In: The tasks T_i ($i = 1..n$); Out: Fail if there is no solution otherwise return Delay.
<pre> 1: $L_{events} \leftarrow \emptyset$; 2: for $i = 1$ to n do /* For all the tasks */ 3: Compute $mP(T_i)$; 4: for $j = 1$ to $\ mP(T_i)\$ do /* Generate mP-events */ 5: if $(sh_{mP(T_i)^j} \neq 0 \vee eh_{mP(T_i)^j} \neq 0)$ then /* i.e., not a null sub-task */ 6: $sl = (eh_{mP(T_i)^j} - sh_{mP(T_i)^j}) / d_{mP(T_i)^j}$; 7: Add $\langle i, Start, st_{mP(T_i)^j}, sh_{mP(T_i)^j}, sl \rangle$ to L_{events}; 8: Add $\langle i, End, et_{mP(T_i)^j}, -eh_{mP(T_i)^j}, -sl \rangle$ to L_{events}; 9: Sort L_{events} in increasing lexicographically order according to $\langle date, kind \rangle$; 10: for $r = 1$ to q do if Sweep_Algorithm$(L_{events}, r) = Fail$ then return Fail; 11: return Delay; </pre>

The next result gives the complexities for computing all the minimum cumulated resource's profiles using a sweep algorithm.

Result 4 *The complexities for computing all the minimum cumulated resource's profiles using a sweep algorithm is $O(p \cdot (\log p + q))$ where q is the number of resources, $p = \sum_{i=1}^n (p_i)$ is the total number of trapezoid sub-tasks of the n tasks.*

Proof. of Result 4. The complexity for computing (see 3.4) all the $mP(T_i)$ and generating all the events is of $O(\sum_{i=1}^n (p_i \log p_i))$. The complexity for sorting the at most $2 \cdot p + n$ events, using a sort algorithm with a complexity in $O(p \log p)$, is $O(p \log p)$. Finally, for each resource r , the Sweep_Algorithm computes $mcrP(r)$ in $O(p)$ (in the worst case). Therefore, we obtain a complexity in the worst case of $O(\sum_{i=1}^n (p_i \log p_i) + p \cdot \log p + \sum_{i=1}^q p)$ i.e., $O(p \cdot (\log p + q))$. \square

5 Conclusion

We have introduced a new task model that takes advantages of two models: in the first one, several resources could be handled, but only a constant consumption or production of resource could be expressed. In the second one, only positive piecewise linear resource functions and a single resource were considered. For this new task model, we came up with a polynomial algorithm to compute, for each resource, its minimum and maximum cumulated resource's profiles.

References

1. Beldiceanu, N. and Carlsson, M. (2002). A New Multi-Resource cumulatives Constraint with negative heights. In: P. Van Hentenryck, ed. *8th Int. Conf. on Principles and Practice of Constraint Programming - CP'2002*, Ithaca, NY, USA, Sept. 8-13, 2002. Springer-Verlag: LNCS 2470, pp 63–79.
2. Caseau Y, Laburthe F (1996) Cumulative Scheduling with Task Intervals. In *Proceedings of the Joint International Conference and Symposium on Logic Programming*, The MIT Press.
3. Herroelen, W., Demeulemeester, E. and De Reyck, B. (1998). A Classification Scheme for Project Scheduling Problems. In: Weglarz J, ed., *Project Scheduling Recent Models, Algorithms and Applications*. Kluwer Academic Publishers, pp 1–26.
4. Klein R, Scholl A (1999) Computing lower bounds by destructive improvement: An application to resource-constrained project scheduling, *European Journal of Operational Research*, Vol 112, pp 322-346.
5. Lahrichi, A. (1982). Scheduling: the Notions of Hump, Compulsory Parts and their Use in Cumulative Problems. *C. R. Acad. Sci. Paris*, t. 294, pp 209–211.
6. Laborie, P. (2003). Algorithms for propagating resource constraints in AI planning and scheduling: Existing approaches and new results. *Artificial Intelligence*, Vol 143, pp 151–188.
7. Maravelias, C.T. and I. E. Grossmann, (2004). A Hybrid MILP/CP Decomposition Approach for the Continuous Time Scheduling of Multipurpose Batch Plants, *Computers & chemical engineering*, vol. 28 (10), pp. 1921–1949.
8. Muscettola, N. (2002). Computing the Envelope for Stepwise-Constant Resource Allocations. In: P. Van Hentenryck, ed. *8th Int. Conf. on Principles and Practice of Constraint Programming - CP'2002*, Ithaca, NY, USA, Sept. 8-13, 2002. Springer-Verlag: LNCS 2470, pp 139–153.
9. Poder, E., Beldiceanu, N., Sanlaville E. (2004). Computing a lower approximation of the compulsory part of a task with varying duration and varying resource consumption. *European Journal of Operational Research*, Vol. 153, pp 239–254.
10. Preparata, F. P. and Ian Shamos, M. (1995). *Computational Geometry, An introduction*. Texts and Monographs in Computer Science. Springer Verlag, New-York.
11. Sourd F and Rogerie J (2002) Continuous Filling and Emptying of Storage Systems in Constraint-Based Scheduling. *8th International Workshop on Project Management and Scheduling - MPS 2002*, Valencia, Spain, April 3-5.
12. Wolf, A. (2003). Filtering while Sweeping over Task Intervals. In: F. Ross, ed. *9th Int. Conf. on Principles and Practice of Constraint Programming - CP 2003*, Kinsale, Ireland. Sept./Oct. 2003. Springer-Verlag: LNCS 2833, pp 739–753.