RAIRO. RECHERCHE OPÉRATIONNELLE

MHAND HIFI
VASSILIS ZISSIMOPOULOS

# Une amélioration de l'algorithme récursif de Herz pour le problème de découpe à deux dimensions

*RAIRO. Recherche opérationnelle*, tome 30, n° 2 (1996), p. 111-125

<http://www.numdam.org/item?id=RO_1996__30_2_111_0>

Recherche opérationnelle/Operations Research
(vol. 30, n° 2, 1996, pp. 111-125)

# UNE AMÉLIORATION DE L'ALGORITHME RÉCURSIF DE HERZ POUR LE PROBLÈME DE DÉCOUPE À DEUX DIMENSIONS (*)

par Mhand HIFI ([1]) et Vassilis ZISSIMOPOULOS ([2])

Communiqué par Gérard PLATEAU

Résumé. – *L'algorithme récursif de Herz est l'algorithme le plus efficace pour résoudre le problème de découpe à deux dimensions. Nous proposons des modifications pour rendre cet algorithme plus performant en utilisant le problème du sac à dos unidimensionnel et la programmation dynamique.*

Mots clés : Problème de découpe, problème du sac à dos, algorithmes récursifs, programmation dynamique.

Abstract. – *The recursive algorithm of Herz is the better actually known algorithm for solving two-dimensional cutting stock problems. We propose some modifications relied on one-dimensional knapsack and dynamic programming principles which improve significantly the computational power of the algorithm.*

Keywords: Cutting stock problems, knapsack, recursive algorithms, dynamic programming

## 1. INTRODUCTION

Le problème de découpe est un problème *NP-complet* avec de nombreuses applications en industrie, en systèmes multiprogrammés et multiprocesseurs, en placement de circuits, etc. Ce problème se présente lorsque l'on veut optimiser l'utilisation d'une entité disponible de taille limitée en y plaçant des sous-entités prédéterminées. Une politique optimale peut ne pas placer certaines des entités disponibles. Dans d'autres cas, le placement de toutes les sous-entités peut être indispensable (« strip packing » ou « bin packing »).

Dans ce papier, on s'intéresse à la première version du problème. On considère que l'entité disponible est donnée sous forme rectangulaire de

(*) Reçu en mars 1993.
([1]) CERMSEM, Université de Paris 1-Sorbonne, 12, place du Panthéon, 75005 Paris, France.
([2]) LRI, URA 410 CNRS, Université de Paris-Sud, Centre d'Orsay, 91405 Orsay, France.

Recherche opérationnelle/Operations Research, 0399-0559/96/02/$ 7.00

dimensions fixées, qu'on appellera par la suite rectangle initial. Les sous-entités ont aussi des formes rectangulaires qu'on appelera pièces. On se propose de découper le rectangle initial en pièces appartenant à l'ensemble des pièces disponibles tout en minimisant la chute, surface des pièces produites n'appartenant pas à cet ensemble. Très fréquemment dans les applications réelles la découpe doit s'effectuer en une seule fois d'un côté à l'autre. Cette manière de découper est dite découpe guillotine.

Le processus de découpe du rectangle initial en rectangles de dimensions inférieures en appliquant des découpes guillotines, est un processus récursif.

Pour ce problème Gilmore et Gomory [2] avaient proposé deux algorithmes itératifs dont le deuxième comme l'a remarqué Herz [3] était erroné. L'algorithme récursif de Herz est plus simple et aussi d'une complexité spatiale et temporelle inférieure à celui de Gilmore et Gomory. Dans ([5], [6]), une heuristique efficace a été présentée pour le même problème avec des bornes théoriques sur la qualité des solutions fournies. D'autre part, il a été prouvé expérimentalement que cette heuristique fournie une solution presque optimale. Dans ce papier, on propose des modifications de l'algorithme récursif de Herz en utilisant certaines caractéristiques de l'heuristique décrite dans [6]. L'algorithme récursif qui en résulte s'avère particulièrement efficace.

Nous rappelons dans la deuxième section l'algorithme récursif initial et ses principales caractéristiques. Dans la troisième section nous présentons les modifications apportées, qui permettent d'accélérer les calculs en exploitant des avantages offerts par la programmation dynamique. Ensuite, nous présentons une étude expérimentale qui justifie l'efficacité du nouvel algorithme.

Par la suite, nous supposons que les découpes sont du type guillotine.

## 2. L'ALGORITHME RÉCURSIF DE HERZ

Soit un rectangle $(L, H)$ de longueur $L$ et de hauteur $H$. Soit $S$ un ensemble fini de $n$ pièces rectangulaires de dimensions fixées. On note $(l_i, h_i)$ la $i$-ème pièce de l'ensemble $S$, avec $l_i$ sa longueur et $h_i$ sa hauteur. On se propose de découper le rectangle $(L, H)$ en effectuant des découpes du type guillotine afin de réaliser des pièces appartenant à $S$ tout en minimisant la chute. Soit $P$ l'ensemble fini de vecteurs représentant les différents plans de découpe, tel que $a = (a_1, ..., a_n) \in P$ avec $a_i$ le nombre de répétitions de la $i$-ème

pièce et $F$ une application définie par : $F : P \rightarrow N | F(a) = \sum_{i=1}^{n} \pi_i a_i$ où $\pi_i$ dénote la surface de la $i$-ème pièce. Le but du problème est de trouver le vecteur $a$ qui maximise $F$. Chaque pièce de $S$ peut être répétée plusieurs fois dans une solution optimale.

PROPRIÉTÉ : Une solution optimale est soit constituée d'une pièce appartenant à $S$, soit sa première découpe produit deux sous-rectangles qui sont aussi découpés d'une manière optimale.

Naturellement, la propriété récursive n'est pas vérifiée si la répétition d'une pièce est bornée par un entier $b_i$ tel que $b_i < \left\lfloor \frac{L}{l_i} \right\rfloor \left\lfloor \frac{H}{h_i} \right\rfloor$, $i = 1, ..., n$. Dans ce cas une solution optimale peut être associée à deux solutions partielles qui ne sont pas optimales.

Cette propriété est à la base de l'algorithme de Herz. Il consiste à essayer toutes les premières découpes et ensuite à retenir celle qui donne la valeur totale maximum pour les deux sous rectangles produits. Ceci est appliqué récursivement sur chaque sous-rectangle.

Pour garantir un nombre fini de découpes, celles-ci sont limitées sur l'ensemble de points $P$ (resp. : $Q$) qui sont des combinaisons linéaires des longueurs (resp. : hauteurs) des pièces de l'ensemble $S$, pour les découpes verticales (resp. : horizontales) ([3, 5]). Les solutions optimales produites ont une structure particulière telle que chaque pièce est placée toujours à gauche et en bas. On peut démontrer que chaque solution optimale est équivalente à une solution optimale ayant une telle structure [3]. D'autre part, les effets de la symétrie nous 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.

Pour accélérer le processus, l'algorithme utilise certaines bornes facilement calculées. Les plus importantes sont les suivantes :

*a*) Pour le rectangle initial ou un sous-rectangle quelconque produit au cours du processus, on calcule la valeur de la meilleure découpe homogène, composée d'une seule pièce de l'ensemble $S$.

*b*) Au cours du processus, si pour un (sous)-rectangle une découpe est pleine c'est-à-dire la chute est égale à zéro, alors on arrête la division de ce (sous-)rectangle car la solution partielle est optimale.

*c*) Si pour un (sous-)rectangle la valeur d'une découpe est connue et qu'elle est supérieure à la somme de la solution optimale du sous-rectangle gauche et de la surface du sous-rectangle droit (on suppose une découpe verticale),

alors il est inutile d'envisager la découpe du sous-rectangle droit. De même, lorsqu'on effectue une découpe horizontale on peut envisager d'ignorer la découpe du sous-rectangle du haut.

L'algorithme est présenté dans l'encadré 1. On note par $P_{\alpha\beta}$ l'ensemble des points représentant les combinaisons linéaires des longueurs des pièces rentrantes, dans le (sous-)rectangle $(\alpha, \beta)$, limités à la moitié de $\alpha$. De manière similaire, on définit $Q_{\alpha\beta}$ l'ensemble des points limités à $\beta/2$.

**Encadré 1 : Algorithme de Herz**

**Entrée :** $L$, $H$ : les dimensions du rectangle initial,
$n$ : le nombre de pièces à découper,
$(l_i, h_i)$, $1 \leq i \leq n$ : les dimensions des pièces.
**Sorties** : La solution optimale notée par **OPT**.
1. Construire les ensembles $P$ et $Q$
2. **OPT** $= R(L, H, 0)$
3. Fonction $R(\alpha, \beta, v_0)$ : *entier*
si $v_0 \geq S_{\alpha\beta}$ alors **sortir** avec $R = 0$
sinon
poser $\alpha_0 = \sup\{x | x \leq \alpha,\ x \in P\}$ et $\beta_0 = \sup\{y | y \leq \beta,\ y \in Q\}$
si la valeur de $(\alpha_0, \beta_0)$ est connue alors **sortir** avec cette valeur
sinon
calculer la *meilleure découpe homogène* $h$
si elle est *pleine* alors **sortir** avec $R = h$
sinon poser $V = h$
pour chaque $c \in P_{\alpha\beta}$
calculer $w = R(c, \beta, \max(V, v_0) - S_{(\alpha-c)\beta})$
et $V_1 = w + R(\alpha - c, \beta, \max(V, v_0) - w)$
si $V_1$ est pleine alors **sortir** avec $R = V_1$
sinon $V = \max(V, V_1)$
répéter la boucle pour tout $d \in Q_{\alpha\beta}$
**sortir** (avec la meilleure solution)

La surface du (sous-)rectangle $(\alpha, \beta)$ est notée par $S_{\alpha\beta}$. On note par $V$ la valeur de la meilleure solution en cours.

La valeur $v_0$ est la différence entre la valeur $V$ et la surface d'un sous-rectangle considérée comme une borne inférieure pour le deuxième sous-rectangle. Elle peut être aussi la différence entre $V$ et la meilleure solution d'un des deux sous-rectangles qui détermine la valeur à atteindre par le deuxième sous-rectangle.

Si au cours du processus on atteint l'optimum d'un sous-rectangle $(\alpha, \beta)$ alors on sauvegarde la valeur de la solution optimale ainsi que les dimensions du sous-rectangle $(\alpha_0, \beta_0)$ tels que $\alpha_0 = \sup\{x | x \leq \alpha, x \in P\}$ et $\beta_0 = \sup\{y | y \leq \beta, y \in Q\}$, puisque les deux sous-rectangles ont la même solution optimale.

On présente, sur la figure 1, l'application de l'algorithme de Herz sur une instance décrite par : $(L, H) = (8, 7)$ et $S = \{(3, 3), (3, 4), (5, 3), (4, 6)\}$ l'ensemble des pièces à découper. Les ensembles $P$ et $Q$ sont : $P = \{3, 4, 5, 6, 7, 8\}$ et $Q = \{3, 4, 6, 7\}$. Les premières découpes à effectuer sur le rectangle initial, en prenant compte la symétrie, sont sur les points $x = 3$, $x = 4$ et $y = 3$ (branchement sur la racine).

La solution homogène [deux fois la pièce (4, 6)] au niveau zéro [*fig.* 1 (a)] est de 48. La valeur homogène du sous-rectangle gauche, après avoir effectué la découpe sur le point $x = 3$, est de 18. Sur ce même rectangle, la seule découpe à effectuer est au point $y = 3$. Ainsi les deux sous-rectangles produits (3, 3) et (3, 4) sont exactement une pièce de l'ensemble $S$. Les solutions optimales de ces derniers, en remontant, permettent de mettre à jour la solution du sous-rectangle (3, 7) qui passe de 18 à 21. Sur le sous-arbre droit, la dernière feuille représente le sous-rectangle (5, 4). Les ensembles $P_{54}$ et $Q_{54}$ sont vides et donc la solution homogène [pièce (5, 3)] constitue une solution optimale. En faisant la mise à jour, la solution du rectangle initial passe de 48 à 51. Le processus est répété de la même manière sur l'arbre [*fig.* 1 (b)] qui donne une solution en cours égale à 51. Après

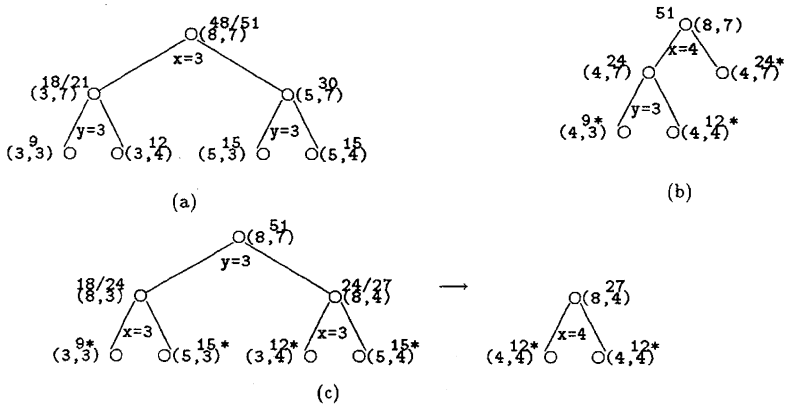


**Figure 1. – Application de l'algorithme de Herz sur l'instance : $(L, H) = (8, 7)$ et l'ensemble $S = \{(3, 3), (3, 4), (5, 3), (4, 6)\}$. $(*)$ indique une solution trouvée en mémoire. "→" découpe sur $x = 4$ après avoir effectué en premier la découpe sur $x = 3$.**

le parcours du dernier arbre [*fig.* 1 (c)], on obtient finalement la solution optimale donnée par la valeur 51.

## 3. MODIFICATIONS DE L'ALGORITHME

Notre algorithme conserve la structure récursive de l'algorithme de Herz et tente d'apporter des améliorations concernant chaque étape. Avant de décrire les modifications introduites, on donne les définitions suivantes :

DÉFINITIONS : Une bande *générale horizontale* est un ensemble de pièces rectangulaires, placées de telle façon qu'il existe une ligne horizontale ayant les propriétés suivantes :

– elle touche toutes les pièces sur leur longueur,

– un des demi-plans défini par celle-ci contient toutes les pièces (*fig.* 2).

Une bande *uniforme* est une bande pour laquelle existe exactement deux lignes horizontales ayant les propriétés précédentes. Dans ce cas, toutes les pièces participantes sont de même hauteur.

Une bande *homogène* est une bande uniforme où il n'y a qu'un seul type de pièce.

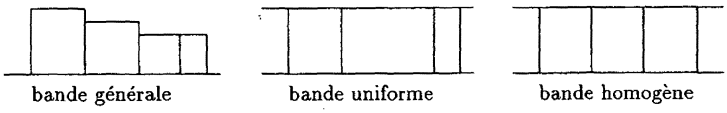


**Figure 2.**

Une bande *optimale* de longueur $\alpha$ et de hauteur $h$ est une bande générale occupant la surface maximum du rectangle $(\alpha, h)$ (*fig.* 3).

Si on note par $S(H^*)$, $S(U^*)$ et $S(G^*)$ la surface maximum occupée dans une bande homogène, uniforme et générale, de longueur $\alpha$ et de hauteur $h$, alors on a $S(H^*) \leq S(U^*) \leq S(G^*)$. La figure 3 montre un exemple de différentes bandes de longueur 11 et de hauteur 5. L'ensemble des pièces est $S = \{(6, 5), (4, 5), (1, 4)\}$. La surface maximum occupée est égale à 54, donnée par la bande générale constituée par une pièce (6, 5), une pièce (4, 5), et une pièce (1, 4).

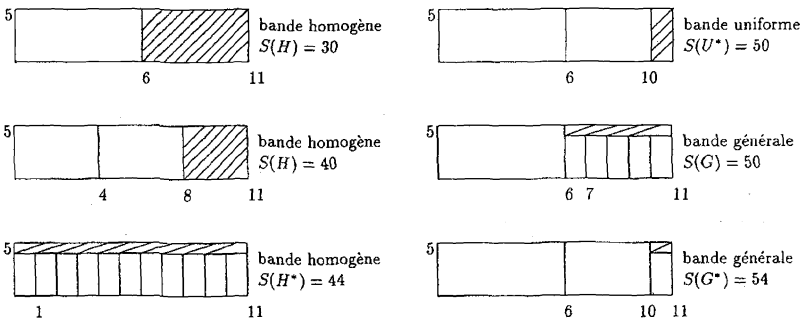


**Figure 3. – Six bandes de longueur 11 et de hauteur 5. $S(.)$ dénote la surface, et $S(H^*)$, $S(U^*)$, $S(G^*)$ sont respectivement les meilleures bandes homogènes, uniformes et générales.**

L'heuristique proposée dans [5], utilise une série de problèmes du sac à dos unidimensionnels ([1, 4]) qui génèrent un ensemble de bandes horizontales et un ensemble de bandes verticales. Ensuite, ces bandes sont combinées verticalement et horizontalement à l'aide d'un autre problème du sac à dos unidimensionnel, afin de produire un plan de découpe. Le processus se répète sur un nombre fini de sous-rectangles pour améliorer la solution. La solution ainsi obtenue est souvent optimale. La rapidité de la méthode est dûe à certains avantages offerts par la programmation dynamique [4]. En effet, toutes les bandes horizontales de différentes hauteurs peuvent être générées en résolvant un seul problème du sac à dos. De la même manière toutes les bandes verticales de différentes largeurs sont générées par un seul problème de sac à dos. D'autre part, toutes les sous-bandes de longueurs inférieures et de différentes hauteurs sont disponibles par le même problème du sac à dos. Cette propriété constitue l'idée de notre amélioration qui est fondée sur les résultats suivants :

LEMME : *Soient* $(L, H)$ *le rectangle initial et* $S = \{(l_1, h_1), ..., (l_n, h_n)\}$ *l'ensemble de pièces prises par ordre croissant selon les hauteurs* $h_1 \leq h_2 \leq ... \leq h_n$. *Toutes les bandes optimales de différentes hauteurs et de longueurs* $\alpha$, *avec* $0 \leq \alpha \leq L$ *sont obtenues en résolvant par la programmation dynamique un seul problème du sac à dos unidimensionnel défini par :*

$$(\mathbf{K}) \quad \begin{cases} \text{Max} \displaystyle\sum_{(l_u, h_u) \in S} \pi_u . x_u \\ \displaystyle\sum_{(l_u, h_u) \in S} l_u . x_u \leq L, \quad x_u \text{ entier} \end{cases}$$

*où $x_u$ dénote le nombre de répétitions de la longueur $l_u$ et $\pi_u$ dénote la surface de la pièce $(l_u, h_u)$.*

*Preuve :* Soit $r$ le nombre de hauteurs différentes de pièces avec $h_{k_1} < h_{k_2} < ... < h_{k_r}$. Pour chaque $i = 1, 2, ..., r$ on définit les ensembles

$$S_{k_i} = \{l_j | (l_j, h_j) \in S,\ h_j \leq h_{k_i}\}.$$

Une bande optimale de longueur $\alpha$, avec $0 \leq \alpha \leq L$, et de hauteur $h_{k_i}$, est obtenue par la résolution du problème du sac à dos défini par :

$$\left(K_\alpha^{k_i}\right) \quad \begin{cases} \text{Max} \displaystyle\sum_{l_u \in S_{k_i}} \pi_u x_u \\ \displaystyle\sum_{l_u \in S_{k_i}} l_u . x_u \leq \alpha, \quad x_u \text{ entier} \end{cases}$$

Si l'on résout le problème $\left(K_\alpha^{k_i}\right)$, pour $\alpha = L$, à l'optimum en utilisant la programmation dynamique, alors les solutions optimales de tous les problèmes du sac à dos $\left(K_\alpha^{k_i}\right)$, avec $\alpha \leq L$ seront disponibles. Par conséquent, toutes les bandes optimales de longueur $\alpha$ et de hauteur $h_{k_i}$ sont créées.

De même, en supposant que les pièces soient rangées par ordre croissant des hauteurs, on peut obtenir les solutions de tous les problèmes $\left(K_L^{k_i}\right)$, $i = 1, ..., r$ en résolvant seulement le problème $\left(K_L^{k_r}\right)$ qui est identique au problème (**K**), puisque $S_{k_r} = S$. ■

On peut remarquer que toutes les bandes de la figure 3 sont des solutions réalisables du problème du sac à dos suivant :

$$\left(K_{11}^5\right) \quad \begin{cases} \text{Max } 30\, x_1 + 20\, x_2 + 4\, x_3 \\ 6\, x_1 + 4\, x_2 + x_3 \leq 11, \quad x_1, x_2, x_3 \text{ entiers} \end{cases}$$

tandis que la bande optimale donnée par la valeur $S(G^*)$ est la solution optimale de ce problème. La bande optimale de longueur 10 et de hauteur 5 est aussi obtenue par le problème $K_{10}^5$, qui est une bande uniforme sans chute (surface occupée 50). Remarquons que la meilleure bande homogène est composée par 10 pièces de (1, 4) occupant une surface égale à 40.

THÉORÈME : *Soient $E = \{(\alpha_1, \beta_1), ..., (\alpha_m, \beta_m)\}$ un ensemble fini de rectangles indépendants et $S = \{(l_1, h_1), ..., (l_n, h_n)\}$ un ensemble de pièces. Si le nombre d'apparitions de chaque pièce n'est pas borné supérieurement alors une solution constituée par des bandes optimales*

*(meilleure qu'une solution homogène) pour le problème de découpe sur chaque rectangle est obtenue en résolvant seulement $m + 1$ problèmes du sac à dos unidimensionnels.*

*Preuve :* Soit $(L, H)$ le rectangle fictif tels que $L = \max_{1 \leq j \leq m} \alpha_j$ et $H = \max_{1 \leq j \leq m} \beta_j$.

Soient $r$ le nombre de différentes hauteurs des pièces $(h_{k_1} < h_{k_2} < ... < h_{k_r})$. Notons par $F_{k_i}(\alpha)$ la surface occupée dans chaque bande $k_i$ de longueur $\alpha$ et de hauteur $h_{k_i}$. L'application du lemme précédent sur $(L, H)$ fournit :

– toutes les bandes optimales de longueur $L$ et de hauteur $h_{k_i} \leq H$, $i = 1, ...r$ avec ses valeurs respectives $F_{k_i}(L)$ ;

– toutes les sous-bandes optimales de longueur $\alpha_j \leq L$ et de hauteur $h_{k_i} \leq \beta_j \leq H$, $i = 1, ..., s$, $s \leq r$ avec ses valeurs respectives $F_{k_i}(\alpha_j)$.

Soit le rectangle $(\alpha, \beta) \in E$. La résolution du problème $(\mathbf{K}_{\alpha,\beta})$ suivant produit un plan de découpe (découpe générale) meilleur qu'un plan de découpe composé de bandes uniformes qui est lui-même meilleur qu'un plan de découpe homogène, pour le rectangle $(\alpha, \beta)$ :

$$(\mathbf{K}_{\alpha,\beta}) \quad \begin{cases} \text{Max} \sum_{i=1}^{s} F_{k_i}(\alpha)\, z_{k_i} \\ \sum_{i=1}^{s} h_{k_i}\, z_{k_i} \leq \beta, \quad z_{k_i} \text{ entier}, \quad \alpha \leq L, \beta \leq H, \; s \leq r \leq m \end{cases}$$

où $s$ est le nombre de bandes optimales de hauteurs $h_{k_1}, ..., h_{k_s} \leq \beta$, $\alpha$ la longueur, $\beta$ la hauteur du rectangle à traiter et $F_{k_i}(\alpha)$ la surface occupée dans la bande $k_i$.

Il est évident que le problème $(\mathbf{K}_{\alpha,\beta})$ donne la meilleure combinaison linéaire de (sous-)bandes optimales réalisant une surface maximale. ■

Si l'on considère l'algorithme de Herz appliqué sur un rectangle initial $(L, H)$ et $E$ l'ensemble de (sous-)rectangles générés, alors le rectangle initial est le rectangle fictif défini dans le théorème précédent. Pour un sous-rectangle $(\alpha, \beta)$ quelconque une découpe générale (solution constituée par des (sous-)bandes optimales) est obtenue en résolvant uniquement le problème $(\mathbf{K}_{\alpha,\beta})$.

Les résultats précédents sont utilisés dans l'algorithme de Herz de la façon suivante :

On résout le problème du sac à dos défini dans le lemme précédent qui produit la bande horizontale la plus longue (longueur du rectangle initial) et la

plus haute (la plus grande hauteur des pièces appartenant à $S$). La résolution de ce problème par la programmation dynamique nous permet d'obtenir toutes les autres bandes (leur structure et leur surface) de même longueur et de hauteurs différentes. Le problème $(\mathbf{K}_{\alpha,\beta})$ nous permet d'obtenir un plan de découpe. Ainsi à chaque étape de l'algorithme nous disposons d'une solution très satisfaisante.

Une découpe homogène utilisée dans l'algorithme de Herz est la répétition verticalement d'une bande horizontale homogène. Parmi l'ensemble de découpes homogènes, on prend celle qui réalise la valeur maximale. Nous proposons de remplacer la meilleure découpe homogène par la découpe générale. Puisque les bandes générales sont déjà disponibles, alors la résolution du problème $(\mathbf{K}_{\alpha,\beta})$ produit une découpe générale tout en combinant ces meilleures bandes. La solution obtenue pour un sous-rectangle est meilleure que la solution homogène. Cela aura les conséquences suivantes :

1. On diminue sensiblement le développement de l'arborescence car l'efficacité de la borne donnée par (*b*) (section 2) est élargie produisant très souvent des découpes pleines et donc des solutions partielles optimales (solutions obtenues par des bandes uniformes).

2. Comme l'apparition de découpes pleines est très fréquente et qu'elles sont obtenues plus rapidement que dans l'algorithme de Herz, on augmente l'efficacité de la borne donnée par (*c*) (section 2), ce qui permet de ne pas traiter fréquemment le sous-rectangle droit (ou le sous-rectangle du haut).

3. L'utilisation de la solution obtenue sur chaque sommet de l'arborescence (solution obtenue par une bande générale) amplifie les conséquences du point 2 précédent. C'est-à-dire, si on dispose d'une bonne solution pour un sous-rectangle donné à un sommet quelconque et une solution optimale pour un sous-rectangle produit, alors on peut se passer du traitement du sous-rectangle droit.

*Remarque :* Les problèmes du sac à dos unidimensionnels introduits sont plutôt de petite taille, même pour des instances de problème de découpe de grande taille. Par conséquent, la résolution exacte de ces problèmes par la programmation dynamique est particulièrement rapide.

Le développement de l'arborescence en profondeur dans l'algorithme de Herz s'arrête lorsqu'une découpe partielle optimale est obtenue. Particulièrement, lorsqu'un sous-rectangle coïncide avec une pièce appartenant à $S$. En introduisant les problèmes du sac à dos unidimensionnels, le critère d'arrêt peut être amélioré en vertu des deux corollaires suivants :

COROLLAIRE 1 : *Tout sous-rectangle* $(\alpha, \beta)$ *tel que* $\beta/2 < h_k$, *avec* $h_k$ *la hauteur minimale des pièces qui rentrent dans ce sous-rectangle est résolu de manière optimale.*

En effet, la solution optimale est une bande générale optimale qui est déjà disponible par la résolution du problème (**K**) tout au début de l'algorithme. En conséquence, il est inutile de résoudre le problème $(\mathbf{K}_{\alpha,\beta})$ ou de continuer la découpe, ce qui n'est pas le cas de l'algorithme actuel.

COROLLAIRE 2: *Tout sous-rectangle* $(\alpha, \beta)$ *tel que* $\alpha/2 < l_r$, *avec* $l_r$ *la longueur minimale des pièces qui rentrent dans ce sous-rectangle est résolu de manière optimale.*

Dans ce cas la solution optimale est une combinaison de bandes horizontales « triviales » (chacune incluant une seule pièce). Cette solution est obtenue par la résolution de $(\mathbf{K}_{\alpha,\beta})$. Ainsi on évite la découpe du sous-rectangle $(\alpha, \beta)$.

Nous présentons dans l'encadré 2 les différentes étapes de l'algorithme amélioré.

La figure 4, représente l'arborescence développée par l'algorithme amélioré sur l'instance décrite dans la figure 1. La solution obtenue au niveau zéro est la solution optimale.

La découpe des sous-rectangles (3, 7), (5, 7) [*fig.* 4 (a)] et (4, 7) [*fig.* 4 (b)] est arrêtée, car les solutions obtenues sont optimales en vertu du corollaire 2. Le traitement sur les sous-rectangles (8, 3) et (8, 4) [*fig.* 4 (*c*)] est aussi arrêté grâce au corollaire 1.

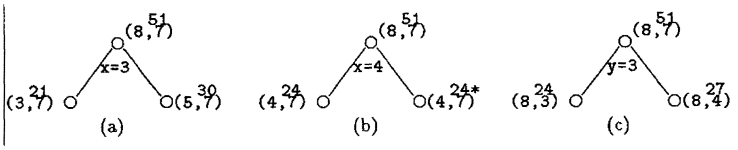


**Figure 4. – Application de l'algorithme amélioré sur l'instance de la figure 1.**

## 4. RÉSULTATS NUMÉRIQUES

Des expérimentations numériques effectuées par Herz montraient une supériorité de 20 % en faveur de l'algorithme récursif, par rapport à l'algorithme de Gilmore et Gomory. Sur le problème donné par Herz dans [3], nos modifications apportent une amélioration de 21 % sur le temps d'exécution par rapport à l'algorithme de Herz, tandis que le nombre de

sommets explorés de l'arborescence est diminué de 35 %. Plus précisément, l'algorithme de Herz nécessite 5630 ms pour donner la solution optimale de cette instance en générant 3727 sommets. En revanche, notre algorithme la fournit en 4 440 ms tout en créant une arborescence de 2 422 sommets.

**Encadré 2 : Algorithme amélioré**

**Entrée :** $L$, $H$ : les dimensions du rectangle initial,
$n$ : le nombre de pièces à découper,
$(l_i, h_i)$, $1 \leq i \leq n$ : les dimensions des pièces.
**Sortie** : La solution optimale notée par **OPT**.
1. Construire les ensembles $P$ et $Q$
2. **Résoudre le problème (K)**
3. **OPT** $= R(L, H, 0)$
4. Fonction $R(\alpha, \beta, v_0)$ : *entier*
si $v_0 \geq S_{\alpha\beta}$ alors **sortir** avec $R = 0$
sinon
poser $\alpha_0 = \sup\{x | x \leq \alpha, x \in P\}$ et $\beta_0 = \sup\{y | y \leq \beta, y \in Q\}$
si la valeur de $(\alpha_0, \beta_0)$ est connue alors **sortir** avec cette valeur
**sinon**
**si** $\min_{h_j / l_j \leq \alpha_0} \{h_j\} > \beta_0/2$
**alors sortir avec $R$ égale à la valeur de la bande optimale**
sinon
**résoudre le problème** $(K_{\alpha_0, \beta_0})$ ; **meilleure découpe générale** $g$
si elle est pleine alors **sortir** avec $R = g$
sinon
**si** $\min_{l_j / h_j \leq b_0} \{l_j\} > \alpha_0/2$
**alors sortir avec** $R = g$ { solution optimale composée de bandes triviales }
sinon poser $V = g$
pour chaque $c \in P_{\alpha\beta}$
calculer $w = R(c, \beta, \max(V, v_0) - S_{(\alpha - c)\beta})$
et $V_1 = w + R(\alpha - c, \beta, \max(V, v_0) - w)$
si $V_1$ est pleine alors **sortir** avec $R = V_1$
sinon $V = \max(V, V_1)$
répéter la boucle pour tout $d \in Q_{\alpha\beta}$
**sortir** avec la meilleure solution.

D'autre part, on a testé l'algorithme amélioré sur deux groupes d'instances, générées aléatoirement. Chaque groupe est composé de 130 instances. La longueur et la hauteur des pièces varient de manière uniforme entre $(0, L)$ et $(0, H)$, où $L$ et $H$ sont les dimensions du rectangle initial.

Le premier groupe est composé des petites instances avec $L$ et $H$ variant entre 30 et 80. Le nombre de pièces à découper varie entre 10 et 30.

Le deuxième groupe est composé des grandes instances avec $L$ et $H$ variant entre 80 et 150. Le nombre de pièces à découper varie entre 30 et 80.

TABLEAU 1
*Performance de l'algorithme amélioré sur différentes instances générées aléatoirement.*

| Taille des instances | Petites | Grandes | Total |
|---|---|---|---|
| Temps Moy. Herz (ms) | 3275,1 | 11536,7 | 7405,9 |
| Nb. Sommets Herz (Moy.) | 2192,6 | 13132,4 | 7662,3 |
| Temps Moy. Alg. Amélioré (ms) | 2702,3 | 8341,0 | 5521,6 |
| Nb. Sommets Alg. Amélioré (Moy.) | 1809,1 | 8562,3 | 5185,7 |
| Gain Temps (%) | 17,5 | 27,7 | 25,4 |
| Sommets non-traités (%) | 20,0 | 34,8 | 32,3 |

En ce qui concerne le premier groupe, l'algorithme modifié donne les solutions optimales avec une amélioration de 17,5 % (tableau 1) sur le temps moyen d'exécution par rapport à l'algorithme de Herz. Le nombre de sommets moyen explorés est diminué de 20 %.

Pour le deuxième groupe, constitué par les grandes instances, les améliorations respectives sont plus importantes. En effet, le temps moyen est réduit de 27,7 % et le nombre de sommets moyen de 34,8 %.

On remarque que le temps moyen d'exécution sur l'ensemble des instances est amélioré de 25,4 % et le nombre de sommets moyen explorés est réduit de 32,3 %.

Nous donnons sur la figure 5, la solution optimale d'une instance décrite de la façon suivante : la longueur du rectangle initial est 78 et sa hauteur 67. Le nombre de pièces à découper est égal à 6 et leurs dimensions exactes sont données comme suit :

$\{p_1 = (7, 10), p_2 = (32, 54), p_3 = (6, 13), p_4 = (14, 54), p_5 = (9, 5), p_6 = (24, 13)\}$. La solution optimale obtenue est composée de deux bandes uniformes optimales incluant 3 pièces de dimensions (24, 13), deux pièces de dimensions (32, 54), une pièce de dimensions (6, 13) et une pièce de dimensions (14, 54). Cette solution est donnée par l'algorithme de Herz en explorant 1237 sommets tandis que notre algorithme fournie cette solution dès le départ (niveau 0). L'algorithme de Herz nécessite 2190 ms pour l'obtention de cette solution, en revanche notre algorithme la fournit en 210 ms. On constate que le temps d'exécution est réduit de 90,4 % par

rapport à l'algorithme initial de Herz. L'ensemble des instances a été exécuté sous IBM/PC 286.

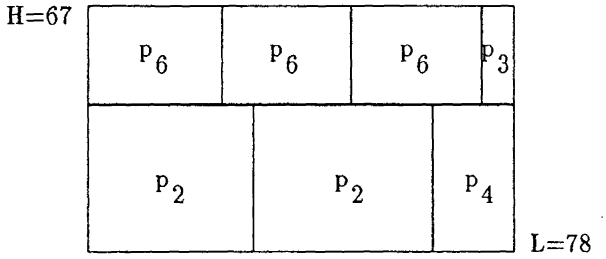


**Figure 5. – Solution optimale d'une instance du problème de découpe obtenue par l'algorithme amélioré au niveau zéro, constituée par deux bandes uniformes optimales.**

Comme il a déjà été indiqué, les deux algorithmes sauvegardent les sous-rectangles résolus à l'optimum. Nos expérimentations numériques ont été réalisés en utilisant un tableau de dimension $(L, H)$. L'algorithme de Herz sur l'exemple de la figure 1 profite neuf fois de la recherche en mémoire, tandis que le nouveau algorithme ne profite qu'une seule fois (*fig.* 4). En général, la sauvegarde des solutions des sous-rectangles apporte un gain considérable pour l'algorithme de Herz. Par conséquent, si la place mémoire occupée est un facteur déterminant, alors la suppression de ce tableau affecterait sensiblement le temps d'exécution de l'algorithme de Herz. Par contre, ceci affecterait peu le nouvel algorithme, le nombre de sommets explorés étant largement diminué.

## 5. CONCLUSION

On a considéré un problème de découpe NP-complet et les deux algorithmes exacts existants de résolution dûs à Gilmore and Gomory et Herz. Le deuxième algorithme s'appuie sur la propriété récursive des solutions optimales et il est plus rapide que l'algorithme itératif de Gilmore et Gomory. En utilisant, certaines caractéristiques de l'heuristique proposée dans ([5, 6]) pour résoudre ce même problème, on a amélioré l'algorithme de Herz en le rendant sensiblement plus rapide (en moyenne 25,4 % plus rapide). L'amélioration repose sur le problème du sac à dos unidimensionnel et la programmation dynamique. Cette amélioration sera particulièrement sensible pour des instances de grande taille, comme il a été constaté sur les expérimentations numériques.

Les résultats théoriques présentés, pourraient être significatifs pour l'élaboration d'algorithmes efficaces pour la résolution du problème général de découpe (problème du carnet de commandes).

## RÉFÉRENCES

1. D. Fayard et G. Plateau, An Algorithm for the Solution of the 0-1 Knapsack Problem, *Computing*, 1982, *28*, p. 269-287.
2. P. Gilmore et R. Gomory, The Theory and Computation of Knapsack Functions, *Opns. Res.*, 1966, *14*, p. 1045-1074.
3. J. Herz, A Recursive Computing Procedure for Two-Dimensional Stock Cutting, *IBM J. Res. Develop.*, 1972, *16*, p. 462-469.
4. P. Toth, Dynamic Programming Algorithms for the Zero-One Knapsack Problem, *Computing*, 1980, *25*, p. 29-45.
5. V. Zissimopoulos, Problèmes de Découpe : Algorithmes $\varepsilon$–Approchants, Thesis, L.R.I., Orsay, 1984.
6. V. Zissimopoulos, Heuristic Methods For Solving (Un)Constrained Two Dimensional Cutting Stock Problems, *Methods of Operations Research*, 1984, *49*, p. 345-357.