N° d'ordre : 15/2007-M/MT

République Algérienne Démocratique et Populaire

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ DES SCIENCES ET DE LA TECHNOLOGIE HOUARI BOUMEDIENNE

FACULTÉ DES MATHÉMATIQUES

Département de recherche opérationnelle

==========

MEMOIRE

Présenté pour l'obtention du diplôme de MAGISTER

En : MATHEMATIQUES

Spécialité : Recherche Opérationnelle : Mathématiques Discrètes et Optimisation

Par : AIT ABDESSELEM Maya

Thème

Découpe Guillotine, Modèles et Méthodes

Soutenu publiquement, le 25/10/2007, devant le jury :

K. BOUKHETALA	Professeur	Président.
R. OUAFI	Maître de Conférences	Directeur de thèse.
A. BERRACHEDI	Professeur	Examinateur.
M. MEDJDEN	Maître de Conférences	Examinateur.

REMERCIEMENTS

En premier lieu, je remercie Dieu le tout puissant de m'avoir prêté main pour la réalisation de ce travail.

À **Mr OUAFI Rachid,** j'exprime toute ma gratitude et ma reconnaissance pour la patience et la générosité avec les quelles il a su me guider durant les travaux de recherches. Je le remercie vivement pour son aide, son assistance et sa sympathie dont il a fait preuve.

Mes sincères remerciements vont aussi à **Mr K. BOUKHETALA,** professeur à l'U.S.T.H.B, qui m'a fait l'honneur de présider le jury.

Mes remerciements s'adressent également à **Mr A. BERRACHEDI,** professeur à l'U.S.T.H.B et **Mr M. MEDJDEN**, maître de conférences à l'U.S.T.H.B pour leur gratitude et leur dévouement qui ont permis d'honorer le jury.

À **Mr HIFI M'hand,** je présente tous les remerciements pour son aide précieux grâce auquel plusieurs de nos résultats ont vus lumière.

Table des matières

Liste des abréviations

abréviations	A quoi elles correspondent
R	Le support initial
RG	Le rectangle guillotine
l_{RG}	Longueur du rectangle guillotine RG
h_{RG}	hauteur du rectangle guillotine RG
N	Le nombre de pièces
r	Le nombre des différentes longueurs
r'	Le nombre des différentes hauteurs
L	Longueur du support initial R
H	Hauteur du support initial R
p_i	La pièce i
l_i	Longueur de la pièce i
h_i	Hauteur de la pièce i
c_i	Poids de la pièce i
λ_j	Le nombre de fois où la longueur l_j a été découpée de L
μ_k	Le nombre de fois où la hauteur h_k a été découpée de H
a_{ijk}	Le nombre de rectangle de dimension $l_j \times h_k$ contenant la pièce de type i
$l_{\min}$	correspond à la plus petite longueur des pièces
$h_{\min}$	Correspond à la plus petite hauteur des pièces.
P	le nombre maximum de bandes verticales
Q	le nombre maximum de bandes horizontales
L_j	Longueur de la j^{eme} bande verticale
h_k	hauteur de la k^{eme} bande horizontale pour
B_{hor}	Valeur du modèle horizontale
B_{ver}	Valeur du modèle verticale
$B_{\inf}$	Valeur de la borne inférieure initiale
$B_{\sup}$	Valeur de la borne supérieure initiale
ED	Etape de découpe
DGDD	Problème de découpe guillotine à deux dimensions
DGDD-NC	Problème de découpe guillotine à deux dimensions non contraint
DGDD-C	Problème de découpe guillotine à deux dimensions contraintes
K-DGDD	Problème de découpe guillotine à deux dimensions avec K étape de découpe (ED)

Liste des figures

Introduction

Les problèmes d'optimisation combinatoire suscitent beaucoup d'intérêts. Malgré les progrès considérables de l'outil informatique, les méthodes d'énumération, exhaustives ou partielles sont encore peu satisfaisantes en temps d'exécution ou en efficacité. Comme ces problèmes contiennent souvent beaucoup de solutions à intérêts pratiques acceptables, les spécialistes de l'optimisation combinatoire ont orienté leur recherche vers le développement des méthodes heuristiques dont le but est de trouver une solution de qualité satisfaisante en un temps de calcul raisonnable. D'autant plus que pour des problèmes réels, il n'est pas toujours impératif de trouver la solution optimale, mais des solutions dont la qualité et le temps pour les obtenir restent dans l'acceptable. Ces performances étant de natures opposées, il s'agit alors de trouver un compromis suivant le contexte du problème.

Parmi les problèmes d'optimisation combinatoire on s'intéresse au problème de découpe. Ce problème intervient dans divers processus industriels : textile, bois, métal, papier, etc.…, il est considéré comme étant un problème NP-complet [10]. Il possède différentes variantes liées aux supports, aux matériels utilisés, à la forme des objets et aux contraintes imposées.

Le plan de découpe entraîne forcément des conséquences qui se traduisent par un profit ou une perte. L'objectif d'un problème de découpe revient donc à maximiser le profit ou minimiser la chute. Ces problèmes sont résolus par des méthodes exactes telle que la programmation dynamique ou l'exploration arborescente mais aussi par des méthodes approchées.

Le thème abordé dans nos travaux est celui de la découpe guillotine à deux dimensions qui consiste à effectuer des coupes verticales ou horizontales sur un support rectangulaire en allant d'une arête à son opposé tout en étant parallèle au deux arêtes restantes. Parmi les divers travaux effectués dans ce domaine on a examiné en détail les méthodes exactes suivantes : l'algorithme récursive de Herz [20] pour le cas non contraint et non pondéré, et une méthode constructive [21], ainsi que la méthode approchée de Beasley [3].
Nous nous sommes inspirés principalement de ces méthodes exactes pour concevoir nos deux heuristiques basées sur le principe des points de découpe et les techniques de programmation dynamique. Leur but est d'éviter le calcul inutile de certains modèles de découpe dont la chute est importante, et par conséquent la diminution du temps de calcul.

Cette thèse comporte cinq chapitres. Dans le chapitre 1, la première partie est consacrée à quelques notions générales et concepts de base pour le problème de découpe ainsi que sa modélisation et ces différents types. Dans la deuxième partie on étudie particulièrement le problème de découpe de type guillotine, ses modèles de bande et de découpe ainsi que ses différentes variantes.

Le deuxième chapitre traite de la classification des problèmes de découpe. Une première classification a été établie par Dychoff (1995) [7] et qui fut améliorée en 2004 par Gerhard Wäscher, Heike Haußner et Holger Schumann [36].

Dans le troisième chapitre nous passons en revue quelques travaux réalisés dans le domaine de la découpe guillotine à deux dimensions sans contrainte. Nous verrons en premier lieu l'algorithme exact récursif de Herz [20] pour le cas non contraint basé sur le principe des points de découpe applicable uniquement au cas non pondéré. En suite nous étudierons une méthode constructive exacte basée sur des techniques de programmation dynamique et une recherche arborescente avec critère de sélection du meilleur d'abord [21] et enfin nous verrons la méthode approchée de Beasley [3] pour la résolution du problème DGDD sans aucune contrainte.

Dans le quatrième chapitre nous présentons deux nouvelles approches inspirées de deux méthodes exactes pour la résolution du problème de découpe guillotine à deux dimensions non contraint. Ces deux approches sont basées sur le principe des points de découpe représentant les différentes longueurs et les différentes hauteurs ainsi que sur des techniques de programmation dynamique. La première approche consiste a effectuer une seule coupe engendrant deux sous- rectangles qui seront traités chacun séparément en utilisant la programmation dynamique, alors que la deuxième approche consiste à effectuer deux coupes successives : l'une horizontale et l'autre verticale, ce qui engendre trois sous- rectangles qui seront traités aussi chacun séparément en utilisant la programmation dynamique.

Dans le cinquième et dernier chapitre nous présentons quelques résultats numériques de l'application des deux approches sur des instances de la littérature et d'autres générées aléatoirement, ainsi qu'une analyse de ces résultats afin de déterminer l'efficacité des deux approches présentées.

Chapitre 1

Notions générales sur les problèmes de découpe

1.1. Introduction :

Le problème de découpe à été étudié pour la première fois par Kantorovick [22] dont le but était de le modéliser sous une forme cohérente. Gilmore et Gomory [13] ont repris ces travaux pour la résolution de quelques unes de ces variantes et les ont généralisé aux problèmes de découpe à deux dimensions.

Ces problèmes différent entre eux suivant le support, le matériel utilisé et le type de découpes effectuées. Parmi ces différents types on s'intéresse particulièrement à la découpe guillotine.

I .1 . Notions générales sur les problèmes de découpe :

I.1.1. La famille des problèmes de découpe :

Le problème de découpe est un problème courant que l'on rencontre particulièrement dans certaines industries tels que : la boiserie, la métallurgie et le prêt-à-porter….Un mauvais plan de découpe peut entraîner d'importantes pertes matérielles.

Le problème de découpe est constitué par les éléments suivants :

I .1.1.1. Le support initial :

Le support constitue l'un des éléments de base du problème de découpe. les supports différent entre eux par leur :

- *Forme géométrique* :

  Si la forme du support est circulaire, il est appelé cercle initiale de dimension (R) où R est son rayon.

  Si le support est de forme rectangulaire, il est appelé dans ce cas rectangle initiale, qui se caractérise par ses dimensions (L, H) où L représente la longueur et H sa hauteur.

  Dans les deux cas on parle de problème de découpe à deux dimensions.

  La longueur (ou la hauteur) est négligée si elle est très grande par rapport aux dimensions des pièces à produire. Dans ce cas le problème peut être représenté par une bande, on parlera donc de problème de découpe à une dimension.

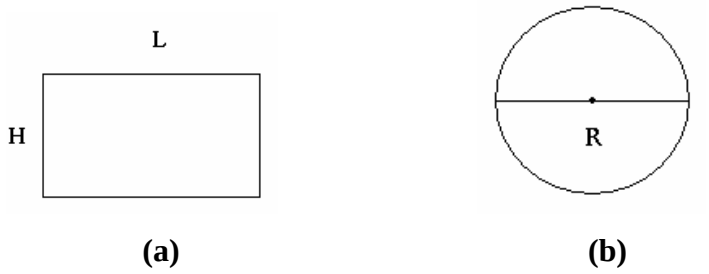
  Un problème de découpe est à trois dimensions si son support est caractérisé par une base rectangulaire de dimension (L, H) et d'une épaisseur E.

- *Homogénéité et régularité* :

  Un support est homogène si sa surface est partout la même et il est régulier s'il ne présente aucun défaut.

- Dissymétrie :

  Sur certain support comme les tissus à motifs, les pièces à découper ont souvent une orientation imposée, on parlera alors de pièce fixée.



i. Problème de découpe à deux dimensions : (a) rectangle.
(b) cercle.

ii. Problème de découpe à trois dimensions. **iii.** Problème de découpe à une dimension

Figure1: les différents types de supports.

I.1.1.2. Les pièces à découper :

Les caractéristiques principales d'une pièce à découper sont :

- Sa forme géométrique.
- Et le respect des contraintes liées au support et au coût de production.

Les pièces sont dites fixées si l'orientation est imposée sinon les rotations sont permises et les pièces peuvent être pivotées de 90°.

I.1.1.3. Les différents types de découpe :

Les découpes diffèrent entre elles suivant le matériel utilisé, on cite :

i. *La découpe guillotine* :

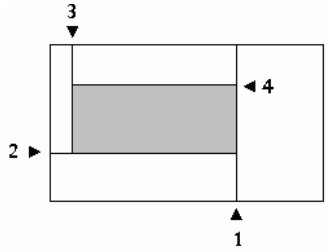
C'est une coupe qui s'effectue d'une seule tranche sur un rectangle en allant d'une arête à son opposé tout en étant parallèle aux deux autres restantes, ce qui engendre deux sous rectangles.

ii. *La découpe non guillotine* :

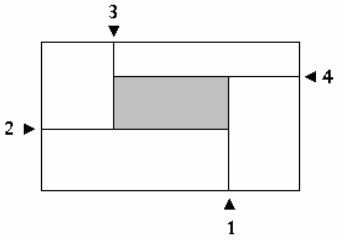
Elle consiste à effectuer le même procédé que dans la découpe guillotine, mais elle peut être effectuée tout en alternant coupe verticale et coupe horizontale. Elle fournie de meilleures solutions que la découpe guillotine.

iii. *La découpe non orthogonale* :

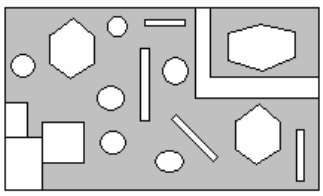
Les rotations sur les pièces sont permises ; ce type de découpe est typique à la découpe au laser (un bras pivotant, se déplaçant dans tous les sens à une vitesse variable et qui effectue les découpes).



(a) découpe guillotine.



(b) découpe non guillotine



(c) découpe non orthogonale.

Figure 2: les différents types de découpe

II.1. Problème de découpe guillotine (DGDD) :

Parmi les différents types de découpe nous allons détailler le problème de découpe guillotine. Soit les hypothèses suivantes :

- Le support à découper est de forme rectangulaire.
- Les découpes sont de type guillotine.
- Les pièces à découper ont une orientation fixée.
- Les dimensions du support et des pièces sont toutes des entiers.

II.1.1. Formulation et modélisation du problème :

Une instance du problème de découpe guillotine à deux dimensions consiste à découper *n* petites pièces rectangulaires à partir d'un rectangle initial *R* de dimensions (*L*, *H*), où chaque pièce *i*, *i=1,…, n*, est caractérisée par sa longueur l_i , sa hauteur h_i et un profit c_i , *i=1,…, n*. L'objectif est de maximiser la somme des utilités des pièces produites.

Soit le vecteur $(x_1,...,x_n)$ constitué d'entiers naturels correspondant à un modèle de découpe guillotine s'il est possible de découper x_i pièce de type i, i=1,…,n, à partir du support initial de telle sorte que les pièces ne se chevauchent pas.

Le problème de découpe guillotine à deux dimensions consiste à maximiser la valeur du modèle de découpe, ie,

$$DGDD\begin{cases}\max \sum_{i=1}^{n} c_i x_i \\ tel \quad que \quad (x_1,...,x_n) \quad correspand \quad à \quad un \quad \text{mod}\, el \quad de \quad DG\end{cases}$$

Toutes les pièces produites qui différent des pièces à découper sont considérées comme *des chutes*

Définition 1.1 :

Le problème DGDD est dit **non pondéré** si le profit de chaque pièce est égale à sa surface C à d :

$$c_i = l_i h_i \quad \text{pour} \quad i=1,\ldots, n.$$

L'objectif du problème de découpe guillotine (DGDD) dans ce cas est de **minimiser** la chute

Définition 1.2 :

Le problème DGDD est dit **pondéré** s'il existe un type de pièce pour lequel le profit est différent de sa surface c à d :

$$\exists \ l \ , \ 1 \le l \le n \quad tel\ que \quad c_l \neq l_l h_l$$

L'objectif dans ce cas est de **maximiser** la somme des utilités sur l'entité en stock.

Définition 1.3 :[28]

Soit M l'ensemble fini des vecteurs représentant les différents modèles de découpe réalisable pour le problème DGDD, soit λ un élément de M tel que $\lambda = (\lambda_1,...,\lambda_n)$, où chaque λ_i désigne le nombre d'apparition de la i^{eme} pièces de S dans le modèle de découpe λ.

(a) version non pondérée :

La fonction objectif est représentée par l'application F définie par :

$$F : M \ \rightarrow \ N$$

$$\text{Tel que } F^* = F(\lambda) = L.H - \sum_{i=1}^{n} c_i.\lambda_i$$

Où $c_i = l_i h_i$ désigne la surface de la pièce i =1,…, n.

La solution optimale du problème DGDD dans ce cas consiste à trouver le couple (λ^*, F^*) tel que :

$$F(\lambda^*) = \min_{\lambda \in M} F(\lambda)$$

L'utilité du modèle de découpe réalisable λ^* est égale à F^*, λ^* est le modèle de découpe optimale qui minimise la chute sur le rectangle initial.

(b) version pondérée

La fonction objectif est représentée cette fois comme suit :

$$F : M \rightarrow N$$

$$\text{Tel que } F^* = F(\lambda) = \sum_{i=1}^{n} c_i . \lambda_i$$

Où $c_i \neq l_i h_i$ désigne le profit associé à la pièce i =1,…, n.

La solution optimale revient à déterminer le couple (λ^*, F^*) tel que :

$$F(\lambda^*) = \max_{\lambda \in M} F(\lambda)$$

L'utilité du modèle de découpe réalisable λ^* est égale à F^*, λ^* est le modèle de découpe optimale qui maximise la somme des utilités sur l'entité en stock.

Définition 1.4 :

Lorsqu'une étape de découpe (ED) supplémentaire nécessaire pour extraire une pièce est permise on parle alors de problème de découpe guillotine (DGDD) non exact, et dans le cas contraire il s'agit d'un problème de découpe guillotine exact.

II.1.2. Modèles de bande :

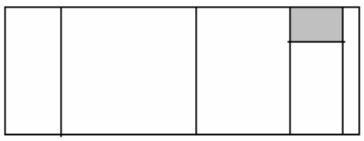
Pour pouvoir obtenir un modèle de DGDD on est contraint de générer d'abord des modèles de bandes qui peuvent être classés comme suit :

a) *Une bande générale* horizontale (resp. verticale) est composée au moins de deux pièces de hauteur (resp. longueur) différente.

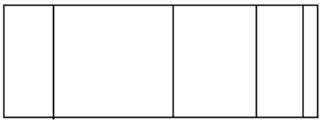
b) *Une bande uniforme* horizontale (resp. verticale) est composée uniquement de pièces ayant la même hauteur (resp. la même longueur).

c) *Une bande homogène* (resp. verticale) est une bande où il n'y a qu'un seul type de pièces participant.

d) *Une bande optimale* de longueur α et de hauteur h (resp. de longueur l et de hauteur β) est une bande générale occupant la surface maximale de la bande (α , h) (resp. (l, β)).



(a) bande générale.



(b) bande uniforme.

(c) bande homogène.

Figure3 : Les différents types de bandes.

II.1.3. Modèles de découpe :

Un modèle de découpe réalisable s'obtient en combinant un certain nombre de bandes horizontales ou verticales. Comme pour les modèles de bandes il existe différents modèles de découpe ;

a) Un modèle de découpe uniforme :

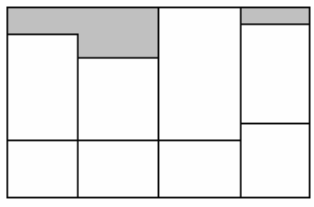
C'est un modèle de découpe réalisable constitué uniquement de bandes uniformes.

b) *Un modèle de découpe homogène* :

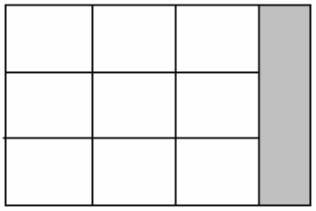
C'est une dissection uniforme caractérisée par la répétition d'un seul type de pièces.

c) *Un modèle de découpe générale* :

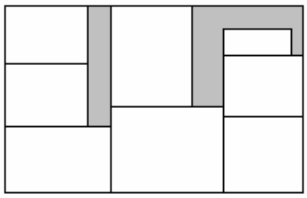
C'est un modèle de découpe réalisable constitué par la combinaison de bandes générales horizontales ou verticales.



(a)modèle de découpe uniforme



(b) modèle de découpe homogène



(c) modèle de découpe générale

Figure 4 : Les différents types de modèles de découpe**.**

II.1.4. Les points de découpe :

Les points de découpes représentent les abscisses (respectivement les ordonnés) sur lesquelles on peut effectuer des coupes verticales (respectivement horizontales). Ce procédé a été introduit par Herz [20] et repris ensuite par un certain nombre d'auteurs.

Les découpes verticales sont limitées par l'ensemble P qui est constitué des combinaisons linaires entre les différentes longueurs de chaque type de pièces de S, idem pour les découpes horizontales qui sont limitées par l'ensemble Q des combinaisons linéaires sur les différentes hauteurs des pièces de S. Ces ensembles sont définis par :

$$P = \left\{ p \in N \,/\, p = \sum_{i=1}^{n} l_i x_i \le L, \quad x_i \in N \right\}$$

$$Q = \left\{ q \in N \,/\, q = \sum_{i=1}^{n} h_i x_i \le H, \quad x_i \in N \right\}$$

II.1.5. Les différentes variantes du problème de découpe guillotine à deux dimensions (DGDD):

Le problème de DGDD peut engendrer plusieurs variantes dues aux différentes contraintes imposées et à leurs combinaisons. Dans ce qui suit nous donnons un bref aperçu de ces différentes variantes ainsi que les travaux effectués pour chacune d'elles :

II.1.5.1. Le problème de découpe guillotine contraint (DGDD-C)

Il consiste à déterminer un modèle de découpe tel que le nombre d'apparition de chaque type de pièce i, i=1,…,n ne doit pas excéder une valeur qu'on

notera b_i . Ce problème peut être modélisé comme suit sous forme non linéaire :

$$\max \quad \sum_{i=1}^{m}\sum_{j=1}^{J}\sum_{k=1}^{K} c_{ijk} a_{ijk} \qquad (1)$$

$$\sum_{j=1}^{J} l_j \lambda_j \le L \qquad (2)$$

$$\sum_{k=1}^{K} w_k \mu_k \le W \qquad (3)$$

$$\sum_{i=1}^{m} a_{ijk} \le \lambda_j \mu_k \text{ , pour chaque j, k} \qquad (4)$$

$$\sum_{j=1}^{J}\sum_{k=1}^{K} a_{ijk} \le b_i \text{ , pour chaque i} \qquad (5)$$

$$\textbf{avec} \quad \lambda_j, \mu_k, a_{ijk} \ge 0, \text{ entier} \quad , i = 1, ..., m; j = 1, ..., J; k = 1, ..., K. \qquad (6)$$

Tel que :

J, k représente le nombre des différentes longueurs et hauteurs.

$$c_{ijk} = \begin{cases} c_i & si \quad l_i \le l_j \quad et \quad w_i \le w_k \\ 0 & \text{sinon} \end{cases}$$

λ_j : Le nombre de fois où la longueur l_j a été découpée de L

μ_k : Le nombre de fois où la hauteur h_k a été découpée de H

a_{ijk} : Le nombre de rectangle de dimension $l_j \times h_k$ contenant la pièce de type i.

La fonction objectif (1) maximise la valeur totale des pièces à découper ; les contraintes (2) et (3) garantissent le non dépassement des dimensions du support initial ; la contrainte (4) limite la variable a_{ijk} à $\lambda_j . \mu_k$; la cinquième contrainte concerne l'accessibilité des pièces ; et la sixiéme et dernière contrainte fait référence à l'intégrité des variables.

On trouve dans la littérature plusieurs travaux réalisés dans ce domaine entre autre Christodides et Whitlock [6] qui ont présentés la méthode « Depth first search » pour résoudre des instances de petite taille.

II.1.5.2. Le problème de découpe guillotine non contraint (DGDD-NC)

Le problème est noté ainsi lorsque la valeur b_i n'est pas imposée. Ce problème peut être modélisé comme pour le cas contraint en suppriment les contraintes (4) et (5) ce qui permet d'avoir le modèle quadratique en nombre entier suivant : (Arenales et morabito 2000):

$$\max \quad \sum_{j=1}^{J}\sum_{k=1}^{K} c_{jk}\lambda_j \mu$$

$$\sum_{j=1}^{J} l_j \lambda_{\cdot} \le L$$

$$\sum_{k=1}^{r} w_k \mu_k \le W$$

$$\text{avec} \quad \lambda_j, \mu_k \ge 0, \text{integer}, j = 1, ..., J; k = 1, ..., K.$$

Cette variante du problème de DGDD a été résolue à l'optimum par J.E.Beasley [3] et Gilmores et Gomory [14]. Alors que Morales et al ont opté pour une méthode approche et ont présenté la stratégie du « Depth first search » et du « Hill Climbing »pour la résolution des grandes instances.

II.1.5.3. Le problème de découpe guillotine à deux dimensions avec contrainte de niveau (K-DGDD)

Une autre variante du problème est de considérer une contrainte sur le nombre totale de coupe, i.e. la somme des coupes guillotine verticales et horizontales engendrant une pièce de type i, i=1,…, n ne doit pas dépasser une certaine constante $k < \infty$. Dans ce genre de cas le problème est appelé *K*-DGDD et K représente le nombre d'étape de découpe (ED). la variante 2-DGDD avec la supposition que la première coupe de la première étape est parallèle à l'axe des longueurs et la première coupe de la deuxième étape est parallèle à l'axe des hauteurs peut être modélisée comme suit (Scheithauer 2002) [29] :

$$\max \quad \sum_{i=1}^{m}\sum_{j=1}^{P}\sum_{k=1}^{Q} c_i x_{ijk} \tag{7}$$

$$\sum_{k=1}^{Q} W_k \le W \tag{8}$$

$$\sum_{i=1}^{m} x_{ijk} \le 1, \quad \text{for all } j,\ k \tag{9}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{P} l_i x_{ijk} \le L, \quad \text{for all } k \tag{10}$$

$$\sum_{i=1}^{m} w_i x_{ijk} \le W_k, \quad \text{for all } j,\ k \tag{11}$$

$$\sum_{j=1}^{J}\sum_{k=1}^{Q} x_{ijk} \le b_i, \quad \text{for all } i \tag{12}$$

$$W_k \ge W_{k+1}, \quad \text{for all } k \tag{13}$$

$$\text{avec} \quad x_{ijk} \in \{0,1\},\ W_k \ge 0,\ i = 1, ..., m;\ j = 1, ..., P;\ k = 1, ..., Q. \tag{14}$$

Tel que :

$l_{\min}$ correspond à la plus petite longueur des pièces.

$h_{\min}$ Correspond à la plus petite hauteur des pièces.

P : le nombre maximum de bandes verticales tel que $P = \left\lfloor L / l_{\min} \right\rfloor$.

Q : le nombre maximum de bandes horizontales tel que $Q = \left\lfloor H / h_{\min} \right\rfloor$

L_j : Longueur de la j^{eme} bande verticale pour j=1,…, P.

h_k : hauteur de la k^{eme} bande horizontale pour k=1,…, Q.

$$x_{ijk} = k \begin{cases} 1_i \ \ si \ \ la \ \ piéce \ \ de \ \ type \ \ i \ \ est \ \ placée \\ \qquad\qquad dans \ \ le \ \ rec\tan gle \ \ l_j \times h_k \\ 0 \ \ \sin on \end{cases}$$

La fonction objectif (7) maximise la valeur totale des pièces à découper. La contrainte (8) garantie le fait que les hauteurs des bandes ne dépassent pas la hauteur du support initial ; la contrainte (9) impose le placement d'une seule pièce dans chaque rectangle $l_j \times h_k$; les contraintes (10) et (11) garantissent que les dimensions de chaque pièce placé dans le rectangle $l_j \times h_k$ ne dépassent pas les dimensions de ce dernier ; la 12^{e} contrainte fait référence à l'accessibilité des pièces ; la contrainte (13) élimine quelque symétrie et la 14^{e} et dernière contrainte précise que les valeurs l_j *et* h_k ne sont pas forcément des entiers.

La figure 5 montre une solution *2*-DGDD qui est produite en appliquant les phases suivantes :

Phase1 : le rectangle est découpé suivant sa longueur en un ensemble de bandes verticales.

Phase 2 : chacune de ces bandes verticales est prise individuellement et elle est découpée suivant sa longueur.

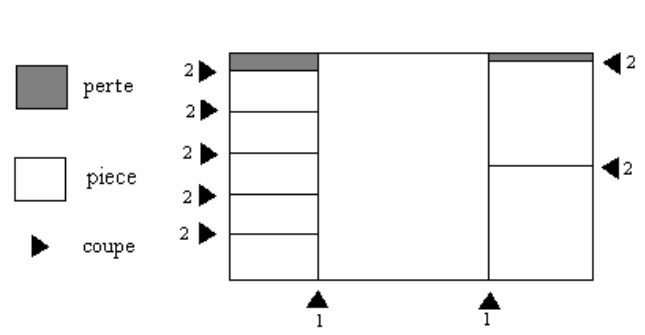


Figure 5 : Représentation d'une solution quand k est fixée à 2.

Parmi les travaux réalisés dans ce sens on cite Morabito. R. and Arenales. M. [26] qui ont traité le cas K-DGDD-C. Lodi et Monaci (2003) [24] ont proposé deux programmes linéaires en nombre entier pour la résolution du problème 2-DGDD.

Chapitre 2

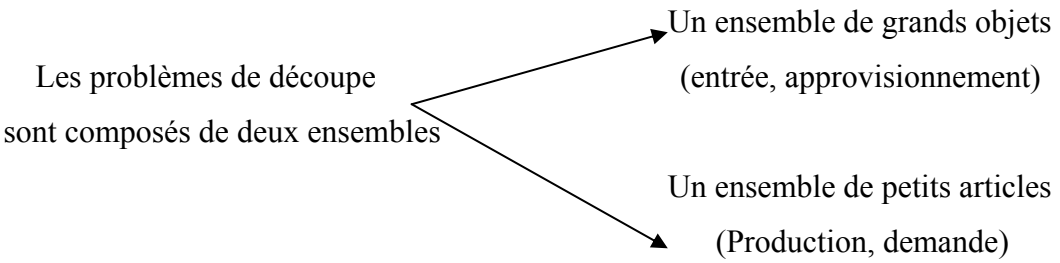
Les problèmes de découpe et leur classification

2.1. Introduction:

La typologie (classification) des problèmes de découpes présentée par Dyckhoff (1990) [7] a fourni initialement un excellent instrument pour l'organisation de la littérature existante et nouvelle.

Cependant, au cours des années quelques insuffisances de cette typologie sont devenues évidentes. En 2004 Gerhard Wäscher, Heike Haußner et Holger Schumann [36] ont présenté une nouvelle typologie partiellement basée sur les idées originales de Dyckhoff [7], mais qui présente un nouveau critère de classification, qui définit de nouvelles catégories de problème.

2.2. Structure des problèmes de découpe :



Ces ensembles sont définis dans une dimension ou plus.

- On sélectionne quelques-uns ou tout les petits articles.
- On les regroupe dans un sous-ensemble ou plus.
- On affecte chacun des sous-ensembles résultants à un des grands objets tel que l'on maintienne *l'état géométrique* càd :
  - Tous les petits articles du sous-ensemble se trouvent entièrement au dessous du grand objet.
  - Les petits articles ne se chevauchent pas.
- On optimise une fonction objective (unidimensionnelle ou multidimensionnelle) donnée.

Cinq sous problèmes peuvent être distingués, et doivent être résolus simultanément afin de réaliser un optimum " global ":

- La sélection d'un problème concernant les grands objets;
- La sélection d'un problème concernant les petits articles;
- Regrouper les problèmes concernant les petits articles choisis;
- Problème d'allocation concernant les tâches affectées des sous-ensembles des petits articles aux grands objets;
- Problème de la disposition concernant l'arrangement des petits articles sur chacun des grands objets sélectionnés tout en respectant les conditions géométriques.

2.3. Critère de classification :

2.3.1. Dimensions

Nous distinguons des problèmes à une, deux et trois dimensions. Les Problèmes de type ($n > 3$) sont considérés comme des variantes.

2.3.2. Genre de tâche affecté :

Nous présentons deux situations de base :

- **La m*aximisation du rendement (valeur de sortie)***

  Dans le cas de la maximisation du rendement, un ensemble de petits articles doit être assigné à un ensemble donné de grands objets qui est insuffisant pour contenir tous les petits articles. Tous les grands objets doivent être utilisés, et auxquels un choix (un sous-ensemble) de petits articles de valeur maximale doit être assigné.

- **La m*inimisation des entrées (valeur)***

  Un ensemble donné de petits articles sera assigné à un ensemble de grands objets. À la différence de ce qui précède, l'ensemble de grands objets est suffisant pour contenir tout les petits articles qui doivent être assignés à un choix (un sous-ensemble) de grand objet(s) de " valeur " minimale. Il n'y a aucun problème de choix concernant les petits articles.

  La" valeur " des objets / des articles doit être définie avec plus de précision et peut être représentée par des coûts, des revenus, ou des quantités matérielles. Il est également possible de traduire « la maximisation du rendement » et « la minimisation des entrées » par « la minimisation des pertes ».

2.3.3. Assortiment de petits articles

En ce qui concerne l'assortiment des petits articles nous distinguons trois cas :

- ***les Petits articles sont identiques :***

Tous les articles ont la même forme et la même taille. Dans le cas de la maximisation du rendement ,on peut supposer que ce type (unique) d'article a une demande illimitée.

- ***Assortiment faiblement hétérogène :***

Les petits articles peuvent être groupés dans peu de classes (par rapport à tout le nombre d'articles), dans lesquelles les articles sont identiques. La demande de chaque type d'article est relativement grande, et peut ou ne pas être limitée par une borne.

- ***Assortiment fortement hétérogène***

L'ensemble de petits articles contient très peu d'éléments de forme et de taille identiques. Si cela se produit, les articles sont traités comme différents éléments. En conséquence, la demande de chaque article est égale à un.

2.3.4. Assortiment de grands objets

Pour l'assortiment des grands objets nous présentons les cas suivants :

- ***Un seul grand objet***

L'ensemble des grands objets se compose d'un unique élément. Les dimensions de l'objet peuvent être fixé ou variable.

- ***Plusieurs grands objets***

Dans le cas de plusieurs grands objets, seules les dimensions fixes seront considérées. Par analogie aux catégories présentées pour l'assortiment des petits articles, nous distinguons :

- *De grands objets identiques* faiblement hétérogènes
- *Un assortiment fortement hétérogène de grands objets*

2.3.5. Forme des petits articles :

On distingue entre :

- *les petits articles réguliers* (rectangles, cercles, boîtes, cylindres, boules…)
- *les petits articles irréguliers*

2.4. Les Types de problème de découpe

2.4.1. Les types de base du problème

Les types de problèmes de base des problèmes de découpe sont développés par la combinaison des deux critères " type d'affectation " et " assortiment de petits articles ".

2.4.1.1. Types de la maximisation des sorties (production)

Pour les problèmes de ce type, les grands objets sont assurés seulement en quantité limitée ce qui ne permet pas une adaptation de tous les petits articles. Comme la valeur des articles adaptés doit être maximisée, tous les grands objets seront employés. En d'autres termes il y a un problème de choix concernant les petits articles, mais aucun concernant les grands objets. Nous distinguons parmi les types de problème de base de la maximisation des valeurs de sortie le problème suivant :

• **Problème De Sac à dos**

Il représente une catégorie de problème qui est caractérisée par un assortiment fortement hétérogène des petits articles qui doivent être assignés à un ensemble donné de grands objets. La disponibilité des grands objets est limitée tels que quelques petits articles seulement peuvent être adaptés et leur valeur doit être maximisée.

2.4.1.2. Types de la minimisation des entrées

Les problèmes de ce type sont caractérisés par un approvisionnement en grands objets assez grand pour contenir tous les petits articles. La valeur des grands objets nécessaires pour les adapter à tous les petits articles doit être réduite au minimum.

- **Problème de découpe d'un stock :**

Cette catégorie exige qu'un assortiment faiblement hétérogène de petits articles soit complètement assigné à un choix de grands objets de valeur, de nombre, ou de taille minimal.

2.4. 2. Problème de Types Intermédiaires

Ils tiennent compte de l'assortiment des grands objets comme critère de différentiation supplémentaire. La Figure 6 représente les types intermédiaires de problème liés à la maximisation du rendement. Les types intermédiaires liés à la minimisation d'entrée sont exposés dans la Figures 7.

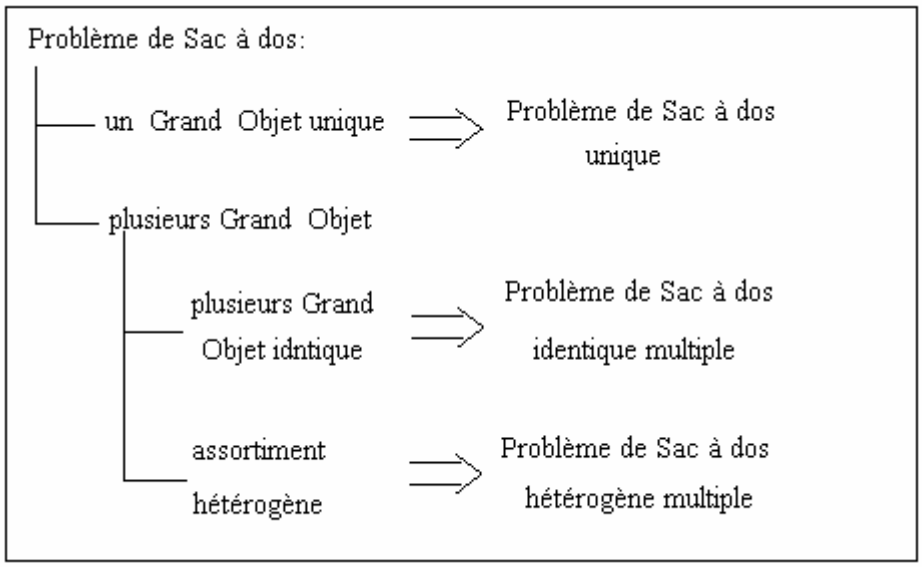


Figure. 6: Problème de sac à dos de type intermédiaire

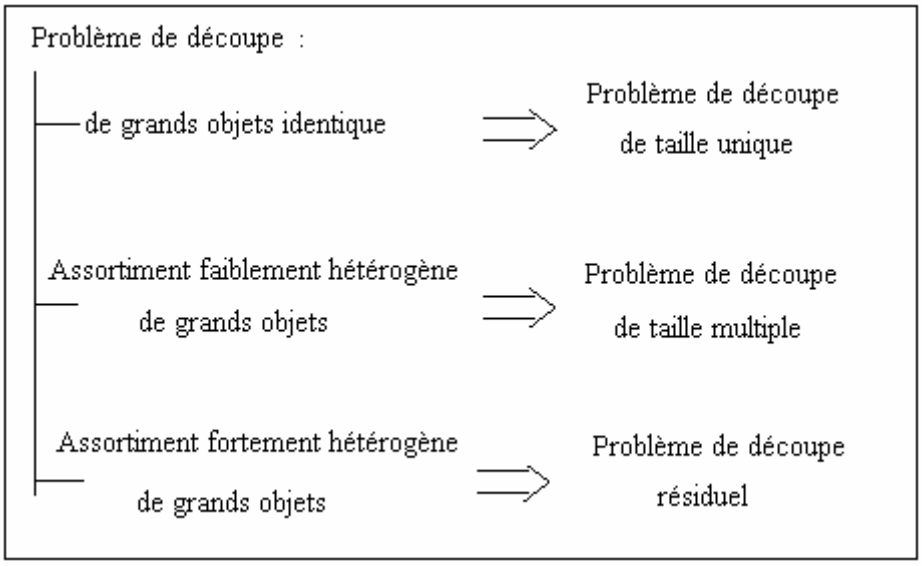


Figure. 7: Problème de découpe de type intermédiaire

2.4. 3. Les problèmes combinés :

Les problèmes de type combiné sont obtenus par l'application du critère « Dimension » alors que pour les problèmes à deux et trois dimensions ils sont obtenus par le critère « Forme des petits éléments». Les sous catégories résultantes sont caractérisées par des adjectifs qui sont ajoutés aux noms des types de problème intermédiaires selon le système suivant:

{1, 2, 3}-dimensionnel {Φ, rectangulaire, circulaires..., irréguliers}

2.5. Catégories des Problème de découpe :

2.5.1. Problème de Sac à dos Unique

- Problème (unidimensionnel) classique de sac à dos (Martello, Pisinger & Toth 2000 [27], Martello & Toth 1990 [25])
- Le problème de sac à dos à m-Contraintes (Schilling 1990) [34]
- Problème de sac à dos à deux dimensions (orthogonal unique) (Caprara & Monaci 2004 [5]; Healy & Moll 1996 [19].
- Problème de sac à dos à trois dimensions (orthogonal unique) (Pisinger 2002[31], Scheithauer 1999 [33]).
- Problème de sac à dos n-dimensionnel (orthogonal unique) (George, George & Lamar 1995 [15])

2.5.2. Problème de Sac à dos Identique Multiple

- Problème d'assemblage de coffre (boite) de cardinalité maximum (Labbé, Laporte & Martello 2003 [23]).

2.5.3. Problème de Sac à dos Hétérogène Multiple

- Problème multiple de sac à dos en 0-1 (Martello & Toth 1990 [25]; Pisinger 1999[30])

2.5.4. Problème de Découpe de Taille Unique

- Le problème de découpe unidimensionnel classique (Gilmore & Gomory 1963 [12], Wäscher & Gau 1996[35]).
- Le problème de découpe à deux dimensions classiques (Gilmore & Gomory 1965[13]).

2.5.5. Problème de découpe de taille Multiple :

- Problème de découpe de Papier (Gilmore & Gomory 1961 [11], Scheithauer 1991 [32], Belov & Scheithauer 2002 [4], Golden 1976 [16]).

2.5.6. Problème de Découpe Résiduel

- Problème de découpe unidimensionnel hybride Gradišar, Resinovič & Kljajić 2002 [18], Gradišar, Kljajić & Resinovič 1999 [17]).

2.6. Conclusions :

Les problèmes de découpe fournissent un champ d'application très vaste et très riche en nouvelles idées ce qui a poussé un grand nombre de chercheur à orienter leurs travaux vers ce domaine.

On remarque aussi qu'il existe une forte dualité entre les problèmes de découpe et d'assemblage. pour les problèmes de découpe, les grands objets sont pleins et doivent être vidés en découpant de petits articles. Par contre pour les problèmes d'assemblage, les grands objets sont vides et doivent être remplis par de petits articles.

Chapitre 3

Etude de quelques méthodes exactes pour le problème de DGDD-NC

3. 1. Introduction :

Durant ces dernières décennies plusieurs chercheurs se sont intéressés au problème de découpe. Certains ont opté pour des méthodes exactes préférant avoir une solution optimale quelque soit le temps d'exécution, alors que d'autre ont plutôt opté pour des méthodes approchées sacrifiant la solution optimale au profit du temps d'exécution.

Dans ce chapitre nous allons citer quelque méthode exacte de la littérature pour la résolution du problème de DGDD-NC : la méthode récursive de herz [20] pour la version non pondérée du problème, une méthode constructive qui s'appuie sur des techniques de programmation dynamique et une recherche arborescente avec critère de sélection du meilleur d'abord [21]. Nous citerons également l'heuristique de Beasley [3].

3.2. Les méthodes exactes :

Les méthodes de résolution exactes sont nombreuses et se caractérisent par le fait qu'elles permettent d'obtenir une ou plusieurs solutions dont l'optimalité est garantie. Parmi ces méthodes on cite :

3.2.1. La méthode d'exploration arborescente

Cette méthode construit itérativement une structure d'arbre qui est constituée de sommets reliés par des arcs orientés. Chaque sommet de l'arbre correspond à une solution partielle du problème considéré. L'arborescence évolue tant que des sommets sont sélectionnés. Cette méthode pratique une énumération intelligente de l'espace des solutions, puis elle partage cet espace en sous ensembles de plus en plus petits ; la plus grande partie étant éliminées par des calculs de borne avant d'être construits.

Appliquées aux problèmes NP –complet ces méthodes restent exponentielles, mais leur complexité est bien plus faible que pour une énumération complète. Pour les problèmes de grande taille, leur durée d'exécution devient exorbitante, il faudra alors ce rabattre sur les heuristiques.

Les méthodes arborescentes ont toute trois composantes communes :

- Une règle de séparation (stratégie de branchement).
- Une fonction d'evaluation.
- Une stratégie d'exploitation.

3.2.2. La programmation dynamique

La programmation dynamique peut être vue comme une amélioration ou une adaptation de la méthode diviser et régner. Ce concept a été introduit par Bellman, dans les années 50, pour résoudre des problèmes d'optimisation.

Dans la programmation dynamique, une solution d'un problème dépend des solutions précédentes obtenues des sous problèmes. Cette méthode permet aux sous problèmes de se superposer. Autrement dit, un sous problème peut être utilisé dans la solution de deux sous problèmes différents.

Dans la Figure 8, le problème à résoudre est à la racine, et les descendants sont les sous problèmes, plus faciles à résoudre. Les feuilles de ce graphe constituent des sous problèmes dont la résolution est triviale ; elles constituent souvent les données de l'algorithme. La programmation dynamique est une méthode dont les calculs se font de bas en haut : on commence par résoudre les plus petits sous problèmes, en combinant leur solution, pour obtenir celle des problèmes de plus grande taille.

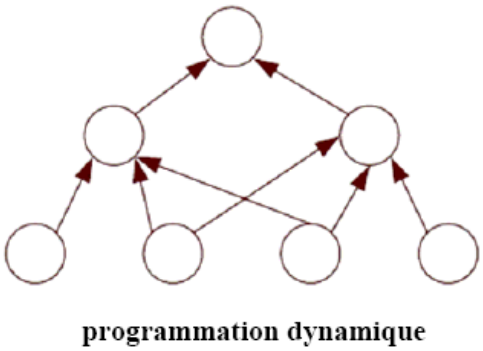


Figure 8 : Illustration de la technique de programmation dynamique.

3.3. Méthode récursive pour la résolution du problème de DGDD-NC pour la version non pondérée :

3.3.1. La propriété récursive de Herz :

N'importe quelle solution optimale issue de l'algorithme de Herz [20] a la propriété récursive suivante:
Ou bien le rectangle initial est dans **S,** ou bien la première ligne de dissection fournie deux rectangles, dont chacun est disséqué de façon optimale. L'algorithme consiste à essayer toute les premières lignes possibles de dissection et retenir celle qui donne la valeur maximale pour les deux sous rectangles.

3.3.2. Structure de l'algorithme de Herz :

La procédure générale de l'algorithme avec la propriété récursive principale est présentée sous forme d'une structure arborescente, imposant toutes les coupes possibles sur le rectangle initial, ensuite une méthode de limitation est imposée sur les découpes horizontales et verticales afin d'aboutir à une méthode de séparation et d'évaluation utilisant la stratégie de développement en profondeur.
Les solutions de cet algorithme ont aussi une structure particulière, chaque pièce est placée en bas à gauche.

Soit p et q les vecteurs colonne des coordonnées p_i, q_i respectivement, Z un vecteur ligne avec n coordonnées entières non négatives. Un nombre fini de tels vecteurs satisfait les conditions $zp \le a$ et $zq \le b$.

P et Q les ensembles finis correspondants aux valeurs zp et zq c à d les ensembles des combinaisons linéaires des longueurs et des hauteurs.

$$P = \left\{ zp / p_i \in N / p_i = \sum_{j=1}^{n} l_j x_j \le L, \quad x_j \in N \right\}$$

$$Q = \left\{ zq / q_i \in N / q_i = \sum_{j=1}^{n} h_j x_j \le H, \quad x_j \in N \right\}$$

Théorème 3.1 :

Pour n'importe quelle dissection *de R,* il existe une dissection de valeur supérieure ou égale avec toutes les abscisses dans *P* et toutes les ordonnées dans Q. Ici nous indiquons la distance d'une ligne de dissection verticale à l'arête gauche de *R* comme abscisse et la distance d'une ligne de dissection horizontale à l'arête inférieure comme ordonnée.

La procédure consiste à forcer les rectangles utilisables au fond à gauche. Une dissection avec toutes les coordonnées dans *P* et *Q* s'appelle ***une dissection* canonique**.

Démonstration :

Soit a et b les points de découpe horizontale et verticale respectivement**.** Si ***a*** est inférieur à chaque p_i alors cette dissection ne donne aucun rectangle utilisable et par conséquent elle est de même valeur qu'une dissection" vide " (idem si ***b*** est inférieur à chaque q_i)**.**

Supposons que le théorème est vrai pour chaque rectangle ***R'*** de dimensions *l' et h'* tels que $l' \le a$ et $h' \le b$ (mais $a + l' < b + h'$) . Considérons une dissection *D de R.*

Nous pouvons supposer que la première ligne de dissection L est verticale, l'autre cas étant tout à fait semblable.

Par l'hypothèse d'induction, il existe une dissection D_1 du rectangle gauche R_1 de valeur égale ou supérieure à la partie gauche *de D* et avec toutes les abscisses dans ***P*** et toutes les ordonnées dans Q; il existe alors une dissection D_2 du rectangle droit R_2 de valeur égale ou supérieure à la partie droite *de D* et avec toutes les abscisses (compté ***L)*** dans ***P*** et toutes les ordonnées dans Q.

Si l'abscisse L est dans P**,** l'union *de* D_1 et *de* D_2 est une dissection satisfaisant la conclusion, que P est additif. Plus précisément, $x \in P$, $x' \in P$ et $x + x' \leq a$ implique que $x + x' \in P$**.**

Si l'abscisse ***L*** n'est pas dans ***P*** alors chaque rectangle *de* D_1 avec son arête droite sur ***L*** est non utilisable; l'abscisse de l'extrême droite des arêtes gauches correspondantes est dans ***P.*** Maintenant nous obtenons la dissection de ***R*** en traçant la ligne verticale ***L '*** à cette abscisse, supprimant les lignes *de* D_1 du côté droit *de L '* et translater R_2 avec D_2 *de L à L '* comme le montre la figure 9.

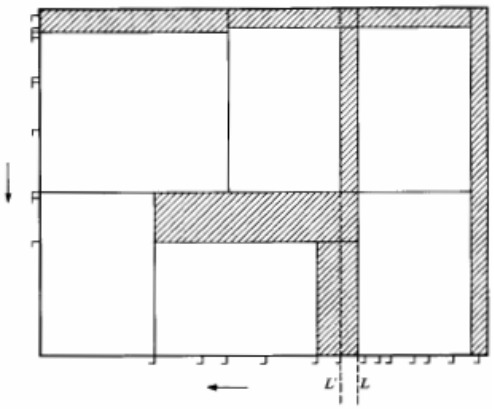


Figure 9 : la dissection canonique des rectangles non utilisables sont ombragés. Chacune des deux pièces principales est canoniquement disséquée. P et Q sont représentés le long des arêtes.

3.3.2. Les effets de symétrie :

Ils permettent de limiter les points de découpe à la moitié de la longueur et de la hauteur du rectangle initial ou d'un sous rectangle en cour de dissection.

Théorème 3.2 : [20]

Pour chaque dissection non homogène canonique *de* ***R*** il existe une dissection canonique de valeur égale tel que la coordonnée de la première ligne est au maximum égale à la moitié de la dimension *de* ***R.***

Ce théorème réduit l'analyse *de P* et de *Q* de moitié. Il s'applique, naturellement, à chaque sou rectangle.

3.3.5. Mémorisation des solutions :

Si au cours du calcul, le même rectangle partiel s'avère être considéré plusieurs fois, il est évidemment utile de stocker sa valeur plutôt que de la calculer à chaque fois.

3.3.6. Les bornes de l'algorithme :

- ***Calcul de la meilleure découpe homogène :***

Pour le rectangle initial ou pour chaque sous rectangle produit on calcule la meilleure découpe homogène constituée d'une unique pièce de l'ensemble S.

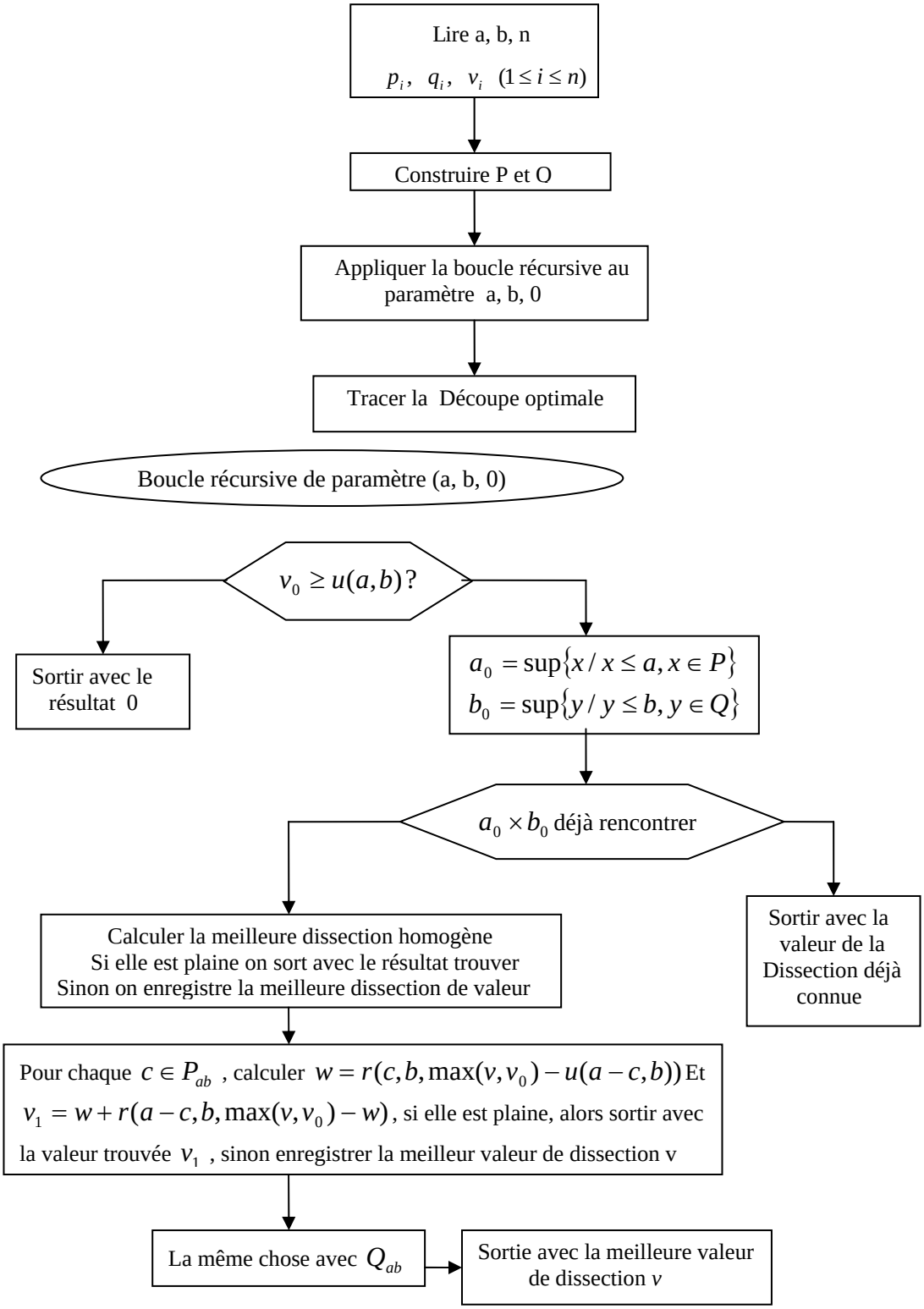
- ***Dissection pleine***

En recherchant une dissection optimale nous pouvons nous arrêter lorsqu'on trouve une dissection de valeur égale à la surface : ceci correspond à une solution sans perte.

- ***Rectangle déjà rencontré***

Supposons qu'à un certain moment du calcul, la meilleure valeur de dissection à déjà été trouvée pour un rectangle de dimension $\alpha \times \beta$ après qu'une coupe a été effectuée à l'abscisse γ. *Donc* la valeur de dissection optimale du rectangle gauche $\gamma \times \beta$ a déjà été calculée il est alors inutile d'étudier les dissections du rectangle droit $(\alpha - \gamma) \times \beta$, ce qui permet de passer à la prochaine étape de l'algorithme.

Algorithme récursif



3.3.7. Déroulement de l'algorithme sur une instance :

On a déroulé cet algorithme sur l'exemple :

(L, H) = (8,7) avec $S = \{(3,3),(3,4),(5,3),(4,6)\}$.

On a obtenue les résultats suivants :

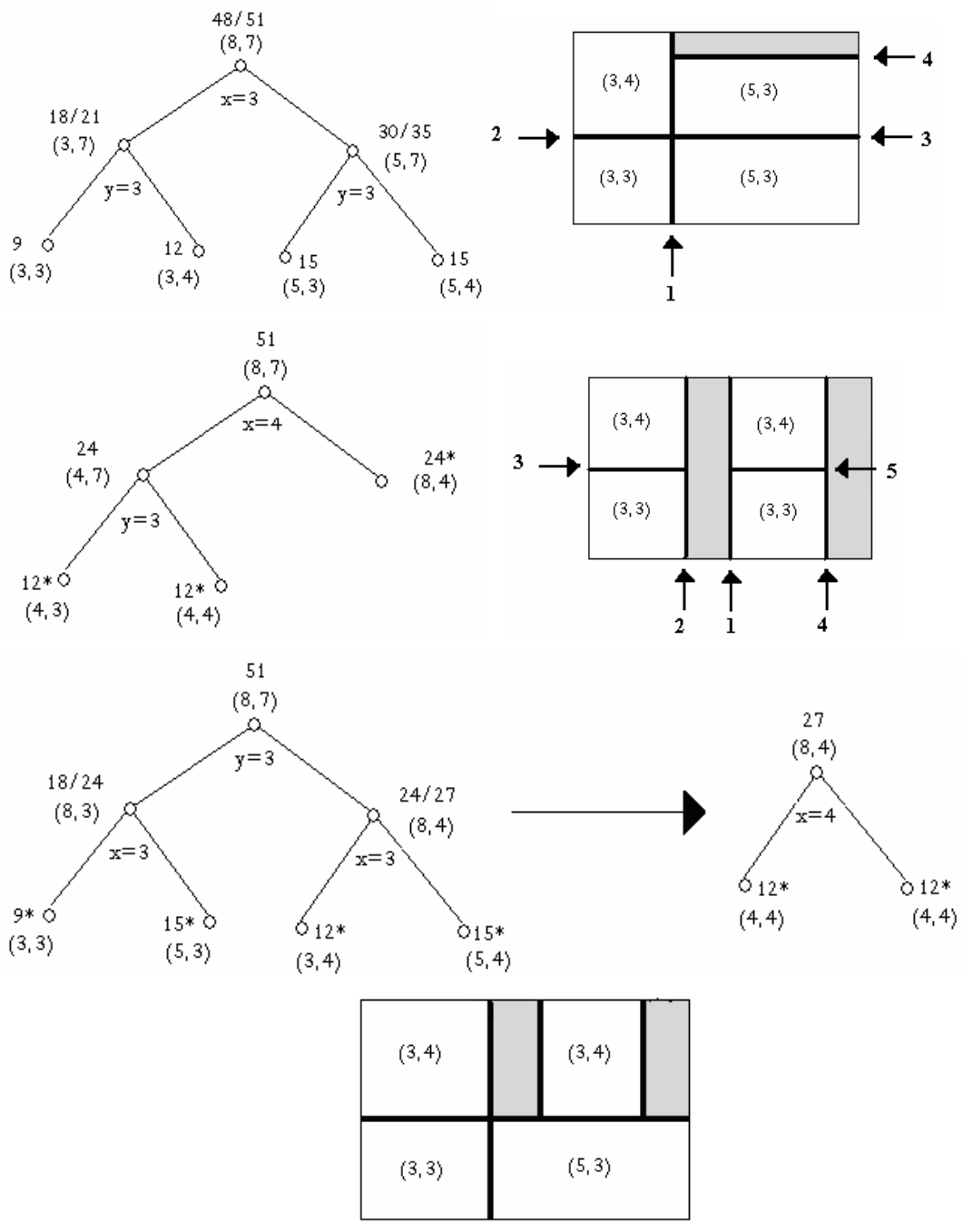


Figure 10 : représentation graphique de la solution

3.4. Etude d'une méthode constructive exacte basée sur le principe de recherche du meilleur d'abord et de la programmation dynamique:

3.4.1. Principe de l'algorithme :

Cet algorithme se base sur une procédure constructive qui permet d'obtenir des modèles de découpe guillotine sur le support initial R et ceci grâce à des constructions verticales et horizontales.

Définition 3.1 :

Une construction horizontale de deux rectangles (α_1, β_1) et (α_1, β_1) produit un nouveau rectangle de dimension $(\alpha_1 + \alpha_2, \max\{\beta_1, \beta_2\})$.

Définition 3.2:

Une construction verticale de deux rectangles (α_1, β_1) et (α_1, β_1) produit un nouveau rectangle de dimension $(\max\{\alpha_1, \alpha_2\}, \beta_1 + \beta_2)$.

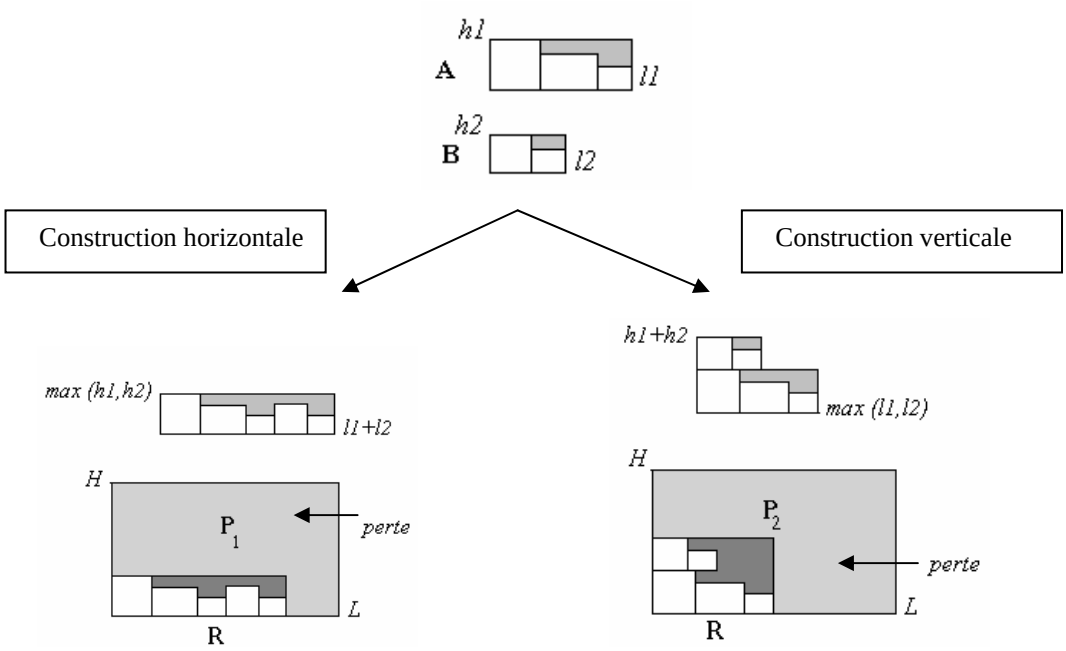


Figure 11 : Représentation des constructions horizontale et verticale

Chaque rectangle résultant d'une construction horizontale ou verticale et satisfaisant la condition $(l_R, h_R) \leq (L, H)$ est considéré comme étant un rectangle guillotine.

L'approche utilise le principe de type « branch and bound » : à partir d'un rectangle guillotine ***RG*** (placé à l'intérieur du rectangle initiale ***R***) on construit un autre rectangle guillotine de dimension supérieure qui est une combinaison entre ***RG*** et un autre rectangle déjà construit et ceci en utilisant les constructions horizontales ou verticales.

A chaque rectangle guillotine on fait correspondre :

Une fonction intermédiaire ***g (RG) :***
Représentant la somme des profits des pièces contenues dans ***RG.***

Une fonction complémentaire ***h (RG) :***
Représentant la valeur optimale du modèle de découpe sur le reste de la surface ***F***.

On désigne par :

$$f(RG) = g(RG) + h(RG)$$

La valeur maximale de l'ensemble des modèles de découpe contraint de contenir le rectangle guillotine ***RG.***

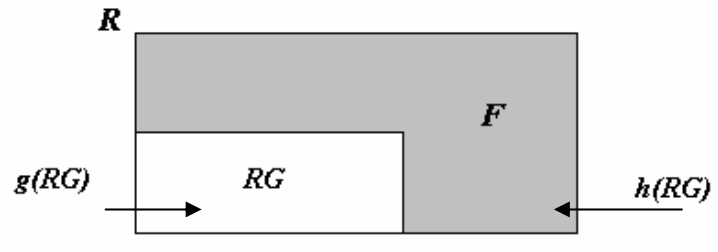


Figure 12 : Paramètre d'évaluation d'un rectangle guillotine.

Comme le temps de calcul de la valeur ***h(RG)*** est assez long on se limite à l'estimation d'une borne supérieure :

$$f`(RG) = g(RG) + h`(RG)$$

Sur le rectangle guillotine ***RG*** .

On peut schématiser ce processus de construction par une arborescence dans laquelle chaque nœud correspond à un niveau de construction. En modélisant le problème sous forme de problèmes de sac à dos unidimensionnels résolus par la programmation dynamique on peut évaluer les bornes inférieures et supérieures.

3.4.2. Critères d'optimalité :

3.4.2.1. Borne supérieure sur les sommets :

Théorème 3.3: [28]

Soit $\boldsymbol{RG} = (l_{RG}, h_{RG})$ un rectangle guillotine, une borne supérieure pour *f (RG)* est donnée par :

$$f`(RG) = g(RG) + h`(RG)$$

Où :

$$h'(RG) = S(F).\max\left\{ c_j \Big/ l_j h_j \, , j \in F \right\}$$

et *S(F)* : la surface de *F*.

Corollaire 3.1 : [28]

Une borne supérieure initiale pour le rectangle R est donnée par :

$$B_{\sup}(\mathrm{R}) = L.H.\max\left\{ c_j \Big/ l_j h_j \right\}$$

3.4.2.2. Bornes inférieures :

La borne de départ de cette approche nécessite la résolution de quatre problèmes de sac a dos unidimensionnels :

-deux engendrent les différentes bandes optimales horizontales et verticales.

- les deux autres permettent la construction de deux modèles de découpe réalisable :

  - l'un est un modèle de découpe horizontale, obtenue comme combinaison Verticale des bandes optimales horizontale de différentes hauteurs, sa valeur est notée B_{hor}
  - l'autre est un modèle de découpe verticale, obtenue comme combinaison horizontale des bandes optimales verticales de différentes longueurs. Sa valeur est noté B_{ver} .

➢ **Borne inférieure initiale :**

$$B_{\inf}(R) = \max\{B_{hor}, B_{ver}\}$$

➢ **Borne inférieure intermédiaire :**

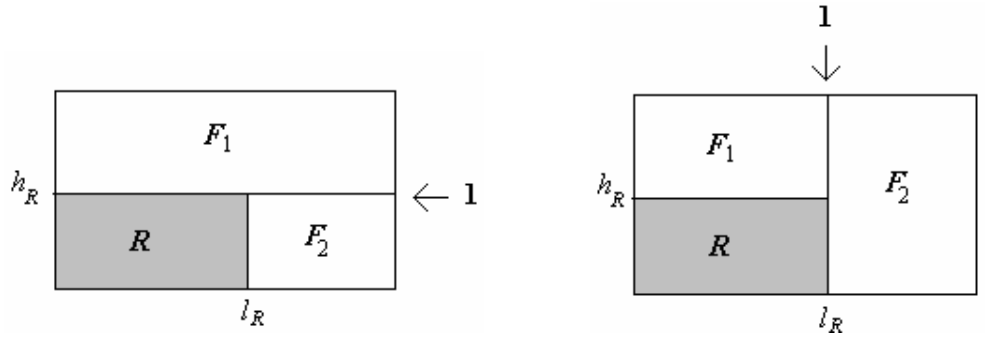
Les sommets intérieurs de l'arborescence de recherche sont associés aux rectangles Guillotine *RG*. A chaque rectangle guillotine construit on lui associe deux solutions faisables obtenues en subdivisant la région F en deux sous rectangles F1 et F_2 , afin de déterminer si on doit commencer par une coupe verticale ou horizontale pour obtenir le rectangle guillotine *R*

- on essaye de découper le rectangle initial R= *(L, H)* verticalement à l'abscisse l_{RG} on obtient deux sous rectangles et on choisit le plus grand suivant sa surface $F_{l_{RG}}$.
- on essaye ensuite de découper le rectangle initial R= *(L, H)* horizontalement à l'ordonnée h_{RG} on obtient deux sous rectangles et on choisit le plus grand suivant sa surface $F_{h_{RG}}$.

Donc :

- si $F_{h_{RG}} > F_{l_{RG}}$ alors la première coupe effectuée afin d'obtenir le rectangle *RG* est horizontale (figure 13(a)).

- si $F_{h_{RG}} < F_{l_{RG}}$ alors la première coupe effectuée pour obtenir le rectangle *RG* est Verticale (figure 13(b)).



(a) (b)

Figure 13 : le choix de la première coupe guillotine.

Pour ces deux sous rectangles F_1 et F_2 on effectue les opérations suivantes :

1- on détermine le modèle de découpe de valeur L_1 du plus grand rectangle par la résolution des problèmes de sac à dos en utilisant la programmation dynamique.
2- La solution faisable est complétée par la meilleure solution homogène L_2 sur le petit rectangle.

Donc la valeur du meilleur modèle de découpe faisable associé au sous rectangle ***RG*** est donnée par :

$$\text{Lower}(RG) = g(RG) + L_1 + L_2$$

3.4.3. Enoncé de l'algorithme

Entrée : une instance du problème DGDD-NC.

Sortie : la valeur de la solution optimale $B_{\inf}(R)$;

Initialisation :

Ouvert = $\left\{ p_i, \quad i = 1,...,n \right\}$

Avec p_i une composante de la liste Ouvert qui contient les dimensions et les valeurs : g , L_1, L_2 , h' et f' de la i^{eme} pièce pour i=1,…,n.

Fermé = $\{\ \}$; *Fin = faux* ;

Calculer la borne inférieure initiale de *R* notée $B_{\inf}(R)$;

Calculer la borne supérieure initiale de *R* notée $B_{\sup}(R)$;

Etape principale :

Si ($B_{\sup}(R)$- $B_{\inf}(R)$=0) alors sortir avec $B_{\inf}(R)$

Sinon

Répéter

Choisir un rectangle P tel que $écart = \max_{P \in ouvert} \{ f'(P) - \mathrm{B}_{\inf}(\mathrm{R}) \}$

Si ($écart \leq 0$) alors *fin* := vrai

Sinon

Transférer P de la liste Ouvert vers la liste Fermé ; construire le rectangle guillotine G tel que :

1- le rectangle G est construit par le placement vertical ou horizontal de P avec les éléments de la liste Fermé dont les valeurs sont supérieures ou égales à $B_{\inf}(R)$

2- les dimensions de chaque élément de G sont plus petites que celle de *R*; Poser $\mathrm{Ouvert} = \mathrm{Ouvert} \cup \mathrm{G}$ avec les valeurs appropriées g , L_1, L_2 , h' et f' (chaque élément de G vérifie $f'(P) > B_{\inf}(R)$) ;

Effectuer la mise à jour de $B_{\inf}(R)$.

Jusqu'à (fin) ou ($ouvert = \phi$) ;

Fin : Sortir avec la solution optimale $B_{\inf}(R)$

3.4.4. Déroulement de l'algorithme :

Soit l'instance suivante du problème DGDD-NC non pondéré :

$R = (L, H) = (13,9)$

$S = \{(5,5),(3,4)\}$;

	l_i	h_i	c_i
p_1	5	5	25
p_2	3	4	12

- ***Initialisation :***

1)- Calcul de la borne supérieure initiale $B_{\sup}(R)$:

$$B_{\sup}(R) = \text{L.H} = 13.9 = 117$$

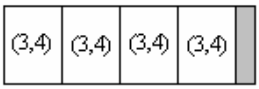
$$B_{\sup}(R) = 117$$

2)- Calcul de la borne inférieure initiale $B_{\inf}(R)$:

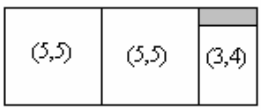
$$B_{\inf}(R) = \max\{B_{hor}, B_{ver}\}$$

C'est le maximum entre les deux modèles de découpe horizontale et verticale obtenues à partir de la combinaison des bandes optimales horizontales pour le modèle de découpe horizontale et verticale pour le modèle de découpe verticale.

- Générations des bandes optimales horizontales de longueur L = 13 et de différente hauteur.

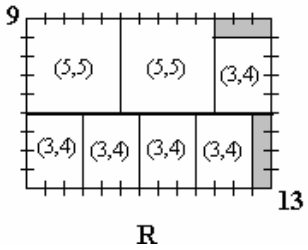


$F_1(13) = 48$



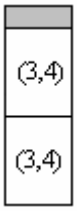
$F_2(13) = 62$

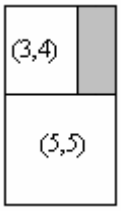
- Génération du modèle de découpe obtenue par combinaison verticale des bandes optimales horizontales



$B_{hor} = 110$

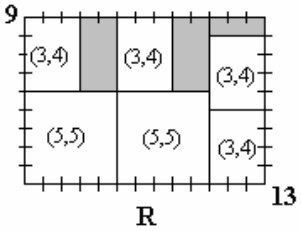
- Génération de bandes optimales verticales de hauteur H et de longueur l_i, i=1,…, r'





$F_1(9) = 24$ $F_2(9) = 37$

- Génération du modèle de découpe obtenu par combinaison verticale des bandes optimales horizontales



$B_{ver} = 98$

Donc $B_{\inf} = 110$

➢ ***Etape principale :***

$B_{\sup}(R) - B_{\inf}(R) = 7 \neq 0$

$\rightarrow$ sinon

- *Itération1 :*

$Ouvert = \{p_1, p_2\}$

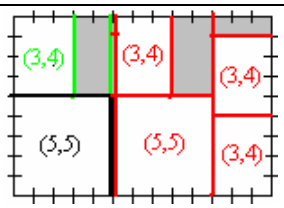
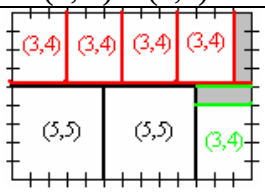
$écart = \max_{p \in ouvert} f'(p_1) - B_{\inf}(R) = 117 - 110 = 7$

$p_1 \rightarrow Fermé = \{p_1\}$

$G = \{R_1 = H(p_1, p_1),\ \ R_2 = V(p_1, p_1)\ \ (sommet\ \ \ stirilisé)\}$

$Ouvert = ouvert \cup G$

$B_{\inf}(R) = 110$

p_1	 $B_{\inf}(R) = 110$ (l , h) = (5,5)	$g' = 25 = 5.5$ $h' = 92 = (13.9) - 5.5 = 117 - 25$ $f' = 117 = 25 + 92$ $L_1 = 61 = K_H^{hor}(13 - 5)$ $L_2 = 12 = 3.4$ $lower = 98 = 25+61+12$
R_1	 $B_{\inf}(R) = 110$ R=(l , h) = (10,5)	$g' = 50 = 10.5$ $h' = 67 = 117 - 10.5 = 117 - 50$ $f' = 117 = 50 + 67$ $L_1 = 48 = K_L^{ver}(9 - 5)$ $L_2 = 12 = 3.4$ $lower = 110 = 50 + 48 + 12$

- *Itération 2 :*

$Ouvert = \{p_2, R_1\}$

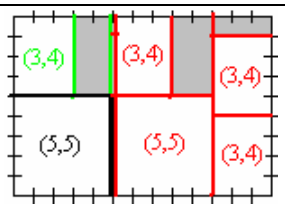
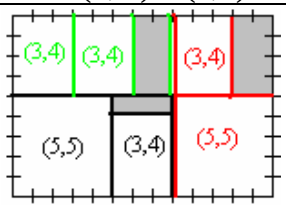
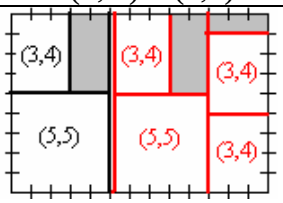
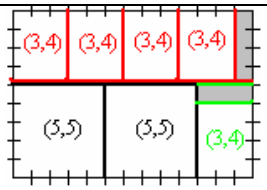
$écart = \max_{p \in ouvert} f'(p_2) - B_{\inf}(R) = 117 - 98 = 7$

$p_2 \rightarrow Fermé = \{p_1; p_2\}$

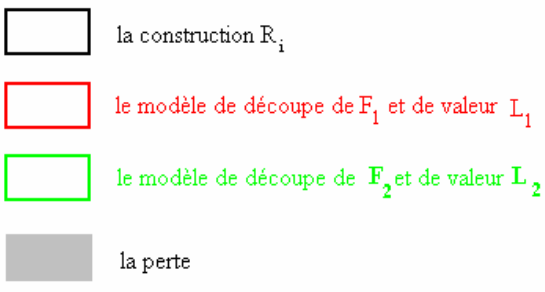
$G = \{R_3 = H(p_2, p_1); R_4 = V(p_2, p_1); R_5 = H(p_2, p_2); R_6 = V(p_2, p_2)(stirilisé)\}$

$Ouvert = ouvert \cup G$

$B_{\inf}(R) = 98$

p_2	 $B_{\inf}(R) = 110$ (l , h) = (5,5)	$g' = 25 = 5.5$ $h' = 92 = (13.9) - 5.5 = 117 - 25$ $f' = 117 = 25 + 92$ $L_1 = 61 = K_H^{hor}(13-5)$ $L_2 = 12 = 3.4$ $lower = 98 = 25+61+12$
R_3	 $B_{\inf}(R) = 110$ R= (l , h) = (8,5)	$g' = 37 = 5.5 + 3.4$ $h' = 77 = 117 - 8.5$ $f' = 114 = 37 + 77$ $L_1 = 37 = K_H^{hor}(13-8)$ $L_2 = 24 = 3.4 + 3.4$ $lower = 98$
R_4	 $B_{\inf}(R) = 110$ R= (l , h) = (5,9)	$g' = 37 = 5.5 + 3.4$ $h' = 72 = 117 - 5.9$ $f' = 109 = 37 + 72$ $L_1 = 49 = K_H^{hor}(13-5)$ $L_2 = 0$ $lower = 98 = 37+61+0$
R_5	 $B_{\inf}(R) = 110$ R=(l , h) = (10,5)	$g' = 50 = 10.5$ $h' = 67 = 117 - 10.5 = 117 - 50$ $f' = 117 = 50 + 67$ $L_1 = 48 = K_L^{ver}(9-5)$ $L_2 = 12 = 3.4$ $lower = 110 = 50 + 48 + 12$

On sort avec une solution optimale $B_{\inf}(R)=110$ à l'issue de **41** itérations engendrant **88** constructions



3.5. Algorithmes approché pour le problème DGDD :

Dans son article, Beasley [3] a soumis deux propositions : l'une concerne la correction d'une erreur détectée dans l'algorithme de Gilmore et Gomory [14] basée sur la procédure de programmation dynamique pour la résolution de la variante K-DGDD, et l'autre consiste à développer une nouvelle heuristique pour la résolution du problème DGDD sans aucune contrainte.

3.5.1. Correction de l'erreur de l'algorithme de Gilmore et Gomory pour la résolution des problème K-DGDD :

Lorsqu'on a affaire a la variante K-DGDD on alterne les coupes (horizontale et verticale) à chaque ED.

Soit $P = [1,2,...,L-1]$ l'ensemble des longueurs possibles pour les coupes parallèles à l'axe y, $Q = [1,2,...,H-1]$ l'ensemble des hauteurs possibles pour les coupes parallèles à l'axe x et k le nombre d'ED. On définit alors :

$F(k,x,y)$ = la valeur optimale d'une k^{ieme} étape de découpe sur un rectangle de dimension *(x, y)* tel que la 1er ED soit parallèle à l'axe *x.*

$G(k,x,y)$ = la valeur optimale d'une k^{ieme} étape de découpe sur un rectangle de dimension *(x, y)* tel que la 1er ED soit parallèle à l'axe *y.*

Lorsque k est de valeur nulle dans les définitions précédente ceci correspond au fait qu'une ED fournie au maximum une seule pièce à partir du rectangle (x, y). On peut définir alors quatre situations :

S=1- Aucune ED n'est permise :

$$F(0,x,y) = \max(0, c_i / l_i = x, \quad h_i = y \quad i = 1,...,m) \text{ ..... (1)}$$

S=2- On ne permet que les ED parallèles à l'axe x:

$$F(0,x,y) = \max(0, c_i / l_i = x, \quad h_i \le y \quad i = 1,...,m) \text{ ……..(2)}$$

S=3- On ne permet que les ED parallèles à l'axe y :

$$F(0,x,y) = \max(0, c_i / l_i \le x, \quad h_i = y \quad i = 1,...,m) \text{ ……..(3)}$$

S=4- On permet les ED parallèles aux axes x et y :

$$F(0,x,y) = \max(0, c_i / l_i \le x, \quad h_i \le y \quad i = 1,...,m) \text{ ……..……..(4)}$$

Les valeurs de $F(0, x, y)$ et $G(0, x, y)$ sont identiques.

A présent on peut développer une procédure de programmation dynamique pour $F(n,L,H)$ et $G(n,L,H)$.

Supposons que la première ED soit verticale, et qu'elle n'existe pas, on passe alors à la deuxième ED qui par alternance est horizontale, dans ce cas la deuxième ED sera considérée comme étant la première, et le nombre d'ED passe de k à k-1, par conséquent nous obtenons :

$$F(k,x,y) = \max[F(0,x,y); F(k,x,y_1) + F(k,x,y-y_1), \\ y_1 \in Q, y_1 \le K_1(y); G(k-1,x,y)] \text{.....(5)}$$

$$\text{Où} \quad K_1(y) = y/2 \text{ si S=2 ou S=4}$$

$$= y\text{-}1 \text{ sinon}$$

De même si la première ED est horizontale :

$$G(k,x,y) = \max[G(0,x,y); G(k,x_1,y) + G(k,x-x_1,y), \\ x_1 \in P, x_1 \le K_2(x); F(k-1,x,y)] \text{.....(6)}$$

$$\text{Où} \quad K_2(x) = x/2 \text{ si S=3 ou S=4}$$

$$= x\text{-}1 \text{ sinon}$$

Les équations (5) et (6) s'appliquent pour chaque $k \geq 1$ et chaque rectangle (x, y). Ces équations sont les procédures de base pour la programmation dynamique récursive, tel que la solution optimale est le maximum entre $F(n, L, H)$ et $G(n, L, H)$, et les équations (1)… (4) fournissent les conditions initiales de la récursion.

Une erreur existe car le développement des fonctions des différentes directions de la première ED a été mal élaboré. Alors que la direction de la 1er ED est fixée, la récursion donnée par Gilmore et Gomory alterne le sens des coupes à chaque ED.
La procédure de programmation dynamique récursive peut être améliorée en utilisant un modèle normal.

3.5.2. Modèle normal :

Le modèle normale fut utilisé par Herz [20] et qu'il appela dissection canonique mais aussi par Christofides et Whitlock [6]. Chaque pièce (ou chaque coupe) peut être déplacé (figure 14) jusqu'à l'arête gauche et le rebord inférieur de la pièce adjacente ou du support initial R.

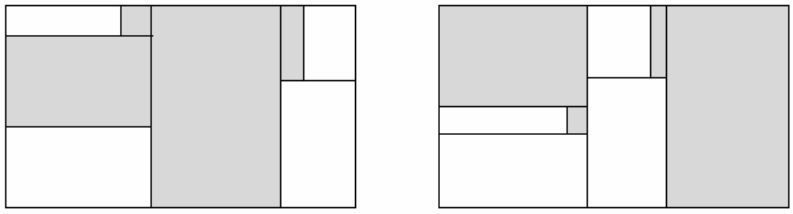


Figure14 : Modèle normal

Notons qu'une coupe à l'abscisse $x > L - \min(l_i / i = 1,...,n)$ peut entraîner le fait que l'on ne puisse pas extraire de pièce de la partie droite.
Par conséquent l'ensemble P des différentes longueurs possibles des coupes peut être remplacée par l'ensemble

$$P = \left[x / x = \sum_{i=1}^{n} l_i a_i, 1 \le x \le L - K_3, a_i \ge 0 \quad pour \quad i = 1,...,n \right] \ldots\ldots\ldots\ldots (7)$$

Où $K_3(x) = \min(l_i / i = 1,...,n)$ si T = 3 ou T = 4

= 1 sinon

De même pour l'ensemble des differentes hauteurs :

$$Q = \left[y / y = \sum_{i=1}^{n} h_i b_i, 1 \le y \le H - K_4, b_i \ge 0 \quad pour \quad i = 1,...,n \right] \ldots\ldots\ldots\ldots (8)$$

Où $K_4(x) = \min(h_i / i = 1,...,n)$ si T = 2 ou T = 4

= 1 sinon

Le modèle normal peut être utilisé pour améliorer la procédure de programmation dynamique récursive.

Soit

$$p(x) = \max(0, x_1 / x_1 \le x, x_1 \in L) \qquad x<L \ldots\ldots\ldots\ldots\ldots\ldots (9)$$

$$q(y) = \max(0, y_1 / y_1 \le y, y_1 \in H) \qquad y<H \ldots\ldots\ldots\ldots\ldots\ldots (10)$$

p(x) est la longueur la plus proche de x dans l'ensemble *P [p(x) ≤x],* et nous posons p(x)=0 s'il n'existe pas de telle longueur (resp. pour q(x)).

Afin de simplifier l'algorithme on définit p (L)=L et p (H)=H.

Soit

$$F[k, p(x), q(y)] \ge F(k, x, y) \quad k \ge 0; \quad x = 1,2,...,L; \quad y = 1,2,...,H \ldots\ldots (11)$$

$$G[k, p(x), q(y)] \ge G(k, x, y) \quad k \ge 0; \quad x = 1,2,...,L; \quad y = 1,2,...,H \ldots\ldots (12)$$

F (k, x,y) est remplacée par :

$$F(k, x, y) = \max[F(0, x, y); F(k, x, y_1) + F[k, x, q(y - y_1)],$$

$$y_1 \in Q, y_1 \le K_1(y); G(k-1, x, y)] \ldots\ldots\ldots\ldots\ldots\ldots (13)$$

Et celle de G(k,x,y) est remplacée par

$$G(k,x,y) = \max[G(0,x,y); G(k,x_1,y) + G[k, p(x-x_1), y],$$
$$x_1 \in P, x_1 \le K_2(x); F(k-1,x,y)]..... (14)$$

Il devient évident qu'en calculant $F(n,L,H)$ et $G(n,L,H)$ en utilisant les équations (13) et (14) il est inutile de calculer F(k,x,y) et G(k,x,y) pour toute les valeurs de x et y et tout les k ($\le n$), il suffit de se restreindre à $x \in L^0$ ($L^0 = P \cup \{L\}$) et $y \in H^0$ ($H^0 = Q \cup \{H\}$).

Le modèle optimal pour chaque rectangle (x, y) est dominé par le modèle optimal du rectangle (p(x), q(x)) (équation 11 et 12). Nous obtenons ainsi la version modifiée de la procédure de programmation dynamique récursive :

$$F(k,x,y) = \max[F(0,x,y); F(k,x,y_1) + F[k,x,q(y-y_1)], y_1 \in Q, y_1 \le K_1(y); G(k-1,x,y)]$$
$$k \ge 1; \quad x \in L^0, \quad y \in H^0 \text{ ……..……..(15)}$$

$$G(k,x,y) = \max[G(0,x,y); G(k,x_1,y) + G[k, p(x-x_1), y], x_1 \in P, x_1 \le K_2(x); F(k-1,x,y)]$$
$$k \ge 1; \quad x \in L^0, \quad y \in H^0 \text{ ……..…….(16)}$$

Il est clair que la version modifiée est plus efficace que la version de base.

La solution optimale d'un problème n-DGDD, est obtenue suite à $o\left[n.|P|.|Q|.(|P|+|Q|)\right]$ opérations. Il est évident aussi que si $|P|$ et $|Q|$ sont très grand alors le calcul devient impossible.

3.5.3. Heuristique pour le problème DGDD :

Le problème DGDD peut être considéré comme un problème K-DGDD, si l'on suppose que le nombre d'ED K est inconnu, on obtient alors l'équation :

$$F(-,x,y) = \max[F(0,x,y);\ \ F(-,x,y_1) + F[-,x,q(y-y_1)], y_1 \in Q, y_1 \le y/2;$$

$$F(-,x_1,y) + F[-,p(x-x_1),y)], x_1 \in P, x_1 \le x/2]$$

$$x \in L^0,\ \ y \in H^0 \text{ ……..…….(17)}$$

Cette équation devient impossible à calculer lorsque $|P|$ et $|Q|$ deviennent trop grand. Afin d'éviter cela, on a redéfinit les ensembles P et Q de telle sorte que leurs cardinaux soit plus petit.

Soit M une borne supérieure pour $|P|$, qui peut être considéré comme étant le nombre maximum d'opération nécessaire pour la résolution de l'équation (17), et d'établir ainsi le nombre d'itération maximal pour lesquels le calcul de l'équation (17) soit faisable, on a proposé alors la procédure suivante pour redéfinir P :

(1) soit N = [1,2,…,n]

(2) calculer P

$$P = \left[x / x = \sum_{i \in N} l_i a_i, 1 \le x \le L - \min(l_j / j \in N), a_i \ge 0 \quad pour \quad i \in N \right] \text{……...(18)}$$

(3) si $|P| \le M$ alors fin avec l'ensemble P demandé, sinon aller à (4)

(4) définir

$$l_i = \min(l_j / i \in N) \text{ ……..……..(19)}$$

Alors N = N - [j] et aller en (2).

Cette procédure permet de déplacer les pièces de plus petite longueur de N vers P.

On ne peut plus garantir de solution optimale ce qui permet d'obtenir donc l'heuristique de programmation dynamique récursive suivante :

$$F(-,x,y) = \max[F(0,x,y);\quad F(-,x,y_1) + F[-,x,q(y-y_1)], y_1 \in Q, y_1 \le y-1;$$

$$F(-,x_1,y) + F[-,p(x-x_1),y)], x_1 \in P, x_1 \le x-1]$$

$$x \in L^0,\quad y \in H^0 \text{……...…….(20)}$$

Ce qui permet de redéfinir aussi F(0,x,y) par

$$F(0,x,y) = \max(0, \lfloor x/l_i \rfloor \lfloor y/h_i \rfloor c_i \mid l_i \le x,\quad h_i \le y,\quad i = 1,...,n) \text{ …….…..(21)}$$

Chapitre 4

Nouvelles approches pour la résolution du problème DGDD-NC pour les deux versions pondérée et non pondéré

4.1. Introduction :

Apres avoir étudié quelques méthodes exactes de la littérature et les avoir déroulé sur des instances on remarque que le nombre d'itération nécessaire pour obtenir la solution optimale est assez grand ainsi que le nombre de constructions engendrées, sachant que la plus part d'entre elles ont une chute importante.

C'est justement pour pallier à ce problème et minimiser l'espace mémoire utilisé qu'on a pensé à deux nouvelles approches combinant le principe des poins de découpe de l'algorithme de Herz [20] et la procédure de calcul de la borne inférieure initiale de la méthode constructive [21].

Ces deux heuristiques traitent les deux cas : **pondéré** et **non pondéré**

4.2. Le principe général :

Les deux heuristiques consistent à effectuer des coupes horizontales ou verticales suivant les ensembles des points de découpe, ce qui engendre des sous rectangles qui seront traités chacun individuellement.

4.2.1. Les points de découpe

On définit l'ensemble des points de découpe verticale comme étant l'ensemble PH constitué de différentes longueurs.

$$PH = \left\{ l_i, \quad i = 1,...r \right\}$$

Et l'ensemble des points de découpe horizontale est défini comme étant l'ensemble QV constitué de différentes hauteurs.

$$QV = \left\{ h_i, \quad i = 1,...r' \right\}$$

4.2.2. Génération des bandes :

Lemme 4.1 : [9]

*Soit R= (L, H) le r*ectangle initial, $S = \{(l_1, h_1),...,(l_n, h_n)\}$ l'ensemble des différentes pièces et r le nombre des différentes hauteurs sachant que les hauteurs sont ordonnées par ordre croissant

$$h_1 \le h_2 \le ... \le h_r$$

Et soit S_i l'ensembles des pièces dont la hauteur est inférieure ou égale à la hauteur h_i

$$S_i = \{(l_j, h_j) / h_j \le h_i\} \quad i = 1,...,r$$

Toutes les bandes horizontales de différentes hauteurs et de longueurs $0 \le \alpha \le L$ sont obtenues en résolvant par la programmation dynamique un seul problème de sac à dos défini par :

$$k_{\alpha}^{i}\begin{cases} F_i(\alpha) = \max \sum_{(l_j,h_j)\in S_i} c_j x_j \\ \sum_{(l_j,h_j)\in S_i} l_j x_j \le \alpha \\ x_j \in N \quad j = 1,...,n \end{cases}$$

c_j : Profit de la pièce j.

x_j : Nombre d'apparition de la pièces de type j.

$F_i(\alpha)$: Profit (valeur) de la bande i de longueur α et de hauteur h_i , $i=1,\ldots, r$

Preuve :

Une bande optimale de longueur α, avec $0 \le \alpha \le L$ et de hauteur h_i pour i=1,…, r est obtenue par la résolution du sac à dos unidimensionnel

$$k_{\alpha}^{i}\begin{cases} F_i(\alpha) = \max \sum_{(l_j,h_j)\in S_i} c_j x_j \\ \sum_{(l_j,h_j)\in S_i} l_j x_j \le \alpha \\ x_j \in N \quad j = 1,...,n \end{cases}$$

Si on résout le problème k_{α}^{i} en remplaçant la variable α par L à l'optimum en utilisant la méthode de la programmation dynamique, alors les solution optimale de tous les problèmes de sac à dos k_{α}^{i} avec $0 \le \alpha \le L$ seront disponibles, en d'autres termes toutes les bandes optimales de longueur α et de hauteur h_i pour i=1,…, r

Lemme 4.2 : [9]

Soit r' le nombre des différentes longueurs ordonnées par ordre croissant

$$l_1 \leq l_2 \leq ... \leq l_{r'}$$

Et soit S_i' l'ensembles des pièces dont la longueurs est inférieure ou égale à la longueur l_i

$$S_i' = \{(l_j, h_j) / l_j \leq l_i\} \quad i = 1,..., r'$$

Toutes les bandes verticales de différentes longueurs et de hauteurs $0 \leq \beta \leq H$ sont obtenues en résolvant par la programmation dynamique un seul problème de sac à dos définit par :

$$k_\beta^i \begin{cases} F_i'(\beta) = \max \sum_{(l_J, h_J) \in S_i'} c_j x_j \\ \sum_{(l_j, h_j) \in S_i'} h_j x_j \leq \beta \\ x_j \in N \qquad j = 1,..., n \end{cases}$$

c_j : Profit de la pièce j.

x_j : Nombre de pièces de type j.

$F_i'(\beta)$: Profit (valeur) de la bande i de hauteur β et de longueur l_i , i=1,…, r'

***Proposition 4.1* :** [9]

Toutes les bandes optimales horizontales de différentes hauteurs et de longueur L sont générées par la résolution d'un seul problème de sac à dos.

$$K_{LH}^{r}\begin{cases}\max F_r(L) = \max \sum_{i=1}^{n} c_i x_i \\ \sum_{i=1}^{n} l_i x_i \le L \\ x_i \in N, \quad i = 1,...,n\end{cases}$$

r : nombre des différentes hauteurs.

n : nombre des différentes pièces.

Proposition 4.2 : [9]

Toutes les bandes optimales verticales de différentes longueurs et de hauteur L sont générées par la résolution d'un seul problème de sac à dos

$$K_{LH}^{r'}\begin{cases}\max F_{r'}(H) = \max \sum_{i=1}^{n} c_i x_i \\ \sum_{i=1}^{n} h_i x_i \le H \\ x_i \in N, \quad i = 1,...,n\end{cases}$$

r' : Nombre des différentes longueurs.

n **:** nombre des différentes pièces.

4.2.3. La combinaison des bandes :

Proposition 4.3 : [9]

Un modèle de découpe horizontale pour le rectangle initial $R= (L, H)$ est obtenu par la résolution du problème de sac à dos suivant :

$$K_{H}^{hor}\begin{cases}B_{hor} = \max \sum_{i=1}^{r} F_i(L) y_i \\ s.c \quad \sum_{i=1}^{r} h_i y_i \le H \\ y_i \in N\end{cases}$$

Où r : la cardinalité de l'ensemble des bandes de longueur L.

y_i : Le nombre d'apparition de la i^{eme} bande horizontale de hauteur h_{k_i} et de profit $F_i(L)$.

Proposition 4.4 : [9]

Un modèle de découpe verticale pour le rectangle initial *R= (L, H)* est obtenu par la résolution du problème de sac à dos suivant :

$$K_L^{ver}\begin{cases} B_{ver} = \max \sum_{i=1}^{r'} F_i(H) y_i \\ s.c \quad \sum_{i=1}^{r} L_i y_i \le l \\ y_i \in N \end{cases}$$

Où r : la cardinalité de l'ensemble des bandes verticales de hauteur H.

y_i : Le nombre d'apparition de la i^{eme} bande verticale de longueur l_{k_i} et de profit $F_i(L)$.

Remarque :

La résolution des problèmes de sac à dos unidimensionnel par la programmation dynamique est particulièrement rapide. Nous utilisons la formule suivante :

$$F_k(\alpha) = \max_{x_k = 0,1,\ldots,\left\lfloor \frac{\alpha}{t_K} \right\rfloor} (c_k x_k + F_{k-1}(\alpha - t_k x_k))$$

$F_0(\alpha) = 0$ Pour α =0,1,…, *T*, $t_0 = t_1$

4.3. Développement de la première heuristique pour la résolution du problème de DGDD-NC

4.3.1. Notation :

Dans ce qui suit nous allons noté cette première heuristique H1-DGDD-NC.

4.3.2. Principe de l'heuristique H1-DGDD-NC

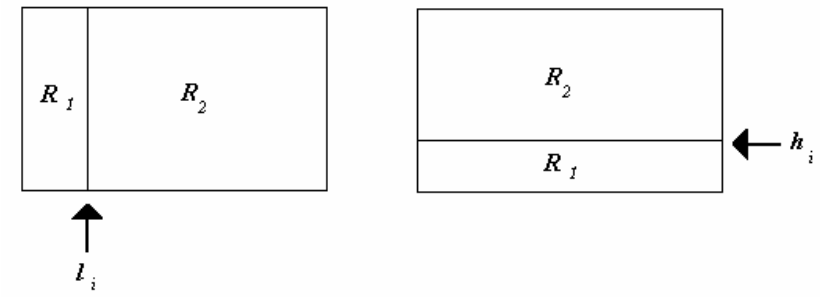
Les différents points de découpe verticale sont pris de l'ensemble *PH* représentant les différentes longueurs tel que

$$PH = \{l_i, i = 1, ..., r'\}$$

où r' est le nombres des différentes longueurs, alors que les points de découpe horizontales sont pris de l'ensemble *QV* représentant les différentes hauteurs tel que

$$QV = \{h_i, i = 1, ..., r\}$$

où *r* est le nombre des différentes hauteurs,



(a) (b)

Figure 15 : (a) la première coupe réalisée est verticale.
(b) la première coupe réalisée est horizontale

- La valeur du rectangle R_1 (proposition 4.2) de la figure 15 (a) (resp. de la figure 15 (b)) est la valeur de la bande optimale verticale de longueur l_i et de hauteur H (resp. est la valeur de la bande optimale horizontale de longueur L et de hauteur h_i (proposition 4.1).

- La valeur du rectangle R_2 de la figure 15(a) est obtenue en combinant verticalement les sous bandes horizontales de longueur $L - l_i$ et de différentes hauteurs (proposition 4.4).

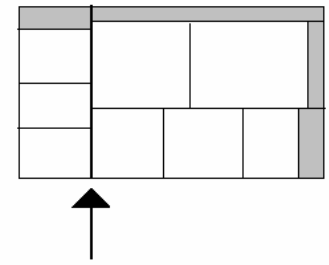


Figure 16 : découpe du rectangle R_2 engendré par une première coupe verticale.

- La valeur du rectangle R_2 de la figure 15(b) est obtenue en combinant horizontalement les sous bandes verticales de différentes longueurs l_i et de hauteurs $H - h_i$ (proposition 4.4)

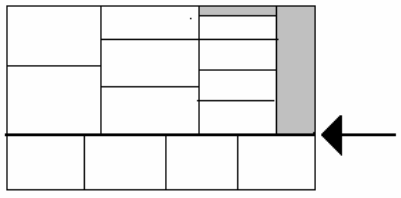


Figure 17: découpe du rectangle R_2 engendré par une première coupe horizontale.

Ainsi nous pouvons calculer la valeur du modèle en sommant les valeurs des deux sous rectangles R_1 et R_2.

4.3.3. Enoncé de l'heuristique :

Entré **:** *(L, H), n,* l_t **,** h_t (pour *t*= 1,…,*n*).

Sortie **:** $B_{\inf}(R)$**.**

❖ **Initialisation :**

1)- Calcul de la borne supérieure initiale :

$$B_{\sup}(R) = L \times H$$

2)- Calcul de la borne inférieure initiale :

$$B_{\inf}(\mathrm{R}) = \max\{B_{hor}, B_{ver}\}$$

❖ **Etape principale :**

Si ($B_{\sup}$(R)- $B_{\inf}$(R)=0) alors sortir avec $B_{\inf}$(R) ;

Sinon

➢ $P = \{ l_i, \ i = 1,...r \}$, l'ensemble des différentes longueurs.

Pour chaque $l_i \in P$ on effectue une coupe verticale à l'abscisse l_i ,

il en résulte deux sous rectangles $R_1 = (l_i, H)$, $R_2 = (L - l_i, H)$:

- La valeur V_1 du premier rectangle est la valeur de la bande Optimale verticale de longueur l_i et de hauteur H.
- La valeur V_2 du deuxième rectangle est obtenue en combinant verticalement les sous bandes horizontales issues de la résolution du problème de sac à dos K_L^i, *i=1,…,r*

$$k_{L-l_i H}^{H} \begin{cases} V_2 = \max \sum_{k=1}^{r} F_k (L - l_i) y_k \\ \sum_{k=1}^{r} h_k y_k \le H \\ y_k \in N \quad k = 1,..., r \end{cases}$$

- $V = V_1 + V_2$, **si** $V > B_{\inf}$ on effectue alors la mise à jour suivante

$$B_{\inf} = V$$

➢ $Q = \left\{ h_i,\ i = 1,...r' \right\}$.

Pour chaque $h_i \in Q$ on effectue une coupe horizontale à l'ordonnée h_i, il en résulte deux sous rectangles $R_1 = (L, h_i)$, $R_2 = (L, H - h_i)$.

- La valeur V_1 du premier rectangle est la valeur de la bande optimale horizontale de hauteur h_i et de longueur L.
- La valeur V_2 du deuxième rectangle est obtenue en combinant Horizontalement les sous bandes verticales issues de la résolution du problème de sac à dos K_H^i

$$k_{LH-h_i}^{H} \begin{cases} V_2 = \max \sum_{k=1}^{r} F_k (H - h_i) y_k \\ \sum_{i=1}^{r} l_k y_k \le L \\ y_k \in N \quad k = 1,..., r' \end{cases}$$

- $V = V_1 + V_2$, **si** $V > B_{\inf}$ on effectue alors la mise à jour suivante

$$B_{\inf} = V$$

Fin :

Sortir avec la solution $B_{\inf}$

4.3.4. Déroulement de l'heuristique Sur un exemple :

$R = (L, H) = (7,5)$;

$S = \{(3,1),(2,2)\}$;

	l_i	h_i	c_i
p_1	3	1	3
p_2	2	2	4

1)- Calcul de la borne supérieure initiale $B_{\sup}(R)$:

$B_{\sup}(R)$ = L.H = 7.5 =35 (cas non pondéré)

$B_{\sup}(R) = 35$

2)- Calcul de la borne inférieure initiale $B_{\inf}(R)$:

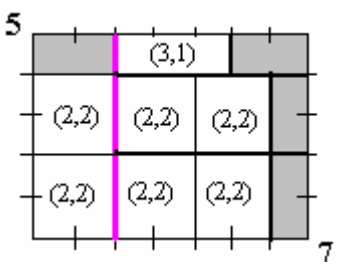
$B_{\inf}(\ R) = \max\{B_{hor}, B_{ver}\} = 31$

Etape principale :

$B_{\sup} - B_{\inf} = 4 \neq 0$;

Sinon

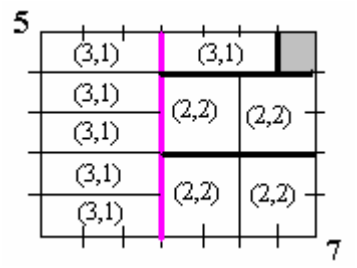
- $P = \{2,3\}$.
  - Pour $l_1 = 2$



De valeur $V = V_1 + V_2 = 8 + 19 = 27 < B_{\inf}$,

$\boxed{B_{\inf} = 31}$

- Pour $l_2 = 3$



De valeur $V = V_1 + V_2$ =15 + 19 = 34 > $B_{\inf}$,

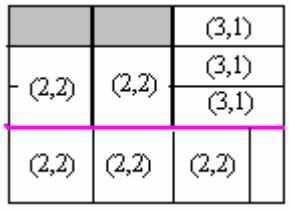
$$\boxed{B_{\inf} = 34}$$

- $Q = \{1,2\}$.
  - Pour $h_1 = 1$

De valeur $V = V_1 + V_2$ =6 + 28 = 34 = $B_{\inf}$,

$$\boxed{B_{\inf} = 34}$$

  - Pour $h_1 = 2$



De valeur $V = V_1 + V_2$

= 8 + 17 = 25 < $B_{\inf}$,

$$\boxed{B_{\inf} = 34}$$

Fin :

Sortir avec la solution $B_{\inf} = 34$

4.4. Développement de la deuxième heuristique pour la résolution du problème de DGDD-NC

4.4.1. Notation :

Dans ce qui suit nous allons noté cette deuxième heuristique H2-DGDD-NC.

4.4.2. Principe de l'heuristique H2-DGDD-NC

Les différents points de découpe verticale sont pris de l'ensemble *PH* représentant les différentes longueurs tel que

$$PH = \{l_i, i = 1,..., r'\}$$

où r' est le nombres des différentes longueurs, alors que les points de découpe horizontales sont pris de l'ensemble *QV* représentant les différentes hauteurs tel que

$$QV = \{h_i, i = 1,..., r\}$$

où *r* est le nombre des différentes hauteurs,

Soit l'ensemble « Liste » représentant les différents rectangles guillotine à découper tel que

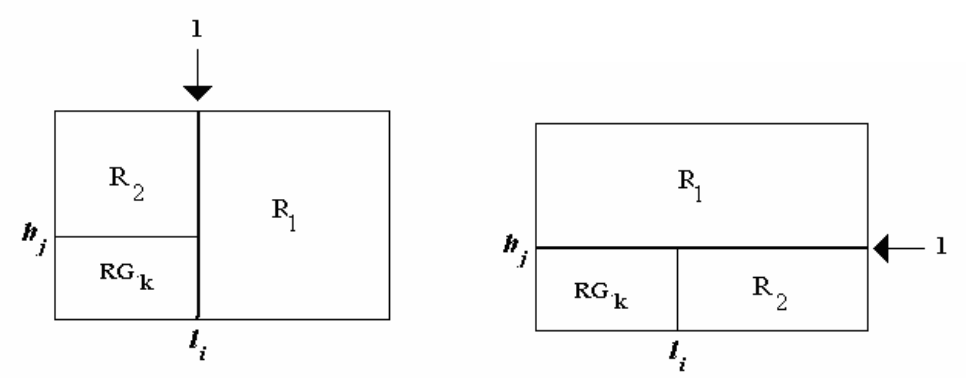
$$Liste = \left\{ RG_K = (l_i, h_j),\ \ i = 1,..., r',\ \ j = 1,...; r,\ \ k = 1,..., r \times r' \right\}$$

RG_K :le rectangle guillotine de dimension (l_i, h_j)

Pour chaque rectangle guillotine $RG_K \in Liste$ on obtient les deux rectangles de la figure 18 .

Les valeurs des rectangles RG_k, R_1 , R_2 sont obtenues en utilisant la procédure de calcul de la borne inférieure initiale c'est à dire calculer les différentes sous bandes horizontales (resp. verticale) en utilisant le lemme 4.1 et le lemme 4.2 ; ensuite les combiner verticalement (resp. horizontalement) en se basant sur les propositions 4.3 et 4.4 .

La valeur du modèle résulte de la somme des valeurs des trois sous rectangles.



(a) (b)

Figure 18 : (a) la première coupe a été effectuée verticalement à l'abscisse l_i,

(b) la première coupe a été effectuée horizontalement à l'ordonnée h_i,

4.4.3. Enoncé de l'heuristique :

Entré **:** *(L, H), n,* l_t **,** h_t (pour t= 1,…,n).

Sortie **:** $B_{\inf}(R)$**.**

❖ **Initialisation :**

1)- Calcul de la borne supérieure initiale :

$$B_{\sup}(R) = L \times H$$

2)- Calcul de la borne inférieure initiale :

$$B_{\inf}(\mathrm{R}) = \max\{B_{hor}, B_{ver}\}$$

❖ **Etape principale :**

Si ($B_{\sup}$ (R)- $B_{\inf}$ (R)=0) alors sortir avec $B_{\inf}$ (R) ;

Sinon

- $Liste = \left\{ RG_K = (l_i, h_j),\ \ i = 1,...,r',\ \ j = 1,...;r,\ \ k = 1,...,r \times r' \right\}$

  RG_K :le rectangle guillotine de dimension (l_i, h_j)

- Pour chaque $R_K \in Liste$.
  - Les valeurs des rectangles RG_k, R_1 , R_2 sont obtenues en utilisant la procédure de calcul de la borne inférieure initiale.
  - La valeur du modèle résulte de la somme des valeurs des trois rectangles

Fin :

Sortir avec la solution $B_{\inf}$

4.4.4. Déroulement de l'heuristique Sur un exemple :

$R = (L, H) = (7,5)$;

$S = \{(3,1),(2,2)\}$;

	l_i	h_i	c_i
p_1	3	1	3
p_2	2	2	4

1)- Calcul de la borne supérieure initiale $B_{\sup}(R)$:

$B_{\sup}(R)$ = L.H = 7.5 =35 (cas non pondéré)

$B_{\sup}(R) = 35$

2)- Calcul de la borne inférieure initiale $B_{\inf}(R)$:

$B_{\inf}(\ R) = \max\{B_{hor}, B_{ver}\} = 31$

Etape principale :

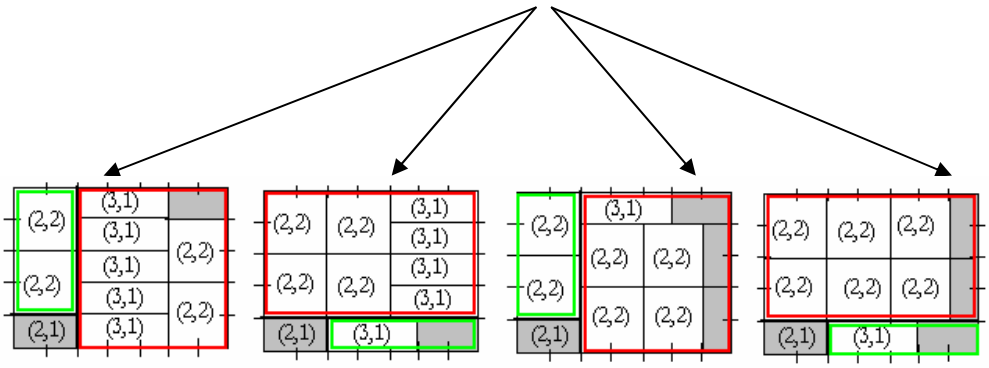
$B_{sup} - B_{inf} = 4 \neq 0$;

Sinon

- $liste = \{RG_1 = (2,1), RG_2 = (2,2), RG_3 = (3,1), RG_4 = (3,2)\}$.

▪ 1^{ere} itération :

$RG_1 = (2,1)$.



V= 0+**23**+**8** =31 **V**=0+**28**+**3**=31 **V**=0+**19**+**8**=27 **V**=0+**24**+**3**=27

$\boxed{B_{inf} = 31}$

▪ 2^{e} itération :

Pour $RG_2 = (2,2)$ la valeur de B_{inf} reste inchangée

$\boxed{B_{inf} = 31}$

▪ 3^{e} itération

pour $RG_3 = (3,1)$ la valeur de B_{inf} est mise à jour car une meilleure valeur a été trouvée

$\boxed{B_{inf} = 34}$

- 4^{e} itération :

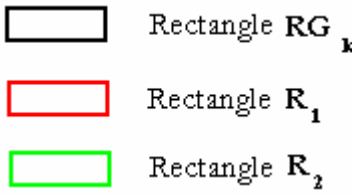
Pour $RG_4 = (3,2)$ la valeur de $B_{\inf}$ reste inchangée.

$$\boxed{B_{\inf} = 34}$$

Fin : sortir avec la solution

$$\boxed{B_{\inf} = 34}$$

Avec



4.5. Conclusion :

Pour les deux approches présentées ,on peut facilement calculer le nombre d'itérations et le nombre de constructions engendrées.

Le nombre d'itération pour l'heuristique H1-DGDD-NC se limite à $r + r' + 2$ et pour l'heuristique H2-DGDD-NC à $r \times r'$. En ce qui concerne le nombre de constructions engendrées, il est égale à $r + r' + 2$ pour l'heuristique H1-DGDD-NC et il est égale à $4.(r \times r') + 2$ pour l'heuristique H2-DGDD-NC.

On remarque que le nombre d'itération pour l'heuristique H2-DGDD-NC est plus grand que celui de l'heuristique H1-DGDD-NC , ce qui peut prévoir un temps d'exécution plus long.

Chapitre 5

Implémentation et résultats numériques

5.1. Introduction :

Les heuristiques destinées à la résolution des problèmes d'optimisation combinatoire et particulièrement les problèmes de découpe ont un même but : trouver une solution proche de l'optimum en un temps de calcul raisonnable tout en utilisant un minimum d'espace mémoire.

Pour prétendre que les deux heuristiques exposées précédemment répondent à ces critères, il est indispensable de les tester sur plusieurs instances de la littérature et sur celles générées aléatoirement. On comparera les résultats des deux heuristiques à ceux obtenus à partir des méthodes exactes, puis on les comparera entre elles.

Les deux heuristiques H1-DGDD-NC et H2-DGDD-NC ont été programmées en Delphi, version 5, et testées sur un micro-ordinateur Intel Pentium 4 (CPU 2.00 GHz et 256 Mo de RAM). Les instances testées sont disponibles à partir du site : ftp://www.univparis1.fr/pub/CERMSEM/hi"/2Dcutting.

5.2. Evaluation des heuristiques :

L'évaluation d'une heuristique est très importante car les méthodes approchées n'offrent aucune garantie d'optimalité : elles peuvent être très proches de l'optimum pour certaines instances comme elles peuvent en être très loin pour d'autres.

Définition 5.1 :

Soit I une instance, H une heuristique, $H(I)$ le coût de la solution heuristique et $OPT(I)$ le coût de la solution optimale. On appelle performance relative de l'heuristique $H(I)$ le quotient :

$$R_H(I) = \frac{H(I)}{OPT(I)}$$

Pour un problème de maximisation $R_H(I) \leq 1$.

5.3. Comparaison des deux heuristique H1-DGDD-NC et H2-DGDD-NC avec des méthodes exactes :

5.3.1. En utilisant des instances connues de la littérature :

Pour déterminer l'efficacité des deux approches H1-DGDD-NC et H2-DGDD-NC , nous avons choisie de les comparer à une méthode exacte qui s'appuie sur des techniques de programmation dynamique et une recherche arborescente avec critère de sélection du meilleur d'abord . En utilisant les instances de la littérature suivante :

- L'instance de Christofides & Whitlock [6] (40,70) pour le cas **pondéré**.
- L'instance de Herz [20] (127,98) pour le cas **non pondéré**.
- L'instance de Morabito & Arenales [26] (253,294) pour le cas **non pondéré**.
- L'instance de beasley [3] (3000,3000) pour le cas **non pondéré**.

Les résultats des testes sont disposés dans la table 1.

INSTANCE	Christofides & Whitlock	Herz	Morabito & Arenales	Beasley
(L, H)	(40,70)	(127,98)	(253,294)	(3000,3000)
N	10	5	10	32
Val OPT	3076	12348	73176	8997780
Val (H1-DGDD-NC*)*	3006	12192	72564	8977332
(%) solution optimale	**97,71**	**98 ,73**	**99,16**	**99,77**
Val (H2-DGDD-NC)	3076	12192	72618	8997780
(%) solution optimale	**100**	**98,73**	**99,23**	**100**

Table 1 : performances des heuristiques H1-DGDD-NC et H2-DGDD-NC comparées à une méthode constructive.

D'après la table 1, pour toutes ces instances les algorithmes H1-DGDD-NC et H2-DGDD-NC produisent de bonnes solutions en un temps de calcul très raisonnable, particulièrement l'heuristique H2-DGDD-NC dont la solution pour certaines instances est égale à la solution optimale

5.3.2. En utilisant des instances de taille moyenne :

➢ **Cas non pondéré :**

Nous utiliserons dans ce qui suit quelques instances de Fayard et al.[8] dont la solution optimale est connue. Les instances sont générées comme suit :

- Les dimensions du rectangle initial (L,H) sont prises de l'intervalle [500,4000].
- Les dimensions des pièces à découper (l_i, h_i) appartiennent à l'intervalle [0,01L ; 0,7H].
- Le nombre de pièces à découper *n* est compris entre 25 et 60.

Les résultats des tests effectués sur des instances dans le cas non pondéré sont exposés dans la table 2.

EXEMPLE	Sol opt	valeur H1-DGDD	Valeur H1-DGDD par rapport à la sol opt (%)	valeur H2-DGDD	Valeur H2-DGDD par rapport à la sol opt (%)
UU 1	242919	239969	**98,78**	242373	**99,77**
UU 2	595288	595288	**100**	595288	**100**
UU3	1072764	1072764	**100**	1072764	**100**
UU 4	1179050	1178295	**99,93**	1178295	**99,93**
UU 11	13157811	13127726	**99,77**	13132331	**99,80**

Table 2: comparaison des résultats obtenus à partir des deux heuristiques avec ceux obtenus par Fayard et al.[8] pour le cas non pondéré

➢ **Cas pondéré :**

Nous utiliserons aussi dans le cas pondéré quelques instances de Fayard et al.[8]. Les dimensions du support initial et des pièces à découper ainsi que le nombre n sont générés de la même façon que pour le cas non pondéré ; par contre le coût c_i d'une pièce i est compris entre 100 et 1000.

Les résultats obtenus pour le cas pondéré sont montrés dans la table 3

EXEMPLE	Sol opt	valeur H1-DGDD	Valeur H1-DGDD par rapport à la sol opt (%)	valeur H2-DGDD	Valeur H2-DGDD par rapport à la sol opt (%)
UW 1	6036	6036	**100**	6036	**100**
UW 3	6302	6226	**98,79**	6226	**98,79**
UW 4	8326	7900	**94,88**	7900	**94,88**
UW 5	7780	7780	**100**	7780	**100**
UW 6	6615	6615	**100**	6615	**100**

Table 3: comparaison des résultats obtenus à partir des deux heuristiques avec ceux obtenus par Fayard et al.[8] pour le cas pondéré

5.3.3. Instance de grande taille :

Les instances de grande taille sur lesquelles on a testé nos deux heuristiques et dont la solution optimale est connue sont obtenues à partir de Alvarez-Valdés. R.[1].

- **Cas non pondéré :**

Les instances sont générées de la façon suivante :

- Les dimensions du rectangle initial (L,H) appartiennent à l'intervalle [1500,3000].
- Les dimensions des pièces à découper sont prises comme suit :

$$l_i \in [0{,}05L ; 0{,}4H].$$

$$h_i \in [0{,}05L ; 0{,}4H].$$

- Le nombre de pièce à découper *n* est compris entre 30 et 60.

Les résultas des tests sont reportés dans la table 4.

EXEMPLE	Sol opt	valeur H1-DGDD	Valeur H1-DGDD par rapport à la sol opt (%)	valeur H2-DGDD	Valeur H2-DGDD par rapport à la sol opt (%)
ATP 10	3589703	3586557	**99,91**	3586557	**99,91**
ATP 11	4188915	4183056	**99,86**	4183056	**99,86**
ATP 12	5156065	5140351	**99,69**	5148302	**99,84**
ATP 13	3498302	3493410	**99,86**	3493410	**99,86**
ATP 14	4463550	4454550	**99,79**	4459262	**99,9**
ATP 15	6047188	6042363	**99,92**	6042363	**99,92**
ATP 16	7566719	7547220	**99,74**	7549879	**99,77**
ATP 17	4535302	4531594	**99,91**	4534815	**99,98**
ATP 18	5825956	5815968	**99,82**	5815968	**99,82**
ATP 19	6826674	6801936	**99,63**	6811338	**99,77**

Table 4: comparaison des résultats obtenus à partir des deux heuristiques avec ceux obtenus par Alvarez-Valdés. R.[1] pour des instances de grande taille dans le cas non pondéré.

- **Cas pondéré :**

Les instances sont générées comme pour le cas non pondéré, sauf le coût c_i d'une pièce i est égale à

$$c_i = \rho\, l_i h_i \quad \text{où} \quad \rho \in [0.25, 0.75]$$

Les résultas des tests sont reportés dans la table 5.

EXEMPLE	Sol opt	valeur H1-DGDD	Valeur H1-DGDD par rapport à la sol opt (%)	valeur H2-DGDD	Valeur H1-DGDD par rapport à la sol opt (%)
ATP 22	4145317	4110313	**99,15**	4111542	**99,18**
ATP 25	3507615	3490017	**99,5**	3501998	**99,83**
ATP 26	2683689	2656729	**99**	2656729	**99**
ATP 27	2438174	2429052	**99,62**	2429052	**99,62**

Table 5: comparaison des résultats obtenus à partir des deux heuristiques avec ceux obtenus par Alvarez-Valdés. R.[1] pour des instances de grande taille dans le cas pondéré

5.4. Comparaison des heuristiques H1-DGDD-NC et H2-DGDD-NC

Pour pouvoir comparer les deux algorithmes H1-DGDD-NC et H2-DGDD-NC on se base sur le temps d'exécution et les solutions obtenues.

En ce qui concerne le temps d'exécution, le pourcentage de H1-DGDD-NC par rapport à H2-DGDD-NC est infiniment petit et frôle les 0%

Quand aux solutions obtenues, d'après la table 6, on remarque qu'elles sont assez bonnes pour les deux heuristiques, mais la méthode H2-DGDD produit pour certaines instances des solutions plus proches de l'optimum.

	H1-DGDD			H2-DGDD		
	Nombre de sol opt	Pourcentage moyen	Pourcentage minimum	Nombre de sol opt	Pourcentage moyen	Pourcentage minimum
Instances connues	0	98,8425	**97,71**	2	99,49	**98,73**
UU	2	99,696	**98,78**	2	99,9	**99,93**
UW	3	98,734	**94,88**	3	98,734	**94,88**
ATP	0	99,813	**99**	0	99,863	**99**

Table 6: comparaison des résultats obtenus avec les deux heuristiques H1-DGDD-NC et H2-DGDD-NC

5.5. Conclusion :

Apres avoir déroulé sur un certain nombre d'instances les heuristiques **H1-DGDD-NC** et **H2-DGDD-NC** qui s'inspirent de deux méthodes exactes, applicables aux deux versions pondérée et non pondérée du problème de découpe guillotine DGDD-NC. Et suite aux résultats obtenus, les deux heuristiques H1-DGDD-NC et H2-DGDD-NC peuvent être considérées comme de bons algorithmes. L'approche H1-DGDD-NC est meilleure si on a uniquement pour but de minimiser le temps d'exécution, alors que l'approche H2-DGDD-NC fournie pour certaines instances une meilleure solution en un temps de calcul plus long.

La solution trouvée est assez proche de la solution optimale, le calcul du quotient $R_H(I)$ nous le prouve. Ces deux approches nous permettent aussi de gagner sur le temps de calcul.

Conclusion générale
et perspectives de recherche

Parmi la grande famille des problèmes de découpe on a choisi de traiter l'une de ses variantes qui est le problème de découpe guillotine à deux dimensions non contraint (DGDD-NC), c'est-à-dire qu'il n'existe aucune limitation sur le nombre d'apparition de chaque type de pièces.
Il existe plusieurs méthodes de résolution pour ce genre de problèmes. Nous avons étudié quelques unes d'entre elles :

- La méthode récursive de Herz [20] pour la version non pondérée, basée sur le principe des points de découpe. Toutes les solutions optimales ont la propriété récursive suivante:

Soit le rectangle initial est dans S**,** soit la première coupe produit deux sous rectangles, dont chacun est disséqué de façon optimale. L'algorithme consiste à essayer toutes les premières dissections possibles et retenir celle qui donne la valeur maximale pour les deux sous rectangles.

Quand à la structure de l'algorithme, la propriété récursive principale est présentée sous forme d'une structure arborescente, imposant toutes les coupes possibles sur le rectangle initial. Ensuite une méthode de limitation est appliquée sur les découpes horizontales et verticales afin d'aboutir à une méthode de séparation et d'évaluation utilisant la stratégie de développement en profondeur. Les solutions de cet algorithme ont aussi une structure particulière, chaque pièce est placée en bas à gauche.

▪ La méthode constructive [28] basée sur les techniques de programmation dynamique et une recherche arborescente avec critères de sélection du meilleur d'abord pour les deux versions pondérée et non pondérée est une fusion entre une méthode de séparation et d'évaluation et une recherche locale. Le principe de cette méthode consiste à :

- Limiter la taille de l'arborescence.
- Accélérer le parcours de celle-ci en se limitant à une procédure de recherche locale pour le calcul de l'évaluation sur un sommet.
- Exploiter au maximum l'information résultante de chaque calcul en un sommet dans la génération des bandes horizontales et verticales en résolvant des problèmes de sac à dos par la programmation dynamique.

Suite à l'étude des méthodes citées, nous avons pensé à deux nouvelles méthodes approchées H1-DGDDG-NC et H2-DGDDG-NC basées sur les principes suivants :

• Le principe des points de découpe inspiré de la méthode récursive de Herz [20]. Dans nos approches on se limite uniquement aux points de découpe représentant les différentes longueurs et hauteurs.

• La procédure de calcul de la borne inférieure initiale de la méthode constructive [28]. Elle est utilisée pour déterminer les valeurs des modèles des sous rectangles engendrés par des coupes effectuées suivant les ensembles des points de découpe horizontale et verticale.

L'approche H1-DGDDG-NC, consiste à effectuer une seule coupe horizontale ou verticale, engendrant deux sous rectangles traités chacun séparément. Par contre l'approche H2-DGDDG-NC consiste à effectuer deux coupes successives l'une horizontale et l'autre verticale, engendrant trois sous rectangles traités aussi chacun séparément.

En procédant de la sorte, il parait évident que le nombre de sommets à parcourir est réduit considérablement. Les solutions obtenues sont très proches de l'optimum. Nos approches ont été comparées aux meilleures méthodes exactes sur des instances de la littérature et d'autres générées aléatoirement. Nous avons constaté de bons résultats en un temps de calcul très raisonnable.

Enfin, ces deux approches peuvent être généralisées au problème de découpe à deux dimensions contraint DGDD-C , ainsi que pour la variante avec contrainte de niveau k-DGDD.

Références Bibliographiques

[1] Alvarez-Valdés. R, Parajon. A, Tamarit. J.M (2002) :
A tabu search algorithm for large-scale guillotine (un)constrainedtwo-d imensional cutting problems, Computers & Operations Research 29 (2002) 925}947.

[2] Arenales, M. and Morabito. (1995):
An AND/OR-graph approach to the solution of two dimensional non-guillotine cutting problems, European Journal of Operational Research 84, 599- 617.

[3] Beasley, J. E. (1985):
Algorithms for unconstrained two-dimensional guillotine cutting. Journal of the Operational Research Society 36, 297-306.

[4] Belov, G.; Scheithauer, G. (2002):
A cutting plane algorithm for the one-dimensional cutting stock problem with multiple stock lengths. European Journal of Operational Research 141, 274-294.

[5] Caprara, A.; Monaci, M. (2004):
On the two-dimensional Knapsack Problem. Operations Research Letters 32, 5-14.

[6] Christofides, N.; Whitlock, C. (1977):
An algorithm for two-dimensional cutting problems. Operations Research 25, 30-44.

[7] Dyckhoff, H. (1990):
A typology of cutting and packing problems. European Journal of Operational Research 44, 145-159.

[8] Fayard D, Hifi M, Zissimopoulos V. (1998) :
An eficient approach for large-scale two-dimensional guillotine cutting stock problems. Journal of the Operational Research Society 1998;49:1270}7.

[9] Feo. T.A and Resende. M.G.C. (1989):
A probabilistic heuristic for computationally difficult set covering problem , Operational Research letters, vol.8, p.p.67-71.

[10] Fowler. R.J, Paterson. R.M., and Tanimoto. S.T. (1981):
Optimal packing and covering in the plane are NP-complete, Information processing Letters, 12, p.p. 133-137.

[11] Gilmore, P. C.; Gomory, R. E. (1961):
A linear programming approach to the cutting-stock problem part I. Operations Research 9, 849-859.

[12] Gilmore, P. C.; Gomory, R. E. (1963):
A linear programming approach to the cutting-stock problem part II. Operations Research 11, 864-888.

[13] Gilmore, P. C.; Gomory, R. E. (1965):
Multistage cutting stock problems of two and more dimensions. Operations Research13, 94-120.

[14] Gilmore, E, and Gomory, R. (1966):
The theory and computational of knapsack functions, Operations Research 14 , 1045-1074.

[15] George, J. A.; George, J. M.; Lamar, B. W. (1995):
Packing different-sized circles into a rectangular container. European Journal of Operational Research 84, 693-712.

[16] Golden, B. L. (1976):
Approaches to the cutting stock problem. AIIE Transactions 8, 265-274.

[17] Gradišar, M.; Resinovič, G.; Kljajić, M. (1999):
A hybrid approach for optimization of one-dimensional cutting. European Journal of Operational Research 119, 719-728.

[18] Gradišar, M.; Resinovič, G.; Kljajić, M. (2002):
Evaluation of algorithms for one-dimensional cutting. Computers & Operations Research 29, 1207-1220.

[19] Healy, P.; Moll, R. (1996):
A Local Optimization-based Solution to the Rectangle Layout Problem. Journal of the Operational Research Society 47, 523-537.

[20] Herz, J. C. (1972):
Recursive computational Procedure for two-dimensional stock cutting. IBM Journal of Research Development 16, 462-469.

[21] Hifi, M.; Ouafi, R. (1998):
A best-first branch-and-bound algorithm for orthogonal rectangular packing problems. International Transactions in Operational Research 5, 345-356.

[22] Kantorovich. L.V.(1960)
"Mathematical methods of organizing and planning production," *Management Science*, vol. 6, pp. 363–422.

[23] Labbé, M.; Laporte, G.; Martello, S. (2003):
Upper bounds and algorithms for the maximum cardinality bin packing problem. European Journal of Operational Research 149, 490-498.

[24] Lodi, A. and Monaci, M. (2003)
"Integer linear programming models for 2-staged two-dimensional knapsack problems", Mathematical Programming, Series B, vol. 94, pp. 257-278.

[25] Martello, S.; Toth, P. (1990):
Knapsack problems – Algorithms and computer implementations. John Wiley & Sons, Chichester et al..

[26] Morabito. R. and Arenales. (1996):
Staged and constrained two-dimensional guillotine cutting problems: An and-or graph approach," European Journal of Operational Research, vol. 94, no. 3, pp. 548–560.

[27] Martello, S.; Pisinger, D.; Toth, P. (2000):
New trends in exact algorithms for the 0-1 knapsack problem. European Journal of Operational Research 123, 325-332.

[28] Ouafi. R. (2004) :
Approche algorithmique pour une classe de problème de découpe. Thèse de Doctorat d'état, USTHB,

[29] Scheithauer, G., 2002:
On a two-dimensional guillotine cutting problem. Presented at IFORS 2002, Edinburgh, UK.

[30] Pisinger, D. (1999):
An exact algorithm for large multiple knapsack problems. European Journal of Operational Research 114, 528-541.

[31] Pisinger, D. (2002):
Heuristics for the container loading problem. European Journal of Operational Research 141, 382-392.

[32] Scheithauer, G. (1991):
A three-dimensional bin packing algorithm. Journal of Information Processing and Cybernetics 27, 263-271.

[33] Scheithauer, G. (1999):
LP-based bounds for the container and multi-container loading problem. International Transactions in Operations Research 6, 199-213.

[34] Schilling, K. E. (1990):
The growth of m-constraint random knapsacks. European Journal of Operational Research 46, 109-112.

[35] Wäscher, G.; Gau, T. (1996):
Heuristics for the Integer one-dimensional cutting stock problem: a computational study. OR Spectrum 18, 131-144.

[36] Gerhard Wäscher · Heike Haußner · Holger Schumann (2004):
An Improved Typology of Cutting and Packing Problems. Faculty of Economics and Management Magdeburg, Working Paper No. 24.2004.