Une contrainte cumulative continue multi-ressources avec des conso mmations - Productions en ressources Positives - Négatives

Nicolas Beldiceanu, Emmanuel Poder

Une contrainte cumulative continue multi-ressources avec des consommations - Productions en ressources Positives - Négatives

Nicolas Beldiceanu **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

Résumé

Cet article introduit une extension de la contrainte *cumulative* classique : une tâche n'est plus représentée par un simple rectangle mais par une suite de sous-tâches trapézoïdales de durées et hauteurs variables. La fonction de ressource n'est plus une constante, mais une fonction du temps, linéaire par morceaux, positive ou négative. Enfin, une tâche n'est pas pré-affectée à une ressource mais à une tâche correspond un ensemble d'affectations possibles. Dans ce contexte, cet article propose un algorithme en $O(p \cdot (\log p + q))$ pour calculer les profils minimum et maximum d'utilisation des ressources par les tâches où q est le nombre de ressources et p le nombre total de sous-tâches de toutes les tâches.

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 $O(p \cdot (\log p + q))$ algorithm for computing all the cumulated resource profils where q is the number of resources and p is the total number of trapezoid sub-tasks of all the tasks.

1 Introduction

La plupart des travaux sur l'ordonnancement de projet sous contraintes de ressources (problème RCPSP) sont liés à des problèmes où les tâches utilisent des quantités constantes de ressources durant toute leur exécution [3]. Cependant, dans de nombreux problèmes pratiques, les profils de consommations ou de productions en ressources des tâches sont plus complexes et varient dans le temps de façons discrètes (besoin en main d'œuvre) ou continues (électricité, pétrole, produits chimiques) [8], [6], [11] et [7].

Une première contribution de cet article est un modèle de tâche qui réunie les deux modèles introduits dans [1] et [9]. Chaque tâche n'est plus seulement un rectangle mais une séquence continue de sous-tâches trapézoïdales. La durée positive et les hauteurs positives ou négatives d'une sous-tâche trapézoïdale sont variables dans des intervalles et expriment une fonction de ressource linéaire par morceaux positive ou négative. Une tâche est non–préemptive, et nécessite pour son exécution d'être affectée à exactement une ressource. A chaque tâche correspond un ensemble d'affectations possibles. Enfin, une tâche doit démarrer pendant un intervalle de temps donné (date de disponibilité). De même, elle doit finir pendant un intervalle de temps donné.

La principale contribution de cet article est un algorithme efficace en $O(p \cdot (\log p + q))$ pour calculer, pour chaque ressource, une estimation inférieure et une estimation supérieure de son profil d'utilisation final où q est le nombre de ressources et p est le nombre total de sous-tâches trapézoïdales de toutes les tâches. Ces deux estimations seront appelées respectivement *profil cumulé de ressource minimum* et *maximum*. Le profil cumulé de ressource minimum permet de vérifier facilement qu'un ordonnancement partiel (i.e., un or-

donnancement où les variables des tâches et des sous-tâches ne sont pas encore toutes fixées) n'est pas réalisable par rapport au fait qu'on ne doive pas dépasser une capacité donnée de ressource.

De même, le profil cumulé de ressource maximum permet de vérifier qu'on atteint, sur des intervalles de temps spécifiques, un niveau minimum d'utilisation de la ressource. De plus, ces deux profils coïncident quand toutes les variables des tâches sont toutes fixées. Le profil cumulé minimum d'une ressource est construit à partir de la somme de tous les profils minimum des tâches, où le *profil minimum d'une tâche* représente, à chaque instant, la plus petite contribution possible de la tâche au profil de ressource.[1] Pour une tâche donnée T avec des hauteurs qui peuvent être positives et négatives, le calcul de son profil minimum de tâche est plus complexe que de seulement déterminer une partie obligatoire[5]. En fait, on commence par fixer les hauteurs de chaque sous-tâches de T à leur valeur minimum et on considère alors séparément la suite de ses sous-tâches positives T^+ et la suite de ses sous-tâches négatives T^-[2]. Pour la première, nous calculons sa *partie obligatoire (compulsory part)* [9] (i.e., l'intersection de tous les ordonnancements réalisables de T^+) et pour l'autre suite son *enveloppe* (i.e., l'union de tous les ordonnancements réalisables de T^-) et on les recombine. Notons que cette méthode est basée sur la propriété que la partie obligatoire de T^+ et l'enveloppe de T^- ne se superposent pas dans le temps.[3]

Ce papier est organisé comme suit : La section 2 introduit la contrainte linéaire par morceaux *cumulatives_pwl* et le modèle de tâche associé. La section 3 présente différentes contributions possibles d'une tâche à l'utilisation d'une ressource : la partie obligatoire, l'enveloppe, et finalement, les profils minimum et maximum de tâche. La section 4 définie d'abord, pour chaque ressource, ses profils cumulés de ressources minimum et maximum. Ensuite, elle décrit un algorithme de balayage pour les calculer à partir de tous les profils minimum et maximum. Tous les algorithmes donnés sont polynomiaux en le nombre total de sous-tâches trapézoïdales.

2 La contrainte linéaire par morceaux cumulatives_pwl

Nous considérons un ensemble de q ressources renouvelables où la k^{eme} ressource à une capacité maximum $C_k \geq 0$ et un ensemble de n tâches non–preemptives. Chaque tâche requière pour son exécution d'être affectée à exactement une ressource parmi un sous-ensemble donné des q ressources (ce sous-ensemble dépend de la tâche considérée). Finalement, chaque tâche est composée d'une séquence de sous-tâches trapézoïdales contiguës qui expriment une fonction de ressource linéaire par morceaux. Dans ce contexte, cette section introduit le modèle de tâche ainsi que le contrainte correspondante.

2.1 Modèle de tâche

Après avoir introduit le modèle de tâche utilisé tout au long de cet article, cette section présente la notion d'instance réalisable d'une tâche ainsi que la notion de fonction de ressource.

Définition 1 *Une tâche T_i est définie par le quintuple $(s_{T_i}, td_{T_i}, e_{T_i}, Seq_{T_i}, a_{T_i})$ où :*
- *Les variables*[4] *s_{T_i}, td_{T_i} et e_{T_i} représentent respectivement le* début*, la* durée totale *et la* fin *de T_i.*
- *Seq_{T_i} est une séquence de p_i sous-tâches trapézoïdales contiguës $\langle T_i^1, T_i^2, \cdots, T_i^{p_i}\rangle$ où la sous-tâche trapézoïdale T_i^j $(j = 1..p_i)$ est définie par les variables $sh_{T_i^j}$, $d_{T_i^j}$ ($\underline{d}_{T_i^j} \geq 0$) et $eh_{T_i^j}$ qui représentent respectivement la* hauteur au début *de T_i^j (i.e., la quantité de ressource nécessaire à son début), la* durée *de T_i^j et la* hauteur en fin *de T_i^j (i.e., le besoin en ressource à sa fin).*
- *La variable a_{T_i} représente la* ressource d'affectation *de la tâche T_i et prend sa valeur dans un ensemble fini d'entiers $dom(a)$.*

$\underline{s}_{T_i}$ est appelée la *date de disponibilité* (début au plus tôt) tandis que $\overline{e}_{T_i}$ est la *date de délivrance* (date de fin au plus tard) de T_i. $\overline{s}_{T_i}$ et $\underline{e}_{T_i}$ sont respectivement la *date de début au plus tard* et la *date de fin au plus tôt* de T_i.

Exemple 1 *La tâche fixée T, définie par le quintuple $(s, td, e, Seq, a) = (1, 7, 8, \langle(2,3,3),(3,1,1),(0,1,-1),(0,1,0),(3,1,0)\rangle, 1)$, commence à l'instant 1, a une durée totale de 7, se termine à l'instant 8 et est constituée de 5 sous-tâches. De plus, T est affectée à la ressource 1 et sa fonction de ressource linéaire par morceaux h_T est définie par : $h_T(t) = 1/3 \cdot (t-1) + 2$ pour $t \in [1,4[$, $h_T(t) = -2 \cdot (t-4) + 3$ pour $t \in [4,5[$, $h_T(t) = -(t-5)$ pour $t \in [5,6[$, $h_T(t) = 0$ pour $t \in [6,7[$ et $h_T(t) = -2 \cdot (t-7) + 2$ pour $t \in [7,8[$.*

[1] Le profil cumulé maximum d'une ressource et le profil maximum d'une tâche sont définis de manière similaire.

[2] Une sous-tâche dont une des hauteurs minimum est positive est l'autre négative peut être divisée en deux sous-tâches, l'une positive et l'autre négative.

[3] Le calcul du profil maximum d'une tâche s'opère de façon similaire.

[4] Une variable v prend ses valeurs dans l'intervalle d'entiers consécutifs $[\underline{v}..\overline{v}]$. Dans cet article, $[a..b]$ désigne l'intervalle des entiers compris entre a et b inclus tandis que $[a, b]$ (resp. $[a, b[$) désigne l'intervalle des rationnels compris entre a inclus et b inclus (resp. b non inclus).

Les définitions 2 et 3 introduisent, pour une tâche T, les notions de séquence réalisable des sous-tâches trapézoïdales de T et d'instance réalisable de T.

Définition 2 *Etant données une tâche T_i définie par $(s_{T_i}, td_{T_i}, e_{T_i}, Seq_{T_i}, a_{T_i})$ et une durée totale $td \in [\underline{td}_{T_i}.. \overline{td}_{T_i}]$, une* séquence réalisable des sous-tâches de T_i de durée totale *td est telle que toutes les variables $sh_{T_i^j}$, $d_{T_i^j}$, $eh_{T_i^j}$ $(j = 1..p)$ sont fixées dans leur domaine respectif et satisfont $\sum_{j=1}^{p_i} d_{T_i^j} = td$.*

Définition 3 *Etant données une tâche T_i définie par $(s_{T_i}, td_{T_i}, e_{T_i}, Seq_{T_i}, a_{T_i})$ et une séquence réalisable Seq des sous-tâches de T_i,* une instance réalisable *de T_i est telle que, d'une part, s_{T_i}, td_{T_i} et e_{T_i} sont fixées dans leur domaine de respectif et vérifient $s_{T_i} + td_{T_i} = e_{T_i}$ et, d'autre part, Seq_{T_i} est fixée à Seq. Nous notons Φ_{T_i} l'ensemble de toutes les instances réalisables de la tâche T_i.*

Exemple 2 *Tout au long de cet article, nous considérons l'exemple suivant composé des quatre tâches T_1, T_2, T_3 et T_4 définies comme suit :*

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

Tous les débuts et fins des tâches T_1 à T_4 sont non fixés. La première sous-tâche de T_1 a sa hauteur en début et sa durée non fixées. T_1 et T_4 sont non encore affectées. Fig. 1 fournit toutes les séquences réalisables des sous-tâches des tâches T_1 à T_4. Notons que T_1 a quatre séquences réalisables (deux de durée 4 ($Seq_{1,1}$ et $Seq_{1,3}$) et deux de durée 5 ($Seq_{1,2}$ et $Seq_{1,4}$)) tandis que les autres tâches n'ont seulement chacune qu'une seule séquence réalisable. Finalement, Fig. 2 fournit les instances réalisables ($I_{1,k}$ $(k = 1..6)$) de la tâche T_1.

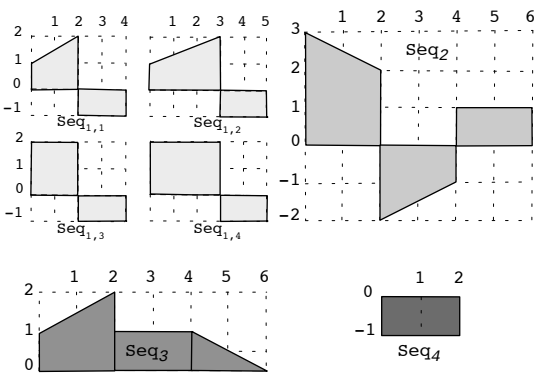

FIG. 1 – Séquences réalisables des sous-tâches trapézoïdales des tâches T_1 à T_4.

Définition 4 *Nous notons $st_{T_i^j}$ le début de T_i^j $(st_{T_i^1} = s_{T_i})$. A chaque $I \in \Phi_{T_i}$ est associée une* fonction de ressource *$h_I(t)$ définie sur $[s_{T_i}, e_{T_i}[$ par : $\forall\ t \in [s_{T_i}, e_{T_i}[, \exists! j \in \{1, 2, ..., p_i\}$ tel que $t \in [st_{T_i^j}, st_{T_i^j} + d_{T_i^j}[$. Alors $h_I(t) = \frac{eh_{T_i^j} - sh_{T_i^j}}{d_{T_i^j}} \left(t - st_{T_i^j}\right) + sh_{T_i^j}$ et $sl_{T_i^j} = \frac{eh_{T_i^j} - sh_{T_i^j}}{d_{T_i^j}}$ est la* valeur de la pente *de la sous-tâche T_i^j. Si Φ_{T_i} est réduit à une seule instance (i.e., T_i est fixée) alors nous notons h_{T_i} la fonction de ressource de T_i.*

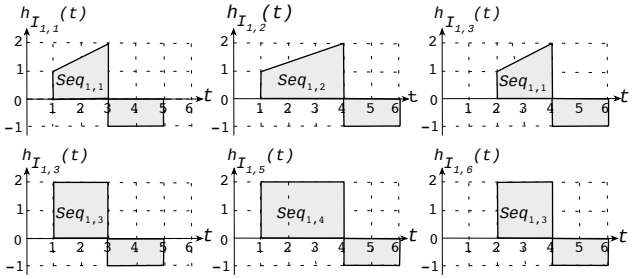

FIG. 2 – Instances réalisables de la tâche T_1.

Exemple 3 *(suite de l'exemple 2) L'instance réalisable $I_{1,1}$ (voir Fig. 2) associée à T_1 est $(1, 4, 5, \langle(1, 2, 2), (-1, 2, -1)\rangle, \{1, 2\}) = (1, 4, 5, Seq_{1,1}, \{1, 2\})$. Sa fonction de ressource est définie par $h_I(t) = 1/2 \cdot (t - 1) + 1$ pour $t \in [1, 3[$ et $h_T(t) = -1$ pour $t \in [3, 5[$.*

La prochaine définition introduit la notion de sous-tâche trapézoïdale positive, négative, nulle et indéterminée ainsi que la notion de tâche positive et négative.

Définition 5 *Une sous-tâche trapézoïdale T_i^j de T_i est* positive *(resp.* négative*) si ses deux hauteurs sont positives i.e., $\underline{sh}_{T_i^j} \geq 0 \wedge \underline{eh}_{T_i^j} \geq 0$ (resp. négatives i.e., $\overline{sh}_{T_i^j} \leq 0 \wedge \overline{eh}_{T_i^j} \leq 0$) et non toutes les deux nulles. Elle est* nulle *si $\underline{sh}_{T_i^j} = \overline{sh}_{T_i^j} = \underline{eh}_{T_i^j} = \overline{eh}_{T_i^j} = 0$. Autrement, elle est* indéterminée. *Une tâche T_i est* positive *(resp.* négative*) si les hauteurs de toutes ses sous-tâches sont positives (resp. négatives).*

Exemple 4 *(suite de l'exemple 3) T_1^1 est une sous-tâche trapézoïdale positive tandis que T_1^2 est négative. T_3 est une tâche positive tandis que T_4 est négative.*

2.2 La contrainte cumulative linéaire par morceaux cumulatives_pwl

La *contrainte cumulative linéaire par morceaux* est de la forme *cumulatives_pwl*(*Taches, Ressources, Contrainte*) où :

– *Taches* est une collection de tâches $\langle T_1, T_2, \cdots, T_n\rangle$ où la tâche T_i $(i = 1..n)$ est définie par le quintuple $(s_{T_i}, td_{T_i}, e_{T_i}, Seq_{T_i}, a_{T_i})$.

– *Ressources* est un ensemble de q entiers $C_1, C_2, \cdots, C_q$ où $C_k \geq 0$ $(k = 1..q)$ est la capacité de la k^{eme} ressource si *Contrainte* $=' \leq'$ ou le niveau minimum de la k^{eme} ressource si *Contrainte* $=' \geq'$.
– *Contrainte* est la contrainte plus petit ou égal (i.e., $\leq$) ou plus grand ou égal (i.e., $\geq$).

La contrainte *cumulatives_pwl* est vérifiée si les conditions suivantes sont vraies :

1. $\forall\, i = 1..n$, $s_{T_i} + td_{T_i} = e_{T_i}$ (i.e., la fin de la tâche est égale à la somme de son début et de sa durée totale),
2. $\forall\, i = 1..n$, $\sum_{j=1}^{p_i} d_j = td_{T_i}$ (i.e., la durée totale de la tâche est égale à la somme des durées de toutes ses sous-tâches trapézoïdales),
3. Cas *Contrainte* $= ' \leq'$: $(\forall\ k = 1..q)$, $(\forall\ t)$, $\sum_{i/a_{T_i}=\{k\}} h_T(t) \leq C_k$ (i.e., pour chaque ressource k et pour chaque instant t, la somme des valeurs prises à l'instant t par les fonctions de ressource des tâches qui sont affectées à cette ressource k, est plus petite ou égale à la capacité C_k de cette ressource k).
   Cas *Contrainte* $= ' \geq'$: $(\forall\, k = 1..q)$, $(\forall\, t/(\exists h/t \in [s_{T_h}, e_{T_h}[))$, $\sum_{i/a_{T_i}=\{k\}} h_T(t) \leq C_k$ (i.e., pour chaque ressource k et pour chaque instant t tel que au moins une tâche est exécutée à cet instant, la somme des valeurs prises à t par les fonctions de ressource des tâches qui sont affectées à cette ressource k, est plus grande ou égale au niveau minimum C_k de cette ressource k).

Exemple 5 *(suite de l'exemple 2) La contrainte* 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)*) $\rangle$

$\langle 2, 2\rangle$ *, ≤) est vérifiée et le résultat est illustré par la Partie (A) de la Fig. 3 pour la ressource 1 et la Partie (B) pour la ressource 2 (les utilisations des ressources sont indiquées par une ligne pointillée).*

3 Profils de tâche minimum et maximum

Cette section montre comment calculer, pour une tâche donnée, ses profils minimum et maximum. Pour ce faire, nous commençons par rappeler (3.1 et 3.2) quelques résultats, démontrés dans [9], pour calculer la partie obligatoire d'une tâche et ensuite nous les étendons au calcul de l'enveloppe (3.3).[5] Puis, dans la section 3.4, nous expliquons comment utiliser ensemble les notions de partie obligatoire et d'enveloppe pour calculer les profils minimum et maximum d'une tâche. Pour ce faire, nous avons besoin d'introduire d'abord deux instances réalisables spécifiques de T qui sont ses ordonnancements au plus tôt et au plus tard.

[5]Dans le cas d'une tâche positive, sa partie obligatoire et son enveloppe peuvent respectivement être interprétées comme une estimation inférieure et supérieure de la tâche (à chaque instant ils fournissent les hauteurs minimum et maximum de la tâche).

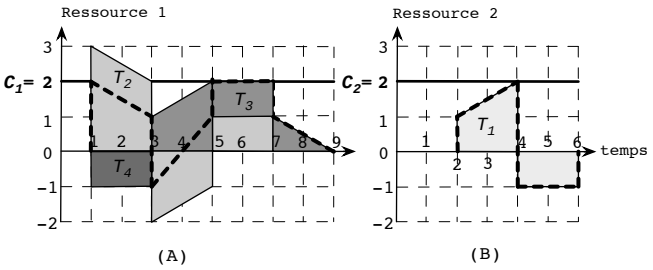

FIG. 3 – Exemple d'un ordonnancement réalisable des tâches T_1 à T_4.

3.1 Ordonnancement au plus tôt et au plus tard de la tâche T

Soient T^{min} et T^{max} respectivement, les ordonnancements au plus tôt et au plus tard de T et $\underline{st}_{T^j}$ et $\overline{st}_{T^j}$ $(j = 1..p)$ respectivement les débuts de la sous-tâche trapézoïdale T^j dans les ordonnancements T^{min} et T^{max}. Poder *et al.* ont montré dans [9] comment calculer $\underline{st}_{T^j}$ et $\overline{st}_{T^j}$ avec une compléxité en $O(p)$. Pour obtenir T^{min}, la tâche T commence à sa date au plus tôt i.e., $st_{T^1} = \underline{s}_T$; puis, $st_{T^2}, st_{T^3}, \cdots, st_{T^{p+1}}$ $(st_{T^{p+1}} = e_T)$ sont fixées successivement les plus petites possibles en respectant la réalisabilité de la contrainte de fin de la tâche. La formule récursive suivante calcule $\underline{st}_{T^j}$ $(j = 1..p+1)$:

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

T^{max} est obtenue de façon similaire i.e., :

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

Notons que $\underline{st}_{T^{p+1}} = \underline{e}_T$ et $\overline{st}_{T^{p+1}} = \overline{e}_T$.

3.2 Partie Obligatoire d'une tâche positive

La *partie obligatoire* a été initialement introduite par Lahrichi [5], pour une tâche rectangle comme l'intersection de tous les instances réalisables d'une tâche. Au fur et à mesure que les domaines des variables de la tâche se réduisent, la partie obligatoire augmente jusqu'à devenir une instance réalisable de la tâche. Pour des tâches rectangle, [4] utilise la notion de partie obligatoire pour serrer des bornes inférieures pour des problèmes d'ordonnancement sous contraintes de ressource, tandis que [2] utilise la partie obligatoire dans

un ensemble de règles de propagation pour résoudre des contraintes cumulative.
Ici, nous considérons une tâche positive T. Néanmoins, le calcul de la partie obligatoire d'une tâche négative est symétrique : en fait, si T est définie, pour tout t, par $h_T(t) = -h_{T'}(t)$ alors, pour tout t, $h_{PO(T')}(t) = -h_{PO(T)}(t)$.
Notons qu'il suffit, pour le calcul de la partie obligatoire de T, de ne considérer seulement que les instances réalisables de T où les hauteurs sont minimum. Par conséquent, par la suite, une tâche T a ses hauteurs fixées à leur minimum.

Définition 6 *La partie obligatoire $PO(T)$ de la tâche T est non vide si et seulement si $\overline{s}_T < \underline{e}_T$. Sa fonction de ressource $h_{PO(T)}$ vérifie pour tout $t \in [\overline{s}_T, \underline{e}_T[$ $h_{PO(T)}(t) = inf_{I \in \Phi_T}(h_I(t))$ et $h_{PO(T)}(t) = 0$ ailleurs.*

Afin de calculer la partie obligatoire d'une tâche, nous devons d'abord introduire la notion de vallée dans une séquence de sous-tâches trapézoïdales et la tâche associée appelé Tâche des Vallées.

Définition 7 *Soit $H_T^{min} = h_1 h_2 \cdots h_{2p} = \underline{sh}_{T^1}\underline{eh}_{T^1}\underline{sh}_{T^2}\underline{eh}_{T^2}\cdots\underline{sh}_{T^p}\underline{eh}_{T^p}$ la séquence de toutes les hauteurs minimum des débuts et fins des sous-tâche trapézoïdales de T. La hauteur $h_k \in H_T^{min}$ $(1 < k < 2p)$ définie* une fin de vallée *si et seulement si $\exists\ j$ $(1 < j \leq k)$ tel que $h_{j-1} > h_j$ et $h_j = h_{j+1} = \cdots = h_k$ et $h_k < h_{k+1}$ (une diminution stricte dans la fonction de ressource est suivie par une augmentation stricte).*

Résultat 1 *Soit $h_{j_1} h_{j_2} \cdots h_{j_v}$ la sous-séquence (peut être vide) de H_T^{min} de toutes les hauteurs des sous-tâches trapézoïdales de T qui correspondent aux fins des v vallées. Soit $TV(T)$, appelée* Tâche des Vallées *de T, la tâche définie pour tout $t \in [\overline{s}_T, \underline{e}_T[$ par $h_{TV(T)}(t) = \min\left(+\infty, \left\{h_{j_k} \ / \ t \in \left[\underline{st}_{T^{\lfloor \frac{j_k}{2} \rfloor + 1}}, \overline{st}_{T^{\lfloor \frac{j_k}{2} \rfloor + 1}}\right[\right\}\right)$*
Alors, $PO(T) = T^{min} \bigcap T^{max} \bigcap TV(T)$ i.e., $\forall t \in [\overline{s}_T, \underline{e}_T[$, $h_{PO(T)}(t) = \min\left(h_{T^{min}}(t),\ h_{T^{max}}(t),\ h_{TV(T)}(t)\right)$ et $h_{PO(T)}(t) = 0$ ailleurs. Si T n'a pas de vallée ($v = 0$) alors, $\forall t \in [\overline{s}_T, \underline{e}_T[$ $h_{PO(T)}(t) = \min\left(h_{T^{min}}(t), h_{T^{max}}(t)\right)$.

Exemple 6 *Fig. 4 illustre le calcul de $PO(T)$ où T a deux fins de vallée (voir Partie (A)). Elles correspondent au début st_{T^2} de T^2 ($h_1 = \underline{sh}_{T^2}$) et à la fin st_{T^4} de T^3 ($h_2 = \underline{eh}_{T^3}$). Les Parties (A) et (B) donnent respectivement T^{min} et T^{max}. Ensuite, la Partie (C) montre $TV(T)$ qui est definie par $h_{TV(T)}(t) = +\infty$ pour $[\overline{s}_T, \underline{st}_{T^2}[$, $h_{TV(T)}(t) = h_1$ pour $t \in [\underline{st}_{T^2}, \underline{st}_{T^4}[$, et $h_{TV(T)}(t) = h_2$ pour $t \in [\underline{st}_{T^4}, \overline{st}_{T^2}[\bigcup[\overline{st}_{T^2}, \underline{e}_T[$ (puisque $h_1 > h_2$). Finalement, la Partie (D) montre le calcul de $PO(T)$ comme l'intersection de T^{min}, T^{max} et $TV(T)$.*

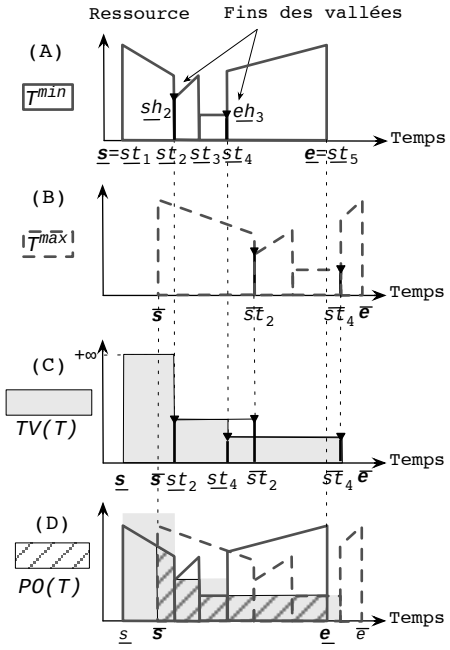

FIG. 4 – Calcul de la partie obligatoire d'une tâche.

Complexité du calcul de la partie obligatoire. Dans [9], Poder *et al.* fournissent un algorithme pour calculer la partie obligatoire d'une tâche, composée de p sous-tâches trapézoïdales et $v > 0$ vallées, en $O(p + v \cdot log v)$. En fait, T^{min} et T^{max} sont calculées en $O(p)$ (i.e., la complexité pour calculer les deux séquences $\underline{st}^j$ et $\overline{st}^j (j = 1..p+1)$), $TV(T)$ est calculée en utilisant une structure de tas en $O(v \cdot \log v)$. Ensuite, l'intersection de T^{min}, T^{max} et $TS(T)$ est obtenue, en les parcourant en parallèle, en $O(p + v)$. Ainsi, on obtient une complexité en $O(p + v \cdot \log v)$ c'est à dire en $O(p \log p)$. Si $v = 0$ alors la complexité est en $O(p)$.

3.3 Enveloppe d'une tâche positive

L'*enveloppe* d'une tâche est l'union de tous les instances réalisables de la tâche. Au fur et à mesure que les domaines des variables de la tâche se réduisent, l'enveloppe diminue jusqu'à devenir une instance réalisable de la tâche. Dans le contexte de ressources multiples, une tâche a la même enveloppe sur chaque ressource où elle peut être potentiellement affectée et une enveloppe vide ailleurs. Ici, nous considérons une tâche positive T et nous calculons son enveloppe $Env(T)$. Néanmoins, le calcul de l'enveloppe d'une tâche où les hauteurs de chaque sous-tâche trapézoïdale sont négatives est symétrique. En fait, si T est définie,

pour tout t, par $h_T(t) = -h_{T'}(t)$ alors, pour tout t, $h_{Env(T')}(t) = -h_{Env(T)}(t)$.

Définition 8 *L'enveloppe $Env(T)$ de la tâche T est telle que sa fonction de ressource $h_{Env(T)}$ satisfasse $h_{Env(T)} = sup_{I\in\ \Phi_T}(h_I(t))$ pour tout $t \in\ [\underline{s}_T, \overline{e}_T[$ et $h_{Env(T)}(t) = 0$ ailleurs.*

Notons qu'il est suffit, pour le calcul de l'enveloppe de T, de ne considérer seulement que les instances réalisables de T où les hauteurs de chaque sous-tâche trapézoïdale sont fixées à leur maximum. Donc, dans cette section, une tâche T a ses hauteurs fixées à leur maximum.

Comme nous avons introduit, pour le calcul la partie obligatoire d'une tâche, la notion de vallée dans une séquence de trapézoïdales, nous introduisons maintenant, pour le calcul de l'enveloppe, la notion de sommet et la tâche associée appelée la Tâche des Sommets.

Définition 9 *Soit $H_T^{max} = h_1h_2\cdots h_{2p} = \overline{sh}_{T^1}\overline{eh}_{T^1}\overline{sh}_{T^2}\overline{eh}_{T^2}\cdots\overline{sh}_{T^p}\overline{eh}_{T^p}$ la séquence des hauteurs maximum de tous les débuts et fins des sous-tâches trapézoïdales de T. La hauteur $h_k \in\ H_T^{max}(1 \le k \le 2p)$ définie un* fin de sommet *si et seulement si $\exists\ j(1 \le j \le k)$ tel que $h_{j-1} < h_j$ et $h_j = h_{j+1} = \cdots = h_k$ et $h_k > h_{k+1}$ avec la convention que $h_0 = h_{2p+1} = 0$.*

Résultat 2 *Soit $h_{j_1}h_{j_2}\cdots h_{j_w}$ la sous-séquence de HT^{max} de toutes les hauteurs de T qui sont des sommets. La* Tâche des Sommets *$TS(T)$ de T est la tâche définie par :*
$\forall\ t \in [\underline{s}_T, \overline{e}_T[, h_{TV(T)}(t) = \max\left(0, \left\{h_{j_k}\ \ t \in \left[\underline{st}_{T^{\lfloor\frac{j_k}{2}\rfloor+1}}, \overline{st}_{T^{\lfloor\frac{j_k}{2}\rfloor+1}}\right[\right\}\right)$
*(avec la convention que $s_{T_p+1} = e$). L'*enveloppe *de T est obtenue par l'union de T^{min}, T^{max} et $TS(T)$ i.e., $h_{Env(T)}$ vérifie $h_{TS(T)}(t) = sup(h_{T^{min}}(t), h_{T^{max}}(t), h_{TS(T)}(t))$ pour tout $t \in\ [\underline{s}_T, \overline{e}_T[$ et $h_{TS(T)}(t) = 0$ ailleurs.*

La démonstration du Résultat 2 est similaire à celle du Résultat 1 et est par conséquent omise.

Exemple 7 *Fig. 5 illustre le calcul de l'enveloppe de la tâche T qui a trois fins de sommet de hauteurs $h_1 = \overline{sh}_{T^1}$, $h_2 = \overline{eh}_{T^2}$ et $h_3 = \overline{eh}_{T^4}$ (voir Partie (A)). Ils correspondent au début $s = st_{T^1}$ de la tâche, à la fin st_{T^3} de la seconde sous-tâche et à la fin $e = st_{T^4}$ de la tâche T. Les Partie (A) et (B) donnent respectivement T^{min} et T^{max}. Ensuite, la Partie (C) montre $TS(T)$ qui est définie par $h_{TS(T)}(t) = h_1$ pour $t \in\ [\underline{s}_T, \overline{s}_T[$, $h_{TV(T)}(t) = h_2$ pour $t \in\ [\underline{st}_{T^3}, \underline{e}_T[$ (puisque $h_2 < h_3$ sur $[\underline{e}_T, \overline{st}_{T^3}[$), et $h_{TV(T)}(t) = h_3$ pour $t \in\ [\underline{e}_T, \overline{e}_T[$. Finalement, la partie (D) montre le calcul de $Env(T)$ comme l'union de T^{min}, T^{max} et $TS(T)$.*

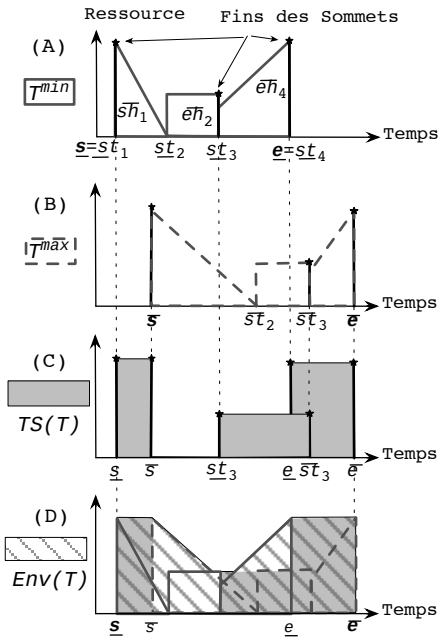

FIG. 5 – Calcul de l'enveloppe d'une tâche.

Complexité du calcul de l'enveloppe d'une tâche. Une tâche T a toujours au moins un sommet sauf si T vérifie $\forall\ t, h_T(t) = 0$. L'algorithme du calcul de l'enveloppe est similaire à celui donné pour le calcul de la partie obligatoire. Il a, par conséquent, une complexité en $O(p+w\cdot\log w)$ c'est à dire en $O(p\log p)$.

3.4 Profils minimum et maximum d'une tâche T

Dans cette section, nous nous plaçons dans le cadre de multi-ressources et définissons la notion de profils minimum et maximum d'une tâche et comment les obtenir à partir des parties obligatoire et des enveloppes.

Définition 10 *Soit $Pm(T)$ (resp. $PM(T)$) le* profil minimum *(resp.* maximum*) de la* tâche *T. Sa fonction de ressource $h_{Pm(T)}$ (resp. $h_{PM(T)}$ satisfait, pour tout $t \in [\underline{s}_T, \overline{e}_T[$, $h_{Pm(T)}(t) = inf_{I\in\ \Phi_T}(h_I(t))$ (resp. $h_{PM(T)}(t) = sup_{I\in\ \Phi_T}(h_I(t))$.*

Notons qu'il suffit, pour le calcul de $Pm(T)$ (resp. $PM(T)$) de ne considérer seulement que les instances réalisables où toutes les hauteurs des sous-tâches trapézoïdales de T sont fixées à leur minimum (resp. maximum).

Nous supposons donc maintenant que toutes les hauteurs de T sont fixées à leur minimum (resp. maximum) respectif pour le calcul de $Pm(T)$ (resp. $PM(T)$).

Soit T^+ (resp. T^-) la tâche obtenue à partir de T en remplaçant chaque sous-tâche trapézoïdale négative

(resp. positive) T^j de T par une sous-tâche nulle de même durée que la sous-tâche remplacée (nous fusionnons des sous-tâches consécutives nulles en une seule sous-tâche). Une sous-tâche T^j telle que ses deux hauteurs minimum $\underline{sh}^j$ et $\underline{eh}^j$ sont de signes différents peut être divisée en deux sous-tâches, l'une positive et l'autre négative. La première avec une hauteur en début égale à $\underline{sh}^j$ et une hauteur en fin nulle et la deuxième avec une hauteur au début nulle et une hauteur en fin égale à $\underline{eh}^j$.

Le prochain résultat exprime la fonction de ressource du profil minimum (resp. maximum) de la tâche T en fonction de celles de la partie obligatoire de T^+ (resp. T^-) et de l'enveloppe de T^- (resp. T^+).

Résultat 3 *Soit t un instant donné.*
Si T est affectée à une ressource alors, sur cette ressource

$$\begin{cases} h_{Pm(T)}(t) = h_{PO(T^+)}(t) + h_{Env(T^-)}(t), \\ h_{PM(T)}(t) = h_{PO(T^-)}(t) + h_{Env(T^+)}(t). \end{cases}$$

sinon, sur chaque ressource où T peut être affectée

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

De plus, $h_{PO(T^+)}(t) \neq 0 \wedge h_{Env(T^-)}(t) \neq 0$ et $h_{Env(T^+)}(t) \neq 0 \wedge h_{PO(T^-)}(t) \neq 0$.

Preuve 1 du Résultat 3

Puisque les définitions de $Pm(T)$ et $PM(T)$ sont symétriques, nous ne prouvons le résultat que pour $Pm(T)$. Notons que, par définition de T^+ et T^-, pour une instance $I \in \Phi_T$, les instances associées I^+ et I^- sont telles que $h_{I^+}(t) = \max(0, h_I(t))$ et $h_{I^-}(t) = \min(0, h_I(t))$. Par conséquent $h_{PO(T^+)}(t) = \inf_{I \in \Phi_T}(h_{I^+}(t))$ et $h_{Env(T^-)}(t) = \inf_{I \in \Phi_T}(h_{I^-}(t))$.

Nous distinguons deux cas : $\forall I \in \Phi_T, h_I(t) > 0$ (1) et son contraire $\exists I \in \Phi_T / h_I(t) \leq 0$ (2). Dans le premier cas, $\forall I \in \Phi_T, h_I(t) = h_{I^+}(t)$. Alors $\inf_{I \in \Phi_T}(h_I(t)) = \inf_{I \in \Phi_T}(h_{I^+}(t)) > 0$ i.e., $h_{Pm(T)}(t) = h_{PO(T^+)}(t) > 0$. De plus, $\forall I \in \Phi_T, h_{I^-}(t) = 0$ et donc $h_{Env(T^-)}(t) = 0$. Dans le deuxième cas : $\inf_{I \in \Phi_T}(h_I(t)) = \inf_{I \in \Phi_T}(h_{I^-}(t)) > 0$. i.e., $h_{Pm(T)}(t) = h_{Env(T^-)}(t)$. De plus, $\inf_{I \in \Phi_T}(h_{I^+}(t)) = 0$ i.e $h_{PO(T^+)}(t) = 0$.

Nous avons prouvé qu'on ne peut avoir en même temps $h_{PO(T^+)}(t) \neq 0$ et $h_{Env(T^-)}(t) \neq 0$ (par définition $h_{PO(T^+)}(t) \geq 0$ et $h_{Env(T^-)}(t) \leq 0$) et que si $h_{PO(T^+)}(t) > 0$ alors $h_{Pm(T)}(t) = h_{PO(T^+)}(t)$ sinon $h_{Pm(T)}(t) = h_{Env(T^-)}(t)$. □

Complexité du calcul des profils de tâche minimum et maximum. Puisque la complexité pour calculer la partie obligatoire ou l'enveloppe d'une tâche avec p sous-taches est $O(p \log p)$, le résultat 3 nous permet de déduire que la complexité pour calculer le profil minimum ou maximum est aussi en $O(p \log p)$.

Exemple 8 *(suite de l'exemple 2) Puisque T_1 et T_4 peuvent être affectées aux ressources 1 et 2 ($a_{T_1} = a_{T_4} = \{1, 2\}$), seulement leur enveloppe sont prises en compte et donc $h_{Pm(T_1)} = h_{Env(T_1^-)}$ et $h_{Pm(T_4)} = h_{Env(T_4)}$ ($T_4^- = T_4$ car T_4 est négative). Puisque T_2 et T_3 sont affectée à ressource 1 ($a_{T_2} = a_{T_3} = \{1\}$), leur partie obligatoire et leur enveloppe sont prises en compte donc $h_{Pm(T_2)} = h_{PO(T_2^+)} + h_{Env(T_2^-)}$ et $h_{Pm(T_3)} = h_{PO(T_3)}$ ($T_3^+ = T_3$ et $T_3^- = \emptyset$ puisque T_3 est positive). Les Parties (A) à (D) de Fig. 6 donnent respectivement $Pm(T_1)$, $Pm(T_2)$, $Pm(T_3)$ et $Pm(T_4)$. Finalement, le tableau suivant résume les différent profils (une sous-tâche est décrite par $\langle$ début, hauteur au début, fin, hauteur en fin $\rangle$).*

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

4 Calcul des profils cumulés de ressource minimum et maximum en utilisant un algorithme de balayage

Cette section commence par définir les notions de profils cumulés de ressource minimum et maximum. Ensuite, il décrit un algorithme de balayage polynomial (en fonction du nombre de sous-tâches et de ressources) pour calculer, pour chaque ressource r, son profil cumulé de ressource minimum $Pcm(r)$. Ce profil est calculé à partir de tous les profils de tâche $Pm(T_i)$ $i = 1..n$ de la contrainte *cumulatives_pwl*.

4.1 Profils de ressource cumulés minimum et maximum

Nous montrons maintenant comment calculer les profils cumulés de ressource minimum et maximum à partir des profils minimum et maximum des tâches.

Définition 11 *Etant donnée une ressource r, $Pcm(r)$ (resp. $PcM(r)$) désigne le* profil cumulé minimum (resp. maximum) *de la ressource r. Pour tout t, $h_{Pcm(r)}(t) = \sum_{T_i / r \in a_{T_i}} h_{Pm(T)}(t)$ (resp. $h_{PcM(r)}(t) = \sum_{T_i, r \in a_{T_i}} h_{PM(T)}(t)$).*

Au fur et à mesure que les domaines des variables de toutes les tâches sont progressivement réduits (jusqu'à devenir fixés), $Pcm(r)$ et $PcM(r)$ respectivement augmente et diminue (jusqu'à devenir un

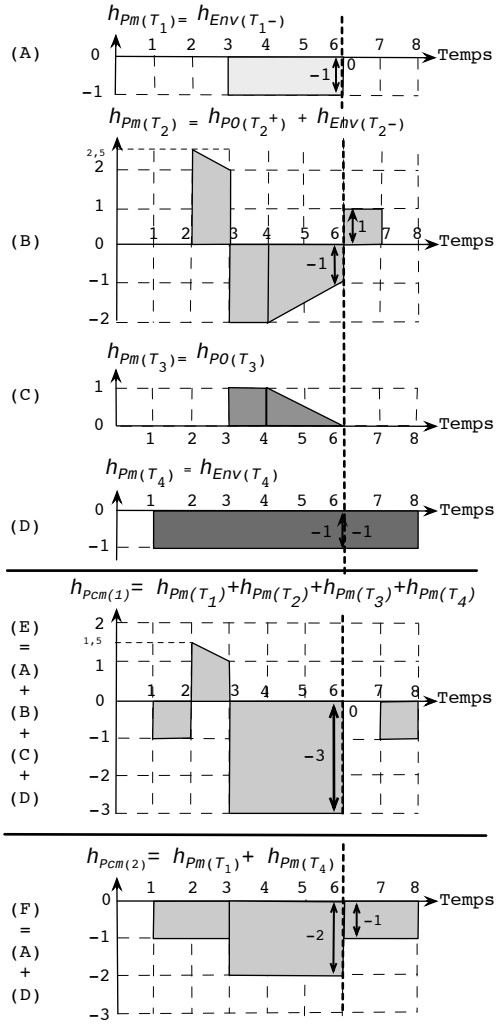

FIG. 6 – Calcul de $Pcm(1)$ et $Pcm(2)$.

même unique profil). A partir de maintenant, puisque $Pcm(r)$ et $PcM(r)$ ont des définitions similaires, nous nous concentrons sur le calcul de $Pcm(r)$

Exemple 9 *(suite de l'exemple 8) Les parties (E) et (F) de Fig. 6 et le tableau suivant fournissent* $Pcm(1)$ *et* $Pcm(2)$ *(*$h_{Pcm(1)} = \sum_{i=1}^{4} h_{Pm(T_i)}$ *et* $h_{Pcm(2)} = h_{Pm(T_1)} + h_{Pm(T_4)}$*).*

Ressource	*Trapèzes de* $mrP(Ressource)$
1	$\langle 1,-1,2,-1\rangle, \langle 2,3/2,3,1\rangle, \langle 3,-3,6,-3\rangle,$ $\langle 6,0,7,0\rangle, \langle 7,-1,8,-1\rangle$
2	$\langle 1,-1,3,-1\rangle, \langle 3,-2,6,-2\rangle, \langle 6,-1,8,-1\rangle$

4.2 Algorithme de balayage

Pour caculer $Pcm(r)$, pour une ressource donnée r, nous avons étendu l'algorithme de balayage présenté dans [1] de manière à prendre en compte des sous-tâches trapézoïdales. Une introduction générale sur le balayage est donné dans [10] et un exemple de son utilisation, dans le contexte de l'ordonnancement disjonctif, est montré dans [12].

L'algorithme de balayage étendu (voir Table 2) déplace une ligne verticale Δ d'un évènement à son prochain - dans notre contexte, un évènement correspond au début ou à la fin d'une sous-tâche trapézoïdale (i.e. un trapèze) de $Pm(T_i)$ $(i = 1..n; r \in a_{T_i})$ - sur l'axe temporel et construit $Pcm(r)$ de façon incrémentale. Il utilise deux structures de données :

- La première, appelée *état de la ligne de balayage*, contient pour la ressource r des informations relatives à la position courante δ de la ligne verticale Δ : sum_h_r et sum_sl_r, respectivement la hauteur et la pente de $Pcm(r)$ à la position δ.
- La seconde structure de données, appelée *suite des évènement*, contient les évènements associés aux trapèzes utilisés pour construire $Pcm(r)$ (i.e. les trapèzes de $Pm(T_i)$ tels que T_i peut être affectée à r (voir Résultat 3 et Définition 11). Nous représentons ces évènements par les champs suivants : $\langle tache, type, date, hauteur, pente\rangle$, où :
  - *tache* indique quelle tâche génère l'évènement.
  - $type \in \{\text{Deb}, \text{Fin}\}$ exprime si l'évènement correspond à un début ou à une fin du trapèze qui a généré l'évènement.
  - *date* spécifie la localisation dans le temps de l'évènement : si $type =$ Deb alors contient le début du trapèze sinon sa fin.
  - *hauteur* donne la quantité à ajouter à sum_h_r : si $type =$ Deb alors contient la hauteur du trapèze à son début sinon l'opposé de sa hauteur en fin.
  - *pente* donne la quantité à ajouter à sum_sl_r : si $type =$ Deb alors contient la pente du trapèze sinon l'opposé de sa pente.

Chaque trapèze non nul $Pm(T_i)^j$ [6] de $Pm(T_i)$ génère la paire d'évènements :

$\langle i, \text{Deb}, st_{Pm(T_i)^j}, sh_{Pm(T_i)^j}, sl_{Pm(T_i)^j}\rangle$
$\langle i, \text{Fin}, et_{Pm(T_i)^j}, -eh_{Pm(T_i)^j}, -sl_{Pm(T_i)^j}\rangle$
où $sl_{Pm(T_i)^j} = \frac{eh_{Pm(T_i)^j} - sh_{Pm(T_i)^j}}{d_{Pm(T_i)^j}}$.

Notons que cette paire d'évènements participe à la construction de tous les profils cumulés $Pcm(r)$ tels que $r \in a_{T_i}$. Tous les évènements sont initialement calculés et triés (par l'algorithme principal (AP) - voir Table 1), dans la liste L_{events}, en ordre lexicographique croissant par rapport à la paire $\langle date, type\rangle$ où $type = Fin$ est considéré être plus petit que $type = Deb$.

[6] $Pm(T_i)^j = (st_{Pm(T_i)^j}, sh_{Pm(T_i)^j}, d_{Pm(T_i)^j}, et_{Pm(T_i)^j}, eh_{Pm(T_i)^j})$ (début, hauteur au début, durée, fin, hauteur en fin) désigne la j^{eme} sous-tâche (trapèze) du profil $Pm(T_i)$.

TAB. 1 – **Algorithme_Principal (AP)** - Calcule tous les $mcrP$

In : Les tâches T_i $(i = 1..n)$; **Out :** Fail si il n'y a pas de solution autrement retourne Delay.	
1 :	$L_{events} \leftarrow \emptyset$;
2 :	**pour** $i = 1$ à n **faire** /* Pour toutes les tâches */
3 :	| Calcule $Pm(T_i)$;
4 :	| **pour** $j = 1$ à $\|Pm(T_i)\|$ **faire** /* Génère les mP-évènements ; $\|Pm(T_i)\|$ est le nombre de trapèzes de $Pm(T_i)$*/
5 :	| | **si** $(sh_{Pm(T_i)^j} \neq 0 \vee eh_{Pm(T_i)^j} \neq 0)$ **alors** /* i.e., $Pm(T_i)^j$ n'est pas un trapèze nul */
6 :	| | | $sl = (eh_{Pm(T_i)^j} - sh_{Pm(T_i)^j})/d_{Pm(T_i)^j}$;
7 :	| | | Ajoute $\langle i, \text{Deb}, st_{Pm(T_i)^j}, sh_{Pm(T_i)^j}, sl\rangle$ à L_{events} ;
8 :	| | | Ajoute $\langle i, \text{Fin}, et_{Pm(T_i)^j}, -eh_{Pm(T_i)^j}, -sl\rangle$ à L_{events} ;
9 :	Trier L_{events} en ordre lexicographique croissant par rapport à $\langle date, type\rangle$;
10 :	**pour** $r = 1$ à q **faire si Algorithme_Balayage**$(L_{events}, r) =$ Fail **alors retourne** Fail ;
11 :	**retourne** Delay ;

sum_h_r et sum_sl_r sont initialisées à 0 (ligne 1 de AB) et δ à la date du premier évènement, associé à la tâche T_i telle que $r \in a_{T_i}$, sur l'axe temporel (ligne 3 de AB). Quand la position courante de Δ change (ligne 5) de δ_k à δ_{k+1}, l'algorithme de balayage :

- Calcule d'abord le k^{eme} trapèze $Pcm(r)^k$ de $Pcm(r)$ (ligne 6).
- Puis (ligne 7) vérifie que $Pcm(r)^k$ ne dépasse pas la capacité de la ressource C_r, sinon la contrainte n'a pas de solution.
- Alors sum_h_r prend la valeur de la hauteur en fin du dernier trapèze retourné $(Pcm(r)^k)$ et δ est mis à jour à la valeur δ_{k+1} (ligne 9).

Dans tous les cas, les contributions (hauteurs et pentes) des trapèzes qui commencent ou se finissent à δ_{k+1} sont prises en compte dans sum_h_r et sum_sl_r (ligne 10).

Exemple 10 *(suite de l'exemple 8) Le trapèze* $\langle 3, -1, 6, -1\rangle$ *de* $Pm(T_1)$ *génère la paire d'évènements (*$e_1 = \langle 1, \text{Deb}, 3, -1, 0\rangle$, $e_2 = \langle 1, \text{Fin}, 6, 1, 0\rangle$*) tandis que le trapèze* $\langle 1, -1, 8, -1\rangle$ *de* $Pm(T_4)$ *génère la paire d'évènements (*$e_3 = \langle 4, \text{Deb}, 1, -1, 0\rangle, e_4 = \langle 4, \text{Fin}, 8, 1, 0\rangle$*). Les évènements triès de* L_{events} *utiles pour calculer* $Pcm(2)$ *à partir de* $Pm(T_1)$ *et* $Pm(T_4)$ *sont* $\langle e_3, e_1, e_2, e_4\rangle$*. Quand* e_1 *devient l'évènement courant alors, la ligne de balayage se déplace de la position 1 à 3 : l'algorithme calcule d'abord le* 1^{er} *trapèze* $(1, -1, 3, -1)$ *de* $Pcm(2)$ *(voir Fig. 6 Partie (F)). Puis, l'état de la ligne de balayage est mis à jour :* $sum\_h_r \leftarrow -1$ *et* $\delta \leftarrow 3$*. Finalement, la contribution de* e_1 *est ajoutée à* sum_h_r *qui décroît de* -1 *à* -2 *et à* sum_sl_r *qui ne change pas.*

Le prochain résultat donne la complexité du calcul de tous les profils cumulé de ressource minimum en utilisant un algorithme de balayage.

Résultat 4 *La complexité pour calculer tous les profils cumulés de ressource minimum en utilisant un algorithme de balayage est en* $O(p \cdot (\log p + q))$ *où* q *est le nombre de ressources,* $p = \sum_{i=1}^{n}(p_i)$ *est le nombre total de sous-tâches trapézoïdales des* n *tâches.*

Preuve 2 *du Résultat 4. La complexité du calcul (voir 3.4) de tous les* $Pm(T_i)$ *et de la génération de tous les évènements est en* $O\left(\sum_{i=1}^{n}(p_i \log p_i)\right)$.

La complexité pour trier les $O(p)$ *évènements, en utilisant un algorithme de complexité en* $O(p \log p)$*, est en* $O(p \log p)$*. Finalement, pour chaque ressource* r*, l'algorithme de balayage AB calcule* $Pcm(r)$ *en* $O(p)$ *(dans le pire cas). Par conséquent, nous obtenons une complexité dans le pire cas en* $O\left(\sum_{i=1}^{n}(p_i \log p_i) + p \cdot \log p + \sum_{i=1}^{q} p\right)$ *i.e., en* $O(p\ (\log p + q))$. □

5 Conclusion

Nous avons d'abord introduit un nouveau modèle de tâche qui combine les avantages de deux modèles. Dans le premier modèle ne prenait en compte que des tâches avec des consommations ou des production constantes et permettait le multi-ressources. Dans le second, les tâches pouvaient exprimer des consommations en ressource linéaires par morceaux mais avec une seule ressource. Ensuite, pour ce nouveau modèle de tâche, nous fournissons un algorithme polynomial pour le calcul, pour chaque ressource, de ses profils de ressource cumulés minimum et maximum.

Références

[1] Beldiceanu, N. and Carlsson, M. (2002). A New Multi-Resource cumulatives Constraint with negative heights. In : P. Van Hentenryck, ed.

TAB. 2 – **Algorithme_Balayage (AB)** - Calcule $Pcm(r)$ pour une ressource donnée r

In : La liste L_{events} de tous les évènements triés $\langle tache, type, date, hauteur, pente\rangle$ et une ressource r.
Out : Fail si $Pcm(r)$ dépasse la capacité de la ressource C_r ; Autrement retourne Delay.

```
1 :  k ← 1 ; sum_h_r ← 0 ; sum_sl_r ← 0 ;
2 :  Extrait, s'il existe, le premier évènement e de L_events tel que r ∈ a_{T_e.tache} ;
3 :  δ ← e.date ;
4 :  tant que e est défini faire
5 :  | si e.date ≠ δ alors /* Δ vient juste de bouger : Calcule le k^eme trapèze de Pcm(r) */
6 :  | | st ← δ; sh ← sum_h_r; d ← e.date − δ ; eh ← sum_sl_r ∗ (e.date − δ) + sum_h_r ;
7 :  | | si sh > C_r ∨ eh > C_r alors retourne Fail ; /* Pcm(r)^k dépasse C_r */
8 :  | | Pcm(r)^k ← (st, sh, e.date, eh) ; k ← k + 1 ;
9 :  | | sum_h_r ← eh; δ ← e.date ; /* Met à jour sum_h_r et δ */
10 : | sum_h_r ← sum_h_r + e.hauteur; sum_sl_r ← sum_sl_r + e.pente ;
11 : | Extrait, s'il existe, le prochain évènement e de L_events tel que r ∈ a_{T_e.tache} ;
12 : retourne Delay ;
```


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, *CP'2002*, Ithaca, NY, USA, Sept. 8-13, 2002. Springer : LNCS 470, 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. *CP'2003*, Kinsale, Ireland. Sept./Oct. 2003. Springer : LNCS 2833, pp 739–753.