

Filtering for a Continuous Multi-Resources *cumulative* Constraint with Resource Consumption and Production

Emmanuel Poder

Emmanuel.Poder@laposte.net

Nicolas Beldiceanu

École des Mines de Nantes, LINA UMR CNRS 6241,
4 rue Alfred Kastler, La Chantrerie, BP 20722
44307 Nantes Cedex 3, France.
Nicolas.Beldiceanu@emn.fr

Abstract

Within the framework of continuous and multi-resources cumulative constraints, a task T expresses a piecewise linear resource function and is represented by a sequence of p contiguous trapezoid sub-tasks with variable durations and heights. In this context, this paper provides an algorithm in $O(p)$ for filtering the resource assignment and the temporal attributes of such a task, according to a trapezoid of a minimum cumulated resource profile, in order to avoid an overflow of the resource capacity.

Introduction

The vast majority of existing models concerning resource-constrained scheduling deal with tasks (activities) that consume or produce a constant amount of resource during their execution (Herroelen, Demeulemeester and De Reyck, 1998). But tasks that consume or produce resource in a continuous way are ubiquitous in applications areas such as electricity or chemical production (Muscatola 2002), (Sourd and Rogerie 2005) and (Maravelias and Grossmann 2004).

Recently, in (Beldiceanu and Poder 2007) we have extended the task model proposed in (Poder, Beldiceanu and Sanlaville 2004) for modelling piecewise linear consumptions or productions and introduced the corresponding *cumulatives_pwl* constraint: a task is represented by a sequence of contiguous trapezoid sub-tasks with variable durations and heights (positive or negative) and to a task corresponds a set of possible resource assignments. They have given a sweep algorithm to compute for each resource its minimum and maximum cumulated resource's profile.

In this paper, we focus on the central sub-problem of filtering the resource assignment and the temporal attributes (start, duration, end of a task and duration of its sub-tasks) of a task according to one trapezoid of a minimum cumulated resource profile in order to avoid an overflow of the resource capacity. This filtering is absolutely required for implementing the *cumulatives_pwl* (Beldiceanu and Poder 2007) or the *cumulative_trapeze* (CHIP 2001) (Poder 2002) constraints and no result about filtering a sequence of trapezoid sub-tasks has been yet published. The main challenge

comes from the fact that all attributes of a task are variable. This filtering may be done while computing, with the sweep algorithm and for a given resource, the successive trapezoids building its minimum cumulated resource profile.

In this paper, we first recall the *cumulatives_pwl* constraint and the notion of minimum cumulated resource's profile. Then, we give the main applications of this constraint and we describe the contribution of this paper that is, given a task and a trapezoid of a minimum cumulated resource's profile, how to filter the task's attributes according to this trapezoid in order to avoid an overflow of the resource's capacity.¹ Finally, we conclude.

Background

We consider a set of q 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 on 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.

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

- The variables² s_{T_i} , td_{T_i} , and e_{T_i} represent respectively the start, the total duration and the end of task T_i .
- Seq_{T_i} is a sequence of p_i contiguous trapezoid sub-tasks $\langle T_i^1, T_i^2, \dots, T_i^{p_i} \rangle$ where T_i^j is defined by the triple $(sh_{T_i^j}, d_{T_i^j}, eh_{T_i^j})$ which represents its start height (i.e., the resource requirement at its start), duration and end height (i.e., the resource requirement at its end). An height may be positive or negative. Moreover, $d_{T_i^j} \geq 0$ and w.l.o.g., we assume that the two heights of a sub-tasks have the same signs.

¹In the context of both non negative heights and one single resource, a detailed version of the filtering of each attribute of a task (start, total duration, end) and of its sub-tasks (duration, heights) is available in (Poder 2002)).

²A variable v may range over the interval of consecutive integer $[\underline{v}, \bar{v}]$ or over the rational interval $[\underline{v}, \bar{v}]$. The results given in this paper may be used in the two cases. $[a, b]$ (resp. $[a, b[$) denotes the rational interval between a and b included (resp. b not included).

- The variable a_{T_i} represents the resource assignment of task T_i and takes its value in a finite set of integer values.

s_{T_i} is called the *release date* (earliest starting time), e_{T_i} the *due date* (latest ending time), and $[s_{T_i}, e_{T_i}]$ the *time window* of the task T_i . \bar{s}_{T_i} and \bar{e}_{T_i} are respectively the *latest starting time* and the *earliest finishing time* of T_i .

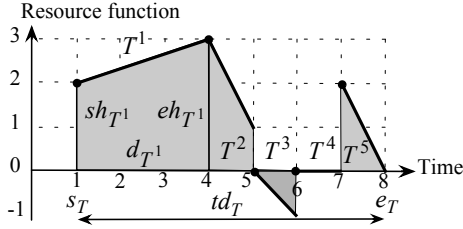


Figure 1: Example of fixed task

Example 1 Figure 1 illustrates a fixed task T defined by the quintuple $(s_T, td_T, e_T, Seq_T, a_T) = (1, 7, 8, \langle(2, 3, 3), (3, 1, 1), (0, 1, -1), (0, 1, 0), (2, 1, 0)\rangle, 1)$. T is made of 5 contiguous trapezoid sub-tasks and is assigned to resource 1.

Definition 2 The piecewise linear cumulative constraint has the form `cumulatives_pwl(Tasks, Resources)` 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 resource k .

Let $h_{T_i}(t)$ denote the resource's consumption of task T_i at instant t . 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 total duration),
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),
3. $(\forall k = 1..q, (\forall t \in \mathbb{R}) \sum_{i/a_{T_i}=k} h_{T_i}(t) \leq C_k$ (i.e., for each resource k and for each instant t , the cumulated consumptions of resource k , at instant t , of all the tasks assigned to the resource k , is less than or equal to C_k).

Definition 3 A feasible sequence of the sub-tasks of a task is such that all the durations and heights of those sub-tasks are fixed within their respective possible values and satisfy the constraint 2. of Definition 2.

A feasible schedule of a task is such that all its attributes are fixed within their respective possible values and satisfy the constraints 1. and 2. of Definition 2.

Example 2 Throughout this paper, 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:

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

For instance, T_1 starts within $[1..2]$, has a total duration within $[4..5]$ and ends within $[5..6]$. Its first sub-task has a start height within $[1..2]$, a duration within $[2..3]$ and a fixed end height. Its second one is fixed. Finally, T_1 may be assigned to resources 1 or 2. Part (I) of Fig. 2 depicts the four feasible sequences of the sub-tasks of T_1 (two of duration 4 and two of duration 5). As all sub-tasks of T_2, T_3 and T_4 are fixed, those sub-tasks have each only one feasible sequence that are respectively depicted in Part (I) of Fig. 2.

We now present the notions due to (Beldiceanu and Poder 2007) of *minimum profile* $mP(T)$ of a task T and of *minimum cumulated resource's profile* $mcrP(r)$ of a resource r . The former profile represents, for each time point, the smallest possible contribution of a task T to all the resources where T may be assigned. The latter profile, represents for each time point, the smallest possible contribution of all the tasks to the resource r . As we are interested by the smallest possible contribution of a task to all the resources where this task may be assigned:

- It is sufficient to consider that all heights of sub-tasks are fixed at their minimum respective value.
- A positive sub-task (i.e., its two heights are non-negative and not both null) of a task T is taken into account for the computation of $mP(T)$ if and only if T is assigned to a resource.
- A negative sub-task (i.e., its two heights are non-positive and not both null) of a task T is always taken into account for the computation of $mP(T)$.

Note that a task where all its sub-tasks are positive (resp. negative) is said to be positive (resp. negative).

Definition 4 The minimum profile $mP(T_i)$ of a task T_i is such that its resource function $h_{mP(T_i)}$ verifies : $\forall t, h_{mP(T_i)}(t) = \inf_{I \in \Phi(T_i)} h_I(t)$ where $\Phi(T_i)$ is the set of all the feasible schedules of task T_i .

The minimum cumulated profile $mcrP(r)$ of a resource r is such that its resource function $h_{mcrP(r)}$ verifies : $\forall t, h_{mcrP(r)}(t) = \sum_{T_i/r \in a_{T_i}} h_{mP(T_i)}(t)$.

In (Beldiceanu and Poder 2007), we explain how to compute, using a sweep algorithm, the minimum cumulated profiles³ of the q resources in $O(p \cdot (\log p + q))$ where $p = \sum_{i=1}^n p_i$ is the total number of trapezoid sub-tasks of the n tasks.

Example 3 (continuation of example 2) For the tasks T_1 to T_4 , Fig. 2 illustrates in part (II) on the first line the tasks (with their heights fixed at their minimum) at their earliest and latest schedules, on the second line, their minimum profile when they are not yet assigned to a resource (only negative trapezoids of tasks are taken into account for the computation of the minimum profiles) and finally, on the third line their minimum profile when they are assigned to a resource (both positive and negative trapezoids of tasks are taken into account for the computation of the minimum profiles). Fig.

³The exact profiles if the starts, durations and ends of all the tasks are rational variables and a lower approximation of this profile if they are integer variables.

2 illustrates in part (III) the minimum cumulated profiles of resources 1 and 2 when T_1 and T_4 may be assigned to the two resources (i.e., $a_{T_1} = a_{T_4} = \{1, 2\}$) and T_2 and T_3 are assigned to the resource 1 (i.e., $a_{T_2} = a_{T_3} = \{1\}$).

Main uses of the piecewise linear cumulative constraint

The piecewise linear cumulative constraint can be used :

- For multi-resources scheduling problem: Fig.3 shows a cumulative scheduling of the four tasks of Example 2 where the two resource capacities are equal to 2. An interpretation is, for instance, that a positive task represents a consumption over time of the resource where this task is assigned and that a negative task allows to increase locally the capacity of the renewable resource where this task is assigned.

Note that, if some tasks are optional, then we introduce an extra *dummy* resource with a huge capacity. Discarded tasks would be assigned to this dummy resource.

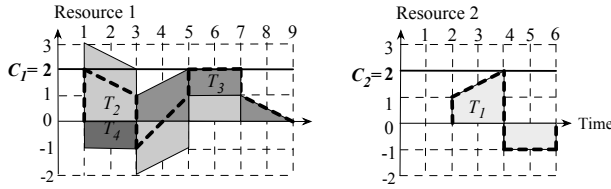


Figure 3: Example of Scheduling of the four tasks T_1 to T_4

- For producer/consumer problem: Given some activities that produce a resource and others that consume that resource, the problem is on one hand to schedule those activities and on the other hand to determine the produced and the consumed quantities in order that, at any time the available resource (stock i.e., what has been already produced minus what has been already consumed) is not negative. Some initial stock may exist. Moreover the production may be discrete (available at a given time point) or continuous (available as it is produced over a period of time). Similarly, the consumption may be discrete (required at a given time point) or continuous (required as it is consumed over a period of time)

The initial stock and the activities that produce the resource are represented by negative tasks and the activities that consume the resource are represented by positive tasks. Moreover, the capacity of the resource is equal to zero.

Fig. 4 illustrates examples of activities and their associated tasks (for the *cumulative_pwl* constraint), that produce or consume, over a period $[0, H[$, a resource in a discrete and in a continuous way. Part (A) shows on the first line an activity that produces the quantities q_1 and q_2 at times t_1 and t_2 and its associated negative task made of two rectangle tasks and, on the second line, an activity that produces the quantities q_3 and q_4 over the periods $[t_3, t_4[$ and $[t_4, t_5[$ and its associated negative task

made of two trapezoid sub-tasks (production periods) and two rectangle sub-tasks (idle periods). Similarly, Part (B) shows, on the first line, an activity that consume the quantities q_3 and q_4 at times t_6 and t_7 and its associated positive task made of two rectangle tasks and, on the second line an activity that produces the quantities q_5 and q_6 over the periods $[t_8, t_9[$ and $[t_{10}, t_{11}[$ and its associated positive task made of two trapezoid sub-tasks (consumption periods) and two rectangle sub-tasks (idle periods). Note that an initial stock of quantity q_0 would be represented by a rectangle over all the scheduling $[0, H[$ with height $(-q_0)$.

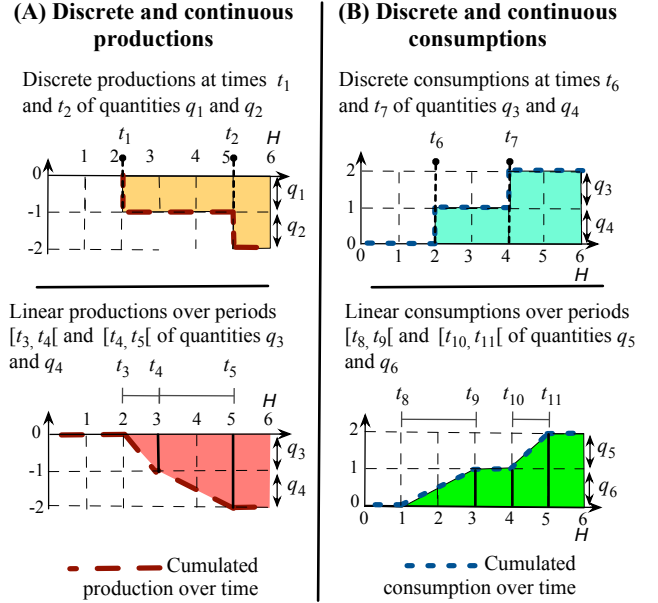


Figure 4: Producer/Consumer problem

Filtering a task according to a minimum cumulated resource's profile

Given a resource r , the sweep algorithm presented in (1) computes trapezoid after trapezoid the minimum cumulated resource profile $mcrP(r)$ of this resource. In the rest of the paper, we consider such a current computed trapezoid $trap_r$ and we explain how to filter each task T_i - that both is (i.e., $a_{T_i} = \{r\}$) or may be assigned (i.e., $r \in a_{T_i}$) to this resource and whose time windows $[s_{T_i}, \bar{e}_{T_i}[$ intersects the support (time window) of this trapezoid - according to $trap_r$ in order not to exceed the capacity C_r . The heights of the sub-tasks of T_i are fixed at their respective minimum value.

We use the following notations:

- $st_{T_i^j}$ (resp. $et_{T_i^j}$) $j = 1..p_i$ denotes the start (resp. the end) of the j^{th} sub-task of T_i . Note that $et_{T_i^j} = st_{T_i^{j+1}}$. The interval $[st_{T_i^j}, \bar{st}_{T_i^j}]$ of $st_{T_i^j}$ is determined while com-

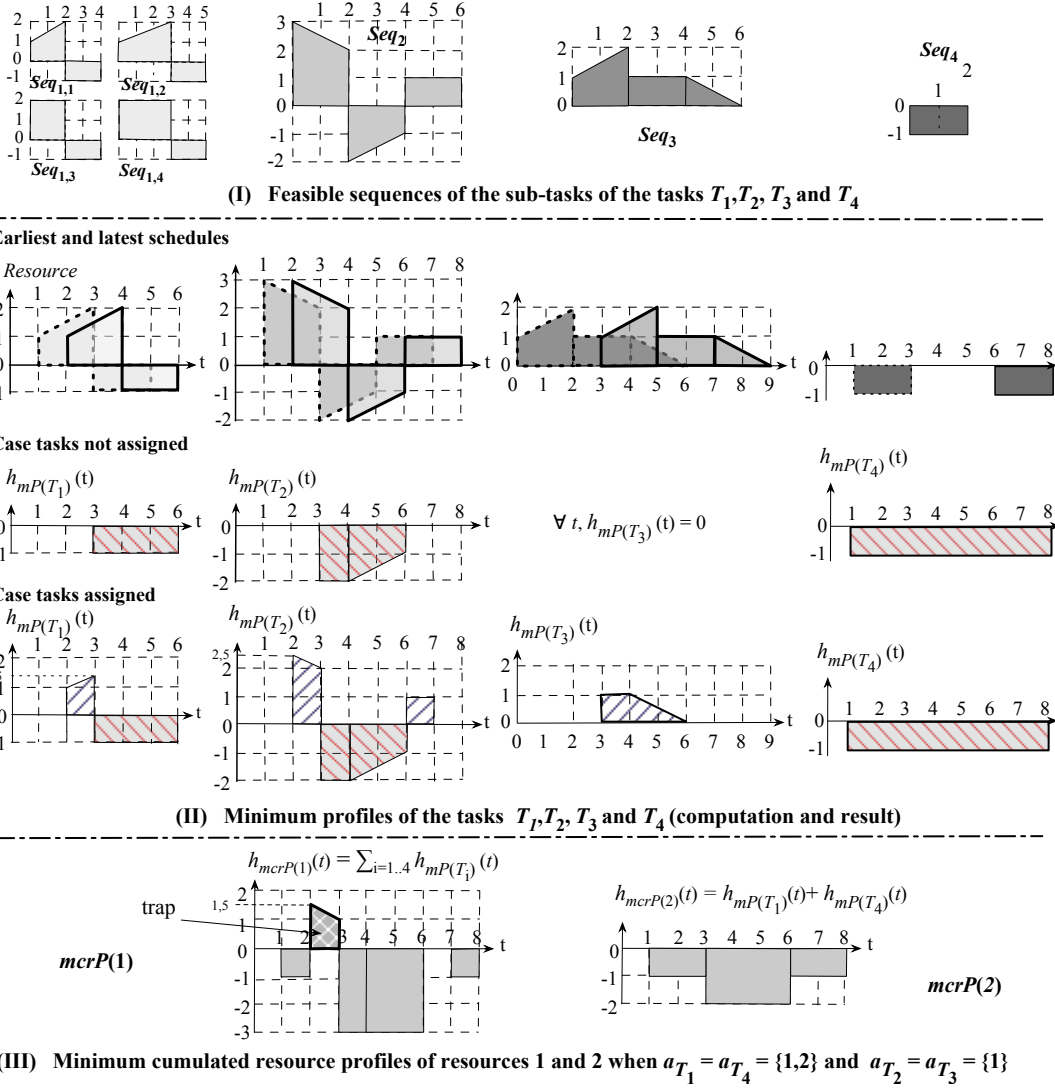


Figure 2: Feasible Sequences, minimum profiles and minimum cumulated resource's profiles

puting $mcrP(r)$.⁴

- Each trapezoid of $mcrP(r)$, of $mP(T_i)$ or of a sub-task T_i^j is represented by a quadruple $(start, startheight, end, endheight)$.
- $trap_r = (st, sh, et, eh)$ denotes the current computed trapezoid of $mcrP(r)$. Its support is $[st, et[$. Note that $st, sh, et, eh \in \mathbb{Q}$ and that $trap_r$ does not exceed the capacity C_r .
- If $mP(T_i)$ has a non-null contribution registered within

⁴Recursive formulae for computing $\underline{st}_{T_i^j}$ and $\overline{st}_{T_i^j}$ in $O(p_i)$ are given in (Poder, Beldiceanu and Sanlaville 2004): $\underline{st}_{T_i^1} = \underline{s}_{T_i}$, $\overline{st}_{T_i^1} = \overline{s}_{T_i}$, $\underline{st}_{T_i^{p+1}} = e_{T_i}$, $\overline{st}_{T_i^{p+1}} = \overline{e}_{T_i}$ and for all $j = 1..p_i$:
 $\underline{st}_{T_i^{j+1}} = \max(\underline{st}_{T_i^j} + \underline{d}_{T_i^j}, e_{T_i} - \sum_{k=j+1}^p \overline{d}_{T_i^k})$,
 $\overline{st}_{T_i^{j+1}} = \min(\overline{st}_{T_i^j} + \overline{d}_{T_i^j}, \overline{e}_{T_i} - \sum_{k=j+1}^p \underline{d}_{T_i^k})$.

$trap_r$ then $(st, h_{mP(T_i)}(st), et, h_{mP(T_i)}(et))$ ⁵ denotes this contribution.

- $trap_r/mP(T_i)$ denotes the trapezoid $trap_r$ where the contribution of $mP(T_i)$ has been removed i.e., the trapezoid $(st, sh - h_{mP(T_i)}(st), et, eh - h_{mP(T_i)}(et))$. Note that if $mP(T_i)$ has a negative contribution registered within $trap_r$ then $trap_r/mP(T_i)$ may exceed the resource capacity. In that case, T_i must be assigned to resource r .

Filtering of the resource assignment a_{T_i} of task T_i

If T_i may be assigned to the resource r then we filter the resource assignment of T_i according to $trap_r$ and C_r . We

⁵If we assume w.l.o.g., that during the construction of $mcrP(r)$, adjacent trapezoids of $mcrP(r)$ with same slopes and same adjacent heights are not merged, there exists at most one trapezoid of $mP(T_i)$ whose support contains $[st, et[$.

distinguish two cases:

- T_i has a known contribution already registered within $trap_r$: By definition of $mcrP(r)$, as T_i is not yet assigned to a resource, this contribution comes from one single negative trapezoid of $mP(T_i)$. If this contribution is absolutely required to avoid an overflow of the capacity C_r (i.e., if its removal leads to an overflow) then we assign T_i to r and enforce the time window of T_i to include the support of $trap_r$ i.e., we filter \overline{st}_{T_i} and \underline{et}_{T_i} so as to $\overline{st}_{T_i} \leq st$ and $et \leq \underline{et}_{T_i}$.
- There exists no registered contribution of T_i within $trap_r$: Remind that as T_i is not assigned to a resource, $mP(T_i)$ is built only from negative trapezoids of T_i . Let $mP^*(T_i)$ denote the minimum profile of T_i if T_i was assigned to the resource r . $mP(T_i)$ is built from positive and negative trapezoid of T_i . If there exists at least one positive trapezoid of $mP^*(T_i)$ whose addition⁶ to $trap_r$ leads to an overflow of C_r then we remove r from a_{T_i} .

Example 4 (continuation of Ex. 3) Let us now consider resource 1 with a capacity $C_1 = 2$ and $trap = (2, 3/2, 3, 1)$ (see Part (III) of Fig. 2). Only tasks T_1 and T_4 are not yet assigned to one of the resource (i.e., $a_{T_1} = a_{T_4} = \{1, 2\}$). Moreover their time windows intersect the support of $trap$ (i.e., $[\underline{st}_{T_i}, \overline{et}_{T_i}] \cap [2, 3] \neq \emptyset, i = 1..2$). So we filter a_{T_1} and a_{T_2} according to $trap$ and C_1 :

- T_1 has no contribution registered within $trap$. The addition of $(2, 1, 3, 1/2)$ (see the third line of Fig. 2 Part (II)) to $trap$ leads to an overflow at time 2 (this addition gives an height of $3/2 + 1 > 2$). So we remove 1 from a_{T_1} and now $a_{T_1} = 2$.
- T_4 has the contribution $(2, -1, 3, -1)$ registered within $trap$ and $a_{T_4} \in \{1, 2\}$. The removal of this contribution would lead again to an overflow at time 2. Consequently, T_4 is assigned to resource 1 and we enforce the time windows of T_4 to include the support of $trap$ i.e., we enforce $\overline{st}_{T_4} \leq st$ and $et \leq \underline{et}_{T_4}$. Hence, now $s_{T_4} \in [1..2]$ (instead of $[1..6]$) and $e_{T_4} \in [3..4]$ (instead of $[3..8]$).

Filtering of the temporal attributes of task T_i

If T_i is assigned to the resource r , we filter its temporal attributes:

- In a first phase, we browse all sub-tasks T_i^j of T_i such that their time windows intersect the support of $trap_r$ (i.e., $[\underline{st}_{T_i^j}, \overline{et}_{T_i^j}] \cap [st, et] \neq \emptyset$). For each such a sub-task T_i^j we compute the interval $IS_{T_i^j}$ of non-feasible values for its start (according to the trapezoid $trap_r/mP(T_i)$) and we filter $s_{T_i^j}$.
- In a second phase, these non-feasible values are propagated back to the origin, total duration, end of T_i as well as to the starts, durations and ends of all T_i^j . They are linked to the temporal attributes of T_i by the following linear constraints (LC):

⁶To determine, if the addition of two trapezoids leads to an overflow or not, we add their respective heights at the start and at the end of their two supports intersection.

- (1) $\sum_{j=1}^{p_i} d_j = td_{T_i}$, (see Definition 2)
- (2) $s_{T_i} + td_{T_i} = e_{T_i}$, (see Definition 2)
- (3) $s_{T_i} + d_{T_i^1} = s_{T_i^2}$, (first trapezoid sub-task T_i^1)
- (4) $\forall j = 2..p_i - 1 \ s_{T_i^j} + d_{T_i^j} = s_{T_i^{j+1}}$, (T_i^2 to $T_i^{p_i-1}$)
- (5) $s_{T_i^{p_i}} + d_{T_i^{p_i}} = e_{T_i}$. (last trapezoid sub-task $T_i^{p_i}$)

Constraint (1) is propagated using the standard incomplete filtering algorithm for linear constraints (Harvey, Stuckey and Peter, 2003). Constraints (2) to (5) of the form $X + Y = Z$ are propagated by using a complete filtering algorithm in order to be able to propagate back the holes created within the origin of one sub-task to the origin and the end of the corresponding task.⁷

In the rest of the paper we focus on the computation of the interval $IS_{T_i^j}$ of non-feasible values for the start of T_i^j (according to the trapezoid $trap_r/mP(T_i)$).

Filtering the starts $s_{T_i^j}$ of a sub-task T_i^j

We now explain how to filter, according to $trap_r = (st, sh, et, eh)$ and C_r the starts $s_{T_i^j}$ of sub-tasks T_i^j whose time windows $[\underline{st}_{T_i^j}, \overline{et}_{T_i^j}]$ intersect $[st, et]$. We assume (otherwise there is no filtering to do) that:

- T_i is assigned to the resource r (i.e., $a_{T_i} = r$),
- T_i^j cannot have a null duration (i.e., $d_{T_i^j} > 0$),
- Scheduling $trap_r$ and T_i^j in parallel can lead to be in excess of the resource capacity C_r (i.e., $\max(sh, eh) + \max(\underline{sh}_{T_i^j}, \underline{eh}_{T_i^j}) > C_r$).

Definition 5 A start s is non-feasible if whatever $d \in [\underline{d}_{T_i^j}..d_{T_i^j}]$ such that $s + d \in [e_{T_i^j}..e_{T_i^j}]$, there exists $t \in [s, s + d] \cap [st, et]$ such that the sum of the two trapezoids $trap_r/mP(T_i)$ and $(s, \underline{sh}_{T_i^j}, s + d, \underline{eh}_{T_i^j})$ leads to be in excess of the resource capacity at time t .

Notations Let $IS_{T_i^j} =]\gamma, \delta[$ denote the interval of non-feasible starts $s_{T_i^j}$ of T_i^j ($s_{T_i^j} \in \mathbb{Q}$) under the hypothesis that there is no constraint on the end $e_{T_i^j}$ of T_i^j (i.e., $\underline{e}_{T_i^j} = -\infty \wedge \overline{e}_{T_i^j} = +\infty$). Note that if $s_{T_i^j} \in \mathbb{Z}$ then $IS_{T_i^j} =]\lceil \gamma \rceil + 1.. \lfloor \delta \rfloor - 1[$.

We first explain how to transform the current problem of scheduling T_i^j in parallel with $trap_r/mP(T_i)$ without exceeding C_r in an equivalent non-overlapping problem. Using this new formulation, we first compute $IS_{T_i^j}$ for a fixed duration of T_i^j then, show how to extend it to the case where T_i^j has a variable duration.

⁷Since it may lead to a better evaluation of the earliest and latest starting time, earliest and latest finishing time of T_i as illustrated by the filtering of task T_3 in Example 5.

Transforming a cumulative scheduling problem involving two trapezoids into a particular non-overlapping problem

- *Transforming a cumulative scheduling problem involving two trapezoids not both positive or null into a cumulative scheduling problem involving two trapezoids positive or null* : Assume that we want to schedule in parallel two trapezoids $trap_1$ and $trap_2$ without exceeding a capacity C . First observe that, if the two trapezoids are not both positive or null (i.e., their two heights are non-negative) then we consider the equivalent problem of scheduling two positive or null trapezoids without exceeding another capacity C' where we transform the heights sh and eh of each non-positive or null trapezoid into respectively the positive heights $sh - \min(sh, eh)$ and $eh - \min(sh, eh)$ and the capacity in $C - \min(sh, eh)$.
- *Transforming a cumulative scheduling problem involving two positive or null trapezoids into a particular non-overlapping problem*: The problem of scheduling the positive trapezoid (or line-segment if T_i^j is a null trapezoid) $trap_2$ in parallel with the positive trapezoid (or line-segment) $trap_1$ without exceeding the capacity C is equivalent to the problem of translating $\overline{trap_2}$ (derived from $trap_2$) as represented by Fig. 5, parallel to the (0x)-axis in order that the two trapezoids $trap_1$ and $\overline{trap_2}$ don't overlap each other. Part (I) of Fig. 5 depicts the two trapezoids $trap_1$ and $trap_2$. Part (II) (resp. (III)) presents a feasible (resp. non-feasible) scheduling instance and the corresponding placement instance.

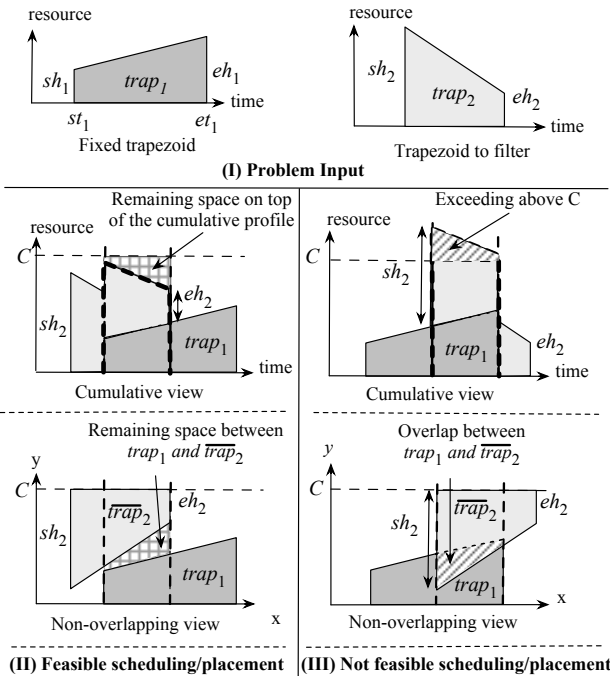


Figure 5: Cumulative and non-overlapping views of the problem of filtering the origin of a trapezoid sub-task according to a fixed trapezoid

Computing $IS_{T_i^j}$ when the duration of T_i^j is fixed Let $trap_1$ (derived from $trap_r/mP(T_i)$), $trap_2$ (derived from T_i^j) and C (derived from C_r) denote the transformed trapezoids and capacity used in the placement problem. To determine the impossible origins of $trap_2$ with a fixed duration d_2 , we move $trap_2$, as shown by Fig. 6 Part (I), parallel to the Ox-axis and we obtain an interval $]\delta(d_2), \gamma(d_2)[$ of values where $trap_2$ overlaps $trap_1$. All the different cases are given by Table 1 and illustrated by Fig. 6 Parts (II) and (III): given the two trapezoids $trap_1$ and $trap_2$, Part (I) (resp. (II)) enumerates all respective positions of $trap_1$ and $trap_2$ associated with their earliest (resp. latest) possible intersection. Within Table 1, Δ_1 (resp. Δ_2) denotes the line-segment corresponding to the slope of $trap_1$ (resp. $trap_2$). In order to significantly reduce the number of cases to consider, a dotted trapezoid denotes the fact that the slope of the corresponding trapezoid may be positive or negative.

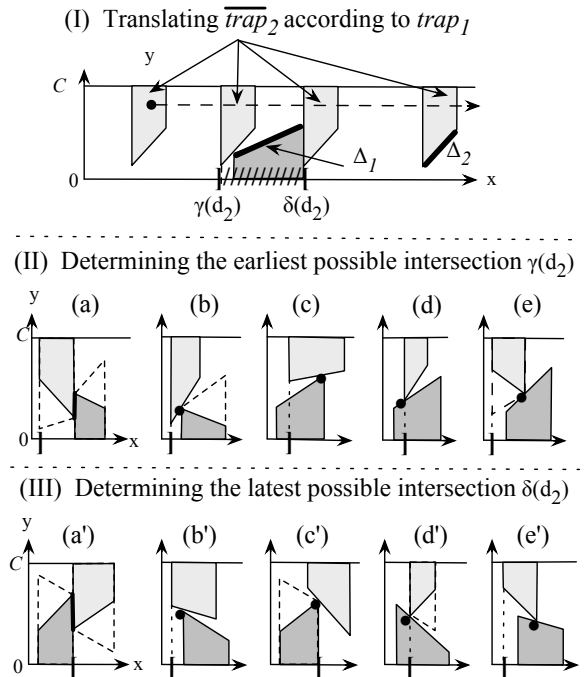


Figure 6: (I) Interval of impossible starts of $trap_2$, (II), (III) All respective positions of $trap_1$ and $trap_2$ associated with their earliest (resp. latest) intersection

Computing $IS_{T_i^j}$ when the duration of T_i^j is variable

When the duration of T_i^j is not already fixed, $IS_{T_i^j} = \bigcap_{d \in [d_{T_i^j}, \bar{d}_{T_i^j}]}]\gamma(d), \delta(d)[$. By considering all the cases of Table 1, we can verify that $\gamma(d)$ is maximum for $d = \underline{d}_{T_i^j}$ (as $\gamma(d)$ is a constant or a decreasing function of d_2) and that $\delta(d)$ is minimum for $d = \bar{d}_{T_i^j}$ (as $\delta(d)$ is a constant or an increasing function of d_2). Note that when d is increasing or decreasing we can switch from a case to another one. So

$$IS_{T_i^j} =]\gamma(\underline{d}_{T_i^j}).. \delta(\bar{d}_{T_i^j})[$$

Table 1: Conditions and values of γ and δ for a given duration of $trap_2$ associated to the respective positions of $trap_1$ and $trap_2$ showed by Fig. 6 Parts (II) and (III) ($sl_1 = (eh_1 - sh_1)/(et - st)$, $sl_2 = (\underline{eh}_{T_i^j} - \underline{sh}_{T_i^j})/d_2$)

Case	sl_1	sl_2	$sl_1 + sl_2$	sh_1, eh_1, sh_2, eh_2	$\gamma(d_2)$
(a) eh_2 meets sh_1	-	-	-	$eh_1 + sh_2 \geq C$	$st - d_2$
(b) sh_1 meets Δ_2	-	< 0	≤ 0	$sh_1 + sh_2 \geq C$	$st - (C - sh_2 - sh_1)/sl_2$
(c) eh_1 meets Δ_2	> 0	< 0	> 0	$eh_1 + eh_2 \leq C$	$et - (C - sh_2 - eh_1)/sl_2$
(d) sh_2 meets Δ_1	> 0	< 0	≤ 0	$sh_1 + sh_2 < C$	$st + (C - sh_2 - sh_1)/sl_1$
(e) eh_2 meets Δ_1	> 0	-	> 0	$eh_1 + eh_2 > C$	$st + (C - eh_2 - sh_1)/sl_1 - d_2$

Case	sl_1	sl_2	$sl_1 + sl_2$	sh_1, eh_1, sh_2, eh_2	$\delta(d_2)$
(a') eh_2 meets sh_1	-	-	-	$eh_1 + sh_2 \geq C$	et
(b') sh_1 meets Δ_2	< 0	> 0	< 0	$sh_1 + sh_2 < C$	$st - (C - sh_2 - sh_1)/sl_2$
(c') eh_1 meets Δ_2	-	> 0	≥ 0	$eh_1 + eh_2 > C$	$et - (C - sh_2 - eh_1)/sl_2$
(d') sh_2 meets Δ_1	< 0	-	< 0	$sh_1 + sh_2 \geq C$	$st + (C - sh_2 - sh_1)/sl_1$
(e') eh_2 meets Δ_1	< 0	> 0	≥ 0	$eh_1 + eh_2 \leq C$	$st + (C - eh_2 - sh_1)/sl_1 - d_2$

Table 2: **Avoid_Trap_Overflow Algorithm (ATO)**

Avoid_Trap_Overflow($i, (st, sh, et, eh), C$)	
Filters all $st_{T_i^j}$ according to (st, sh, et, eh) and C ;	
1:	$max = \max(sh, eh)$; if $max_{T_i} + max \leq C$ then return Delay;
2:	$j \leftarrow 1$; /* Transform (if needed) (st, sh, et, eh) and C : */
3:	if $(sh \geq 0 \wedge eh \geq 0)$ then $sh_1 \leftarrow sh$; $eh_1 \leftarrow eh$; /* (st, sh, et, eh) is positive or null */
4:	else $m \leftarrow \min(sh, eh)$; $sh_1 \leftarrow sh - m$; $eh_1 \leftarrow eh - m$; $C \leftarrow C - m$;
5:	while $j \leq p_i$ do /* Iterates through the p_i sub-tasks of T_i */
6:	if $\underline{d}_{T_i^j} > 0 \wedge [st, et[\cap [\underline{st}_{T_i^j}, \overline{st}_{T_i^{j+1}}[\neq \emptyset \wedge max + \max(\underline{sh}_{T_i^j}, \underline{eh}_{T_i^j}) > C$
7:	then /* Transform (if needed) T_i^j , then compute $IS(s_{T_i^j})$ and filters $s_{T_i^j}$ */
8:	if $(\underline{sh}_{T_i^j} \geq 0 \wedge \underline{eh}_{T_i^j} \geq 0)$ then $sh_2 \leftarrow \underline{sh}_{T_i^j}$; $eh_2 \leftarrow \underline{eh}_{T_i^j}$; /* T_i^j is positive or null */
9:	else $m \leftarrow \min(\underline{sh}_{T_i^j}, \underline{eh}_{T_i^j})$; $sh_2 \leftarrow \underline{sh}_{T_i^j} - m$; $eh_2 \leftarrow \underline{eh}_{T_i^j} - m$; $C \leftarrow C - m$;
10:	$sl_1 \leftarrow (eh_1 - sh_1)/(et - st)$;
11:	$sl_2 \leftarrow (eh_2 - sh_2)/\underline{d}_{T_i^j}$; /* Compute $\gamma(\underline{d}_{T_i^j})$: */
12:	if $eh_2 + sh_1 \geq C$ then /* case (a) */ $\gamma \leftarrow st - \underline{d}_{T_i^j}$ else
13:	if $sl_1 + sl_2 \leq 0$ then /* case (b) or (d) */
14:	if $sh_1 + sh_2 \geq C$ then $\gamma \leftarrow st - (C - sh_2 - sh_1)/sl_2$; /* case (b) */
15:	else $\gamma \leftarrow st + (C - sh_2 - sh_1)/sl_1$; /* case (d) */
16:	else /* case (c) or (e) */
17:	if $eh_1 + eh_2 \leq C_r$ then $\gamma \leftarrow et - (C - sh_2 - eh_1)/sl_2$; /* case (c) */
18:	else $\gamma \leftarrow st + (C - eh_2 - sh_1)/sl_1 - \underline{d}_{T_i^j}$; /* case (e) */
19:	$sl_2 \leftarrow (eh_2 - sh_2)/\overline{d}_{T_i^j}$; /* Compute $\delta(\overline{d}_{T_i^j})$: */
20:	if $eh_1 + sh_2 \geq C$ then /* case a' */ $\delta \leftarrow et$ else
21:	if $sl_1 + sl_2 < 0$ then /* case (b') or (d') */
22:	if $sh_1 + sh_2 < C$ then $\delta \leftarrow st - (C - eh_2 - sh_1)/sl_2$; /* case (b') */
23:	else $\delta \leftarrow st + (C - sh_2 - sh_1)/sl_1 - \overline{d}_2$; /* case (d') */
24:	else /* case (c') or (e') */
25:	if $eh_1 + h_2 > C$ then $\delta \leftarrow et - (C - sh_2 - eh_1)/sl_2$; /* case (c') */
26:	else $\delta \leftarrow st + (C - sh_2 - sh_1)/sl_1 - \overline{d}_{T_i^j}$; /* case (e') */
27:	if $\gamma \leq \delta$ then if Remove ($s_{T_i^j}, [[\gamma] + 1, [\delta] - 1]$) = Fail then return Fail;
28:	$j \leftarrow j + 1$;
29:	return Delay;

Example 5 (continuation of Ex. 4) We now detail the filtering of tasks T_2 , T_3 and T_4 according to the second trapezoid $trap = (2, 11/6, 3, 2/3)$ of $mcrP(1)$ (see Fig. 2 Part (III)) in order not to exceed the capacity $C_1 = 2$ (T_1 was previously assigned to resource 1 so is not considered here):

- T_2 : As the superposition of T_2 and $trap/mP(T_2) = (2, -1, 3, -1)$ can never exceed the capacity ($maxh_{T_2} + maxh_{trap/mP(T_2)} = 5/2 - 1 \leq C_1 = 2$)⁸ there is no filtering to perform.
- T_3 : As $trap/mP(T_3) = mcrP(1)^2$ and $maxh_{T_3} + maxh_{mcrP(1)^2} > C_1$ so we filter T_3 according to $trap$. We have $st_{T_3^1} = st_{T_3} \in [0..3]$, $st_{T_3^2} \in [2..5]$, $st_{T_3^3} \in [4..6]$ and $st_{T_4^3} = e_{T_3} \in [6..9]$ and $d_{T_3^1} = d_{T_3^2} = d_{T_3^3} = 2$.
 - T_3^1 : As $IS(T_3^1) \in [1..2]$ (cases (a) and (a')), we remove $[1..2]$ from st_{T_3} that is now in $\{0, 3\}$. Then, by propagating constraints (LC) we get $st_{T_3} \in \{0, 3\}$, $st_{T_3^2} \in \{2, 5\}$, $st_{T_3^3} \in \{4, 7\}$ and $st_{T_4^3} = e_{T_3} \in \{6, 9\}$.
 - T_3^2 : As $IS(T_3^2) \in [1..2]$ (cases (a) and (d')) we remove 2 from $st_{T_3^2}$ that becomes 5. Then, by propagating constraints (LC), we get $st_{T_3^1} = st_{T_3} = 3$, $st_{T_3^2} = 5$, $st_{T_3^3} = 7$ and $st_{T_4^3} = e_{T_3} = 9$.
 - T_3^3 : As $et_{trap} < st_{T_3^3}$ no more filtering occurs.
- T_4 : We filter T_4^1 according to $trap/mP(T_4) = (2, 5/2, 3, 2)$ and $C_1 = 2$. This is equivalent to the placement problem of $trap_1 = (2, 5/2, 3, 2)$ and a null line-segment $trap_2$ of duration 2 with $C = 3$. We obtain no filtering.

Complexities

Result 1 Given a resource r , a computed trapezoid $trap$ of $mcrP(r)$ and a task T_i with p_i trapezoid sub-tasks, the complexity of filtering the resource assignment and the starts of the sub-tasks of T_i according to $trap$ and the capacity C_r is in the worst case $O(p_i)$. Then, the filtering of all the tasks according to all the minimum cumulated resources' profiles may be processed in $O(q \cdot p^2)$ where p is the total number of trapezoid sub-tasks of all the tasks and q the number of resources.

Proof 1 of Result 1. The complexity of the algorithm *Avoid_Trap_Overflow* (a) and the complexity for filtering the resource assignment (b) are both in worst case in $O(P_i)$. In fact, the algorithm *Avoid_Trap_Overflow* browses all the sub-tasks of T_i and for each relevant sub-task makes operations in $O(1)$ then it has a complexity in the worst case of $O(p_i)$. If T_i is assigned to the resource r , the filtering of the resource assignment is in $O(1)$ and if T_i may be assigned to the resource then then this filtering is in the worst case in $O(p_i)$. Each time a trapezoid ($O(p)$ trapezoids) of the minimum cumulated profile of this resource is built, then (a) and (b) are applied for each task that both may be assigned to

⁸ $maxh_T$ contains the value of the maximum over all minimum heights of T .

this resource and whose time windows intersects this trapezoid so a complexity at most of $O(\sum_{r=1}^q (p \cdot \sum_{i=1}^n p_i))$ i.e., $O(q \cdot p^2)$. □

Conclusion

This paper provides the first algorithm for filtering in a systematic way the attributes of a task, made of a sequence of trapezoid sub-tasks with variable durations and heights, according to a fixed trapezoid in order to respect a resource capacity. Future research may consist in filtering such a task according to task intervals. Observe that the problem of computing the exact minimum intersection of such a task with an interval is an open question.

References

- Beldiceanu, N. and Poder, E. 2007. A continuous Multi-Resources cumulative Constraint with Positive / Negative Resource Consumption/ Production. *CP-AI-OR'07*, May 23-26, 2007 - Brussels, Belgium. Springer: LNCS 4510, pp 214–228.
- CHIP Reference Manual Version 5, 2001, COSYTEC SA, France.
- Harvey, W., and Stuckey, Peter J. 2003. Improving Linear Constraint Propagation by Changing Constraint Representation. *Constraints*, Vol 8 (2), pp 173–207.
- 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.
- 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.
- Muscettola, N. 2002. Computing the Envelope for Stepwise-Constant Resource Allocations. *CP'2002*, Ithaca, NY, USA, Sept. 8-13, 2002. Springer: LNCS 2470, pp 139–153.
- Poder, E. 2002. Programmation Par Contraintes et Ordonnement de Tâches avec Consommation variable de ressource. Thèse de Doctorat. Université Blaise Pascal - Clermont Ferrand II.
- Poder, E., Beldiceanu N. and Sanlaville E. 2004. Computing a lower approximation of the compulsory part of a task with varying duration and varying resource consumption. *EJOR*, Vol. 153, pp 239–254.
- Sourd F and Rogerie J 2005. Continuous Filling and Emptying of Storage Systems in Constraint-Based Scheduling. *EJOR*, Vol 165, pp 510–524.