COST

Communication Science & Technologie N° 3 . Décembre 2004

LA GENERATION AUTOMATIQUE DES PLANS DE COUPE - CAS D'UNE DECOUPE QUELCONQUE -

M. ZENNAKI[1], K. SAADOUNI[2]
Département Informatique, Faculté des Sciences, U.S.T.O.M.B.
BP 1505 El-M'naouer – ORAN – Tél & Fax : 041.53.07.86
[1] E-mail : prosoft99@caramail.com
[2] E-mail: kaddour_sadouni@hotmail.com

ABSTRACT
Cutting Stock Problems (which are NP-hard), have numerous applications in operations research (iron, steel, glass, wood, paper… industry) and in studies of computer operation (multiprogrammed computer systems, multiprocessor systems).
The crucial step in this problem is generating activities (cutting models), especially when we have a non-guillotine cutting. This kind of problem also called Placing problem have several methods: "Bottom Left", "Best Fit", "Difference Process"…
We propose an algorithm of generating activities using "Difference process" method which guarantee improved non-guillotin activities.

Key Words: Combinatorial Optimization, Integer Linear Programming, Cutting Stock Problem, Guillotine and non-guillotine cutting, Placing Problem, Difference Process, Best Fit, Bottom left.

RESUME
Les problèmes de découpe considérés comme des problèmes NP-complets, ont de nombreuses applications dans les industries sidérurgiques, automobiles, textiles, du papier, du bois, du verre etc. Ceux-ci sont également couramment rencontrés en VLSI et en systèmes (Gestion de la mémoire en multiprogrammation...).
La partie cruciale dans ce type de problème est la génération automatique des plans de coupe (activités), surtout lorsqu'il s'agit de découpe non guillotine ou découpe quelconque. Ce type de problème appelé encore problème de placement fait l'objet de recherches intenses, qui ont aboutit à plusieurs méthodes : « Bottom Left », « Best Fit », « Difference Process »…
Nous proposons dans cet article un algorithme de génération automatique des plans de coupe basé sur la méthode « Difference Process » qui garantit l'obtention de plans de coupe de haute qualité. Les résultats obtenus confirment l'efficacité de l'algorithme proposé.

Mots clés : Optimisation combinatoire, Programme linéaire en nombre entiers, Problème de découpe à deux dimensions, Découpe guillotine, Découpe quelconque, Problème de placement, Difference Process, Best Fit, Bottom left.

1. LE PROBLEME

Lorsqu'on se propose, à partir de certaines "entités" disponibles (bandes de papier, plaques d'acier, ...) de disposer des "sous-entités" déterminées, en suivant une politique optimale (minimisation de la chute causée par la découpe des "entités" en "sous-entités"), on rencontre certains problèmes de la programmation linéaire en nombre entiers (PLNE), connus sous le nom de problèmes de découpe formulés comme suit [1] :

Soit M = (L , H) Dimensions du matériel brut.

$d_i = (l_i , h_i)$ $(i = 1,...,m)$ les formats du carnet de commande à satisfaire en quantité q_i.

$$\begin{cases} Min \;\; z = \sum c_j x_j \quad (j = 1,...,n) \\ \sum a_{ij} x_j \geq q_i \quad j = 1,...,n \;\; \text{et} \;\; i = 1,...,m \\ x_j \geq 0 \;\; \text{et} \;\; x_j \in IN \quad \forall j = 1,...,n \end{cases}$$

Avec :

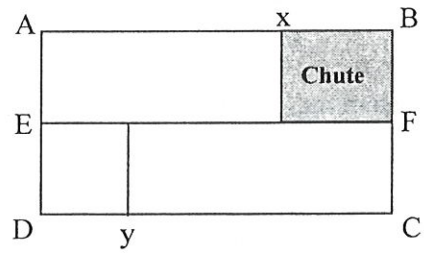
- La matrice A (a_{ij}) des contraintes est construite de telle manière que chaque

colonne constitue une activité (plan de coupe).

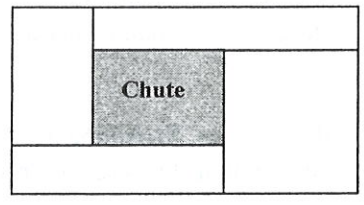
- x_j (j = 1,...,n n étant le nombre d'activités) désigne le nombre de fois que la $j^{ème}$ activité est utilisée et par c_j son coût. Si c_j correspond à la chute produite dans la $j^{ème}$ activité, le PLNE consistera à minimiser la chute totale.

L'utilisation d'heuristiques (recherche tabou, recuit simulé, algorithmes génétiques) ont permis la résolution approchée de ce problème. Sauf que la qualité des solutions dépend essentiellement des activités générées. Le cas d'une découpe guillotine a été largement étudié, notamment dans [1], [2] et [4]. Cependant, les industries actuelles utilisent des outils de coupe sophistiquées pouvant faire la découpe dans tous les sens et générant ainsi une découpe non guillotine.

Exemple :



Cette figure illustre un exemple de découpe guillotine, où le rectangle ABCD est coupé en E (de l'arête AD à l'arête BC). On obtient 2 rectangles (ABEF) et (EFDC) sur lesquels on opère une coupe guillotine respectivement en x et en y.



Contrairement à la découpe guillotine, cette figure montre qu'aucune coupe ne traverse le rectangle de bout en bout. La contrainte des trajectoires de coupe parallèles est levée.

Il est clair que le nombre d'activités générées en respectant la découpe guillotine est inférieur au nombre d'activités obtenues avec une découpe quelconque (guillotine ou non). Nous nous intéressons donc dans cet article à la génération automatique des activités dans le cas d'une découpe quelconque.

2. DIFFICULTES

La difficulté de ce problème tient en deux points essentiels :

- La génération des plans de coupe sans aucune contrainte et minimisant au mieux la chute (Problème de qualité).
- Le nombre d'activités qui accroît d'une manière exponentielle en fonction du nombre de formats à placer (Problème de quantité).

Pour surmonter le premier problème, nous proposons un algorithme basé sur l'approche « Difference Process » permettant d'obtenir des activités de haute qualité. Pour le deuxième problème, une heuristique est en cours de mise au point et qui consiste à générer un ensemble « suffisant » d'activités pour la résolution du problème de découpe.

3. LE PLACEMENT DES FORMES

Le placement des formes rectangulaires a fait l'objet de plusieurs recherches. On peut distinguer les travaux suivants :

3.1. « BOTTOM LEFT »

On peut assimiler cet algorithme au jeu « Tetrice » [3]. Tout d'abord, la forme est placée au coin supérieur droit du format brut, puis elle est déplacée vers le bas le maximum possible (jusqu'elle soit bloquée par une autre forme ou le bord inférieur de la tôle), puis vers la gauche le plus possible. Ce mouvement est répété jusqu'à obtention d'un état stationnaire (la forme ne peut plus bouger dans les deux sens). L'algorithme « Bottom Left » dans sa forme initiale est assez limité ; puisqu'il ne permet pas de rotation des formes. Une variante a été proposée dans [5] pour résoudre cet inconvénient.

3.2. « BEST FIT »

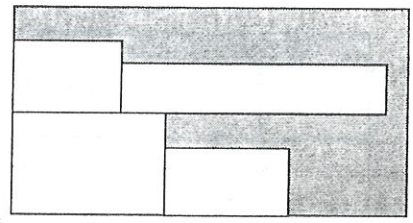
Le « Best fit » s'inspire des techniques utilisées en système d'exploitation pour la gestion de la mémoire. Dans cet algorithme, une liste appelée « Chutes » est tenue à jour comportant les chutes présentes dans la tôle courante. Cette liste est triée par ordre croissant, et contient initialement le format brut. Lorsqu'une forme est candidate au placement, on parcourt la liste (Chutes) pour dégager la meilleure chute qui peut contenir la forme, c'est à dire la plus petite chute en superficie.

3.3. « DIFFERENCE PROCESS » (D.P.)

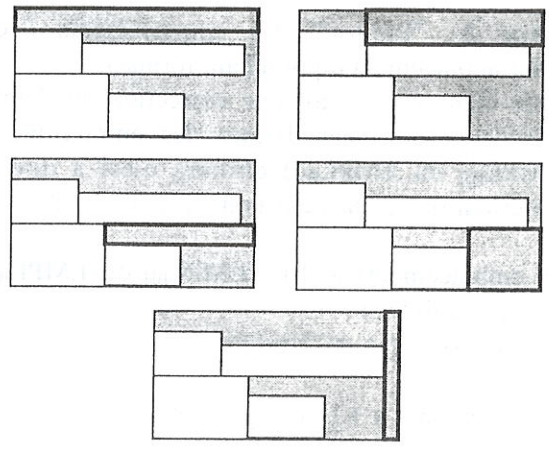
A la différence des algorithmes précédents, la méthode « Difference Process » considère l'espace disponible dans le format brut de telle

manière à former les plus grands espaces rectangulaires.

Exemple :



La chute de l'activité représentée ci-dessus, est considérée comme un ensemble de 5 chutes (les plus grands espaces rectangulaires) :



Après l'insertion d'une nouvelle forme, on doit gérer les espaces générés et recalculer les distances.

Les activités obtenues par l'algorithme « Difference Process » sont par rapport aux algorithmes « Bottom Left » et « Best Fit » d'une part de meilleure qualité et d'autre part sans aucune contrainte dans la découpe.

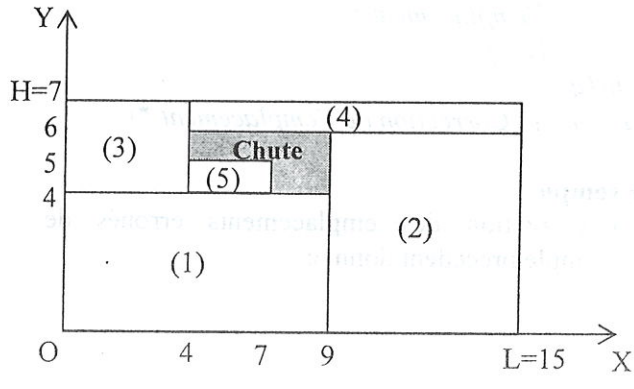
4. ALGORITHME « D.P. »

4.1. MODELISATION D'UNE ACTIVITE

Nous représentons une activité sur un format brut (L,H) par rapport à un repère orthogonal (O, X, Y). Ainsi, on retient pour chaque format placé (l_i, h_i) ses coordonnées (x , y).

Exemple :

Soit l'activité



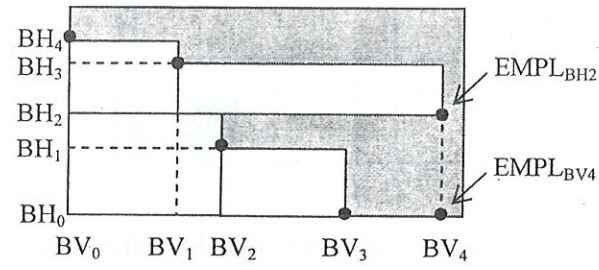
Format Brut (L,H) = (15,7)

L'activité est représentée comme suit :

Coordonnées (x , y)	Format (l_i , h_i)
(0 , 0)	Format n°1 (9 , 4)
(9 , 0)	Format n°2 (6 , 6)
(0 , 4)	Format n°3 (4 , 3)
(4 , 6)	Format n°4 (11 , 1)
(4 , 4)	Format n°5 (3 , 1)

4.2. LOCALISATION

La première partie de l'algorithme est la localisation de tous les emplacements susceptibles au placement du prochain format. Pour cela, nous générons l'ensemble des bandes horizontales (BH=$\{bh_i\}$) et verticales (BV=$\{bv_i\}$) obtenues à partir de l'activité courante, puis on détermine les emplacements correspondants à chaque bande : $EMPL_{BH}$ et $EMPL_{BV}$ respectivement.



Initialement :

BH =$\{0\}$ BV =$\{0\}$

$EMPL_{BH}$ =$\{(0,0)\}$ $EMPL_{BV}$ = $\{(0,0)\}$

indiquant l'origine (0,0) comme seul emplacement.

L'activité est représentée par un tableau ACT (voir § 4.1) :

Coordonnées	Formats
(x , y)	(l_i , h_i)

Les bandes horizontales et verticales sont déterminées comme suit :

$BH = \{0\} \cup \{ (y + h_i) / [(x,y) , (l_i,h_i)] \in ACT \}$

$BV = \{0\} \cup \{ (x + l_i) / [(x,y) , (l_i,h_i)] \in ACT \}$

L'élément $bh_i = H$ (resp. $bv_i = L$) est éliminé de l'ensemble BH (resp. BV).

La génération des emplacements par bandes $EMPL_{BH}$ et $EMPL_{BV}$ est réalisée en deux étapes :

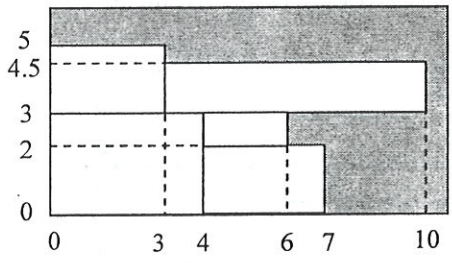
Etape 1 : Génération des emplacements (possibles et impossibles).

Pour chaque bande horizontale bh_i :
- *Chercher l'élément de ACT tel que $y=bh_i$ et ayant le plus grand x.*
- *Si cet élément existe (soit (l_i , h_i)) alors ajouter à $EMPL_{BH}$ l'élément $(x + l_i , bh_i)$*
- *Sinon ajouter l'élément $(0 , bh_i)$.*

Pour chaque bande verticale bv_i :
- *Chercher l'élément de ACT tel que $x=bv_i$ et ayant le plus grand y.*
- *Si cet élément existe (soit (l_i , h_i)) alors ajouter à $EMPL_{BV}$ l'élément $(bv_i , y + h_i)$*
- *Sinon ajouter l'élément $(bv_i , 0)$.*

Exemple : Soit l'activité suivante :

(L , H) = (11 , 6)



Représentée d'une manière tabulaire comme suit :

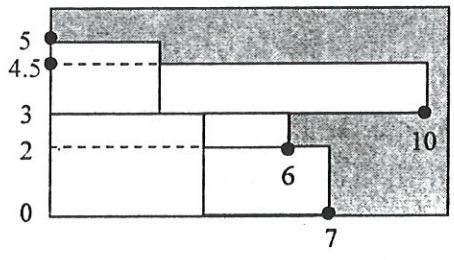
ACT

(x , y)	(l_i , h_i)
(0 , 0)	(4 , 3)
(0 , 3)	(3 , 2)
(4 , 0)	(3 , 2)
(3 , 3)	(7 , 1.5)
(4 , 2)	(2 , 1)

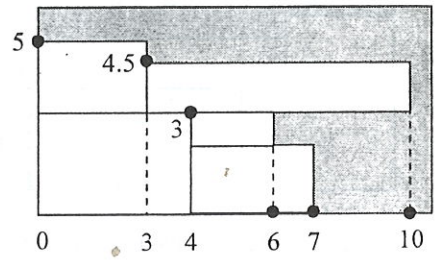
On obtient :
BH = {0 , 3 , 5 , 2 , 4.5 , 3} = {0 , 2 , 3 , 4.5 , 5}
BV = {0 , 4 , 3 , 7 , 10 , 6} = {0, 3 , 4 , 6, 7 , 10}

Après exécution de l'étape 1, on obtient :

- $EMPL_{BH}$ = {(7,0) (6,2) (10,3) (0,4.5) (0,5)}



L'emplacement (0,4.5) est erroné.
- $EMPL_{BV}$ = {(0,5) (3,4.5) (4,3) (6,0) (7,0) (10,0)}



Les emplacements (4,3) et (6,0) sont erronés.

Etape 2 : "Pré-test" des emplacements (Test avant le placement du prochain format).
Cette étape est en fait une correction de l'étape précédente qui peut donner des emplacements ($EMPL_{BH}$ ou $EMPL_{BV}$) erronés, c'est à dire, à l'intérieur de formats existants dans l'activité.

Un emplacement (a , b) ($EMPL_{BH}$ ou $EMPL_{BV}$) n'est pas valide si :
$\exists$ [(x,y) , (l_i,h_i)] $\in$ ACT tel que :

$$x \le a < x + l_i \quad \text{et} \quad y \le b < y + h_i$$

La correction des emplacements erronés se fait comme suit :

- $EMPL_{BH}$ (α , bh_i)

$t := \alpha$
Tant que (t , bh_i) non valide et $t < L$ faire
- *Soit [(x,y) , (l_i,h_i)] l'élément de ACT causant la non validité de l'emplacement :*
  $t := x + l_i$

FinTq
$\alpha := t$ (Correction de l'emplacement *)*

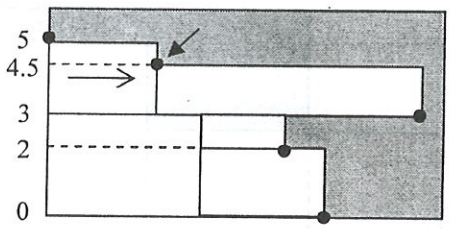
- $EMPL_{BV}$ (bv_i , α)

$t := \alpha$
Tant que (bv_i , t) non valide et $t < H$ faire
- *Soit [(x,y) , (l_i,h_i)] l'élément de ACT causant la non validité de l'emplacement :*
  $t := y + h_i$

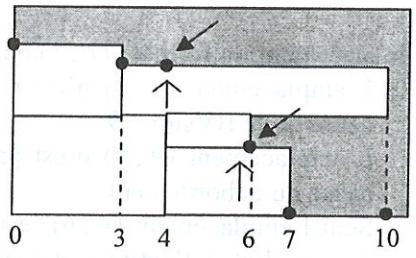
FinTq
$\alpha := t$ (Correction de l'emplacement *)*

Exemple :
La correction des emplacements erronés de l'exemple précédent donne :

- $EMPL_{BH}$: (0 , 4.5) ➔ (3 , 4.5)

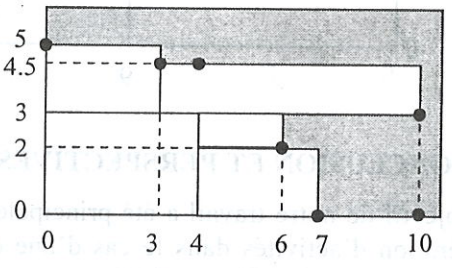


- $EMPL_{BV}$: (4 , 3) ➔ (4 , 4.5) et
(6 , 0) ➔ (6 , 2)



A la fin de cette étape, nous construisons l'ensemble EMPL = $EMPL_{BH}$ U $EMPL_{BV}$

Ainsi EMPL = { (7,0) (6,2) (10,3) (3,4.5) (0,5) (4,4.5) (10,0) }



Remarque : Les emplacements (10,3) et (4,4.5) seront utiles pour la deuxième partie de l'algorithme mais seront écartés lors du placement final.

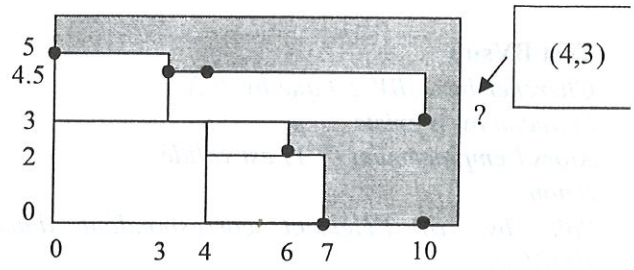
4.3. PLACEMENT DU FORMAT

Après avoir localiser les emplacements susceptibles au placement du format suivant (EMPL), nous continuons les tests, et ceci en essayant de placer le format dans chaque emplacement de EMPL en commençant par le plus proche de l'origine. Pour cela, deux tests sont réalisés :

a) Test du débordement

Permet d'éliminer les emplacements qui provoquent un débordement par rapport au format brut (L , H).

Exemple : Soit à placer le format (4 , 3) dans l'activité de l'exemple précédent (L=11, H=6):



L'ensemble des emplacements classés suivant l'ordre du plus proche de l'origine est donc :
EMPL={ (0 , 5) (7 , 0) (3 , 4.5) (6 , 2) (4 , 4.5) (10 , 0) (10 , 3) }

Nous essayons de placer ce format dans chaque emplacement de EMPL :

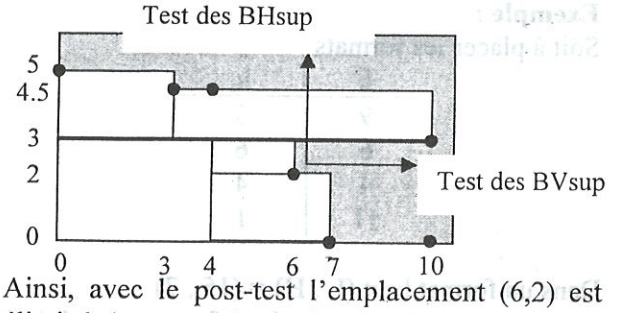
(0 , 5) + (4 , 3) = (4 , 8)	Débordement
(7 , 0) + (4 , 3) = (11 , 3)	OK
(3 , 4.5) + (4 , 3) = (7 , 7.5)	Débordement
(6 , 2) + (4 , 3) = (10 , 5)	OK
(4 , 4.5) + (4 , 3) = (8 , 7.5)	Débordement
(10 , 0) + (4 , 3) = (14 , 3)	Débordement
(10 , 3) + (4 , 3) = (14 , 6)	Débordement

De ce test, deux emplacements sont retenus, l'ensemble EMPL devient égal à {(7 , 0) (6 , 2)}

b) "Post-Test"

Ce test permet d'éliminer les emplacements erronés après placement du format. Dans l'exemple précédent, il est impossible de placer le format à l'emplacement (6,2) malgré qu'il ne provoque pas de débordement.

Pour écarter ce type d'emplacement, le Post-Test consiste à parcourir les bandes verticales et horizontales d'ordre supérieur. Le test se fait d'abord vers la droite dans le sens des BV supérieurs puis vers le haut dans le sens des BH supérieurs.



Ainsi, avec le post-test l'emplacement (6,2) est éliminé à cause de la bande horizontale BH3 ($EMPL_{BH3}$ = (10,3)).

Le post-test se réalise comme suit :
Soit (X , Y) l'emplacement à tester,

Test BVsup
Chercher $bv_i \in BV$ tel que $bv_i > X$
Si aucun bv_i n'existe
Alors l'emplacement (X,Y) est validé
Sinon
Soit (bv_i , α) l'élément correspondant dans $EMPL_{BV}$
Si $\alpha \leq Y$ alors Test BHsup
Sinon Si $X+l_i \leq bv_i$ alors Test BHsup
Sinon L'emplacement est erroné.
Fsi
Fsi
Fsi

Test BHsup
Chercher $bh_i \in BH$ tel que $bh_i > Y$
Si aucun bh_i n'existe
Alors l'emplacement (X,Y) est validé
Sinon
Soit (α, bh_i) l'élément correspondant dans $EMPL_{BH}$
Si $bh_i \geq Y+h_i$ alors tester prochaine BVsup
Sinon Si $\alpha \leq X$ alors Tester prochaine BHsup
Sinon L'emplacement est erroné.
Fsi
Fsi
Fsi

Remarque : Si après exécution de l'algorithme, un format (l , h) ne peut se placer, il faut essayer de placer le format inverse (h , l).

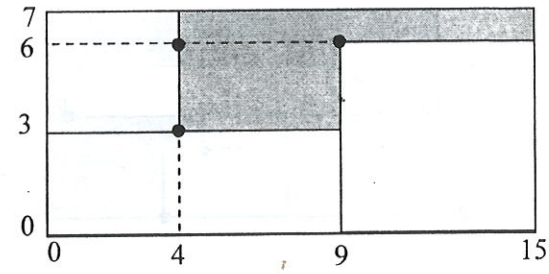
4.4. RESULTATS ET COMMENTAIRES

Les résultats obtenus à travers l'implémentation de cet algorithme montre la haute qualité des activités obtenues. En plus, aucune contrainte n'est imposée dans la découpe (découpe quelconque ou découpe non guillotine). Ceci est bien illustré dans l'exemple suivant :

Exemple :
Soit à placer les formats

l_i	h_i
9	3
6	6
4	4
11	1

Dans un format brut (L , H) = (15 , 7)
Le placement des 3 premiers formats conduit à l'activité suivante :

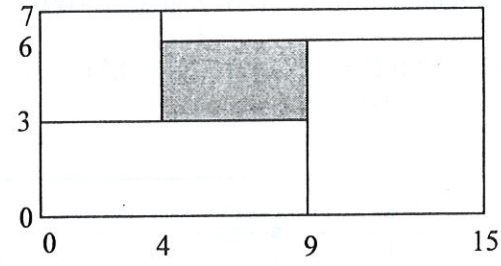


L'application de l'algorithme donne les emplacements suivants :

EMPL = { (4 , 3) (4 , 6) (9 , 6) }

Pour le placement du format (11 , 1) on a :

- L'emplacement (4 , 3) n'est pas valide à cause de la BVsup = 9.
- L'emplacement (9 , 6) n'est pas valide à cause du débordement.
- Seul l'emplacement (4 , 6) est valide, ce qui conduit à l'activité suivante (qui ne peut pas être obtenue avec une découpe guillotine) :



5. CONCLUSION ET PERSPECTIVES

L'objectif de notre travail a été principalement la génération d'activités dans le cas d'une découpe quelconque. L'algorithme « D.P » proposé permet dans le cas d'une découpe de formats rectangulaires à deux dimensions, d'obtenir des activités de haute qualité. Nous essayons actuellement de développer notre travail pour traiter les points suivants :

- Proposer une heuristique efficace permettant de générer un nombre suffisant d'activités. En effet, la génération de toutes les activités par une méthode arborescente ne peut être envisagé dans le cas de problèmes de grande taille, ou lorsque les formats du carnet de commande sont de petite taille par rapport au format brut. Dans ce dernier cas, même si le nombre de formats n'est pas important, la profondeur de l'arbre risque d'augmenter d'une manière considérable. Une solution consiste à faire des regroupements des formats de petite taille afin de réduire la

profondeur de l'arbre, mais provoque une détérioration de la qualité des activités.

- Proposer une méthode de placement adaptée à tout type de formats (rectangulaires ou non). L'algorithme "D.P" proposé ne peut s'exécuter que si les formats du carnet de commande sont rectangulaires, ou ont été "rectangularisés". Nous pensons que la meilleure approche serait de se baser sur l'expertise humaine et d'introduire cet expertise dans la méthode.

BIBLIOGRAPHIE

[1] D. Benhamamouch. Un algorithme de découpe optimale dans IR^2. Thèse de docteur 3ème cycle. Université Pierre et Marie Curie, Paris 6, 1979.

[2] V. Zissimopoulos. Algorithme ε-approchant Thèse docteur 3ème cycle. Orsay 1984.

[3] T.W.Leung, C.H. Yung, M.D. Troutt. Applications of genetic search and simulated annealing to the 2-dimensional non-guillotine cutting stock problem. Graduate School of Management, Kent State University. 1997.

[4] M. Zennaki. La recherche tabou et le recuit simulé pour la résolution des problèmes de découpe à deux dimensions. Thèse de Magister. U.S.T.O.M.B. 1998.

[5] Liu, H. Teng. An improved BL-algorithm for genetic algorithm of the orthogonal packing of rectangles. European Journal of Operational Research, 112, pp. 413-420. 1999.