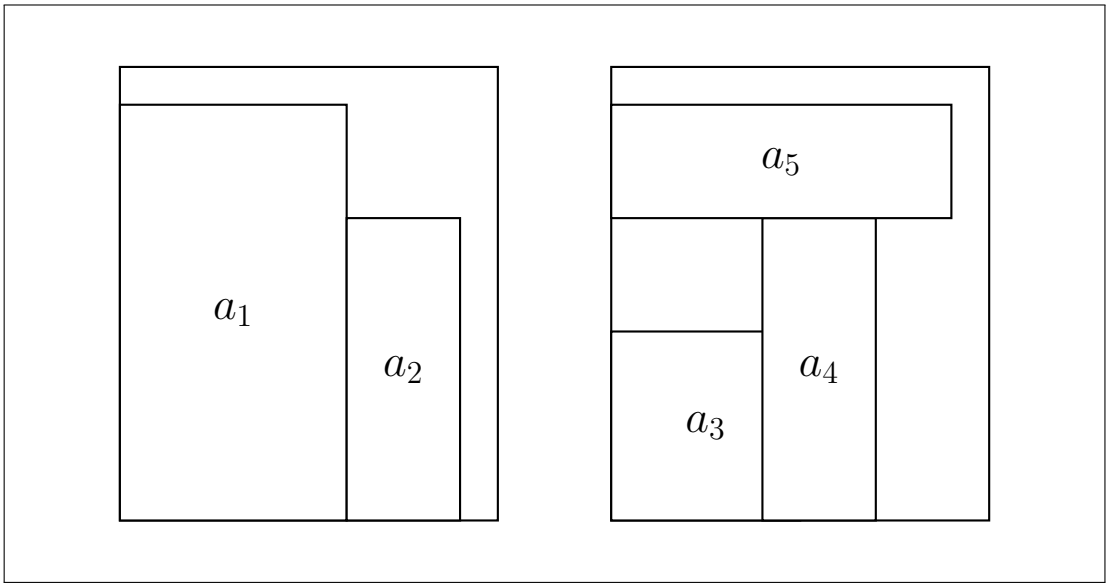
Le problème de bin-packing en deux-dimensions, le cas non-orienté : résolution approchée et bornes inférieures.

Joseph El Hayek

par Joseph El Hayek

Le problème de bin-packing en deux-dimensions, le cas non-orienté : résolution approchée et bornes inférieures

Thèse présentée
pour l'obtention du grade
de Docteur de l'UTC.



Soutenue le : 8 décembre 2006
Spécialité : Technologie de l'information et des systèmes

Le problème de bin-packing en deux-dimensions, le cas non-orienté : résolution approchée et bornes inférieures

Thèse soutenue le 8 décembre 2006 devant le jury composé de :

MM. J. Carlier (Président, Examinateur)
V. Cung (Rapporteur)
J. Hao (Rapporteur)
M. Hifi (Examinateur)
A. Moukrim (Directeur de thèse)
S. Negre (Directeur de thèse)

Remerciements

Je tiens à remercier mes directeurs de thèse, Aziz Moukrim et Stéphane Negre pour leur disponibilité et leur soutien tout au long de ces trois années de thèse. Ce fut une expérience riche mais aussi agréable.

J'adresse mes sincères remerciements à Van-Dat Cung et Jin-Kao Hao pour avoir accepté de rapporter ce travail.

J'exprime ma gratitude à Jaques Carlier ainsi qu'à Mhand Hifi pour m'avoir fait l'honneur de participer à mon jury.

Mes remerciements vont aussi au Conseil Régional de Picardie et au FSE qui ont apporté le financement nécessaire à cette thèse, ainsi qu'à l'ensemble des membres du laboratoire Heudiasyc, pour leur accueil chaleureux.

Je souhaite également remercier François Clautiaux qui m'a fourni plusieurs supports tels des articles, données et sources, qui m'ont été d'une grande utilité, notamment au démarrage de la thèse. Je remercie tous les membres de l'équipe ARO Heudiasyc, en particulier Antoine Jouglet, avec qui j'ai eu l'occasion de collaborer. Je remercie également tous mes collègues qui m'ont permis de travailler dans une ambiance excellente.

Merci à Dima Abi-Abdallah, François Clautiaux, David Savourey et Hermann Bouly pour avoir relu des chapitres de mon rapport de thèse.

Mes remerciements vont aussi à tous mes amis, merci de m'avoir soutenu pendant les moments difficiles, et j'espère avoir su partager avec vous les moments de joie.

Enfin, je salue et remercie mes parents auxquels je dédie ce travail. Je remercie également mes proches, en particulier mes frères Antoine, Pierre, Michel et Dany ainsi que ma soeur Véra, votre soutien m'a été indispensable pour en arriver là.

Table des matières

Liste des tableaux

Liste des figures

Introduction

L'optimisation combinatoire est une discipline qui présente un intérêt majeur tant sur le plan théorique que pratique.

Sur le plan théorique, de nombreux concepts très novateurs ont été proposés lors du siècle dernier. Aujourd'hui, ces concepts continuent d'être enrichis par une bibliographie très riche et très dense.

Sur le plan pratique, cette discipline connaît un regain d'intérêt majeur, notamment sur le plan logistique, discipline dans laquelle l'optimisation joue un rôle essentiel.

Par exemple, les problèmes de type bin-packing ont été abordés dans la littérature dès les années 1950, essentiellement sur des problèmes en une dimension. Aujourd'hui, la littérature est très riche sur ce seul sujet et bons nombres d'articles traitent de problèmes en deux dimensions. Plus récemment (depuis une dizaine d'années), quelques articles sur des problèmes en trois dimensions (ou plus) ont été proposés.

Dans la présente thèse, nous traitons le problème de bin-packing en deux dimension. Ce problème consiste à déterminer le nombre minimum de rectangles identiques (les bins) à utiliser pour ranger un ensemble donné de rectangles plus petits (les objets).

Ce problème, fort simple en apparence, est NP-difficile au sens fort. Il se retrouve de façon directe ou indirecte dans de nombreux problèmes industriels. De façon directe, les problèmes de découpe (Cutting) se modélisent sous la forme de problèmes de bin-packing. L'objectif est alors de minimiser les chutes de matières premières. D'autre part, les problèmes de chargement ou d'assemblage (Loading et Packing) sont aussi directement issus de la même problématique. Dans le domaine de la logistique, on est par exemple amené à minimiser le nombre de containers nécessaires au transport de marchandise. Cela permet d'apporter des améliorations conséquentes aux logiciels consacrés à la SCM (Supply Chain Management) ou aux WMS (Warehouse Management Systems), les systèmes de gestion d'entrepôts. De façon indirecte, le problème de bin-packing permet aussi de modéliser de nombreux problèmes d'affectation avec contraintes ainsi que certains problèmes d'ordonnancement. Dans tous les cas, la nature très riche de cette problématique a donné naissance à plusieurs classifications dans la littérature. On distingue notamment la contrainte guillotine, qui impose des coupes allant de bout en bout des bins pour restituer les

objets. Une autre spécificité est la possibilité de tourner les objets de 90 degrés (le cas non-orienté). Cette possibilité est prise en compte dans le cadre de cette thèse.

Le problème de bin-packing étant NP-difficile, l'énumération de toutes les solutions réalisables pour trouver la meilleure solution s'avère impossible même pour un problème de taille moyenne. Toutefois, on peut aborder la résolution du problème par bornes : des bornes supérieures (évaluation par excès) et des bornes inférieures (évaluation par défaut). Les bornes supérieures sont obtenues par des méthodes de résolution approchée du problème, en particulier des heuristiques et des méta-heuristiques. Ces méthodes fournissent de bons résultats en un temps raisonnable. Les résultats obtenus par ces méthodes sont comparés aux bornes inférieures calculées, on a la valeur optimale si la borne supérieure est égale à la borne inférieure. La borne inférieure la plus connue est la borne continue. Elle consiste dans le cas deux dimensions à sommer la surface des objets et la diviser par la surface du bin, la partie entière supérieure du résultat étant une borne inférieure valide. Des bornes inférieures plus performantes ont été proposées dans la littérature. On peut aussi appliquer des prétraitements sur les instances à résoudre. Les prétraitements permettent en général la réduction de la taille des instances pour pouvoir les résoudre d'une façon exacte et/ou augmenter la taille de certains objets, et ainsi les bornes inférieures.

Le premier chapitre est principalement consacré à la description des bornes les plus connues de la littérature. Nous commençons par une description de la variété de contraintes pratiques qui reflètent la variété des problèmes réels identifiés comme problèmes de bin-packing. Nous citons également les typologies les plus connues. Puis, nous donnons la définition générale du problème et décrivons des modèles proposés. Nous présentons également les prétraitements proposés dans la littérature. Nous décrivons ensuite les heuristiques et les méta-heuristiques ainsi que les bornes inférieures les plus récentes. Enfin, nous décrivons plusieurs méthodes exactes proposées dans la littérature.

Les chapitres 2, 3, 4 et 5, sont consacrés à la description de notre contribution dans la résolution des problèmes de bin-packing en deux dimensions dans le cas non-orienté.

La principale contribution du chapitre 2 est la proposition de trois nouveaux prétraitements. Nous introduisons en particulier un prétraitement basé sur une nouvelle notion que nous appelons *objets à orientation dépendante*. Ces prétraitements permettent d'identifier les espaces perdus et les valoriser en augmentant la taille de certains objets. Si un objet prend la taille d'un bin, son rangement optimal devient trivial et il peut alors être séparé de l'instance à résoudre. Les prétraitements cherchent aussi à trouver des assemblages optimaux pour quelques sous-ensembles d'objets, ce qui permet d'appliquer des transformations sur les objets. Ces transformations consistent en général à augmenter la taille des grands

objets et éliminer les petits objets sans changer la valeur optimale du problème. Les nouveaux prétraitements proposés dans le chapitre permettent de réduire la taille de nombreuses instances. Ils permettent par exemple de réduire à 0 la taille de certaines instances.

Dans le chapitre 3, nous proposons une nouvelle borne inférieure appelée L^R_{CJE}. Nous commençons par la description des bornes inférieures proposées dans la littérature. Nous présentons ensuite le schéma général d'évaluation. Il consiste à transformer l'instance d'un problème non-orienté de taille n en une instance du problème orienté de taille $2n$. Nous montrons que si LB est une borne inférieure valide pour l'instance transformée (de taille $2n$), alors $\lceil LB/2 \rceil$ est une borne inférieure valide de la solution optimale de l'instance initiale (de taille n). La borne L^R_{CJE} consiste à appliquer les meilleures bornes inférieures connues dans la littérature pour le problème orienté à l'instance transformée par le schéma général proposé. Nous montrons enfin que les bornes inférieures obtenues par l'application de L^R_{CJE} dominent théoriquement et pratiquement celles de la littérature.

La principale contribution du chapitre 4 est la proposition d'une nouvelle heuristique de résolution. Cette heuristique est basée sur une stratégie Best-Fit adaptée au problème en deux dimensions. Elle consiste à choisir à chaque étape du rangement, le meilleur couple (objet, espace). Les espaces vides sont gérés par des *surfaces maximales*, une notion définie et étudiée dans le chapitre. Cette façon de gérer les espaces permet de considérer tous les espaces vides présents dans la configuration. Nous démontrons que le nombre de surfaces maximales est en $O(n^2)$ si le rangement considéré est *stable* (*i.e.* chaque objet placé ne peut pas être décalé davantage en bas ou à gauche). Les résultats expérimentaux obtenus sur les benchmarks de la littérature montrent que le nombre de ces surfaces n'excède pas $2n$ en pratique. Nous comparons la nouvelle heuristique proposée (IMA) avec les heuristiques et les méta-heuristiques récentes de la littérature et nous mettons en évidence son efficacité.

Dans le chapitre 5, nous proposons un nouvel algorithme tabou pour le problème étudié, basée sur la notion de *configurations équivalentes* introduite dans le chapitre. Nous commençons par une représentation générale de la méthode tabou. Nous décrivons ensuite l'heuristique de base et le voisinage utilisés dans le nouvel algorithme. Nous introduisons la notion de *configurations équivalentes*, une notion qui nous permet de réduire le voisinage en éliminant les solutions qui présentent certaines *similarités* avec la solution courante, dans le but d'échapper plus vite à des optima locaux. Nous décrivons ensuite la fonction d'évaluation utilisée et nous donnons le schéma général de notre méthode. Nous testons notre algorithme tabou sur les benchmarks de la littérature : il améliore les bornes supérieures pour plusieurs instances de la littérature.

Notre contribution dans le cadre de cette thèse concerne la proposition de prétraitements, de nouvelles bornes inférieures, une nouvelle méthode heuristique et un nouvel algorithme tabou pour la résolution du problème de bin-packing dans le cas non orienté. Nos méthodes, testées sur les benchmarks de la littérature, améliorent les bornes supérieures et inférieures pour plusieurs instances.

Problèmes de bin-packing

1.1 Introduction

Etant donné un ensemble d'objets de formes rectangulaires de dimensions quelconques connues et étant donné un bin de forme rectangulaire plus large de dimensions connues, le problème de bin-packing en deux dimensions ($2BP$) consiste à déterminer le nombre minimum de bins nécessaires pour ranger sans chevauchement l'ensemble de ces objets (les objets ne débordent pas des bins et ne se chevauchent pas). Les objets sont rangés de telle façon que leurs arêtes soient parallèles à celles des bins qui les contiennent (on parle de rangement orthogonal). Ce problème a beaucoup d'applications industrielles. On le retrouve essentiellement dans l'industrie du tissu, de l'acier, du bois, ou du verre sous forme de problème de découpe. On trouve aussi d'autres applications comme la planification de tâches avec contraintes de ressources, ou l'optimisation du placement de publicités dans un journal. Chaque application réelle présente ses propres spécificités et contraintes pratiques. Une extension particulière de ce problème est la possibilité de tourner les objets de 90 degrés.

Le $2BP$ est une généralisation du problème de bin-packing en une dimension ($1BP$) qui est connu comme étant un problème NP-difficile. Il en va de même pour le $2BP$. La bibliographie du problème $2BP$ est riche. En effet, ce problème a connu un vif intérêt durant les dix dernières années. La plupart des travaux portent sur le problème avec orientation fixe des objets.

Dans ce chapitre, nous donnons la définition des problèmes de bin-packing en une, deux et N-dimensions. Nous décrivons aussi les travaux récents portant sur la résolution de ces problèmes.

Nous décrivons, dans la section 1.2, la typologie des problèmes et les contraintes rencontrées souvent dans les problèmes réels. Nous donnons par la suite (Section 1.3) la définition générale du problème de bin-packing. Les modèles les plus connus sont décrits dans la section 1.4. Dans la section 1.5 nous décrivons les prétraitements proposés dans la littérature. Les sections 1.6 et 1.7 sont dédiées aux méthodes de résolution approchée, elles sont respectivement consacrées aux heuristiques et aux méta-heuristiques. Dans la section 1.8, nous exposons les principales bornes

inférieures. La section 1.9 est dédiée à la description de quelques méthodes exactes. Enfin, nous concluons.

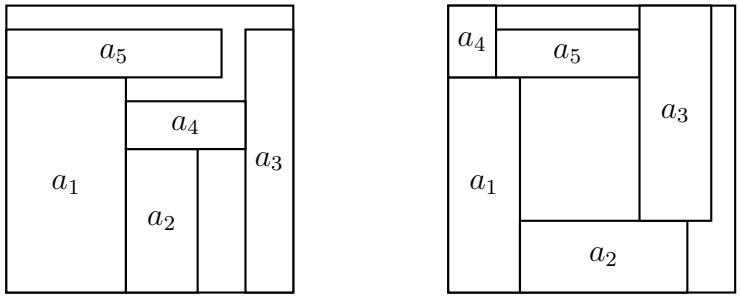
1.2 Classification et contraintes pratiques

De nombreux problèmes pratiques se modélisent sous la forme d'un problème de bin-packing. Cependant, chaque problème réel présente ses propres spécificités telles que :

- les caractéristiques propres aux objets :
  - objets de formes homogènes ou non homogènes ;
  - objets de tailles uniformes ou différentes ;
  - objets déformables ou non déformables ;
  - etc.
- les spécificités propres au problème :
  - le nombre de dimensions du problème ;
  - disposer d'un seul bin (problème de décision ou problème de maximisation) ;
  - chercher à minimiser le nombre de bins à utiliser ;
  - chercher à minimiser la surface ou le volume global des objets à placer ;
  - etc.
- les contraintes propres au problème :
  - contraintes d'équilibre entre les objets ;
  - contraintes sur l'ordre dans lequel les objets doivent être retirés du bin ;
  - contraintes d'orientation d'un objet ;
  - contraintes de poids (par exemple, le poids d'un bin complet ne peut pas excéder une limite donnée) ;
  - contraintes de placement, certains objets très lourds doivent être placés en bas, d'autres fragiles doivent être placés en dessus ;
  - etc.

Dyckhoff et Finke [21] ont proposé une typologie qui permet d'organiser les problèmes de découpes et de placements en tenant compte de quatre caractéristiques principales :

- le nombre de dimensions du problème ;
- le type de tâche : tous les objets et une sélection de bins, ou bien une sélection d'objets et tous les bins ;
- les caractéristiques des bins : 1 seul bin, des bins de tailles identiques, ou bien des bins de tailles différentes ;
- les caractéristiques des objets : objets identiques, peu d'objets de formes différentes, plusieurs objets de formes différentes ou bien des objets de formes relativement identiques.



(a) Contrainte guillotine respectée.

(b) Contrainte guillotine non respectée.

Figure 1.1 – Illustration de la contrainte guillotine.

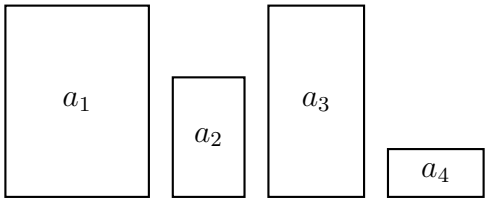
Une typologie plus récente a été proposée par Wäscher *et al.* [51] dans le but d'inclure les problématiques récentes de découpe et de rangement, et établir une catégorisation complète de tous les problèmes connus dans le domaine.

En ce qui concerne le problème de bin-packing en deux dimensions ($2BP$), les deux spécificités les plus rencontrées sont les suivantes :

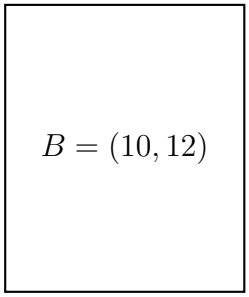
- *l'orientation* : les objets peuvent être à orientation fixe (on parle du cas orienté) ou bien ils peuvent être tournés de 90 degrés (le cas non orienté). C'est cette spécificité qui nous intéresse en particulier. Au cours de ce chapitre nous développons plusieurs travaux effectués autour du problème $2BP$, le cas orienté aussi bien que le non orienté.
- *La contrainte guillotine* : Si elle est imposée, on doit avoir la possibilité de restituer les objets rangés par des coupes bout à bout parallèles aux dimensions de bins. Dans la figure 1.1 nous présentons deux exemples de solutions : la solution illustrée dans la figure 1.1(a) respecte la contrainte guillotine, ce qui n'est pas le cas de la solution illustrée dans la figure 1.1(b).

Dans [45] une classification des problèmes $2BP$ tenant compte des deux spécificités citées ci-dessus a été proposée. Un problème en deux dimensions est alors désigné par 2BP$|C1|C2$. Le champ $C1$ prend la valeur R pour indiquer le cas non orienté et la valeur O pour le cas orienté. Tandis que le champ $C2$ indique la présence ou l'absence de la contrainte guillotine (il prend ainsi la valeur G et F respectivement).

La contrainte guillotine fait l'objet de plusieurs travaux récents. Tout au long de ce document nous ne considérons pas la contrainte guillotine, nous allons omettre le champ $C2$ et nous parlerons de $2BP|O$ (resp. $2BP|R$) pour désigner le problème $2BP$ dans le cas orienté (resp. non orienté).



(a) L'ensemble des objets



(b) Le bin $B = (W, H)$

Figure 1.2 – Un exemple d'instance 2BP.

1.3 Définition du problème et notations

Plus formellement, le problème de bin-packing en deux dimensions (2BP) est défini de la façon suivante : étant donné un ensemble de n objets rectangulaires $A = \{a_1, ..., a_n\}$ et un nombre illimité de rectangles identiques (*les bins*) de dimensions plus larges que celles des objets, le problème consiste à déterminer le nombre minimum de bins à utiliser pour ranger l'ensemble des objets sans chevauchement.

Ce problème appartient à la famille des problèmes de rangement et de découpe (*cutting and packing problems* : C & P). Plus précisément, ce problème est noté $2SBSBPP$ (*Two-Dimensional Single Bin Size Bin-Packing Problem*) selon la typologie de Wäscher *et al.* (*cf.* [51]).

Nous notons (w_i, h_i) les dimensions d'un objet a_i appartenant à l'ensemble des rectangles à ranger A, et (W, H) les dimensions du bin B : $B = (W, H)$. Nous considérons, sans perte de généralité, que les dimensions des objets et du bin sont des entiers. Une instance I de $2BP|R$ est alors définie par la paire (A, B). La valeur optimale du nombre de bins nécessaires pour ranger tous les objets d'une instance I est notée $OPT(I)$. Nous notons l_i la plus petite dimension de l'objet a_i et L_i l'autre dimension. l et L désignent respectivement la plus petite et la plus grande dimension du bin, ainsi :

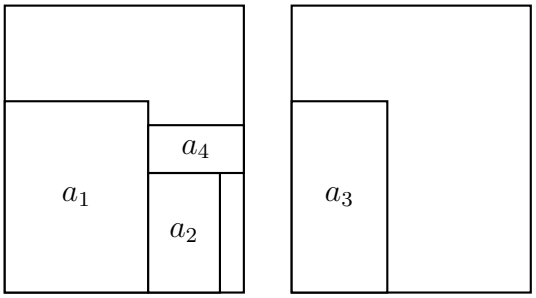


Figure 1.3 – Une solution pour le $2BP|O$

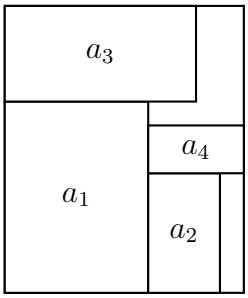


Figure 1.4 – Une solution pour le $2BP|R$

$$l_i = min(w_i, h_i) \; et \; L_i = max(w_i, h_i)$$

$$l = min(W, H) \; et \; L = max(W, H)$$

Un exemple d'une instance 2BP est illustré dans la figure 1.2 où $I = (A, B) = (\{a_1 = (6, 8), a_2 = (3, 5), a_3 = (4, 8), a_4 = (4, 2), \}, B = (10, 12))$. Ainsi pour cet exemple $w_1 = 6$, $h_1 = 8$, $l_1 = w_1$, $L_1 = h_1$, $l = W = 10$ et $L = H = 12$.

Un exemple de solution de cette instance dans le cas non-orienté est donnée dans la figure 1.3. Cette solution est optimale. La figure 1.4 représente une solution pour la même instance, mais en considérant cette fois ci le problème $2BP|R$. La possibilité de tourner les objets de 90 degrés nous a permis de trouver une solution avec un seul bin.

Le problème de bin-packing en une dimension ($1BP$) est défini d'une façon similaire au $2BP$. Une instance $1BP$ peut être définie comme une instance $2BP$ particulière où tous les objets et le bin ont une largeur identique. Alors que la

question d'orientation des objets se pose pour le $2BP$, elle n'a plus de sens dans le cas de $1BP$.

Le problème de bin-packing unidimensionnel est NP-difficile au sens fort [31]. Il en est de même pour le 2BP qui est une généralisation du problème unidimensionnel.

La définition du problème peut se généraliser dans le cas de N-dimensions. Les objets et les bins seront de dimensions N.
Dans la section suivante, nous décrivons quelques modèles proposés dans la littérature pour le problème en deux dimensions.

1.4 Modèles pour le $2BP$

Dans cette section nous décrivons deux exemples de modèles pour le $2BP$.

Un premier modèle dédié au $2BP$ a été proposé par Gilmore et Gomory [34] comme une extension de leur approche $1BP$ [32, 33]. Leur modèle est basé sur l'énumération de tous les sous-ensembles d'objets qui peuvent être rangés dans un même bin. Chacun de ces ensembles est représenté par un vecteur colonne binaire V_j composé de n éléments v_{ij}, $i = 1, \ldots, n$ tel que :

$$v_{ij} = \begin{cases} 1 \text{ si } a_i \text{ appartient au sous-ensemble } j \\ 0 \text{ sinon} \end{cases}$$

Soit M la matrice composée des vecteurs V_j $(j = 1, ..., m)$, m étant le nombre de sous-ensembles qui peuvent être placés dans un même bin. La matrice M représente donc l'ensemble de toutes les configurations de rangements réalisables. Le modèle s'écrit alors

$$\text{Minimiser} \sum_{j=1}^{m} x_j \tag{1.1}$$

En respectant les contraintes suivantes :

$$\sum_{j=1}^{m} v_{ij} x_j = 1 \quad (i = 1, \ldots, n) \tag{1.2}$$

$$x_j \in \{0, 1\} \quad (j = 1, \ldots, m) \tag{1.3}$$

L'inconvénient majeur de ce modèle est le nombre exhaustif de colonnes à générer. Gilmore et Gomory [32,33] ont présenté pour le $1BP$, une méthode pour générer dynamiquement les colonnes au besoin. Elle consiste en la résolution d'un problème de sac à dos unidimensionnel.

Fekete et Schepers [28] ont proposé un modèle basé sur la théorie de graphes pour le problème de décision suivant : est-il possible de placer un ensemble donné d'objets dans un seul bin ?

L'intérêt du modèle proposé est qu'il évite un grand nombre de configurations symétriques. Deux graphes $G_W = (V, E_W)$ et $G_H = (V, E_H)$, chacun de taille n, sont considérés, n étant le nombre d'objets. Chaque sommet v_i de G_W et de G_H est associé à un objet a_i rangé dans le bin. Les deux graphes sont construits de la façon suivante : une arête est rajoutée dans G_W (resp. dans G_H) entre les deux sommets v_i et v_j si et seulement si les projections des objets a_i et a_j sur l'axe horizontal (resp. vertical) se chevauchent. Fekete et Schepers montrent qu'une solution associée à un couple de graphes d'intervalles (G_W, G_H) est réalisable si les trois conditions suivantes sont vérifiées :

1. Pour chaque ensemble stable S_W de G_W, on a $\sum_{v_i \in S_w} w_i \leq W$,
2. Pour chaque ensemble stable S_H de G_H, on a $\sum_{v_i \in S_h} h_i \leq H$,
3. $E_w \cap E_h = \emptyset$.

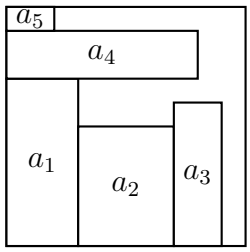
Les deux premières conditions signifient que l'ensemble des objets tient dans le bin, tandis que la troisième condition signifie qu'il n'y a pas de chevauchement entre deux objets quelconques rangés dans le bin.

D'autres modèles ont été proposés pour le problème de bin-packing et ses variantes. Beasley [2] a proposé un modèle linéaire pour la caractérisation des problème de découpe 2D qui associent un profit à chaque objet, l'objectif étant de ranger un sous-ensemble d'objets dans un bin de façon à maximiser la somme des profits des objets rangés (problème de sac à dos). Hadjiconstantinou et Christofides [36] ont travaillé sur un modèle similaire. Une modélisation par programmation linéaire du problème de découpe guillotine de bande (Strip Packing) en deux dimensions a été proposée récemment dans la thèse de Ben Messaoud [49].

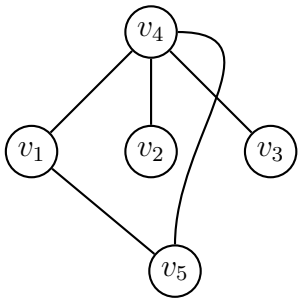
Dans la section suivante nous décrivons différents prétraitements proposés dans la littérature pour le problème de bin-packing.

1.5 Prétraitements

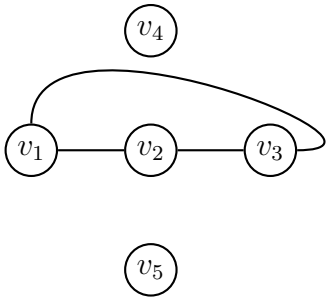
Dans les problèmes difficiles comme le cas du problème de bin-packing, il est souvent intéressant d'appliquer des prétraitements sur l'instance à étudier avant de lancer une méthode de résolution coûteuse en terme de temps de calcul. Les prétraitements peuvent aussi contribuer à l'amélioration des bornes inférieures en valorisant les espaces qui seront perdus indépendamment de la méthode de résolution du problème. Il n'existe pas à notre connaissance des prétraitements dédiés au 2BP|R. Dans cette section, nous décrivons les prétraitements proposés dans la littérature pour le $2BP|O$.



(a) Exemple de solution.



(b) graphe G_W.



(c) graphe G_H.

Figure 1.5 – Exemple d'application du modèle de Fekete et Scheppers.

1.5.1 Prétraitement de Martello et Vigo

Martello et Vigo [48] ont proposé en 1998 un prétraitement qui consiste à chercher à déterminer des assemblages optimaux. Soit a_i un objet donné de l'instance considérée, on cherche à trouver l'ensemble des objets à placer avec a_i tel que le rangement du bin contenant a_i soit optimal.

Soit Q_i l'ensemble des objets compatibles avec a_i :

$$Q_i = \{a_j \in A : (h_i + h_j) \leq H \text{ ou } (w_i + w_j) \leq W\}$$

Deux cas se présentent. Dans le premier cas, $Q_i = \emptyset$, cela signifie qu'aucun objet de $A \setminus a_i$ ne peut être rangé avec a_i. Le rangement optimal de l'objet a_i est trivial, il doit être rangé seul dans un bin. Dans le deuxième cas, $Q_i \neq \emptyset$, une borne supérieure sur le nombre d'objets qui peuvent être rangés avec a_i est alors calculée de la façon suivante. Les objets de Q_i sont considérés par ordre décroissant de surfaces. La valeur maximale l telle que la somme des l derniers objets de Q_i ne dépasse pas la valeur $HW - h_i w_i$, est une borne supérieure sur le nombre d'objets qui peuvent être rangés avec a_i. Soit a_{j*} le premier (le plus large) objet dans q_i, un prétraitement est appliqué dans les deux cas de figures suivants :

- $l = 1$ et pour tout a_j dans $Q_i \setminus a_{j*}$, $h_{j*} \geq h_j$ et $w_{j*} \geq w_j$
- $l = 2$ et toute paire d'objets de $Q_i \setminus a_{j*}$ qui peuvent être placés avec a_i, peuvent être placés dans un bin de dimensions égales aux dimensions de a_{j*}.

Dans les deux cas, le meilleur choix consiste à ranger a_{j*} avec a_i.

1.5.2 Prétraitement de Boschetti et Mingozzi

Boschetti et Mingozzi [5] ont proposé en 2003 un prétraitement qui consiste à mettre à jour les dimensions des objets en tenant compte de la surface qui sera perdue dans le bin où ils seront affectés, quelle que soit la configuration considérée.

Etant donné un objet a_i, il s'agit de calculer la largeur w_i^* obligatoirement perdue lorsque a_i est placé dans un bin. Cette valeur peut être calculée en résolvant une variante du problème de subset-sum :

$$w_j^* = w_j + max \left\{ \sum_{a_i \in A - \{a_j\}} w_i \xi_i : \sum_{a_i \in A - \{a_j\}} w_i \xi_i \leq W - w_j, \xi_i \in \{0, 1\} \right\}$$

w_j^* peut être calculée en $O(nW)$ par l'intermédiaire d'une méthode pseudo-polynomiale classique.

La largeur w_i de l'objet a_i peut être alors mise à jour de la façon suivante :

$$w_j \leftarrow w_j + (W - w_j^*)$$

Le prétraitement est aussi appliqué, d'une façon similaire, à la hauteur h_i de l'objet a_i. Ce prétraitement est appliqué à tous les objets de l'instance traitée, mais l'ordre dans lequel les objets sont considérés est important. En effet, si la largeur d'un objet a_j entre en jeu dans le calcul de la valeur optimale w_i^* d'un objet a_i, w_j^* aura comme valeur W et w_j ne peut plus être mis à jour à son tour. Une règle heuristique qui consiste à considérer les objets par ordre décroissant de hauteur (resp. largeur) avant d'appliquer le prétraitement sur les hauteurs (resp. largeurs) des objets est appliquée dans le but de maximiser le nombre d'objets prétraités.

1.5.3 Prétraitements de Carlier *et al.*

Carlier *et al.* [10] ont proposé en 2005 deux procédures de réductions sous forme des fonctions *identiquement réalisables* (*Identically-Feasible Functions*) définies de la façon suivante. Etant donnée une instance $I = (A, B)$, une fonction f est une IFF associée à la donnée I si et seulement si l'instance $f(D) = (\{a'_1, \ldots, a'_n\}, B)$ obtenue en appliquant f à tous les objets de A est telle que $OPT(I) = OPT(f(I))$.

Dans la première IFF appelée v_1, étant donnée une instance $I = (A, B)$, deux sous-ensembles particuliers sont considérés : A_H le sous-ensemble des objets *hauts* et A_s le sous-ensemble des objets *courts* qui peuvent être placés au dessus d'un objet de A_H.

Soit q un entier tel que $1 \leq q \leq \frac{1}{2}H$.

- $A_H = \{a_i \in A : h_i > H - q\}$
- $A_s = \{a_i \in A : h_i < q\}$

Soit $m = |A_H|$. Pour chaque objet a_i de A_H, un bin B_i de taille $(H - h_i, w_i)$ est considéré. Soit le problème de décision $\mathcal{P}_1$ suivant : est-il possible de ranger tous les objets de A_s dans les bins $B_1, \ldots, B_m$ créés ?

Si $\mathcal{P}_1$ a une solution, les objets de A_s peuvent alors être éliminés et les hauteurs des objets de A_H sont mises à jour, elles prennent la hauteur du bin. La fonction v_1^q définie ci-après est dans ce cas une IFF associée à l'instance I.

$$v_1^q : A \to A'$$

$$a_i \mapsto a'_i \text{ où } \begin{cases} w'_i = w_i \text{ et } h'_i = H \text{ if } h_i > H - q \\ w'_i = 0 \text{ et } h'_i = 0 \text{ if } h_i < q \\ w'_i = w_i \text{ et } h'_i = h_i \text{ sinon} \end{cases}$$

Soit v_1^H l'application itérative de v_1^q. En ne considérant que les valeurs intéressantes de q, celles qui correspondent aux hauteurs des objets, v_1^H peut être implantée en $O(n \times \Gamma(n))$ où $\Gamma(n)$ est la complexité de l'algorithme utilisé pour résoudre le problème de décision $\mathcal{P}_1$. Une fonction v_1^W est définie d'une façon similaire en

remplaçant H par W. La fonction finale v_1 est définie comme l'application itérative de v_1^W et v_1^H tant que l'instance est modifiée.

Dans la deuxième procédure de réduction v_2 proposée par les auteurs, étant donnée une instance $I = (A, B)$, les deux sous-groupes ciblés sont appelés A_G et A_P, ils contiennent respectivement les *grands* objets qui occupent plus de la moitié du bin en hauteur et en largeur et les *petits* objets qui peuvent être rangés dans un même bin avec un objet de A_G. Ils sont définis de la façon suivante :

- $A_G = \{a_i \in A : w_i > W - p \text{ et } h_i > H - q\}$
- $A_P = \{a_i \in A : w_i < p \text{ ou } h_i < q\}$

où p et q sont deux entiers tels que $1 \leq p \leq \frac{1}{2}W$ et $1 \leq q \leq \frac{1}{2}H$.

Soit $\mathcal{P}_2$ le problème de décision suivant : Est-il possible de ranger les objets de $A_P \cup A_G$ dans $|A_G|$ bins de taille B ?

Si $\mathcal{P}_2$ a une solution, cela signifie que tous les objets de A_P peuvent être placés dans les bins qui contiennent les objets de A_G, ils peuvent alors être éliminés, et chaque objet de A_G prend désormais la taille d'un bin. Dans ce cas, la fonction $v_2^{p,q}$ suivante est une IFF associée à l'instance I :

$$v_2^{p,q} : A \to A'$$

$$a_i \mapsto a'_i \text{ où } \begin{cases} w'_i = W \text{ et } h'_i = H \text{ si } w_i > W - p \text{ et } h_i > H - q \\ w'_i = 0 \text{ et } h'_i = 0 \text{ si } w_i < p \text{ ou } h_i < q \\ w'_i = w_i \text{ et } h'_i = h_i \text{ sinon} \end{cases}$$

La fonction v_2 consiste à appliquer la fonction $v_2^{p,q}$ pour toutes les valeurs intéressantes des paramètres p et q (*i.e.* celles qui correspondent à la taille d'un article de A). La méthode est en $O(n^2\Gamma(n))$, où $\Gamma(n)$ est la complexité de l'algorithme utilisé pour résoudre le problème de décision $\mathcal{P}_2$.

Les problèmes de décisions $\mathcal{P}_1$ et $\mathcal{P}_2$ définis dans les deux prétraitements v_1 et v_2 étant des variantes du problème de bin-packing ils sont NP difficiles. Les auteurs utilisent des heuristiques simples pour résoudre les problèmes de décision.

Dans la section suivante nous décrivons quelques méthodes heuristiques proposées dans la littérature de $2BP$ et $1BP$.

1.6 Heuristiques de résolutions

Comme dans tous les problèmes combinatoires, des méthodes heuristiques ont été proposées pour le problème de bin-packing dans le but de trouver des solutions de bonne qualité dans un temps raisonnable.

Dans cette section, nous présentons quelques méthodes heuristiques proposées pour le $1BP$, puis nous décrivons les heuristiques de résolution récentes proposées pour le $2BP$ qui utilisent des critères généralisés du $1BP$.

1.6.1 Le cas 1BP

Dans cette section, nous décrivons quelques heuristiques proposées pour le $1BP$ (*cf.* [17] pour un état de l'art et les résultats d'analyse au pire cas). Les stratégies utilisées dans le $1BP$ ont inspiré plusieurs approches de résolution proposées plus tard pour des problèmes de bin-packing de dimension plus élevée.

Stratégie Next Fit (NF). Dans cette stratégie on ne considère qu'un bin ouvert à la fois. Les objets sont traités selon un ordre donné. Les objets sont rangés successivement dans le bin ouvert tant qu'il y a de la place pour l'objet a_i en cours, sinon le bin en cours est fermé et un nouveau bin est ouvert. Une heuristique qui adopte une stratégie NF a l'avantage d'avoir une complexité temporelle linéaire en fonction du nombre d'objets à placer. Par contre, le fait de ne considérer qu'un seul bin à la fois cause beaucoup de perte d'espaces exploitables. Le Next Fit Decreasing (NFD) consiste à trier les objets par ordre décroissant de hauteurs et appliquer la stratégie NF pour les ranger.

Stratégie First Fit (FF). Initialement un seul bin est considéré, et les objets sont traités selon un ordre donné. Quand il n'y a plus de la place dans le premier bin pour ranger l'objet en cours, un deuxième bin est alors ouvert mais sans fermer le premier. Dans une étape intermédiaire où on dispose de k bins ouverts numérotés de 1 à k selon l'ordre de leur première utilisation, un objet a_i en cours est rangé dans le bin du plus faible numéro qui peut le contenir. Dans le cas où aucun bin ne peut contenir a_i un nouveau bin $k+1$ est alors utilisé sans fermer les autres. L'ordre selon lequel on traite les objets est crucial pour la qualité de la solution. Un choix heuristique consiste à trier les objets par ordre décroissant de hauteurs, on parle dans ce cas d'heuristiques First Fit Decreasing.

Stratégie Best Fit (BF). Comme dans la stratégie FF, les algorithmes BF laissent les bins toujours ouverts. Cependant, le choix du bin dans lequel l'objet a_i en cours va être placé dépend des valeurs des *gaps* (hauteurs non utilisées) présentes dans les bins. Ainsi, a_i sera placé dans le bin qui présente le moindre gap parmi les bins qui peuvent le contenir. Les heuristiques BF et FF peuvent être implantées en $O(nlog(n))$ en utilisant une structure de données appropriée [41]. On parle de Best Fit Decreasing (BFD) quand il s'agit de trier les objets à ranger dans l'ordre décroissant de hauteurs avant de les ranger suivant une stratégie BF.

Nous citons aussi la stratégie Worst Fit (WF) qui est une variante du BF, suivant

laquelle l'objet en cours est placé dans le bin qui peut le contenir et qui présente le plus grand gap. Aussi surprenant que cela puisse paraître, cette stratégie peut produire de meilleures solutions que BF sur certaines instances. Dans une stratégie dite Any Fit (AF), l'objet en cours est affecté à un bin choisi aléatoirement parmi l'ensemble de bins ouverts qui peuvent le contenir.

1.6.2 Le cas 2BP

Plusieurs heuristiques ont été proposées pour le 2BP. Ces heuristiques peuvent être divisées en deus familles différentes [46] : *les algorithmes en une phase* et *les algorithmes en deux phases*. Les algorithmes en une phase consistent à ranger directement les objets dans des bins, tandis que les algorithmes à deux phases commencent d'abord par ranger les objets dans un bin de hauteur infinie en cherchant une solution pour le problème de *strip-packing (2SP)* suivant : étant donné l'ensemble A d'objets à placer et un bin unique de largeur W et de hauteur infinie, l'objectif est de ranger tous les objets de A dans le bin en minimisant la hauteur totale utilisée. En une deuxième phase, un algorithme de résolution pour le problème $1BP$ consiste à utiliser la solution du $2SP$ obtenue pour construire le rangement dans les bins à utiliser effectivement (solution pour le $2BP$).

1.6.2.1 Algorithmes en deux phases

Dans la plupart des approches proposées, la première étape (strip packing) consiste à trier les objets suivant leur hauteur décroissante (Decreasing Height) et les placer successivement dans un container de hauteur infinie, en le remplissant couche par couche, formant ainsi des niveaux définis par la hauteur du plus grand objet par couche. Un exemple de cette première étape est illustré dans la figure 1.6. Le premier objet, le plus long, est placé dans le premier niveau qui correspond à l'arête inférieure du bin. Les autres objets sont rangés suivant une stratégie NF, FF ou BF par rapport à la largeur du bin, on parle respectivement d'une stratégie NFDH, FFDH et BFDH.

Un algorithme en deux phases appelé *Hybrid First Fit* (HFF) a été proposé par Chung *et al.* [13]. Dans la première phase, le problème de strip packing est construit suivant une stratégie FFDH. Soit $H_1, \ldots, H_{NL}$ les hauteurs des NL niveaux résultants. Observons que $H_1 \geq H_2 \geq \ldots \geq H_{NL}$ puisque les objets ont été triés par ordre décroissant des hauteurs. Dans une deuxième étape, une solution du problème $2BP$ initial est obtenue en résolvant un problème $1BP$ avec NL objets a_i ($i = 1, ..., NL$) de taille (W, H_i) chacun correspondant à un niveau (le bin étant de taille (W, H)).

Une variante de l'algorithme HFF a été proposée par Berkey et Wang [4], elle est appelée *Hyprid Best Fit* (HBF). Elle consiste à employer la stratégie BFDH

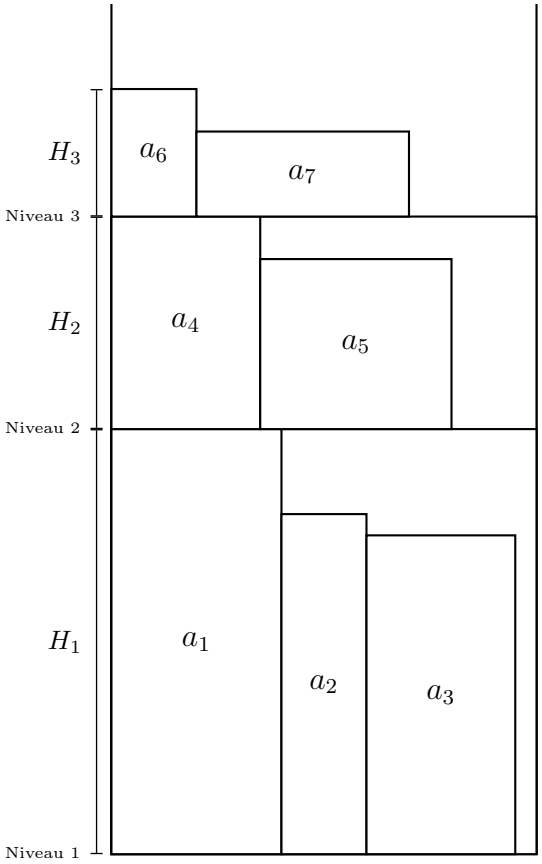


Figure 1.6 – Exemple d'une solution d'un problème de strip packing

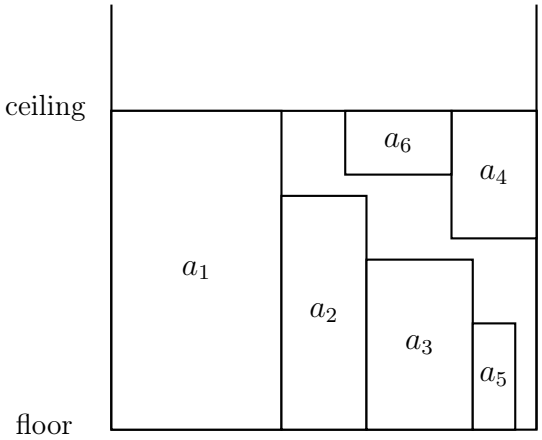


Figure 1.7 – Approche FC

pour résoudre le problème de strip packing (première phase). Pour la résolution du problème $1BP$ de la deuxième phase (ranger les niveaux obtenus dans des bins finis) la stratégie BFD est employée.

Lodi *et al.* [45] ont proposé une approche nouvelle pour le rangement par couche. Cette approche s'appelle *Floor Ceiling* (FC), où la première phase consiste à placer les objets par couche suivant la stratégie BFDH avec la modification suivante. Notons *floor* (resp. *ceiling*) l'horizontale définie par l'arête supérieure (resp. inférieure) d'un niveau donné. Un objet qui ne tient plus sur le *floor* d'un niveau, peut être placé, décalé à droite, et accroché au *ceiling* du niveau (*cf.* 1.7). Dans la deuxième phase, les niveaux sont placés dans des bins finis selon une stratégie BFD, ou bien en utilisant une méthode exacte pour le $1BP$. Une autre approche dite *Knapsack packing* (KP) a été proposée par les auteurs. Dans cette approche, on ne considère qu'un niveau à la fois. Un niveau initialisé est rempli en entier en résolvant un problème de sac à dos en une dimension (1KP). Dans un problème $1KP$, comme pour le $1BP$, on dispose d'un bin de hauteur H et d'un ensemble d'objets de hauteurs données. A chaque objet a_i est associé un profit. L'objectif est de ranger dans le bin un sous-ensemble d'objets qui maximise la somme des profits associés. Le profit associé à chaque objet est la surface dans la méthode KP. La deuxième phase est identique à celle employée dans la méthode FC.

Burke *et al.* [8] ont proposé une nouvelle heuristique pour le problème de strip packing en deux dimensions avec la possibilité de tourner les objets de 90 degrés. Ils adoptent une stratégie BF : à chaque étape du rangement d'un objet, l'espace disponible le plus en bas est considéré. L'objet dont une des dimensions remplit au mieux cet espace y est placé.

1.6.2.2 Algorithmes en une phase

Parmi les algorithmes qui procèdent en une phase, nous trouvons quelques uns qui placent les objets par niveaux comme l'algorithme Finite First Fit (FF) [4]. Cet algorithme consiste à trier les objets par ordre décroissant des hauteurs. L'objet en cours est placé dans le niveau le plus bas du premier bin qui peut le contenir. Si aucun niveau ne peut contenir l'objet en cours un nouveau niveau est créé dans le premier bin approprié ou bien en initialisant un nouveau bin.

Parmi les algorithmes qui ne rangent pas les objets par niveau, nous trouvons essentiellement ceux qui adoptent une stratégie connue sous le non de *Bottom-Left* (BL) : Les objets sont considérés dans un ordre donné, et l'objet en cours est placé dans la position la plus en bas puis le plus à gauche possible. Une implantation efficace a été proposée par Chazelle [11] qui permet de trouver des solutions avec une complexité temporelle en $O(n^2)$.

Lodi *et al.* [45] ont proposé une approche différente appelée *alternative direction* (AD). Soit z une borne inférieure évaluée pour le problème à résoudre (*cf.* Section

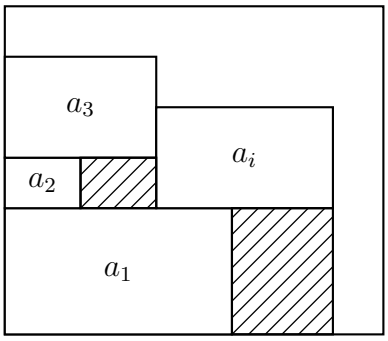


Figure 1.8 – Evaluation du coût du placement de a_i dans une position donnée.

1.8 pour un aperçu sur les bornes inférieures). L'approche AD commence par trier les objets par ordre décroissant de hauteurs, puis elle initialise z bins par un sous-ensemble d'objets suivant une stratégie BFD. Les objets restants sont rangés en bandes en considérant un bin à la fois et allant alternativement de gauche à droite et de droite à gauche dans la position le plus en bas possible. Lorsque l'objet ne peut pas être placé dans le bin courant, on essaie le bin suivant ou bien on initialise un nouveau bin si nécessaire.

Une méthode heuristique appelée *Touching Perimeter* (TP_{RF}) dédiée au $2BP|R$ a été proposée par Lodi *et al.* [45] en 1999. Cette méthode commence par trier les objets par ordre décroissant de leurs surfaces et les orienter horizontalement : chaque objet est orienté tel que son arête de taille maximale soit orientée horizontalement. Ensuite, z bins vides sont initialisés (z correspondant à une borne inférieure connue pour l'instance à traiter, *cf.* Section 1.8). L'algorithme place l'objet en cours dans un bin déjà ouvert, ou en initialisant un nouveau bin. L'objet qui initialise un bin est toujours placé dans le coin bas-gauche du bin. Chaque objet est placé dans une *position normale* (*cf.* Christofides et Witlock [12]) *i.e.* son arête inférieure touche soit l'arête inférieure du bin, soit l'arête supérieure d'un autre objet et son arête gauche touche soit l'arête gauche du bin, soit l'arête droite d'un autre objet. Le choix du bin auquel affecter l'objet et sa position se font en évaluant un *score* défini comme le pourcentage du périmètre de l'objet qui touche le bin et d'autres objets déjà placés. Pour chaque position candidate, le score est évalué deux fois en considérant les deux orientations possibles de l'objet, la valeur la plus élevée est considérée.

Boschetti et Mingozzi [6] ont proposé en 2003 une heuristique itérative appelée HBP qui prend en considération la rotation possible des objets. Ayant un ordre donné des objets, les auteurs proposent une méthode heuristique appelée H_1 qui consiste à placer les objets dans l'ordre en ne considérant qu'un bin à la fois. Lors du placement d'un objet, toutes les positions dites *faisables* sont considérées : l'objet

ne chevauche pas sur d'autres objets et on ne peut pas le décaler davantage vers le bas ou vers la gauche. Une fonction coût est évaluée en considérant les deux orientations possibles de l'objet pour chaque position faisable. La fonction coût est égale à la somme des surfaces non utilisées en dessous et à gauche de l'objet. L'objet est finalement rangé dans la position et l'orientation qui produisent le coût le plus faible. La figure 1.8 illustre les surfaces vides (surfaces hachurées) entrant en jeu dans le calcul du coût du placement de l'objet a_i dans sa position actuelle. Le bin en cours est fermé et un autre est initialisé lorsque l'ensemble de positions faisables par rapport à l'objet courant est vide. L'heuristique HBP consiste à l'application itérative de H_1. Cinq initialisations de priorités d'objets qui dépendent de leurs caractéristiques sont testées : surface, largeur, hauteur et périmètre décroissant. Soit $nbBin$ le nombre de bins utilisés notés de b_1 à b_nbBin suivant leur ordre d'initialisation. Une fois tous les objets sont rangés, les priorités de ces objets sont mises à jour en augmentant celles des objets affectés aux bins b_k si $k > nbBin/2$ et diminuant celles des objets autres objets. L'heuristique H_1 est rappelée après le tri des objets par ordre décroissant de priorités. L'heuristique HBP utilise deux règles de mises à jour des priorités des objets. Les deux règles consistent à multiplier les priorités des objets appartenant aux derniers bins (resp. premiers bins) par $1+a$ (resp. $1-a$). Pour la première règle, a vaut 0.1 tandis que la deuxième règle consiste à tirer la valeur réelle a aléatoirement entre 0 et 1 exclu.

Les algorithmes de résolutions approchées dans la littérature ne se limitent pas aux méthodes heuristiques, certaines approches méta-heuristiques ont été proposées pour le $2BP$, ils font l'objet de la section suivante.

1.7 Méta-heuristiques

Les méta-heuristiques sont des méthodes employées souvent pour la résolution approchée des problèmes fortement combinatoires. On pourrait se référer à [1] pour une introduction aux méta-heuristiques.

Parmi les méta-heuristiques proposées pour le problème de bin-packing et ses variantes, nous citons Dowsland [20] et Jakobs [40] qui ont proposé respectivement un algorithme recuit simulé et un algorithme génétique pour le problème de strip-packing 2D. Une recherche locale guidée a été proposée par Faroe *et al.* [26] pour le problème de bin-packing 3D, une approche qui peut être appliquée pour le cas 2D.

Ci-dessous nous décrivons deux méthodes de recherche tabou qui ont montré leur efficacité dans la résolution des instances $2BP$ et le $2BP|R$ en particulier (voir [35] pour une introduction aux méthodes par recherche tabou).

Lodi *et al.* [45] ont proposé une méthode de recherche tabou que nous notons $TS-TP$. Cette recherche peut être appliquée indépendamment de la méthode de résolution adoptée pour évaluer les voisins.

Ayant une solution courante, la méthode $TS - TP$ cherche à vider un bin pour améliorer la solution. Une première étape consiste alors à désigner le bin le plus facile à vider appelé le bin *cible*. Pour choisir le bin cible, les auteurs utilisent la même stratégie qu'ils avaient utilisée dans leurs travaux précédents [44]. Soit $B(S, i)$ l'ensemble des objets rangés dans le bin i dans la solution courante S. Le bin cible est celui qui minimize la fonction de *remplissage* suivante :

$$\mathcal{R}(i) = \alpha \frac{\sum_{a_j \in B(S,i)} w_j h_j}{WH} - \frac{|B(S,i)|}{n}$$

où α est un coefficient positif déterminé expérimentalement. Cette fonction favorise la sélection du bin dont la surface remplie est petite. En cas d'égalité, le bin qui contient le plus d'objets sera sélectionné car on estime qu'il est plus facile de distribuer des petits objets dans les autres bins que des grands.

Le voisinage utilisé consiste à déplacer un seul objet a_j du bin cible (t) à la fois dans d'autres bins en modifiant le rangement des objets appartenant dans k bins de la solution courante. Les voisins sont évalués en utilisant une méthode de résolution pour ranger les objets des k bins sélectionnés et a_j. L'algorithme maintient k listes tabou. La valeur k est dynamique, sa variation dépend du déroulement de la recherche [45].

Récemment, Harwig et Barnes en 2006 [38] ont proposé une nouvelle méthode de résolution par recherche tabou. L'idée principale de leur méthode consiste à considérer une fonction objectif sélective qui favorise les mouvements des objets de sorte à améliorer la solution en vidant un bin. Cette fonction objectif est basée sur l'approche de l'énergie potentielle, une approche qui a été déjà utilisée par Li et Milenkivic [43] pour le problème de Strip Packing 2D. Le but principal étant de minimiser le nombre de bins, comme dans la méthode précédente, un bin sera désigné comme candidat pour être vidé. Les auteurs considèrent trois types de mouvements qui accélèrent en principe l'atteinte de l'objectif principal. Les trois types de mouvement sont les suivants :

- *excess bin moves*. Le bin désigné comme candidat pour être vidé est noté excess bin. Les mouvements qui réduisent la surface occupée dans ce bin (la somme des surfaces des objets affectés au bin) sont favorisés par la fonction objectif.
- *intra-bin moves*. Si deux mouvements différents produisent deux configurations avec le même rangement d'objet dans un bin, celle qui produit le rangement le plus *compact* est favorisée.
- *inter-bin moves*. Les mouvements qui produisent les configurations les plus compactes sont favorisés. Les configurations compactes présentent des espaces vides plus importants, ce qui facilite le mouvement d'un grand objet du bin excess.

1.8 Bornes inférieures

Nous avons exposé dans les sections précédentes des méthodes de résolution approchée pour le problème de bin-packing. Dans cette section nous décrivons des méthodes d'évaluation par défaut qui fournissent des bornes inférieures pour les instances du bin-packing. Ceci permet de juger la qualité des solutions obtenues par les méthodes de résolution approchée. Les bornes inférieures servent aussi comme critères d'arrêt dans une méthode méta-heuristique ou un algorithme itératif. Avoir des bornes inférieures de bonnes qualités permet aussi de réduire la taille des solutions à énumérer lors de l'application d'une méthode exacte.

Nous synthétisons dans cette section les bornes inférieures proposées dans la littérature pour le $1BP$ et le $2BP$. Certaines de ces bornes seront développées dans le chapitre 3 car elles seront utilisées pour l'étude de la dominance de la nouvelle borne inférieure proposée par rapport à celles de la littérature. Certaines méthodes peuvent être renommées.

1.8.1 Bornes inférieures pour le $1BP$

La borne inférieure la plus connue est la borne continue L_0, elle consiste à sommer la taille des objets de l'instance et la diviser par la taille du bin :

$$L_0 = \lceil \frac{\sum_{a_i \in A} h_i}{H} \rceil$$

Cette borne fournit de bons résultats lorsque l'instance est constituée de petits objets ou bien d'objets uniformément répartis dans [0, H] (*cf.* [3]).

Si l'instance présente un pourcentage élevé d'objets de tailles moyennes, les résultats obtenus par L_0 peuvent vite devenir insatisfaisants.

Martello et Vigo [48] ont montré que la borne continue peut atteindre $\frac{1}{2}OPT(I)$. Ils ont également proposé une borne inférieure pour le $1BP$ appelée L_{1BP}^{MV}. Ils décomposent l'instance en trois sous-ensembles différents : l'ensemble des *grands* objets, des objets *moyens* et des *petits* objets définis respectivement de la façon suivante :

- $A_G = \{a_i \in A : h_i > H - q\}$
- $A_M = \{a_i \in A : H - q \geq h_i > \frac{1}{2}H\}$
- $A_P = \{a_i \in A : \frac{1}{2}H \geq h_i \geq q\}$

où q est un entier tel que, $1 \leq q \leq \frac{1}{2}H$. La borne proposée L_{1BP}^{MV} est composée de deux bornes : (L_α et L_β). La borne L_α a été proposée initialement dans [47]. Les deux bornes L_α et L_β exploitent le fait que les objets de A_P et A_M ne peuvent pas être placés ensemble. La borne L_α intègre aussi une évaluation du nombre de bins nécessaires pour ranger les objets de A_M et A_P, tandis que la borne L_β intègre une

évaluation sur le nombre de bins nécessaires pour ranger les objets de A_P qui ne peuvent pas être placés avec les objets de A_M :

$$L_\alpha = \max_{1\leq q\leq \frac{1}{2}H} \left\{ |A_G \cup A_M| + max \left\{ 0, \left\lceil \frac{\sum_{a_i \in A_M \cup A_P} h_i}{H} - |A_M| \right\rceil \right\} \right\} \tag{1.4}$$

$$L_\beta = \max_{1\leq q\leq \frac{1}{2}H} \left\{ |A_G \cup A_M| + max \left\{ 0, \left\lceil \frac{|A_P| - \sum_{a_i \in A_M} \left\lfloor \frac{H-h_i}{q} \right\rfloor}{\left\lfloor \frac{H}{q} \right\rfloor} \right\rceil \right\} \right\} \tag{1.5}$$

$$L_{1BP}^{MV} = max(L_\alpha, L_\beta) \tag{1.6}$$

Labbé *et al.* [42] ont proposé une borne inférieure pour le $1BP$ qui considère quatre sous-ensembles définis de la façon suivante :

- $A_G = \{a_i \in A : h_i > H - q\}$
- $A_{M1} = \{a_i \in A : \frac{H}{2} < h_i \leq H - q\}$
- $A_{M2} = \{a_i \in A : \frac{H}{3} \leq h_i \leq \frac{H}{2}\}$
- $A_{M3} = \{a_i \in A : k \leq h_i \leq H - q\}$

q étant un entier tel que $(0 \leq q \leq \frac{H}{3})$. Soit A_{M2}^+ le sous-ensemble d'objets de A_{M2} qui ne peuvent être rangés avec aucun objet de A_G, la borne proposée qu'on note L_{1BP}^L s'écrit alors :

$$L_{1BP}^L = \max_{0\leq q\leq \frac{H}{3}} \left\{ |A_G| + max \left\{ |A_{M1}| + \left\lceil \frac{|A_{M2}^+|}{2} \right\rceil, \left\lceil \frac{\sum_{a_i \in A_{M3}} h_i}{H} \right\rceil \right\} \right\} \tag{1.7}$$

Bourjoully et Rebetz [7] Montrent que L_{1BP}^L domine la borne L_α de [47].

Fekete et Schepers [27] ont proposé de nouvelles bornes inférieures pour le $1BP$ en se basant sur le concept des *fonctions dual-réalisables* proposées par Johnson [41] et définies de la façon suivante.

Définition 1.1. *Une fonction* $f : [0,1] \to [0,1]$ *est une* Fonction Dual-Réalisable (DFF) *si pour tout ensemble fini* S *de réels, on a*

$$\sum_{x\in S} x \leq 1 \Rightarrow \sum_{x\in S} f(x) \leq 1$$

Les DFF ont la particularité suivante : une borne inférieure calculée pour une instance transformée par une DFF est une borne valide pour l'instance initiale. Des DFF sont appliqués aux instances dans le but de modifier la taille de certains objets et d'améliorer le rapport de la somme des longueurs des objets sur la longueur d'un

bin (la borne continue). Fekete et Schepers [27] ont proposé trois fonctions dual-réalisables et ont discuté les bornes inférieures obtenues.

Haouari et Gharbi [37] ont proposé une méthode qui permet de mettre au point un processus d'améliorations itératives des bornes inférieures.

Dans la section suivante, nous présentons trois bornes inférieures proposées dans la littérature pour le 2BP|O.

1.8.2 Bornes inférieures pour le $2BP|O$

La borne continue L_0 dans le cas 2D consiste à sommer les surfaces des objets et la diviser par la surface d'un bin. Cette borne est loin d'être satisfaisante, elle peut atteindre un quart de l'optimum au pire des cas.

Martello et Vigo [48] proposent trois bornes inférieures pour le $2BP|O$. La première borne est une généralisation des bornes qu'ils ont proposées pour le 1D. Cette borne est notée L_1^{MV}. Elle est dominée par une deuxième borne $L_2^{MV} = max(L_2^W, L_2^H)$ qui prend explicitement en considération les deux dimensions des objets. La borne L_2^W considère trois sous-ensembles d'objets suivant leur largeur définis par un paramètre entier $1 \leq p \leq \frac{1}{2}W$:

- $A_G = \{a_i \in A : w_i > W - p\}$
- $A_M = \{a_i \in A : W - p \geq w_i > \frac{1}{2}W\}$
- $A_P = \{a_i \in A : \frac{1}{2}W \geq w_i \geq p\}$

En remarquant que les objets de A_P ne peuvent pas être placés à côté des objets de A_G, la borne inférieure L_2^W s'écrit :

$$L_2^W(p) = L_1^W + max\left\{0, \left\lceil \frac{\sum_{a_i \in A_M \cup A_P} w_i h_i - (HL_1^W - \sum_{a_i \in A_G^w} h_i)W}{WH} \right\rceil \right\} \quad (1.8)$$

où L_1^W est obtenue en appliquant la borne L_{1BP}^{MV} à la sous-instance constituée des objets de $A_G \cup A_M$. Le second terme de la borne est une évaluation du nombre de bins additionnels exigés pour ranger les objets de A_P. La borne L_2^H est calculée d'une façon similaire en permutant H et W.

Dans la borne L_3^{MV}, comme dans L_2^{MV} trois sous-ensembles d'objets sont considérés. Cette fois ci, les deux dimensions des objets entrent en jeu dans la définition des trois sous-ensembles A_G, A_M et A_P. Soit p et q deux entiers tels que $1 \leq p \leq \frac{1}{2}W$ et $1 \leq q \leq \frac{1}{2}H$, les trois sous-ensembles sont définis de la façon suivante :

- $A_G = \{a_i \in A : w_i > W - p \text{ et } h_i > H - q\}$
- $A_M = \{a_i \in A \setminus A_G : w_i > \frac{1}{2}W \text{ et } h_i > \frac{1}{2}H\}$
- $A_P = \{a_i \in A : \frac{1}{2}W \geq w_i \geq p \text{ et } \frac{1}{2}H \geq h_i \geq q\}$

La borne L_3^{MV} vaut :

$$\max_{\substack{1\leq q\leq \frac{1}{2}H \\ 1\leq p\leq \frac{1}{2}W}} \left\{ |A_G \cup A_M| + max \left\{ 0, \left\lceil \frac{|A_P| - \sum_{a_i \in A_M} m(a_i,p,q)}{\lfloor \frac{W}{p} \rfloor \lfloor \frac{H}{q} \rfloor} \right\rceil \right\} \right\} \tag{1.9}$$

où $m(a_i, p, q)$ est une évaluation du nombre d'objets de taille (p, q) qui peuvent être rangés avec a_i :

$$\lfloor \frac{W}{p} \rfloor \lfloor \frac{H - h_i}{q} \rfloor + \lfloor \frac{W - w_i}{p} \rfloor \lfloor \frac{H - \lfloor \frac{H-h_i}{q} \rfloor q}{q} \rfloor \tag{1.10}$$

Fekete et Schepers [29] ont proposé une nouvelle borne inférieure qui consiste à appliquer les DFF qu'ils ont proposées pour le $1BP$ sur chaque dimension d'une instance $2BP$ et appliquer la borne continue sur l'instance résultante.

Boschetti et Mingozzi [5] ont proposé une borne inférieure qu'on note L_{BM}^O. Cette borne est composée de trois bornes différentes $L_{BM}^{O,2}$, $L_{BM}^{O,3}$ et $L_{BM}^{O,4}$ qui seront détaillées dans la section 3.2.1.1.

Plus récemment Carlier *et al.* ont proposé une nouvelle borne inférieure qui domine les bornes de la littérature de $2BP|O$. Elle est basée sur la notion des *DFF discrètes* et des *DFF dépendantes de la donnée*. Ces notions, ainsi que la borne inférieure proposée, seront décrites en détail dans le chapitre 3. En effet, cette borne sera utilisée dans l'évaluation de la nouvelle borne que nous proposons pour le $2BP|R$.

1.8.3 Bornes inférieures pour le $2BP|R$

Notons que la borne continue L_0 est une borne valide pour le $2BP|R$, et que comme dans le cas orienté, elle peut atteindre un quart de l'optimum.

Dell' Amico *et al.* [19] ont proposé une méthode d'évaluation des bornes inférieures pour le $2BP|R$ qui consiste à relaxer le problème en décomposant les objets de l'instance initiale en carrés. La borne inférieure (LB_{DA}^R) consiste à appliquer des bornes dédiées aux instances constituées d'objets carrés proposées par les auteurs.

Boschetti et Mingozzi [6] ont proposé une borne inférieure L_{BM}^R qui est composée de deux bornes. La première consiste à transformer les objets en carrés et appliquer une borne dédiée au $2BP|O$. La deuxième borne prend explicitement en considération la caractéristique "non-orienté".

Ces deux bornes inférieures de la littérature seront détaillées dans le chapitre 3.

1.9 Méthodes exactes

On trouve plusieurs méthodes exactes proposées dans la littérature pour le problème de bin-packing et ses variantes. La résolution exacte des problèmes NP-

difficiles adoptent en général des approches programmation linéaire, programmation dynamique, ou bien des méthodes de recherche par séparation et évaluation [2, 18, 36, 39]. Nous décrivons brièvement les méthodes les plus récentes proposées pour le 2BP.

Martello et Vigo [48] ont proposé une méthode exacte par séparation et évaluation pour le 2BP. Leur méthode commence par calculer une borne supérieure z du nombre de bins nécessaires pour ranger l'ensemble des objets. Ensuite, z bins sont créés. Les objets sont considérés par ordre décroissant de surface. La méthode tente d'affecter à chaque niveau i l'objet a_i en cours à chaque bin ouvert ou éventuellement à un nouveau bin. A chaque affectation de a_i dans un bin B_j, la faisabilité du rangement des objets affectés à B_j. Des heuristiques sont utilisées dans une première étape. Si les heuristiques n'arrivent pas à placer les objets considérés dans B_j, des méthodes exactes dédiées au problème de faisabilité sont utilisées. Un bin est fermé si aucun objet, non encore rangé, ne peut être affecté à ce bin. Une borne inférieure est évaluée pour la sous-instance formée des objets affectés aux bins non fermés ainsi que les objets non affectés.

Les auteurs ont aussi proposé une méthode exacte pour la résolution du problème de faisabilité : tester la possibilité de rangement d'un ensemble donné d'objets dans un bin. Elle consiste à énumérer les possibilités du choix de l'objet à ranger et la position dans laquelle le placer. Les positions considérées sont telles que l'objet ne peut plus être décalé vers le bas ou vers la gauche. Certaines des solutions énumérées sont redondantes, Scheithauer [50] a proposé des méthodes pour éviter quelques répétitions.

Fekete et Schepers [30] ont proposé une méthode exacte pour le problème de faisabilité basée sur le modèle graphe décrit dans la section 1.4.

Clautiaux *et al.* [14] ont proposé une méthode exacte pour le problème de faisabilité. Leur méthode consiste dans une première étape à dénombrer les possibilités de placement des objets en ne considérant que les abscisses. Dans cette étape, les contraintes liées aux ordonnées des objets sont levées. Ensuite, pour chaque solution trouvée dans la première étape, les ordonnées des objets sont déterminées en respectant les choix pris pour les abscisses précédemment.

Les auteurs ont également proposé une méthode exacte pour le $2BP$ [15] qui se résume en une recherche dichotomique de la valeur d'une solution optimale. Cette méthode consiste à calculer plusieurs bipartitions de l'ensemble des objets et à résoudre les deux sous-problèmes résultants.

1.10 Conclusion

L'état de l'art des problèmes de bin-packing est riche. Dans ce chapitre nous avons décrit une partie de ces travaux, surtout les plus récents. Nous remarquons

que peu de travaux portent sur le cas non-orienté, que ce soit au niveau des méthodes de résolution ou bien des évaluations par défaut. Cependant la possibilité de tourner les objets est une spécificité fréquemment rencontrée dans les problèmes réels, qui mérite donc d'être étudiée.

Nouveaux prétraitements

2.1 Introduction

Dans ce chapitre, nous nous intéressons aux procédures de prétraitements des objets ; ceci dans le but de réduire la taille des problèmes et de résoudre de façon optimale certains sous-ensembles d'objets. La réduction de la taille des instances est une technique très avantageuse avant l'application d'algorithmes de résolution coûteux en temps de calcul tels que les méta-heuristiques ou les méthodes exactes. Le prétraitement d'une instance donnée consiste à prendre en considération les caractéristiques de ces objets en les comparant avec celles du bin et des autres objets de l'instance afin de trouver des assemblages optimaux de sous-ensembles d'objets et de valoriser les surfaces perdues dans le bin. Ils permettent de réduire théoriquement les tailles des instances et d'améliorer qualitativement les bornes inférieures.

Plusieurs prétraitements ont été proposés pour le cas orienté dans [48], [6] et [10]. A notre connaissance, aucun prétraitement dédié au $2BP|R$ n'est proposé dans la littérature. Nous introduisons dans ce chapitre un nouveau concept dit *objets à orientation dépendante*. Ces objets trouvent leur possibilité de rotation restreinte s'ils sont placés ensemble dans un même bin. Nous proposons des adaptations des prétraitements proposés dans la littérature pour le cas orienté. Nous proposons ces différents prétraitements sous formes de *Fonctions Identiquement Réalisables*, une notion introduite dans [10]. Nous avons testé les prétraitements proposés sur des jeux d'essai issus de la littérature [48] et [4]. Ces tests prouvent l'efficacité des prétraitements. En effet, un nombre considérable d'objets est éliminé pour plusieurs jeux d'essai. De plus, nous obtenons des améliorations significatives de la borne continue pour plusieurs classes d'instances.

2.2 Notion d'objets à orientation dépendante

Dans la version non-orientée du problème 2BP ($2BP|R$), la rotation de 90° des objets est permise. En fonction des caractéristiques des objets et du bin constituant une instance donnée, la rotation de certains objets peut être restreinte. Trivialement, un objet $a_i \in A$ tel que $L_i > l$ aura une orientation fixe. Dans cette section, nous

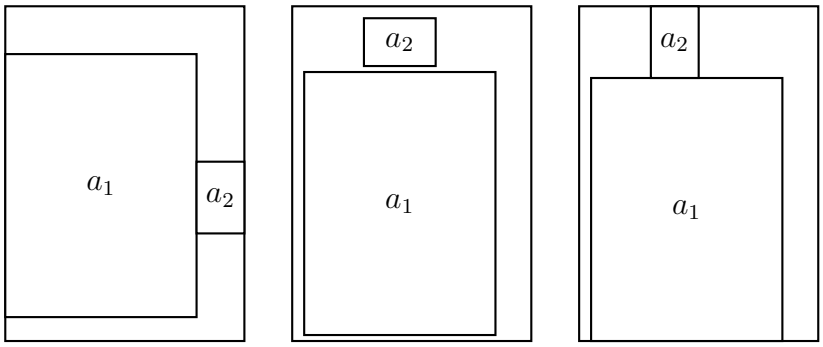


Figure 2.1 – a_1 et a_2 sont à orientation dépendante selon W mais pas selon H

introduisons le concept *d'objets à orientation dépendante*. Les orientations de deux objets à orientation dépendante sont restreintes quand ils sont placés dans un même bin.

Afin d'introduire le concept d'objets à orientation dépendante, nous commençons par définir la notion d'objets à orientation dépendante par rapport à une dimension donnée du bin.

Définition 2.1. *Deux objets $a_i, a_j \in A$ sont dits à orientation dépendante selon W (resp. H)* ssi *a_i et a_j ne peuvent pas être placés côte à côte selon W (resp. H) dans un même bin ou s'il existe une configuration unique dans laquelle les deux objets peuvent être placés côte à côte selon W (resp. H).*

En général, si deux objets sont à orientation dépendante selon une dimension, cela n'implique pas qu'ils soient à orientation dépendante selon l'autre dimension. L'exemple illustré dans la figure 2.1 montre ce fait. Soit une instance $I = (A, B)$ où $B = (10, 14)$, et $a_1 = (8, 11)$, $a_2 = (2, 3)$, $a_3 = (6, 13)$, $a_4 = (3, 4)$ quatre objets dans A. Les deux objets a_1 et a_2 sont à orientation dépendante selon W puisqu'il n'existe qu'une unique configuration dans laquelle les deux objets peuvent être placés côte à côte selon W. Par contre, a_1 et a_2 ne sont pas à orientation dépendante selon H. En effet, il existe deux configurations différentes selon lesquels les deux objets peuvent être placés côte à côte selon H (*cf.* figure 2.1). La figure 2.2 illustre le fait que les objets a_3 et a_4 sont à orientation dépendante selon H mais pas selon W. En effet, a_3 et a_4 ne peuvent pas être placés côte à côte selon H, et il existe deux configurations différentes selon lesquelles les deux objets peuvent être placés côte à côte selon W.

Soit a_i, a_j deux objets dans A, et φ, ψ deux nombres binaires arbitraires. Notons :

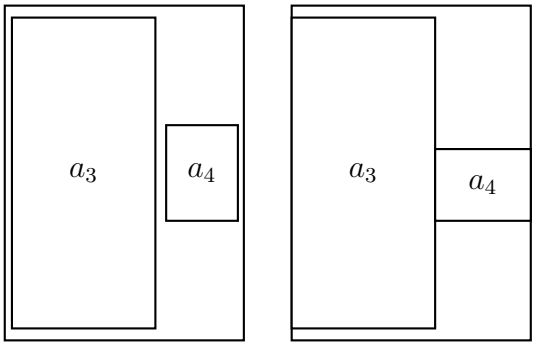


Figure 2.2 – a_3 et a_4 sont à orientation dépendante selon H mais pas selon W

$$M_{(W,H)}(i,j,\varphi,\psi) = (O_i(\varphi) + O_j(\psi) \leq W) \text{ et } (O_i(\overline{\varphi}) \leq H) \text{ et } (O_j(\overline{\psi}) \leq H)$$

et

$$\begin{aligned} N_{(W,H)}(i,j,\varphi,\psi) = &((O_i(\overline{\varphi}) + O_j(\overline{\psi}) > W) \text{ ou } (O_i(\varphi) > H) \text{ ou } (O_j(\psi) > H)) \\ &\text{et } ((O_i(\overline{\varphi}) + O_j(\psi) > W) \text{ ou } (O_i(\varphi) > H)) \\ &\text{et } ((O_i(\varphi) + O_j(\overline{\psi}) > W) \text{ ou } (O_j(\psi) > H)) \end{aligned}$$

Proposition 2.1. *Soit a_i, a_j deux objets dans A. Les objets a_i et a_j sont à orientation dépendante selon W (resp. H)* ssi *: $\forall \varphi, \psi \in \{0,1\}$, si $M_{(W,H)}(i,j,\varphi,\psi)$ (resp. $M_{(H,W)}(i,j,\varphi,\psi)$) est vrai alors $N_{(W,H)}(i,j,\varphi,\psi)$ (resp. $N_{(H,W)}(i,j,\varphi,\psi)$) est vrai.*

Démonstration. Soit a_i et a_j deux objets à orientation dépendante selon W. Soit $\varphi, \psi \in \{0,1\}$ tel que $M_{(W,H)}(i,j,\varphi,\psi)$ est vrai. Cela signifie qu'il existe une configuration $\mathcal{F}_1$ (une orientation donnée de a_i et une orientation de a_j) tel que :

- a_i peut être rangé dans le bin selon cette orientation $(O_i(\overline{\varphi}) \leq H)$
- a_j peut être rangé dans le bin selon cette orientation $(O_j(\overline{\psi}) \leq H)$
- a_i et a_j peuvent être placés côte à côte selon W selon les orientations considérées $(O_i(\varphi) + O_j(\psi) \leq W)$

Soit $\mathcal{F}_2$, $\mathcal{F}_3$ et $\mathcal{F}_4$ les trois autres configurations selon lesquelles les deux objets pourraient être placés côte à côte selon W. $\mathcal{F}_1$ est l'unique configuration réalisable qui permet de placer a_i et a_j côte à côte selon W car les deux objets sont à orientation dépendante selon W. Les configurations $\mathcal{F}_2$, $\mathcal{F}_3$ et $\mathcal{F}_4$ sont donc irréalisables. Cela est équivalent au fait que $N_{(W,H)}(i,j,\varphi,\psi)$ est vrai. Réciproquement, soit a_i et

a_j deux objets qui ne sont pas à orientation dépendante selon W, nous allons montrer que pour une valeur donnée de φ, $\psi \in \{0,1\}$: $M_{(W,H)}(i,j,\varphi,\psi)$ est vrai, et $N_{(W,H)}(i,j,\varphi,\psi)$ est faux. Si a_i et a_j ne sont pas à orientation dépendante selon W cela signifie qu'il existe au moins deux configurations différentes pour ranger les deux objets côte à côte selon W. Autrement dit, il existe une valeur φ, $\psi \in \{0,1\}$ tel que $M_{(W,H)}(i,j,\varphi,\psi)$ est vrai (configuration $\mathcal{F}_1$). $\mathcal{F}_1$ n'est pas unique car au moins une des trois configurations $\mathcal{F}_2$, $\mathcal{F}_3$ ou $\mathcal{F}_4$ est réalisable. Cela est équivalent au fait que $N_{(W,H)}(i,j,\varphi,\psi)$ est faux. □

Ayant défini la notion d'objets à orientation dépendante selon une dimension donnée du bin, on peut définir la notion d'objets à orientation dépendante et de sous-ensemble d'objets à orientation dépendante.

Définition 2.2. *Deux objets a_i, a_j dans A sont dits à orientation dépendante ssi ils vérifient les deux conditions suivantes :*

1. *a_i et a_j sont à orientation dépendante selon les deux dimensions du bin (W et H).*
2. *S'il existe une configuration $\mathcal{F}_W$ selon laquelle a_i et a_j peuvent être placés côte à côte selon W et une configuration $\mathcal{F}_H$ selon laquelle les deux objets peuvent être placés côte à côte selon H, alors $\mathcal{F}_H$ peut être déduite de $\mathcal{F}_W$ par une rotation simultanée des deux objets.*

Un sous-ensemble $Ado \subseteq A$ est dit à orientation dépendante ssi $\forall a_i, a_j \in Ado$, *les objets a_i et a_j sont à orientation dépendante.*

Un exemple de deux objets à orientation dépendante est illustré dans la figure 2.3. Dans cet exemple, nous considérons l'instance suivante, $I = (A, B)$ où $B = (10, 12)$, et $a_1 = (5, 9)$, $a_2 = (4, 8)$, $a_3 = (6, 8)$ trois objets dans A. Les objets a_1 et a_2 sont à orientation dépendante. En effet, il existe deux configurations suivant lesquelles a_1 et a_2 peuvent être placés dans un même bin. Une des deux configurations permet de les placer côte à côte selon W et l'autre permet de les placer côte à côte selon H. Les objets a_1 et a_2 sont alors à orientation dépendante selon W et H. De plus, une configuration peut être déduite de l'autre par une rotation simultanée des deux objets. Considérons maintenant les deux objets a_1 et a_3. Ces deux objets sont aussi à orientation dépendante selon W puisqu'ils ne peuvent pas être placés côte à côte selon cette dimension. Ils le sont aussi selon H car il existe une seule configuration selon laquelle ces deux objets peuvent être placés côte à côte selon H (*cf.* figure 2.3). Les objets a_1 et a_3 sont donc aussi à orientation dépendante.

Remarquons que si deux objets dans A sont à orientation dépendante selon W et selon H, ils ne sont pas nécessairement à orientation dépendante. En effet considérons l'instance suivante. Soit $I = (A, B)$, où $B = (10, 14)$ et $a_1 = (11, 8)$, $a_2 = (2, 9)$ deux objets dans A. Nous remarquons qu'il existe une configuration

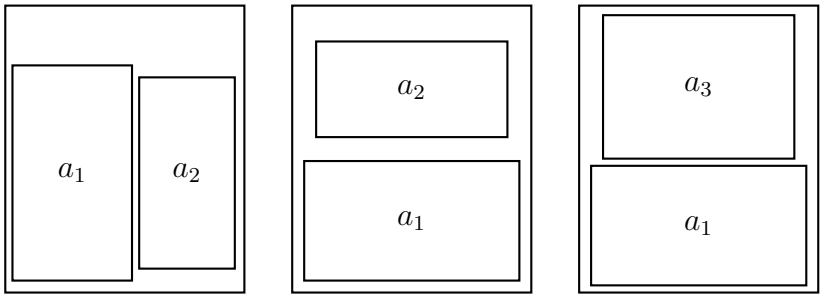


Figure 2.3 – Les sous-ensembles $\{a_1, a_2\}$ et $\{a_1, a_3\}$ sont à orientation dépendante

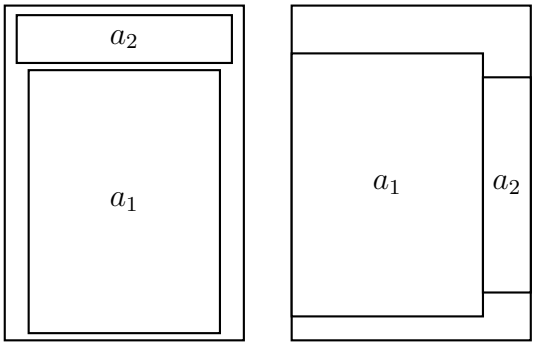


Figure 2.4 – a_1 et a_2 sont à orientation dépendante selon H et W mais ils ne sont pas à orientation dépendante

unique pour placer a_1 et a_2 côte à côte selon H et une configuration unique pour laquelle les objets sont placés côte à côte selon W. Les deux objets sont alors à orientation dépendante selon H et W, mais ils ne sont pas à orientation dépendante puisque les deux configurations possibles ne peuvent pas être déduites l'une de l'autre par une rotation simultanée des deux objets (*cf.* figure 2.4).

En pratique, nous considérons une famille de sous-ensembles d'objets à orientation dépendante paramétrée par un entier. Soit un entier p tel que $1 \leq p < L/2$, nous définissons le sous-ensemble $Ado_p = \{a_i \in A; L_i \geq (L - p)$ *et* $l_i > p\}$.

Proposition 2.2. *Pour une valeur donnée du paramètre p tel que $1 \leq p < L/2$, Ado_p est un sous-ensemble d'objets à orientation dépendante dans A.*

Démonstration. Soit a_i et a_j deux objets de Ado_p, ils vérifient :

$$L_i + L_j \geq L_i + l_j > L \text{ et } l_i + L_j > L \tag{2.1}$$

a_i et a_j sont à orientation dépendante selon W. En effet, deux cas se présentent. Dans le premier cas, les deux objets ne peuvent pas être placés côte à côte selon W, ce qui signifie qu'il n'existe pas φ' et ψ' tel que $M_{(W,H)}(i,j,\varphi',\psi')$ est vrai $(l_i + l_j > W)$. Dans ce cas a_i et a_j sont à orientation dépendante selon W. Dans le deuxième cas on suppose que $\exists(\varphi',\psi')$ tel que $M_{(W,H)}(i,j,\varphi',\psi')$ est vrai. D'après l'équation (2.1), les seules valeurs possibles pour φ' et ψ' pouvant vérifier $M_{(W,H)}(i,j,\varphi',\psi')$ sont telles que :

$$\begin{aligned} M_{(W,H)}(i,j,\varphi',\psi') = {} & (O_i(\varphi') + O_j(\psi') = l_i + l_j \leq W) \\ & \text{et } (O_i(\overline{\varphi'}) = L_i \leq H) \text{ et } (O_j(\overline{\psi'}) = L_j \leq H) \end{aligned}$$

Nous pouvons alors facilement vérifier que $N_{(W,H)}(i,j,\varphi',\psi')$ est vrai :

$$\begin{gathered} (O_i(\overline{\varphi'}) + O_j(\overline{\psi'}) = L_i + L_j > L \geq W) \text{ et } (O_i(\overline{\varphi'}) + O_j(\psi') = L_i + l_j > L \geq W) \\ \text{et } (O_i(\varphi') + O_j(\overline{\psi'}) = l_i + L_j > L \geq W) \end{gathered}$$

Les objets a_i et a_j sont alors à orientation dépendante selon W. Par symétrie, tout couple d'objets de Ado_p est à orientation dépendante selon H.

Supposons maintenant, qu'il existe une configuration $\mathcal{F}_W$ selon laquelle a_i et a_j puissent être placés côte à côte selon W, et une autre configuration $\mathcal{F}_H$ selon laquelle les deux objets puissent être placés côte à côte selon H. D'après l'équation (2.1), la seule possibilité pour placer les objets a_i et a_j côte à côte selon W (resp. H) est de positionner leurs plus petites dimensions l_i et l_j parallèles à W (resp. H) dans la configuration $\mathcal{F}_W$ (resp. $\mathcal{F}_H$). $\mathcal{F}_W$ et $\mathcal{F}_H$ peuvent alors être déduites l'une de l'autre par une rotation simultanée des deux objets. □

2.3 Prétraitements

Les prétraitements, appliqués aux objets d'une instance donnée $I = (A, B)$ permettent d'augmenter la taille de certains objets, et de réduire la taille de l'ensemble A sans modifier le nombre de bins de la solution optimale. De cette façon, les prétraitements permettent d'améliorer les bornes inférieures et de simplifier l'instance. Dans cette section, nous proposons trois prétraitements notés respectivement α, β et γ.

Le premier prétraitement (α) consiste à valoriser les surfaces perdues dans le bin. Il est basé sur une variante du problème de *subset-sum*. Un prétraitement semblable est proposé dans [5] pour le cas orienté. D'autre part, beaucoup de sous-ensembles de l'instance initiale, peuvent être résolus à l'optimum séparément. Ces sous-ensembles contiennent principalement les grands et les petits objets. Les prétraitements β et γ sont basés sur cette idée. Ces prétraitements permettent l'élimination des petits objets qui n'ont aucun impact sur la solution optimale, et l'augmentation de la taille

des grands objets. Deux procédés de réduction basés sur cette idée sont proposés dans [10] pour le cas orienté.

Les trois prétraitements proposés sont basés sur le concept de fonctions identiquement réalisables (*IFF : Identical Feasible Functions*) défini dans [10]. Soit $I = (A, B) = (\{a_1, ..., a_n\}, B)$ une instance 2BP. Une fonction f est dite identiquement réalisable *ssi* l'instance $f(I) = (a'_1, \ldots, a'_n, B)$ obtenue en appliquant f à tous les objets de l'ensemble A est telle que $OPT(I) = OPT(f(I))$.

2.3.1 Prétraitement α

Ce prétraitement consiste à évaluer la surface perdue dans les bins. L'application du prétraitement α à l'instance initiale permet d'augmenter la taille de certains objets. Si un objet donné prend la taille d'un bin (W, H), il peut alors être retiré de l'instance initiale car son packing optimal est alors connu.

L'idée de ce prétraitement est d'évaluer la surface perdue autour de chaque objet $a_j \in A$ placé dans un bin, tout en prenant en compte la possibilité de tourner les objets de 90°. Pour déterminer cette surface condamnée, une variante du problème de *subset-sum* est appliquée sur chaque dimension de l'objet a_j. Etant donné un ensemble d'entiers $\mathcal{E}$ et un entier z, le problème de subset-sum consiste à déterminer le sous-ensemble $\mathcal{E}_z \subset \mathcal{E}$ tel que $\sum_{e_j \in \mathcal{E}_z} e_j = z$.

Pour chaque objet a_j, une valeur notée $S(A, w_j, W)$ est évaluée. Cette valeur représente la largeur maximale inférieure ou égale à W qui peut être atteinte en plaçant un ensemble d'objets les uns à côté des autres, selon la prolongation de w_j (l'objet a_j est supposé orienté tel que l'arête de taille w_j soit parallèle à la largeur du bin de taille W). L'évaluation de la fonction S tient compte de toutes les permutations possibles des orientations des objets $\{a_i \in A - \{a_j\}\}$. Nous posons $S(A, w_j, W) = -\infty$ si $w_j > W$ ou $h_j > H$, autrement :

$$S(A, w_j, W) = w_j + Max \sum_{a_i \in A - \{a_j\}} (\varphi w_i + \psi h_i) \tag{2.2}$$

où :

$$\sum_{a_i \in A - \{a_j\}} (\varphi w_i + \psi h_i) \leq (W - w_j) \tag{2.3}$$

$$\varphi + \psi \leq 1 \tag{2.4}$$

$$\psi w_i \leq H \tag{2.5}$$

$$\varphi h_i \leq H \tag{2.6}$$

$$\varphi \in \{0, 1\} \text{ et } \psi \in \{0, 1\} \tag{2.7}$$

L'équation (2.3) signifie que la somme des dimensions des objets constituant le sous-ensemble entrant en jeu dans le calcul de $S(A, w_j, W)$ ne dépasse pas la largeur

résiduelle $(W - w_j)$ disponible. L'équation (2.4) garantit que le même objet n'est pas considéré deux fois dans la même solution. Les équations (2.5), (2.6) et (2.7) vérifient que l'orientation considérée des objets sélectionnés est réalisable.

De même, $S(A, w_j, H)$ représente la hauteur maximale inférieure ou égale à H, qui peut être atteinte en plaçant un ensemble d'objets les uns au dessus des autres, selon la prolongation de w_j. L'orientation de a_j dans ce cas est telle que w_j est parallèle à H. Notons qu'au moins une des deux valeurs $S(A, w_j, W)$ ou $S(A, w_j, H)$ est finie, le cas contraire signifierait que l'objet a_j ne peut pas être placé dans un bin.

L'évaluation de $S(A, w_j, W)$ est donnée dans l'algorithme 1. Les données de cet algorithme sont l'ensemble A des objets de l'instance considérée, la dimension w_j de l'objet a_j à traiter et la largeur W du bin. Nous introduisons la liste A' d'objets tel que $A' = A \setminus \{a_j\} = \{a'_1, ..., a'_{n'}\}$ où $n' = n - 1$. La matrice M est de taille $[n'][W - w_j + 1]$ (n' lignes et $W - w_j + 1$ colonnes). Chaque ligne de la matrice correspond à un objet du sous-ensemble A' candidat à être placé selon le prolongement de w_j. Les éléments de M sont initialisés à 0.

Algorithme 1 : Evaluation de $S(A, w_j, W)$

$S := -\infty$; $disp := W - w_j$;
si $((w_1 \leq disp)$ *et* $(h_1 \leq H))$ **alors**
  $M[0][w_1] := 1$; $S := \mathbf{Max}(S, w_1)$;
si $((h_1 <= disp)$ *et* $(w_1 \leq H))$ **alors**
  $M[0][h_1] := 1$; $S := \mathbf{Max}(S, h_1)$;
pour $k := 1$ *à* $n' - 1$ **faire**
  pour $l := 0$ *à* $disp$ **faire**
    si $M[k-1][l] = 1$ **alors**
      $M[k][l] := 1$;
      si $(((l + w_{k+1}) <= disp)$ *et* $(h_{k+1} \leq H))$ **alors**
        $M[k][l + w_{k+1}] := 1$; $S := \mathbf{Max}(S, l + w_{k+1})$;
      si $(((l + h_{k+1}) <= disp)$ *et* $(w_{k+1} \leq H))$ **alors**
        $M[k][l + h_{k+1}] := 1$; $S := \mathbf{Max}(S, l + h_{k+1})$;
retourner $S + w_j$;

Proposition 2.3. *Soit un objet $a_j \in A$, la fonction α^j_w définie ci dessous appliquée à l'ensemble A des objets constituant l'instance initiale à traiter est une IFF.*

$$\alpha^j_w : A \to A'$$

$$a_i \mapsto a'_i \text{ où } \begin{cases} h'_i = h_i \\ w'_i = \begin{cases} w_i + \min(W - S(A, w_i, W), H - S(A, w_i, H)) & si\ i = j \\ w_i \quad sinon \end{cases} \end{cases}$$

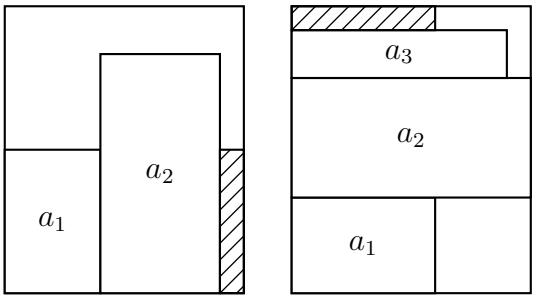


Figure 2.5 – Illustration de la surface perdue dans le bin selon la prolongation de la largeur de l'objet a_1

Démonstration. Supposons qu'une configuration optimale est connue pour l'instance transformée I', il est évident que la même configuration optimale peut être construite pour les objets de l'instance initiale puisque les surfaces ajoutées aux objets sont de toute façon perdues. Il suffit donc de placer les objets dans les mêmes positions. Réciproquement, si une configuration optimale est connue pour I alors une configuration optimale de même coût peut être construite pour I' : il suffit de placer tous les objets dans les mêmes positions. Cela est réalisable puisque les espaces des objets transformés ne sont pas utilisés quel que soit le rangement considéré. □

Nous proposons ci-dessous un exemple d'application du prétraitement α sur une dimension d'un objet d'une instance donnée. Soit l'instance suivante :

$$I = (A, B) : A = \{a_1 = (4, 6), a_2 = (5, 10), a_3 = (2, 9), a_4 = (9, 11)\},\ B = (10, 12)$$

Considérons α^1_w. La figure 2.5 illustre les valeurs $S(A, w_1, W)$ et $S(A, w_1, H)$ obtenues pour cet exemple. Nous obtenons $W - S(A, w_1, W) = 1$ et $H - S(A, w_1, H) = 1$, L'instance transformée après l'application de la fonction α^1_w est alors la suivante :

$$I' = (A', B); A' = \{a'_1 = (5, 6), a'_2 = a_2, a'_3 = a_3, a'_4 = a_4\}$$

Une IFF semblable α^j_h, est appliquée à la dimension h_j de l'objet a_j. Le prétraitement α consiste à appliquer les fonctions α^j_w et α^j_h pour tout $a_j \in A$. Pour tout objet $a_j \in A$, l'évaluation de $S(A, w_j, W)$ est en $O(W)$. Le prétraitement appliqué à tous les objets est donc en $O(nW)$. L'ordre dans lequel les tailles des objets sont majorées est important car si une dimension w_i d'un objet a_i est employée dans la mise à jour d'une dimension d'un autre objet a_j, w_i ne peut plus être mis à jour à son tour. En effet, $\min(W - S(A, w_i, W), H - S(A, w_i, H))$ sera égale à 0. Afin de

maximiser le nombre d'objets qui pourraient être transformés par ce prétraitement, une règle heuristique est employée pour ordonner les objets avant de les traiter. Les objets sont triés selon les largeurs (resp. hauteurs) décroissantes avant d'appliquer α_w^j (resp. α_h^j), $j = 1, ..., n$.

2.3.2 Prétraitement β

Le prétraitement β se situe dans le cadre des procédures de réduction des instances de $2BP|R$. Dans ce prétraitement, les petits objets qui, en général, n'ont pas un grand impact sur la valeur optimale de la solution sont éliminés. La taille d'autres objets est par contre augmentée, ce qui augmente le nombre d'objets contraignants (les grands objets). Ceux ci ont en effet, généralement, un rôle plus important sur la valeur optimale de la solution des instances considérées.

Ce prétraitement est basé sur la présence d'objets à orientation dépendante dans l'instance initiale. Pour cela nous considérons la famille des sous-ensembles d'objets à orientation dépendante Ado_p introduite dans la section 2.2. Pour chaque sous-ensemble d'objets à orientation dépendante paramétré par p, nous considérons le sous-ensemble A_p d'objets qui ont une dimension suffisamment petite pour qu'ils puissent être placés dans la prolongation de la plus grande dimension d'un des objets de Ado_p. L'idée est de chercher à éliminer les objets de A_p et de mettre à jour les dimensions des objets de Ado_p.

Nous rappelons que $Ado_p = \{a_i \in A; L_i \geq (L - p)\ et\ l_i > p\}$ où p est un entier positif, tel que $1 \leq p < L/2$. Le sous-ensemble A_p s'écrit par conséquent : $A_p = \{a_i \in A; l_i \leq p\}$. Remarquons que $Ado_p \cap A_p = \emptyset$. A_p est l'ensemble des objets dans A qui ont une dimension suffisamment petite pour être placés dans la prolongation de la plus grande dimension d'un des objets de Ado_p.

Pour chaque objet a_i dans Ado_p, placé dans un bin, nous considérons la surface vide résultante selon sa plus grande dimension L_i. Si les deux orientations possibles de l'objet dans le bin sont réalisables, un bin correspondant à la plus petite des deux surfaces résultantes est considéré. C'est le cas de la surface S_2 de l'exemple illustré dans la figure 2.6. Dans le cas d'un objet à orientation fixe ($L_i > l$), un bin correspondant à la surface vide obtenue selon sa plus grande dimension L_i est considéré :

$$B_i = \begin{cases} (l - L_i, l_i) \text{ si } l \geq L_i \\ (L - L_i, l_i) \text{ sinon} \end{cases} \tag{2.8}$$

Soit $\mathcal{P}$ le problème de décision suivant : est-il possible de placer tous les objets de A_p dans les surfaces considérées selon les prolongations des plus grandes dimensions des objets de Ado_p dans les bins B_i ?

D'où la proposition suivante :

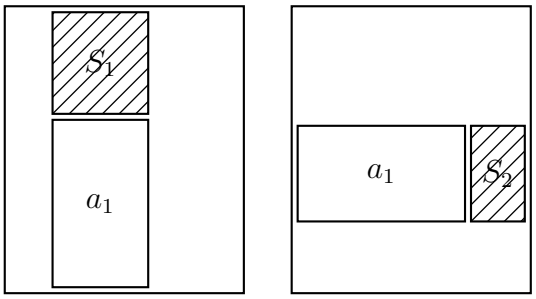


Figure 2.6 – Illustration de la surface vide considérée selon la prolongation de la plus grande dimension de l'objet a_1

Proposition 2.4. *Si $\mathcal{P}$ a au moins une solution, la fonction β^p définie ci dessous est alors une IFF associée à l'instance $I = (A, B)$.*

$$\beta^p : A \to A'$$

$$a_i \mapsto a'_i \text{ où } \begin{cases} l'_i = l_i \text{ et } L'_i = l \text{ si } (a_i \in Ado_p \text{ et } L_i \leq l) \\ l'_i = l_i \text{ et } L'_i = L \text{ si } (a_i \in Ado_p \text{ et } L_i > l) \\ l'_i = 0 \text{ et } L'_i = 0 \text{ si } a_i \in A_p \\ l'_i = l_i \text{ et } L'_i = L_i \text{ sinon} \end{cases}$$

Démonstration. Supposons qu'une configuration optimale est connue pour l'instance transformée I'. Une configuration optimale pour l'instance initiale I peut être construite en plaçant les objets de $A \setminus A_p$ dans les mêmes positions. Puisqu'aucun objet de $A \setminus A_p$ ne peut être placé dans les surfaces considérées suivant le prolongement de la plus grande dimension des objets de Aod_p, ces surfaces sont disponibles. Les objets de A_p sont alors placés dans ces surfaces (cela est réalisable puisque le problème de décision a une solution). Réciproquement, si une configuration optimale est connue pour I, une configuration optimale du même coût peut être construite pour I'. Il s'agit de placer les objets dans les mêmes positions en décalant si nécessaire les objets dans le bin. □

Nous proposons ci après un exemple d'application de la fonction β^p à une instance donnée.

Soit l'instance suivante, $I = (A, B)$: $A = \{a_1 = (4, 8), a_2 = (2, 2), a_3 = (6, 7), a_4 = (3, 4), a_5 = (7, 4), a_6 = (1, 6), a_7 = (2, 5)\}$ et $B = (10, 10)$. pour $p = 3$, $Ado_p = \{a_1, a_3, a_5\}$, les bins considérés pour le problème de décision $\mathcal{P}$ sont :

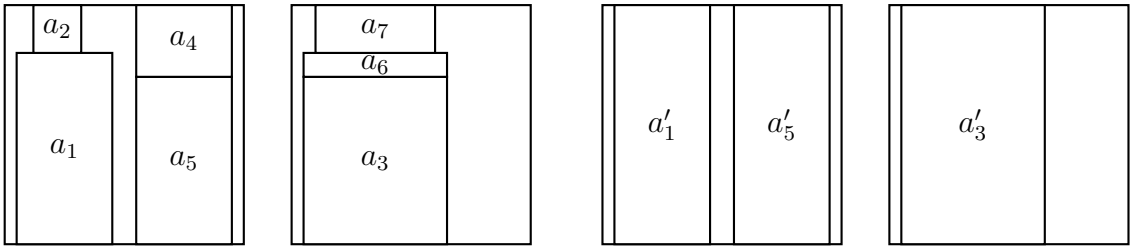


Figure 2.7 – Illustration de la surface réclamée dans le bin selon la prolongation de la largeur de l'objet a_1

$\{(2,4),(3,4),(3,6)\}$. $A_p = \{a_2, a_4, a_6, a_7\}$. La figure 2.7 montre un placement possible des objets de A_p dans les bins considérés. La réponse au problème $\mathcal{P}$ est alors affirmative. L'instance est alors transformée comme suit : $A' = \{a'_1 = (4,10), a'_3 = (6,10), a'_5 = (10,4)\}$, il en résulte alors l'élimination des objets de A_p.

Le problème de décision $\mathcal{P}$ qui consiste à vérifier la faisabilité de placement des objets de A_p dans les bins B_i considérés est une variante du problème de bin-packing, lui aussi NP-difficile. En pratique, une heuristique de résolution est utilisée pour résoudre ce problème.

La même procédure est appliquée à l'instance I pour toutes les valeurs possibles de l'entier p. En ne considérant que les valeurs intéressantes du paramètre p, celles qui correspondent aux dimensions des objets, ce prétraitement est en $O(n * \Gamma(n))$ où $\Gamma(n)$ est la complexité de l'algorithme de résolution du problème de décision.

2.3.3 Prétraitement γ

Comme le prétraitement β, ce prétraitement prend en considération la présence de sous-ensembles d'objets particuliers dans l'instance. Cette fois, les sous-ensembles considérés sont composés des grands et petits objets.

Soit p un entier tel que $1 \leq p \leq \frac{1}{2}L$. Nous définissons le sous-ensemble suivant $A_{large} = A^1_{large} \cup A^2_{large}$, où $A^1_{large} = \{a_i \in A : (L_i \leq l) \text{ et } (l_i > L - p)\}$ et $A^2_{large} = \{a_i \in A : (L_i > l) \text{ et } (l_i > l - p) \text{ et } (L_i > L - p)\}$. Nous posons $A_{small} = \{a_i \in A : l_i < p\}$.

A_{large} représente l'ensemble des "grands" objets dans A : deux objets a_i et a_j dans A_{large} ne peuvent pas être placés dans un même bin quelles que soient les orientations considérées. Le sous-ensemble A_{small} contient tous les objets qui peuvent tenir avec au moins un objet de A_{large} dans un même bin. Si tous les objets de A_{small} peuvent être placés dans les espaces libres disponibles autour des objets de A_{large}, ils

peuvent alors être éliminés puisqu'ils n'auront aucun impact sur la solution optimale, et chaque objet de A_{large} prend la taille d'un bin (W, H).

Soit $A' = A_{large} \bigcup A_{small}$ et I' l'instance de $2BP|R$ suivante : $I' = (A', B)$. Si $OPT(I') = |A_{large}|$, la proposition suivante est alors démontrée d'une façon similaire à la proposition 2.4.

Proposition 2.5. *Si* $(OPT(I') = |A_{large}|)$ *alors, la fonction* γ^p *suivante est une* IFF *associée à l'instance* I*.*

$$\gamma^p : A \rightarrow A'$$

$$a_i \mapsto a'_i \text{ où } \begin{cases} w'_i = l \ et \ h'_i = L \ si \ a_i \in A_{large} \\ w'_i = 0 \ et \ h'_i = 0 \ si \ a_i \in A_{small} \\ w'_i = w_i \ et \ h'_i = h_i \ sinon \end{cases}$$

En limitant les valeurs de p à celles qui correspondent aux dimensions des objets, ce prétraitement est en $O(n * \Gamma(n))$ où $\Gamma(n)$ est la complexité de l'algorithme de résolution de l'instance I'.

2.3.4 Comparaison des trois prétraitements

Les trois prétraitements α, β et γ proposés sont complémentaires. Nous donnons trois exemples qui illustrent cette complémentarité.

Dans le premier exemple, nous considérons l'instance $2BP|R$ suivante. $I = (A, B)$ avec $A = \{a_1 = (3, 5), a_2 = (2, 3), a_3 = (6, 5)\}$ et $B = (10, 10)$. Le prétraitement γ ne peut pas être appliqué aux objets de cette instance à cause de l'absence de grands objets (comme ils sont définis dans la section 2.3.3). Nous pouvons essayer d'appliquer le prétraitement β pour p = 4 : $Aod_4 = \{a_3\}$, $A_p = \{a_1, a_2\}$ mais a_1 et a_2 ne tiennent pas dans l'espace disponible au dessus de a_3, ce qui ne permet pas d'appliquer le prétraitement β. Par contre, L'application du prétraitement α à la dimension w_3 de l'objet a_3 permet de le transformer en $a'_3 = (7, 5)$. En effet $S(A, w_3, W) = S(A, w_3, H) = 9$ (*cf.* Section 2.3.1).

Nous donnons un autre exemple où seul le prétraitement β peut être appliqué. Soit $I = (A, B)$ tel que $A = \{a_1 = (4, 8), a_2 = (2, 2), a_3 = (6, 7), a_4 = (3, 4), a_5 = (7, 4)\}$ et $B = (10, 10)$. Le prétraitement α ne peut être appliqué à aucun objet de l'instance considérée. En effet, pour tout objet a_i de A, l'évaluation de la fonction $S(A, w_i, D)$ ou de $S(A, h_i, D)$ donne comme valeur D $(D = W = H)$. Le prétraitement γ peut être envisagé (pour $p = 5$) mais les objets du sous-ensemble correspondant $A_{small} = \{a_1, a_2, a_4, a_5\}$ ne peuvent pas être tous placés dans la surface disponible autour de a_3, le seul élément de A_{large}. Par contre le prétraitement β peut

être appliqué à I pour $p = 3$, $Aod_3 = \{a_1, a_3, a_5\}$, Les bins correspondants pour le problème de décision à considérer sont de dimensions $\{(2,4),(3,6),(3,4)\}$. Une heuristique simple permet de vérifier que les objets a_2 et a_4 de A_p peuvent être placés dans les bins considérés. Par conséquent, le problème de décision est réalisable et l'instance initiale peut alors être transformée en $A' = \{a'_1 = (4,10), a'_3 = (6,10), a'_5 = (10,4)\}$.

Considérons maintenant l'instance suivante : $I = (A, B)$ tel que $A = \{a_1 = (7,8), a_2 = (9,2), a_3 = (1,3)\}$ et $B = (10,10)$. Comme dans le cas de l'exemple précédent, le prétraitement α ne peut être appliqué à aucun objet de l'instance considérée. En effet, pour tout objet a_i de A, l'évaluation de la fonction $S(A, w_i, D)$ comme $S(A, h_i, D)$ donne comme valeur D. En essayant le prétraitement β, pour $p = 1$, $Aod_1 = \{a_2\}$. Mais a_3, l'unique objet de A_1, ne peut pas être placé sans chevauchement dans le bin $(2,1)$ considéré au dessus de a_2. Pour les autres valeurs possibles de p (2, 3 et 4) on a $Aod_2 = Aod_3 = Aod_4 = \{a_1\}$ et $A_2 = A_3 = A_4 = \{a_2, a_3\}$. Mais les objets a_2 et a_3 ne tiennent pas dans le bin (2,7) considéré au dessus de l'objet a_1. Le prétraitement β ne peut donc pas être appliqué à cette instance. Il reste à tester le prétraitement γ. Pour $p = 4$, $A_{large} = \{a_1\}$, $A_{small} = \{a_2, a_3\}$. Une simple heuristique réussit à placer a_2 et a_3 dans l'espace disponible autour de a_1. Nous pouvons alors éliminer les objets a_2 et a_3 et mettre à jour la taille de a_4. L'objet prend désormais la taille d'un bin $(10,10)$.

Ces trois exemples que nous venons de présenter montrent que les trois prétraitements α, β et γ sont complémentaires. Il sera intéressant d'appliquer d'une façon itérative les trois prétraitements aux différentes instances du problème. L'efficacité de cette méthode est montrée dans la section 2.4 (résultats expérimentaux). En effet, l'application itérative des trois prétraitements domine l'unique application de l'un d'entre eux.

2.4 Résultats expérimentaux

Dans cette section nous présentons les résultats numériques des prétraitements (α, β et γ). Nous avons testé nos algorithmes sur des benchmarks proposés dans la littérature ([4] et [48]). Ces benchmarks se décomposent en dix classes d'instances, générées aléatoirement. Les six premières classes ont été proposées dans [4] tandis que les quatres dernières ont été proposées dans [48]. Chaque classe est composée de cinq groupes qui diffèrent par le nombre d'objets (n = 20, 40, 60, 80, 100). Chaque groupe contient dix instances différentes. Le benchmark contient au total 500 instances différentes.

Les caractéristiques des instances qui constituent les six classes de [4] sont les

Problème			% objets restants				surface totale/surface d'un bin					temps CPU			
Classe	HxW	n	α	β	γ	*all*	*init*	α	β	γ	*all*	α	β	γ	*all*
I	10x10	20	98.50	96.50	73.50	72.50	5.79	5.79	5.82	5.96	6.06	0.00	0.00	0.00	0.01
		40	98.50	97.00	72.50	70.75	11.60	11.60	11.61	12.03	12.17	0.01	0.01	0.02	0.02
		60	99.00	97.50	71.50	71.50	17.93	17.93	17.95	18.42	18.83	0.01	0.02	0.05	0.08
		80	99.00	99.00	74.25	74.25	24.81	24.81	24.81	25.31	25.81	0.02	0.05	0.14	0.20
		100	99.00	99.00	86.30	86.30	29.98	29.98	29.98	30.25	30.46	0.03	0.07	0.30	0.39
II	30x30	20	100.00	100.00	100.00	100.00	0.64	0.64	0.64	0.64	0.64	0.00	0.00	0.00	0.00
		40	100.00	100.00	100.00	100.00	1.29	1.29	1.29	1.29	1.29	0.01	0.00	0.00	0.01
		60	100.00	100.00	100.00	100.00	1.99	1.99	1.99	1.99	1.99	0.03	0.00	0.00	0.03
		80	100.00	100.00	100.00	100.00	2.76	2.76	2.76	2.76	2.76	0.05	0.00	0.00	0.05
		100	100.00	100.00	100.00	100.00	3.33	3.33	3.33	3.33	3.33	0.08	0.00	0.00	0.07
III	40x40	20	100.00	95.50	83.00	80.50	3.81	3.81	3.83	3.96	4.11	0.00	0.01	0.01	0.02
		40	100.00	100.00	76.25	76.25	7.71	7.71	7.71	8.10	8.37	0.01	0.05	0.05	0.10
		60	100.00	100.00	91.00	91.00	12.03	12.03	12.03	12.19	12.27	0.02	0.15	0.26	0.42
		80	100.00	97.50	90.25	87.62	16.67	16.67	16.69	16.82	16.89	0.04	0.30	0.62	0.85
		100	100.00	100.00	81.70	81.70	20.08	20.08	20.08	20.66	20.89	0.06	0.63	0.97	1.53
IV	100x100	20	100.00	100.00	100.00	100.00	0.61	0.61	0.61	0.61	0.61	0.01	0.00	0.00	0.01
		40	100.00	100.00	100.00	100.00	1.23	1.23	1.23	1.23	1.23	0.03	0.00	0.00	0.03
		60	100.00	100.00	100.00	100.00	1.92	1.92	1.92	1.92	1.92	0.06	0.00	0.00	0.06
		80	100.00	100.00	100.00	100.00	2.67	2.67	2.67	2.67	2.67	0.12	0.00	0.00	0.11
		100	100.00	100.00	100.00	100.00	3.21	3.21	3.21	3.21	3.21	0.18	0.00	0.00	0.18
V	100x100	20	100.00	92.50	66.50	61.50	4.78	4.82	4.83	5.14	5.36	0.00	0.01	0.01	0.02
		40	100.00	97.25	56.25	55.50	9.69	9.70	9.74	10.56	10.78	0.02	0.06	0.07	0.14
		60	99.83	97.67	57.67	57.17	15.14	15.16	15.19	16.15	16.47	0.03	0.25	0.47	0.66
		80	99.88	98.38	56.00	55.75	20.97	20.98	20.99	21.94	22.51	0.06	0.68	1.03	1.49
		100	100.00	97.40	76.00	74.80	25.29	25.30	25.33	26.16	26.39	0.09	1.14	2.81	3.81
VI	300x300	20	100.00	100.00	100.00	100.00	0.53	0.53	0.53	0.53	0.53	0.02	0.00	0.00	0.02
		40	100.00	100.00	100.00	100.00	1.08	1.08	1.08	1.08	1.08	0.08	0.00	0.00	0.08
		60	100.00	100.00	100.00	100.00	1.68	1.68	1.68	1.68	1.68	0.17	0.00	0.00	0.17
		80	100.00	100.00	100.00	100.00	2.33	2.33	2.33	2.33	2.33	0.32	0.00	0.00	0.32
		100	100.00	100.00	100.00	100.00	2.81	2.81	2.81	2.81	2.81	0.52	0.00	0.00	0.52
VII	100x100	20	100.00	99.00	98.50	98.50	4.24	4.27	4.26	4.25	4.29	0.00	0.00	0.00	0.01
		40	99.75	100.00	98.00	98.00	9.14	9.15	9.15	9.15	9.18	0.01	0.02	0.03	0.06
		60	100.00	99.50	100.00	99.50	13.41	13.42	13.41	13.41	13.42	0.03	0.05	0.06	0.15
		80	100.00	99.50	99.75	99.38	19.35	19.35	19.36	19.35	19.36	0.06	0.18	0.25	0.48
		100	100.00	99.90	100.00	99.90	23.34	23.34	23.35	23.34	23.35	0.08	0.37	0.49	0.92
VIII	100x100	20	100.00	100.00	100.00	100.00	4.34	4.36	4.35	4.34	4.36	0.01	0.01	0.00	0.01
		40	100.00	100.00	100.00	100.00	9.21	9.22	9.21	9.21	9.22	0.02	0.02	0.03	0.06
		60	100.00	99.83	100.00	99.83	13.60	13.61	13.61	13.60	13.61	0.03	0.06	0.07	0.15
		80	100.00	99.50	100.00	99.50	19.09	19.09	19.09	19.09	19.10	0.06	0.17	0.26	0.48
		100	100.00	100.00	100.00	100.00	23.62	23.62	23.62	23.62	23.62	0.09	0.37	0.46	0.86
IX	100x100	20	97.00	94.50	3.00	0.00	8.91	9.73	9.44	14.03	14.30	0.00	0.01	0.00	0.00
		40	98.75	99.00	7.00	5.00	17.57	17.99	17.68	26.88	27.40	0.01	0.03	0.00	0.02
		60	99.33	97.00	3.83	2.50	27.18	27.63	27.84	42.99	43.45	0.02	0.07	0.01	0.05
		80	99.75	97.75	3.62	3.50	36.52	36.66	36.93	56.77	57.10	0.03	0.22	0.00	0.12
		100	99.90	97.30	2.90	2.60	44.53	44.61	45.16	68.59	69.10	0.06	0.46	0.01	0.26
X	100x100	20	100.00	90.50	68.00	65.50	3.27	3.28	3.32	3.46	3.56	0.01	0.00	0.01	0.01
		40	100.00	97.00	89.25	86.75	6.32	6.32	6.33	6.39	6.42	0.02	0.04	0.05	0.10
		60	100.00	95.67	98.83	95.50	8.99	8.99	9.02	8.99	9.02	0.04	0.09	0.19	0.30
		80	100.00	99.62	99.50	99.12	11.72	11.72	11.72	11.72	11.72	0.08	0.30	0.47	0.83
		100	100.00	99.20	100.00	99.20	14.70	14.70	14.71	14.70	14.71	0.12	0.64	0.96	1.63

Tableau 2.1 – Prétraitements

suivantes :

- Classe 1 : w_i et h_i générés aléatoirement suivant une loi uniforme dans [1, 10], $W = H = 10$.
- Classe 2 : w_i et h_i générés aléatoirement suivant une loi uniforme dans [1, 10], $W = H = 30$.
- Classe 3 : w_i et h_i générés aléatoirement suivant une loi uniforme dans [1, 35], $W = H = 40$.
- Classe 4 : w_i et h_i générés aléatoirement suivant une loi uniforme dans [1, 35], $W = H = 100$.
- Classe 5 : w_i et h_i générés aléatoirement suivant une loi uniforme dans [1, 100], $W = H = 100$.
- Classe 6 : w_i et h_i générés aléatoirement suivant une loi uniforme dans [1, 100], $W = H = 300$.

Les problèmes proposés dans [48] reflètent mieux la réalité puisque les instances d'une même classe ne sont pas générées du même interval comme c'est le cas des six premières classes décrites ci dessus. En effet, les auteurs considèrent quatre types différents d'objets :

- Type 1 : w_j et h_j générés aléatoirement suivant une loi uniforme dans $[2/3W, W]$ et $[1, 1/2H]$ respectivement.
- Type 2 : w_j et h_j générés aléatoirement suivant une loi uniforme dans $[1, 1/2W]$ et $[2/3H, H]$ respectivement.
- Type 3 : w_j et h_j générés aléatoirement suivant une loi uniforme dans $[1/2W, W]$ et $[1/2H, H]$ respectivement.
- Type 4 : w_j et h_j générés aléatoirement suivant une loi uniforme dans $[1, 1/2W]$ et $[1, 1/2H]$ respectivement.

Les objets dans les instances des classes 7, 8, 9 et 10 sont générées de la façon suivante :

- Classe 7 : type 1 avec une probabilité de 70%, types 2, 3 et 4 avec une probabilité de 10% pour chaque type.
- Classe 8 : type 2 avec une probabilité de 70%, types 1, 3 et 4 avec une probabilité de 10% pour chaque type.
- Classe 9 : type 3 avec une probabilité de 70%, types 1, 2 et 4 avec une probabilité de 10% pour chaque type.
- Classe 10 : type 4 avec une probabilité de 70%, types 1, 2 et 3 avec une probabilité de 10% pour chaque type.

La taille des bins pour ces quatre dernières classes est égale à (100,100). Ces jeux d'essai (les dix classes) sont disponibles sur le Web à l'adresse suivante :
http ://www.or.deis.unibo.it/research_pages/ORinstances/ORinstances.htm.

Le tableau 2.1 montre les résultats des trois prétraitements proposés. Les trois premières colonnes du tableau présentent les attributs des groupes d'instances constituant le benchmark : la classe, les dimensions du bin utilisé pour placer les objets et le nombre d'objets par instance. Chaque ligne représente la moyenne des résultats sur les 10 instances qui constituent un groupe dans une classe. Deux résultats sont présentés : le premier est le pourcentage d'objets de taille différente de (W, H) restant dans l'instance. Le deuxième résultat est le rapport de la surface totale des objets restant après prétraitement sur la surface d'un bin (borne inférieure continue).

La partie réservée pour le pourcentage des objets restants est constituée de quatre colonnes. Les trois premières représentent les résultats obtenus, en appliquant respectivement les prétraitements α, β et γ séparément. La dernière colonne (intitulée *all*) représente les résultats obtenus par l'application itérative de α, β et γ : les trois prétraitements sont appliqués l'un après l'autre tant que l'instance est modifiée. La colonne *init* représente la borne inférieure continue initiale (sans prétraitements).

Le prétraitement γ fournit les meilleurs résultats en moyenne aussi bien en terme de rapport de surfaces qu'en terme de pourcentage d'objets restants. Cependant, pour les classes 7 et 8, par exemple, c'est le prétraitement β qui fournit en moyenne les meilleurs résultats. En effet, cette classe est constituée principalement d'objets "longs et assez larges", ce qui favorise le prétraitement β. Dans la plupart des cas, le prétraitement α est moins efficace pour éliminer des objets ou pour augmenter la somme des surfaces et par conséquence la borne continue. Mais, l'application itérative des trois prétraitements (colonne *all*) donne des résultats strictement meilleurs qu'un des trois prétraitements appliqué séparément. En effet, en augmentant par exemple la taille de certains objets, on parvient ainsi à mettre en évidence des assemblages qui n'apparaissaient pas auparavant, les trois prétraitements étant indépendants.

2.5 Conclusion

Nous avons introduit la notion d'objets à orientation dépendante. Cette notion permet de prendre en considération des contraintes de réduction des libertés de rotation de certains objets d'une instance donnée. Nous avons également proposé trois différents prétraitements pour le $2BP|R$. Les résultats numériques montrent l'intérêt de ces prétraitements, en terme de simplification de l'instance, et l'augmentation de la somme de la surface des objets de l'instance et, par conséquent, la borne continue. Ces travaux ont été soumis pour une publication internationale [22] et présentés en

congrès [23].

Bornes inférieures

3.1 Introduction

Plusieurs bornes inférieures ont été proposées pour le problème de bin-packing. La plupart de ces travaux ont été dédiés à la version $2BP|O$: [5], [48], [29], [9] et [10]. Deux bornes inférieures ont été proposées pour le $2BP|R$ dans [19] et [5].

Dans ce chapitre, nous proposons un nouveau schéma général pour l'évaluation des bornes inférieures dans le cas de bins carrés. Ce schéma consiste en la construction d'une instance $2BP|O$ avec $2n$ objets à partir d'une instance de $2BP|R$ de taille n. Chaque objet apparaît deux fois dans l'instance (dans ses deux orientations possibles). Nous montrons que si l'instance initiale a besoin de z bins, l'instance de taille $2n$ aura besoin de $2z$ bins au plus, pour placer l'ensemble des objets de l'instance. Nous pouvons alors appliquer des bornes dédiées au $2BP|O$ à l'instance de taille $2n$. Nous montrons que dans le cas de bins carrés, les bornes proposées dans [10], utilisées dans le schéma général d'évaluation, dominent celles de la littérature. Ces résultats sont confirmés par une étude expérimentale. Dans le cas de bins non carrés, un prétraitement de l'instance est proposé, il permet de se rapporter au cas des bins carrés. Les résultats de dominance ne s'appliquent néanmoins plus.

Ce chapitre est divisé en six sections. La section 3.2 est consacrée aux différentes méthodes d'évaluation par défaut de la littérature utilisées dans ce chapitre, notamment les bornes inférieures proposées dans [10] et [5] utilisées dans le schéma général de l'évaluation des nouvelles bornes inférieures, ainsi que les bornes inférieures proposées pour le *twobpr* ([19] et [5]). Dans la section 3.3 nous proposons une nouvelle borne inférieure pour le $2BP|R$. Les résultats de dominance de la nouvelle borne par rapport à celles de la littérature sont étudiées dans la section 3.4. Puis nous exposons les résultats expérimentaux de notre nouvelle borne sur des instances de la littérature et nous les comparons avec les bornes déjà proposées (Section 3.5). Enfin, nous concluons (Section 3.6).

3.2 Aperçu de la littérature

Nous décrivons dans cette section les différentes bornes inférieures proposées dans la littérature pour $2BP|O$ qui vont nous servir dans ce chapitre, ainsi que les bornes inférieures proposées pour le $2BP|R$ par rapport auxquels nous étudions la dominance théorique et pratique.

3.2.1 Bornes inférieures pour le $2BP|O$

Dans cette section nous décrivons dans un premier temps la borne proposée par Boschetti et Mingozzi [5] et utilisée dans [10] pour renforcer la borne introduite par les auteurs. Puis, nous décrivons la borne proposée par Carlier *et al.* [10] qui sera utilisée dans la nouvelle borne proposée pour le $2BP|R$.

3.2.1.1 La borne de Boschetti et Mingozzi [5]

La borne inférieure L^{O}_{BM} proposée par Boschetti et Mingozzi [5] est composée de trois bornes différentes $L^{O,2}_{BM}$, $L^{O,3}_{BM}$ et $L^{O,4}_{BM}$.

La borne $L^{O,2}_{BM}$ repose sur l'observation de quatre différents types d'objets les *grands*, *hauts*, *larges* et les *petits* que nous notons respectivement $A2_G$, $A2_H$, $A2_L$ et $A2_P$. Etant donné un couple d'entiers (p, q), tel que $1 \leq p \leq \frac{1}{2}W$ et $1 \leq q \leq \frac{1}{2}H$, les quatre groupes sont définis de la façon suivante :

- $A2_G = \{a_i \in A : w_i > W - p \text{ et } h_i > H - q\}$.
- $A2_H = \{a_i \in A \setminus A2_G : w_i \geq p \text{ et } h_i > H - q\}$.
- $A2_L = \{a_i \in A \setminus A2_G : w_i > W - p \text{ et } h_i \geq q\}$.
- $A2_P = \{a_i \in A \setminus A2_G \cup A2_H \cup A2_L : w_i \geq p \text{ et } h_i \geq q\}$.

Remarquons que chaque objet de $A2_G$ nécessite un bin pour le ranger et que les objets de $A2_G$ ne peuvent pas être placés avec des objets de $A2_H$, $A2_L$ et $A2_P$ dans un même bin. La borne $L^{O,2}_{BM}$ s'écrit :

$$L^{O,2}_{BM} = \max_{\substack{1 \leq q \leq \frac{1}{2}H \\ 1 \leq p \leq \frac{1}{2}W}} \{|A2_G| + max\{L'_2(p,q), L''_2(p,q)\}\} \tag{3.1}$$

où $L'_2(p,q)$ et $L''_2(p,q)$ sont deux évaluations sur le nombre minimum de bins requis pour ranger les objets de $A2_H$, $A2_L$ et $A2_P$.

La borne $L'_2(p,q)$ exploite le fait qu'aucun objet de $A2_H$ ne peut être placé au dessus d'un autre objet de $A2_H \cup A2_L \cup A2_P$. Elle est obtenue en calculant la borne inférieure pour l'instance $1BP$ suivante. Le bin est de taille WH. Pour chaque objet $a_i \in A2_H$, un objet de taille w_iH est considéré. Pour chaque objet $a_i \in A2_L$, un objet de taille Wh_i est considéré. Pour chaque objet $a_i \in A2_P$, un objet de taille w_ih_i est considéré.

La borne $L_2''(p,q)$ est basée sur l'observation suivante. Un objet quelconque de $A2_H$ ne peut pas être placé dans le même bin qu'un objet de $A2_L$. De plus, les objets de $A2_H$ ne peuvent pas être placés l'un au dessus de l'autre, et les objets de $A2_L$ ne peuvent pas être placés l'un à coté de l'autre. $L_2''(p,q)$ est alors évalué comme la somme de deux bornes inférieures $L_2^W(p,q)$ et $L_2^H(p,q)$. $L_2^W(p,q)$ est une borne inférieure valide pour le problème $1BP$ où le bin est de taille W, et pour chaque objet $a_i \in A2_H$ un objet de taille w_i est considéré, tandis que $L_2^H(p,q)$ et une borne inférieure pour le problème $1BP$ où le bin est de taille H, et pour chaque objet $a_i \in A2_L$ un objet de taille h_i est considéré.

Théorème 3.1. *(Boschetti et Mingozzi [5]) $L_{BM}^{O,2}$ est une borne valide pour le $2BP|O$.*

La borne $L_{BM}^{O,3}$ prend explicitement en considération les deux dimensions des objets. L'ensemble A des objets est décomposé en trois sous-ensembles qui dépendent de deux paramètres p et q : les grands objets ($A3_G$), les objets moyens ($A3_M$) et les petits objets $A3_P$. Soit p et q deux entiers tel que $1 \leq p \leq \frac{1}{2}W$ et $1 \leq q \leq \frac{1}{2}H$,

- $A3_G = \{a_i \in A : w_i > W - p \text{ et } h_i > H - q\}$.
- $A3_M = \{a_i \in A \setminus A3_G : w_i > \frac{1}{2}W \text{ et } h_i > \frac{1}{2}H\}$.
- $A3_P = \{a_i \in A \setminus A3_G \cup A3_M : w_i \geq p \text{ et } h_i \geq q\}$.

Remarquons que deux objets quelconques de $A3_G \cup A3_M$ ne peuvent pas être placés dans un même bin. $|A3_G \cup A3_M|$ est alors une borne inférieure valide. La borne $L_{BM}^{O,3}$ est complétée par une résolution des problèmes de sac à dos $1KP$ pour déterminer le nombre de bins nécessaires au rangement des petits objets. Ainsi, quatre valeurs $M_W(W, A3_P)$, $M_H(H, A3_P)$, $M_W(W - w_i, A3_P)$ et $M_H(H - h_i, A3_P)$ correspondant chacune à la solution d'un problème $1KP$ formalisé par les équations (3.2) et (3.3) sont calculés (où *dim* est un entier positif). Ce problème peut être résolu en temps linéaire si les objets de $A3_P$ sont triés par ordre croissant de largeur ou de hauteur selon le problème.

$$M_W(dim, A3_P) = Max\{\sum_{a_i \in A3_P} \xi_i : \sum_{a_i \in A3_P} w_i\xi_i \leq dim, \xi_i \in \{0,1\}\} \tag{3.2}$$

$$M_H(dim, A3_P) = Max\{\sum_{a_i \in A3_P} \xi_i : \sum_{a_i \in A3_P} h_i\xi_i \leq dim, \xi_i \in \{0,1\}\} \tag{3.3}$$

$M_W(W, A3_P)$ (resp. $M_H(H, A3_P)$) correspond au nombre maximum d'objets de $A3_P$ qui peuvent être rangés côte à côte selon W (resp. les uns au dessus des autres selon H) dans un bin. $M_W(W - w_i, A3_P)$ (resp. $M_H(H - h_i, A3_P)$) correspond au nombre maximum d'objets de $A3_P$ qui peuvent être rangés côte à côte selon W (resp. les uns au dessus des autres selon H) dans un bin contenant l'objet a_i de $A3_M$.

Soit $m'(a_i, A3_P)$ une borne supérieure pour le nombre d'objets de $A3_P$ qui peuvent être rangés avec l'objet a_i, la borne $L^{O,3}_{BM}$ s'écrit :

$$L^{O,3}_{BM} = \max_{\substack{1\leq q\leq \frac{1}{2}H \\ 1\leq p\leq \frac{1}{2}W}} \left\{ |A3_G \cup A3_M| + max\left\{0, \left\lceil \frac{|A3_P| - \sum_{a_i\in A3_M} m'(a_i, A3_P)}{M_W(W, A3_P)M_H(H, A3_P)} \right\rceil \right\} \right\} \tag{3.4}$$

Notons que, $M_W(W, A3_P)M_H(H, A3_P)$ est une borne supérieure pour le nombre d'objets de $A3_P$ qui peuvent être rangés ensemble dans un bin.

$$\begin{aligned} m'(a_i, A3_P) &= M_W(W - w_i, A3_P)M_H(H - h_i, A3_P) \\ &+ (M_W(W, A3_P) - M_W(W - w_i, A3_P))(M_H(H - h_i, A3_P)) \\ &+ (M_W(H, A3_P) - M_W(H - h_i, A3_P))(M_H(W - w_i, A3_P)) \end{aligned} \tag{3.5}$$

Théorème 3.2. *(Boschetti et Mingozzi [5]) $L^{O,3}_{BM}$ est une borne valide pour le $2BP|O$.*

La borne $L^{O,4}_{BM}$ utilise des techniques d'arrondi qui consiste à réduire les dimensions des petits objets et de croître celles des grands objets. L'ampleur de la réduction de dimensions des petits objets dépend de la valeur de la dimension en question et d'un paramètre p ou q selon qu'on traite la largeur ou la hauteur. Le nombre de petits objets qui peuvent être rangés avec un grand objet a_i devant rester inchangé après l'application des techniques d'arrondi. Ainsi les auteurs considèrent cinq sous-ensembles définis de la façon suivante :

- $A4_G = \{a_i \in A : w_i > W - p \text{ et } h_i > H - q\}$.
- $A4_M = \{a_i \in A \setminus A4_G : w_i > \frac{1}{2}W \text{ et } h_i > \frac{1}{2}H\}$.
- $A4_H = \{a_i \in A : \frac{1}{2}W \geq w_i \geq p \text{ et } h_i > \frac{1}{2}H\}$.
- $A4_L = \{a_i \in A : w_i > \frac{1}{2}W \text{ et } \frac{1}{2}H \geq h_i \geq q\}$.
- $A4_P = \{a_i \in A : \frac{1}{2}W \geq w_i \geq p \text{ et } \frac{1}{2}H \geq h_i \geq q\}$.

Cette borne exploite les observations suivantes. Deux objets quelconques de $(A4_G \cup A4_M)$ ne peuvent pas être placés dans un même bin. Aucun objet de $A4_H \cup A4_G \cup A4_P$ ne peut être placé dans le même bin qu'un objet de $A4_G$. La borne $L^{O,4}_{BM}$ est alors composée de deux termes, le premier est égal à la taille de l'ensemble $A4_G \cup A4_M$, le deuxième est une borne inférieure valide pour le nombre de bins nécessaires pour ranger les objets de $A4_H \cup A4_L \cup A4_P$ qui ne peuvent pas être placés dans le même bin que les objets de $A4_M$.

$$L^{O,4}_{BM} = \max_{\substack{1\leq p\leq \frac{W}{2} \\ 1\leq q\leq \frac{H}{2}}} \left\{ |A4_G \cup A4_M| + max\left\{0, \left\lceil \frac{\sum_{a_i\in A\setminus A4_G} m''(a_i, p, q)}{\left\lfloor \frac{W}{p} \right\rfloor \left\lfloor \frac{H}{q} \right\rfloor} \right\rceil \right\} \right\} \tag{3.6}$$

où

$$m''(a_i,p,q) = \begin{cases} \lfloor \frac{W-w_i}{p} \rfloor \lfloor \frac{H-h_i}{q} \rfloor - \lfloor \frac{W-w_i}{p} \rfloor \lfloor \frac{H}{q} \rfloor - \lfloor \frac{W}{p} \rfloor \lfloor \frac{H-h_i}{q} \rfloor \text{ si } a_i \in A4_M \\ \lfloor \frac{w_i}{p} \rfloor \lfloor \frac{H}{k} \rfloor - \lfloor \frac{w_i}{p} \rfloor \lfloor \frac{H-h_i}{q} \rfloor \text{ si } a_i \in A4_H \\ \lfloor \frac{W}{p} \rfloor \lfloor \frac{h_i}{q} \rfloor - \lfloor \frac{W-w_i}{p} \rfloor \lfloor \frac{h_i}{q} \rfloor \text{ si } a_i \in A4_L \\ \lfloor \frac{w_i}{p} \rfloor \lfloor \frac{h_i}{q} \rfloor \text{ si } a_i \in A4_P \end{cases}$$

Théorème 3.3. *(Boschetti et Mingozzi [5]) $L_{BM}^{O,4}$ est une borne valide pour le $2BP|O$.*

3.2.1.2 La borne de Carlier *et al.* [10]

Carlier *et al.* [10] proposent une nouvelle borne inférieure dédiée au $2BP|O$ qui domine les bornes de la littérature. Cette borne repose sur le concept des fonctions redondantes ou DFF discrètes définies de la façon suivante.

Définition 3.1. *Une Fonction Redondante (RF), ou DFF dicrète f est une application discrète de $[0,X]$ dans $[0,X']$ (X et X' entiers) telle que :*

$$x_1 + x_2 + \ldots + x_k \leq X \Rightarrow f(x_1) + f(x_2) + \ldots + f(x_k) \leq f(X) = X'.$$

A partir de cette section, nous parlons de DFF pour faire référence aux DFF discrètes.

Les DFF sont définies indépendamment de l'instance. Une autre classe de fonctions qui dépendent de la taille des objets dans l'instance considérée a été définie par les auteurs. Il s'agit des DFF dépendantes de la donnée appelées DDFF (*cf.* définition 3.2).

Définition 3.2. *Soit $A = \{a_1, \ldots, a_n\}$, $c_1, c_2, \ldots, c_n$ n valeurs entières et C un entier tel que $C \geq c_i$ pour $i = 1, \ldots, n$. Une* Fonction Dual-Réalisable Dépendante de la Donnée *(DDFF) f associée à C et $c_1, c_2, \ldots, c_n$ est une application discrète de $[0,C]$ dans $[0,C']$ telle que*

$$\forall A_1 \subset A, \sum_{a_i \in A_1} c_i \leq C \Rightarrow \sum_{a_i \in A_1} f(c_i) \leq f(C) = C'$$

Les DDFF ont un comportement similaire aux DFF sur un exemple spécifique donné. Les DDFF peuvent prendre en compte les caractéristiques de l'instance comme le nombre d'occurrences de chaque valeur, et peuvent ainsi conduire à des bornes inférieures qui améliorent strictement celles obtenues par l'application des DFF.

Les auteurs considèrent deux familles de DFF (f_0^k et f_2^k)et une famille de DDFF (f_1^k). f_0^k est directement dérivée d'une famille classique de DFF proposée dans [27].

f_2^k est une amélioration sur une DFF utilisée dans [6]. (f_1^k) est une nouvelle famille de DDFF. Etant donné une valeur entière C et un autre entier $k = 1, ..., C/2$, les trois fonctions sont définies ci après.

Une dimension d'un objet a_i, peut être augmentée jusqu'à la taille d'un bin si aucun autre objet dans l'instance ne peut être rangé avec a_i suivant la prolongation de la dimension considérée. La famille de DFF f_0^k proposée consiste à supprimer les objets de taille inférieure à k et augmenter la taille des objets strictement supérieure à $C - k$.

$$f_0^k : [0, C] \to [0, C]$$
$$x \mapsto \begin{cases} C \text{ si } x > C - k \\ x \text{ si } C - k \geq x \geq k \\ 0 \text{ sinon} \end{cases}$$

La fonction f_1^k est définie pour une valeur de l'entier C et une liste d'entiers $c_1, c_2, \ldots, c_n$ $(A = a_1, ..., a_n)$. Soit A' le sous-ensemble d'objets suivant : $A' = \{a_i \in A : \frac{1}{2}C \geq c_i \geq k\}$. Etant donné un entier X, soit $M(X, J)$ le nombre maximal d'objets dans J qui peuvent être placés côte à côte sans dépasser X. $M(X, J)$ est la valeur optimale pour le problème de sac à dos en une dimension (1KP) induit par l'ensemble A' et la taille X. la nouvelle DDFF f_1^k s'écrit alors :

$$f_1^k : [0, C] \to [0, M(C, J)]$$
$$x \mapsto \begin{cases} M(C, J) - M(C - x, J) & \text{si } x > \frac{1}{2}C \\ 1 & \text{si } \frac{1}{2}C \geq x \geq k \\ 0 & \text{sinon} \end{cases}$$

La fonction f_2^k repose sur des techniques d'arrondi qui consiste à augmenter la taille des grands objets et réduire celle des petits, elle est définie de la manière suivante :

$$f_2^k : [0, C] \to \left[0, 2 \times \left\lfloor \frac{C}{k} \right\rfloor\right]$$
$$x \mapsto \begin{cases} 2 \times (\lfloor \frac{C}{k} \rfloor - \lfloor \frac{C-x}{k} \rfloor) & \text{si } x > \frac{1}{2}C \\ \lfloor \frac{C}{k} \rfloor & \text{si } x = \frac{1}{2}C \\ 2 \times \lfloor \frac{x}{k} \rfloor & \text{si } \frac{1}{2}C > x \end{cases}$$

Carlier *et al.* [10] introduisent une borne inférieure pour le $1BP$ notée L_{CCM}^{1D} [10] et ils montrent qu'elle domine théoriquement la borne L_{MV}^{1D} proposée dans [48]. Ce résultat est utilisé par les auteurs pour montrer la dominance théorique de la nouvelle borne inférieure qu'ils proposent par rapport aux bornes de la littérature, en particulier $L_{BM}^{O,2}$ [5].

$$L_{CCM}^{1D} = \max_{u \in \{0,1,2\}} \left\{ L_{f_u}^{1BP}(c_i) \right\}$$

où L_{f_u} est définie de la façon suivante :

$$L_{f_u} = \max_{1 \leq k \leq \frac{C}{2}} \left\{ \left\lceil \frac{\sum_{a_i \in A} f_u^k(c_i)}{f_u^k(C)} \right\rceil \right\}$$

Puis ils introduisent des nouvelles bornes inférieures pour le $2BP|O$ suivant le schéma défini dans [29]. Pour $u = 0, 1, 2$ et $v = 0, 1, 2$ ils définissent :

$$L^{2D}_{(f_u, f_v)} = \max_{1 \leq k \leq \frac{W}{2}, 1 \leq l \leq \frac{H}{2}} \left\{ \left\lceil \frac{\sum_{a_i \in A} f_u^k(w_i) f_v^l(h_i)}{f_u^k(W) f_v^l(H)} \right\rceil \right\}$$

$L^{2D}1, CCM$ est une borne inférieure valide pour le $2BP$, elle est égale à la borne continue maximale obtenue en appliquant les DDF (ou DDFF) proposées.

$$L^{2D}_{1,CCM} = \max_{u \in \{0,1,2\}, v \in \{0,1,2\}} \left\{ L_{(f_u, f_v)} \right\} \tag{3.7}$$

Cette borne est renforcée par la borne $L^{2D}2, CCM$ obtenue en utilisant la borne L^{1D}_{CCM} à la place de L^{1D}_{MV} dans la borne $L^{O,2}_{BM}$. En effet, la borne $L^{O,2}_{BM}$ ne peut pas être calculée par l'intermédiaire des DFF.

La nouvelle borne inférieure valide pour le $2BP$ est notée L^{O}_{CCM}, Elle domine les bornes de la littérature [10] :

$$L^{O}_{CCM} = max\{L^{2D}_{1,CCM}, L^{2D}_{2,CCM}\} \tag{3.8}$$

Notons que les DFF f_0 et f_2 proposées par les auteurs sont des fonctions croissantes et superadditives, elles vérifient donc la relation suivante : $x + y \leq z$ implique $f(x) + f(y) \leq f(z)$ pour $f \in \{f_0, f_2\}$.

3.2.2 Bornes inférieures pour le $2BP|R$

Dans cette section nous décrivons deux bornes inférieures pour le $2BP|R$ proposées dans la littérature. Remarquons que la borne continue est aussi une borne inférieure valide pour le $2BP|R$. Premièrement, nous décrivons la borne inférieure proposée dans [19] où les auteurs proposent de découper les objets de l'instance initiale en carrés. Une borne inférieure dédiée au problème de placement des objets carrés est alors appliquée à l'instance obtenue. La borne inférieure L^{R}_{BM} proposée dans [6] est composée de deux bornes $L^{R,1}_{BM}$ et $L^{R,2}_{BM}$. La première consiste à transformer les objets en carrés et appliquer par la suite une borne inférieure dédiée au $2BP|O$. La deuxième borne prend explicitement en considération les deux dimensions des objets et la possibilité de rotation de 90 degrés.

3.2.2.1 La borne LB^R_{DA} de Dell'Amico *et al.* [19]

Le calcul de la borne LB^R_{DA} se fait en deux étapes. Dans la première étape, chaque objet $a_i \in A$ est décomposé en plusieurs carrés. Ces objets carrés sont obtenus par un algorithme pseudo-polynomial appelé CUTSQ. Initialement chaque rectangle est considéré dans l'orientation r tel que $w_i^{r(i)} \geq h_i^{r(i)}$ (orientation horizontale). Chaque itération de CUTSQ, consiste à couper de chaque rectangle (w_i, h_i) le nombre maximal de carrés de taille (h_i, h_i), et de tourner le rectangle résiduel de 90 degrés pour conserver une orientation horizontale. Dans une deuxième étape, une borne dédiée au problème de rangement des carrés est appliquée à l'instance résultante notée $I_S = (A_S, B)$. Soit l_j la dimension d'un objet a_j dans A_S. A_S est alors divisé en cinq sous-groupe de la façon suivante (le bin est considéré dans une orientation horizontale : $W \geq H$). Etant donné un entier k tel que $0 \leq k \leq \frac{1}{2}H$, les cinq sous-ensembles d'objets sont définis de la façon suivante :

- $A_1 = \{a_j \in A_S : l_j > W - k\}$
- $A_2 = \{a_j \in A_S : W - k \geq l_j > \frac{1}{2}W\}$
- $A_3 = \{a_j \in A_S : \frac{1}{2}W \geq l_j > \frac{1}{2}H\}$
- $A_4 = \{a_j \in A_S : \frac{1}{2}H \geq l_j \geq k\}$
- $A_{23} = \{a_j \in A_2 \cup A_3 : l_j > H - k\}$

$$LB^R_{DA} = \max_{0 \leq k \leq \frac{1}{2}H} \left\{ |A_1| + \tilde{L} + max\left\{0, \left\lceil \frac{\sum_{a_j \in A_2 \cup A_3 \cup A_4} l_j^2 - (WH\tilde{L} - \sum_{a_j \in A_{23}} l_j(H - l_j))}{WH} \right\rceil \right\} \right\}$$

LB^R_{DA} est une borne inférieure valide pour le $2BP|R$, où $\tilde{L}$ est une borne inférieure valide sur le nombre de bins nécessaire pour ranger les objets de $A_2 \cup A_3$, elle est basée sur les observations suivantes. Chaque objet de $A_1 \cup A_2$ exige un bin, aucun objet de A_3 ne peut être placé avec un objet de A_1 ou au dessus d'un objet de A_2 et un objet au plus de A_3 peut être placé à côté d'un objet de A_2. Les auteurs considèrent alors le sous-ensemble $\overline{A_3}$ des objets de A_3 qui peuvent être rangés dans les bins qui contiennent les objets de A_2. La borne $\tilde{L}$ est alors complétée par une borne inférieure pour le nombre de bins nécessaires pour ranger les objets de $A_3 \setminus \overline{A_3}$:

$$\tilde{L} = |A_2| + max\left\{ \left\lceil \frac{\sum_{a_j \in A_3 \setminus \overline{A_3}} l_j}{W} \right\rceil, \left\lceil \frac{|A_3 \setminus \overline{A_3}|}{\left\lfloor \frac{W}{\lfloor H/2+1 \rfloor} \right\rfloor} \right\rceil \right\}$$

Le dernier terme de la borne LB^R_{DA} est une borne inférieure sur le nombre de bins qu'on doit ajouter pour ranger les objets de A_4.

Dans le cas de bins carrés (W, W) et pour une valeur donnée de k, la borne LB^R_{DA} peut être écrite de la façon suivante :

$$L'^R_{DA}(k) = |A_1 \cup A_2| + max\{0, \lceil \frac{\sum_{a_j \in A_2 \cup A_4} (l_j)^2}{W^2} - |A_2| \rceil\}$$

3.2.2.2 La borne L^R_{BM} de Boschetti et Mingozzi [6]

Boschetti et Mingozzi [6] proposent une nouvelle borne inférieure notée L^R_{BM}. Cette borne est calculée comme le maximum de deux bornes inférieures $L^{R,1}_{BM}$ et $L^{R,2}_{BM}$.

La borne inférieure $L^{R,1}_{BM}$ consiste à transformer les objets de l'instance $2BP|R$ initiale en des objets à orientation fixe. Ainsi, étant donnée une instance initiale $I = (A, B)$, les dimensions des objets de A sont transformées de la façon suivante :

$$a_j \mapsto a'_j = (w'_j, h'_j) \text{ où}$$
$$\begin{cases} w'_j = min(w_j, h_j) \text{ et } h'_j = min(w_j, h_j) \text{ si } w_j \leq H \text{ et } h_j \leq W \\ w'_j = w_j \text{ et } h'_j = h_j \text{ sinon} \end{cases}$$

$a_j \mapsto a'_j = (w'_j, h'_j)$, où $w'_j = min(w_j, h_j)$ si $(w_j \leq H$ et $h_j \leq W)$ et $w'_j = w_j$ sinon. $h'_j = min(w_j, h_j)$ si $(w_j \leq H$ et $h_j \leq W)$ et $h'_j = h_j$ sinon.

Ainsi si un objet donné ne peut pas être tourné (objet à orientation fixe) il est laissé inchangé. Dans l'autre cas, seul le plus grand carré inclus dans l'objet rectangulaire est considéré. Les objets résultants sont alors à orientation fixe. $L^{R,1}_{BM}$ consiste alors à appliquer la borne inférieure L^O_{BM} proposée dans [5] pour le $2BP|O$.

La borne inférieure $L^{R,2}_{BM}$ prend en considération les deux dimensions des objets ainsi que la possibilité de les tourner de 90 degrés. Deux sous-groupes différents sont considérés : $A' = \{a_i \in A : w_i \neq w'_i\}$ et $A'' = A \setminus A'$. Etant donné deux entiers k et l tel que $1 \leq k \leq \frac{1}{2}H$ et $1 \leq l \leq \frac{1}{2}W$, une borne inférieure valide $L^{R,2}_{BM}$ peut être écrite de la façon suivante :

$$L^{R,2}_{BM} = \max_{k,l} \left\{ \left\lceil \frac{\sum_{i=1}^{n} \mu^{k,l}(i)}{\lfloor \frac{H}{k} \rfloor \lfloor \frac{W}{l} \rfloor} \right\rceil \right\}$$

où

$$\mu^{k,l}(i) = \begin{cases} min\big\{\eta^{l,W}(w_i) \times \eta^{k,H}(h_i), \\ \qquad \eta^{l,W}(h_i) \times \eta^{k,H}(w_i)\big\} & \text{si } a_i \in A' \\ \eta^{l,W}(w_i) \times \eta^{k,H}(h_i) & \text{si } a_i \in A'' \end{cases}$$

et

$$\eta^{k,X}(x) = \begin{cases} \lfloor \frac{X}{k} \rfloor \lfloor \frac{X-x}{k} \rfloor & \text{if } x > \frac{X}{2} \\ \lfloor \frac{x}{k} \rfloor & \text{if } x \leq \frac{X}{2} \end{cases}$$

Notons que $\eta^{k,X}$ est une version de la fonction f_2^k décrite dans la section 3.2.1.2, où un objet de taille $\frac{X}{2}$ peut avoir une image plus petite.

La borne finale proposée dans [6] est alors $max(L_{BM}^{R,1}, L_{BM}^{R,2})$. Nous notons cette borne L_{BM}^{R}.

3.3 Nouvelle borne inférieure pour le $2BP|R$

Nous adoptons dans ce chapitre, la notation suivante. Soit une instance $I = (A, B)$ donnée. A représente l'ensemble des objets de l'instance, $A = \{a_1, ..., a_n\}$ tel que $a_i = (w_i, h_i)$ $\forall a_i \in A$. Un objet a_i considéré dans son orientation initiale est de taille $(w_i^1, h_i^1) = (w_i, h_i)$. Si a_i est tourné de 90 degrés, sa taille est alors notée (w_i^2, h_i^2), telle que $w_i^2 = h_i^1$ and $h_i^2 = w_i^1$. Une orientation de l'ensemble des objets est une application de A dans $\{1, 2\}$ qui associe à chaque objet a_i une orientation $r(i)$.

Dans cette section, nous proposons une nouvelle méthode d'évaluation des bornes inférieures pour le $2BP|R$ dans le cas de bins carrés. Cette méthode consiste à construire une nouvelle instance $2BP|O$ à partir de l'instance $2BP|R$. Chaque objet de l'instance $2BP|R$ apparaît deux fois dans la nouvelle instance $2BP|O$ (dans ses deux orientations possibles). Nous proposons par la suite une généralisation de cette borne pour qu'elle puisse être appliquée dans le cas de bins non carrés.

Soit $I^R = (A, B)$ une instance de $2BP|R$. $A = \{a_1, \ldots, a_n\}$ est l'ensemble des objets de l'instance et $B = (W, W)$ (bins carrés). Considérons maintenant le problème $2BP|O$ suivant. $\widehat{I}^O = (\widehat{A}^O, B)$, avec $\widehat{A}^O = \{a_1, \ldots, a_{2n}\}$, tel que

1. $w_i = w_i^1$ et $h_i = h_i^1$ pour $i \in \{1, \ldots, n\}$
2. $w_i = w_{i-n}^2$ et $h_i = h_{i-n}^2$ pour $i \in \{n+1, \ldots, 2n\}$

Le théorème 3.4 établit que $\widehat{I}^O$ a besoin au plus du double du nombre de bins nécessaires pour I^R.

Théorème 3.4. *Si z bins sont nécessaires pour ranger les objets de I^R, il existe alors une solution $\widehat{I}^O$ avec $2z$ bins.*

Démonstration. Considérons la solution de I^R qui a besoin de z bins. Dans cette solution, chaque objet a_i est utilisé une seule fois dans une de ses deux orientations possibles. Une solution avec $2z$ bins peut alors être trouvée pour $\widehat{I}^O$ en utilisant la même configuration pour les objets qui sont dans I^R, et ranger les objets restants en tournant les z premiers bins de 90° (*cf.* Figure 3.1). □

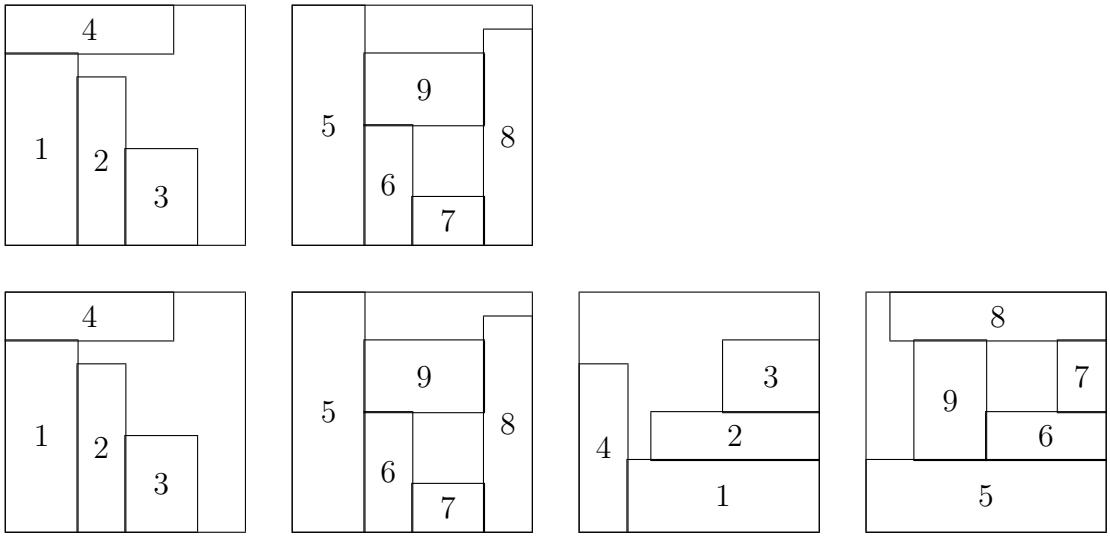


Figure 3.1 – Construction d'une solution pour $\widehat{I}^O$

La figure 3.1 montre la construction d'une solution avec quatre bins pour $\widehat{I}^O$ (les quatre bins du dessous) à partir d'une solution avec deux bins connue pour I^R(les deux bins du dessus). La configuration des objets placés dans les deux premiers bins est identique à la solution du problème initial. La configuration du troisième (resp. quatrième) bin est obtenue en tournant de 90 degrés la configuration du premier (resp. deuxième) bin.

Corollaire 3.1. *Si LB^O est une borne inférieure valide pour le $2BP|O$, alors $\lceil LB^O(\widehat{I}^O)/2 \rceil$ est une borne inférieure valide de la solution optimale de I^R.*

D'après le corollaire 3.1, toute borne inférieure pour le $2BP$ avec orientation fixe des objets peut être utilisée pour calculer une borne inférieure pour le cas non-orienté. Nous utilisons la borne L^O_{CCM} proposée dans [10] après une légère modification. La légère modification sur L^O_{CCM} concerne la famille f_1^k de DDFF : dans la nouvelle borne que nous proposons, les deux objets à orientation fixe issus du même objet d'origine peuvent être placés dans un même bin (si cela n'est pas permis, la borne reste valide). Dans ce cas, dans le sous-groupe dépendant de la donnée entrant en jeu dans le calcul de la fonction f_1^k, nous ne considérons qu'un seul objet par couple (l'objet dont la dimension considérée est la plus petite). La nouvelle borne inférieure obtenue est notée L^R_{CJE}. Elle est de même complexité que la borne inférieure utilisée (L^O_{CCM}). Elle est en $O(n^3)$.

Dans le cas de bins non carrés, la méthode proposée peut toujours être utilisée. Nous supposons sans perte de généralité que $W > H$. Dans ce cas, nous transformons le bin (W, H) en un bin (W, W). Pour éviter une relaxation excessive du problème,

un objet factice de taille $(W, W-H)$ est ajouté à chaque bin pour remplir la surface ajoutée. Cette procédure est mise en place de la façon suivante. z objets factices sont ajoutés à la nouvelle instance $2BP|O$ considérée. Si la borne inférieure est supérieure à z, la méthode est relancée avec la valeur $z+1$. La valeur initiale donnée à z peut être égale à un, ou une borne inférieure connue pour le problème. Lors de l'utilisation de la fonction f_1^k, il faut s'assurer qu'un objet factice au plus entre en jeu dans la solution du problème du sac à dos.

3.4 Dominance par rapport aux bornes de la littérature

Dans cette section, nous montrons que la borne inférieure L^R_{CJE} proposée dans la section précédente domine les méthodes proposées dans [19] et [6], dans le cas de bins carrés. Pour cela, nous introduisons une nouvelle borne notée $L^{R,t}_{CJE}$, qui domine les bornes de la littérature et qui est dominée par L^R_{CJE}.

Soit $\mathcal{F}$ l'ensemble des fonctions f_0^k et f_2^k valides pour une instance $I = (A, B)$ donnée. La borne $L^{R,t}_{CJE}$ généralise la borne $L^{R,2}_{BM}$ proposée dans [6].

$$L^{R,t}_{CJE} = \max_{f,g\in\mathcal{F}} \left\lceil \sum_{a_i\in A} \frac{min\left\{f(w_i^1)g(h_i^1), f(w_i^2)g(h_i^2)\right\}}{f(W)g(W)} \right\rceil$$

Proposition 3.1. *$L^{R,t}_{CJE}$ est une borne inférieure valide pour le $2BP|R$.*

Démonstration. Si nous explorons toutes les orientation possibles des objets de l'instance I et évaluons les bornes inférieures pour les problèmes $2BP|O$ résultants, la valeur minimale obtenue est une borne inférieure pour I. Notons $\mathcal{R}$ l'ensemble de toutes les orientations r possibles des objets.

$$OPT(I) \geq \min_{r\in\mathcal{R}} \left\{ \max_{f,g\in\mathcal{F}} \left\lceil \sum_{a_i\in A} \frac{f(w_i^{r(i)})g(h_i^{r(i)})}{f(W)g(W)} \right\rceil \right\}$$

Cette expression est supérieure ou égale à :

$$\min_{r\in\mathcal{R}} \left\{ \max_{f,g\in\mathcal{F}} \left\lceil \sum_{a_i\in A} \frac{min\{f(w_i^{r(i)})g(h_i^{r(i)}), f(h_i^{r(i)})g(w_i^{r(i)})\}}{f(W)g(W)} \right\rceil \right\}$$

Or l'orientation choisie pour les objets n'affecte pas cette dernière expression. L'expression ci après est alors une borne inférieure pour I.

$$\max_{f,g\in\mathcal{F}} \left\lceil \sum_{a_i\in A} \frac{min\{f(w_i^1)g(h_i^1), f(w_i^2)g(h_i^2)\}}{f(W)g(W)} \right\rceil$$

□

Nous allons maintenant montrer que $L^{R,t}_{CJE}$ est dominée par L^{R}_{CJE}. Ce résultat sera utilisé pour montrer que L^{R}_{CJE} domine les bornes de la littérature.

Proposition 3.2. *La borne $L^{R,t}_{CJE}$ est dominée par L^{R}_{CJE} dans le cas de bins carrés.*

Démonstration. Considérons une version simplifiée de L^{R}_{CJE}, où seules f_0 et f_2 sont utilisées. La borne résultante est la suivante :

$$LB = \left\lceil \frac{1}{2} \max_{f,g \in \mathcal{F}} \Big\{ \sum_{i=1}^{n} \frac{f(w_i^1)g(h_i^1)}{f(W)g(W)} + \sum_{i=n+1}^{2n} \frac{f(w_i^1)g(h_i^1)}{f(W)g(W)} \Big\} \right\rceil$$

Par définition de l'instance $\widehat{I}^O$, pour $i = 1, \ldots, n$, nous avons $w_i^1 = h_{i+n}^2$ et $h_i^1 = w_{i+n}^2$. La dernière équation peut alors être écrite de la façon suivante :

$$LB = \left\lceil \max_{f,g \in \mathcal{F}} \left\{ \sum_{i=1}^{n} \frac{1}{2} \frac{f(w_i^1)g(h_i^1) + f(w_i^2)f(h_i^2)}{f(W)g(W)} \right\} \right\rceil$$

or

$$(f(w_i^1)g(h_i^1) + f(w_i^2)g(h_i^2))/2$$

est supérieure ou égale à

$$min\{f(w_i^1)g(h_i^1), f(w_i^2)g(h_i^2)\}$$

la borne simplifiée domine alors $L^{R,t}_{CJE}$. Il en est de même pour L^{R}_{CJE}. □

Nous allons maintenant montrer que $L^{R,t}_{CJE}$ domine la borne de [19]. Pour cela, nous commençons par décrire un résultat concernant les DFF. Le lemme 3.1 explique que le fait de découper les objets avant l'application d'une borne inférieure ne peut pas améliorer la qualité de la borne quand $L^{R,t}_{CJE}$ est utilisée. Le lemme est vrai pour le cas de découpes horizontales ainsi que verticales.

Lemme 3.1. *Soit deux DFF f et g, et quatre valeurs w_a, w_b, w et h tel que $w_a + w_b = w$, nous avons :*

$$\begin{aligned} &min\{f(w_a)g(h), g(w_a)f(h)\} + min\{f(w_b)g(h), g(w_b)f(h)\} \\ &\leq min\{f(w)g(h), g(w)f(h)\} \end{aligned}$$

Démonstration. Comme f est superadditive, nous avons :

$$f(w_a) + f(w_b) \leq f(w)$$

cela implique que

$$f(w_a)g(h) + f(w_b)g(h) \leq f(w)g(h)$$

d'où :

$$min\{f(w_a)g(h), g(w_a)f(h)\} + min\{f(w_b)g(h), g(w_b)f(h)\} \leq f(w)g(h).$$

En utilisant la même propriété pour g, nous obtenons :

$$g(w_a)f(h) + g(w_b)f(h) \leq g(w)f(h)$$

nous avons alors :

$$min\{f(w_a)g(h), g(w_a)f(h)\} + min\{f(w_b)g(h), g(w_b)f(h)\} \leq g(w)f(h)$$

Le lemme est par conséquent vrai. □

Notons que le résultat s'applique aussi dans le cas où $w_a = h$, comme dans la méthode L^R_{DA} [19].

Proposition 3.3. *L^R_{CJE} domine la borne L'^R_{DA} de [19] dans le cas de bins carrés.*

Démonstration. Dans la borne L'^R_{DA} [19] les objets (rectangles) d'une instance I sont découpés récursivement en rectangles plus petits. Montrons maintenant que la borne inférieure $L'^R_{DA}(k)$ appliquée pour une valeur donnée de k peut être calculée par le biais de DFF.

Si $L'^R_{DA}(k) = |A_1 \cup A_2|$, elle est alors inférieure ou égale à la borne continue calculée pour l'instance transformée en appliquant $f_0^{\frac{W}{2}}$ sur les deux dimensions.

Si $L'^R_{DA}(k) = |A_1| + \lceil \frac{\sum_{a_i \in A_2 \cup A_4} (w_i)^2}{W^2} \rceil$, elle est alors égale à la borne continue calculée pour l'instance transformée en appliquant f_0^k sur les deux dimensions.

Dans les deux cas, la borne peut alors être obtenue en appliquant deux DFF dans $\mathcal{F}$.

Nous montrons maintenant que toute méthode qui consiste à découper les objets de l'instance initiale et calculer par la suite une borne inférieure, par l'intermédiaire des DFF, pour l'instance résultante ne peut pas donner des résultats meilleurs que l'application de la même borne à l'instance initiale.

Soit a_i un objet de A, et J_i l'ensemble des objets obtenus par des découpes récursives de l'objet a_i. D'après le lemme 3.1, nous savons que la surface résultante de a_i transformé après l'application de $L^{R,t}_{CJE}$ est supérieure ou égale à la somme des surfaces de deux premières découpes de a_i. En utilisant ce résultat d'une façon récursive, nous constatons que la surface de a_i modifiée par deux DFF est supérieure à la surface totale des objets de J_i modifié par DFF. En sommant les inégalités obtenues, nous trouvons que $L^{R,t}_{CJE}$ est supérieure à L^R_{DA}. Or L^R_{CJE} domine $L^{R,t}_{CJE}$ dans le cas de bins carrés, elle domine alors de même L^R_{DA}.

□

Nous allons maintenant montrer que notre borne domine la borne L^R_{BM} de [6].

Proposition 3.4. *L^R_{CJE} domine la borne L^R_{BM} de [6] dans le cas de bins carrés.*

Démonstration. Dans le cas de bins carrés, la borne $L^{R,1}_{BM}$ (*cf.* Section 3.2) consiste à considérer l'instance obtenue en gardant le carré le plus large dans chaque objet de l'instance initiale, puis appliquer les trois bornes $L^{O,2}_{BM}$, $L^{F,3}_{BM}$ et $L^{F,4}_{BM}$ décrites dans [5] (*cf.* Section 3.2).

Pour le $2BP|R$, dans le cas de bins carrés, la borne $L^{O,2a}_{BM}$ est équivalente à la création d'une instance $1BP$ telle que chaque objet a_i a la taille suivante $(min\{w_i, h_i\})^2$, la taille du bin étant W^2. Les DFF sont alors appliquées (*cf.* Section 3.2). Dans L^R_{CJE}, l'instance considérée est composée des objets plus larges que ceux de $L^{O,2}_{BM}$. Comme la borne inférieure intérieure L^O_{CCM} utilise la même technique que $L^{O,2a}_{BM}$, avec des DFF qui dominent celles de [5], elle domine alors cette borne (rappelons que les DFF sont superadditives).

Quand la plus petite des deux dimensions est gardée, seuls les objets larges en terme de hauteur et de largeur sont considérés dans la borne $L^{O,2b}_{BM}$. La borne obtenue ne peut donc pas être supérieure la borne obtenue en appliquant $f_0^{\frac{W}{2}}$ à chaque dimension.

Carlier *et al.* [10] ont montré que $L^{O,3}_{BM}$ peut être calculée par l'intermédiaire de deux fonctions f^k_1. La surface totale des objets considérés est évidement supérieure quand L^R_{CJE} est appliquée. Bien que les DFF sont superadditives, les DDFF ne le sont pas, mais rappelons que l'ensemble des valeurs dépendant de la donnée J considérés dans L^R_{CJE} est le même que celui considéré dans $L^{R,1}_{BM}$. En effet, en utilisant l'amélioration sur la borne L^O_{CCM} proposée dans la section 3.3, seule la valeur $min\{w^1_i, h^1_i\}$ est considérée. Elle est identique à la valeur considérée dans $L^{R,1}_{BM}$. La surface obtenue pour chaque objet, est alors supérieure ou égale à la surface totale obtenue en utilisant la méthode de [5,6]. En sommant les inégalités obtenues, nous vérifions la règle de dominance.

Carlier *et al.* [10] ont aussi montré que $L^{O,4}_{BM}$ peut être calculée par l'intermédiaire de deux DFF qui sont dominées par f^k_2. En se référant au lemme 3.1 et Proposition 3.2 nous déduisons que L^R_{CJE} domine la borne $L^{O,4}_{BM}$.

Finalement la borne $L^{R,2}_{BM}$ [6] est obtenue à partir du schéma de la borne $L^{R,t}_{CJE}$ en utilisant une version simplifiée de f^k_2, elle est alors dominée par $L^{R,t}_{CJE}$. D'après la proposition 3.2, la borne $L^{R,2}_{BM}$ est alors dominée par L^R_{CJE}.

La borne L^R_{BM} est donc dominée par L^R_{CJE}. □

Dans le cas de bins non carrés, les résultats de dominance montrés ne sont plus valides. L'exemple donné ci après illustre ce fait. Soit l'instance $I = (A, B)$ suivante tel que : $A = \{(4,1),(4,1),(3,3)\}$ est $B = (6,3)$. La méthode de [6] détermine un ensemble d'objets qui ne peuvent pas être tournés : $(4,1)$ et $(4,1)$. f^3_0 est alors appliquée à la largeur et f^1_0 à la hauteur, La borne inférieure résultante est égale à l'expression suivante : $\lceil (6 \times 1 + 6 \times 1 + 3 \times 3)/(6 \times 3) \rceil = \lceil 21/18 \rceil = 2$. Si

notre méthode était utilisée, la nouvelle instance créée serait la suivante : $A' = \{(6,3),(3,6),(4,1),(4,1),(3,3),(3,3),(1,4),(1,4)\}$ et $B' = (6,6)$. La valeur obtenue par notre borne est égale à un bin, tandis que la borne inférieure calculée par la méthode de [6] est égale à deux.

Le lemme 3.1 ne peut pas être appliqué dans le cas des DDFF, puisque les DDFF ne sont pas superadditives. Les borne obtenues en utilisant f_1^k dans le schéma de [19] peuvent alors aboutir à des résultats qui ne sont pas dominés parL_{CJE}^R. Cependant, des résultats expérimentaux que nous avons obtenus n'ont mené à aucune amélioration sur notre méthode quand f_1^k était utilisée sur des instances transformées en carrés.

3.5 Résultats expérimentaux

Nous avons testé nos algorithmes sur des benchmarks proposés dans la littérature [4, 48] et décrits dans le chapitre 2.

Dans le tableau 3.1, nous comparons nos résultats avec ceux de [6], qui sont les meilleurs résultats connus dans la littérature. Chaque ligne représente la moyenne obtenue pour un groupe de dix instances d'une classe donnée, tandis que les lignes intitulées Avg. sont consacrées aux moyennes par classe. Le tableau 3.1 montre les valeurs des bornes inférieures obtenues (colonne LB). Dans la colonne CPU nous montrons en secondes le temps CPU consommé pour le calcul de la borne inférieure. La colonne intitulée BM concerne les résultats de [6], tandis que les colonnes intitulées CJE concernent les résultats obtenus par notre nouvelle borne quand la version modifiée de L_{CCM}^O [10] est utilisée (*cf.* Section 3.3).

Nous constatons que la nouvelle borne inférieure (L_{CJE}^R) fournit des résultats strictement meilleurs que ceux obtenus par la borne de [6]. Les résultats théoriques sont confirmés par les résultats expérimentaux.

3.6 Conclusion

Nous avons proposé dans ce chapitre un nouveau schéma de calcul de bornes inférieures pour le problème de bin-packing dans le cas non-orienté. Ce schéma consiste à composer une instance $2BP|O$ à partir d'une instance $2BP|R$. En utilisant une borne inférieure interne efficace pour le $2BP|O$, nous avons obtenue une borne inférieure pour le $2BP|R$ qui domine les bornes de la littérature dans le cas des bins carrés. Les résultats expérimentaux confirment la dominance théorique. Ce travail a été accepté pour une publication internationale [16].

Notre schéma de calcul peut être amélioré en tenant compte de la présence des objets à orientation fixe dans l'instance. Cette spécificité n'est pas prise en compte

Problème			LB		CPU
Class	$H \times W$	n	BM	CJE	CJE
I	10×10	20	6.6	6.6	0.00
		40	12.7	12.8	0.00
		60	19.5	19.5	0.00
		80	26.9	27.0	0.00
		100	31.0	31.3	0.00
		Avg.	19.34	19.44	0.000
II	30×30	20	1.0	1.0	0.00
		40	1.9	1.9	0.00
		60	2.5	2.5	0.00
		80	3.1	3.1	0.00
		100	3.9	3.9	0.00
		Avg.	2.48	2.48	0.000
III	40×40	20	4.7	4.7	0.00
		40	9.0	9.1	0.00
		60	13.2	13.2	0.00
		80	18.1	18.2	0.01
		100	21.5	21.5	0.02
		Avg.	13.30	13.34	0.006
IV	100×100	20	1.0	1.0	0.00
		40	1.9	1.9	0.00
		60	2.3	2.3	0.00
		80	3.0	3.0	0.00
		100	3.7	3.7	0.01
		Avg.	2.38	2.38	0.002
V	100×100	20	5.9	5.9	0.00
		40	11.3	11.3	0.00
		60	16.9	17.1	0.02
		80	23.5	23.6	0.04
		100	27.2	27.2	0.09
		Avg.	16.96	17.02	0.030
VI	300×300	20	1.0	1.0	0.00
		40	1.5	1.5	0.00
		60	2.1	2.1	0.00
		80	3.0	3.0	0.03
		100	3.2	3.2	0.04
		Avg.	2.16	2.16	0.014
VII	100×100	20	4.7	4.7	0.00
		40	9.7	9.9	0.01
		60	14.0	14.0	0.03
		80	19.8	20.0	0.06
		100	23.9	23.9	0.11
		Avg.	14.42	14.50	0.042
VIII	100×100	20	4.9	5.0	0.00
		40	9.6	9.7	0.01
		60	14.1	14.2	0.03
		80	19.7	19.7	0.06
		100	24.2	24.2	0.11
		Avg.	14.50	14.56	0.042
IX	100×100	20	14.3	14.3	0.00
		40	27.5	27.5	0.00
		60	43.5	43.5	0.01
		80	57.3	57.3	0.02
		100	69.3	69.3	0.03
		Avg.	42.38	42.38	0.012
X	100×100	20	3.9	3.9	0.00
		40	6.9	7.0	0.00
		60	9.4	9.5	0.02
		80	12.2	12.2	0.04
		100	15.3	15.3	0.07
		Avg.	9.54	9.58	0.026

Tableau 3.1 – Comparaison avec la borne de [6]

actuellement dans notre méthode ce qui pourrait l'affaiblir dans le cas de bins non carrés.

La méthode proposée peut être adaptée au problème de bin-packing en trois dimensions. Dans ce cas, l'instance à construire serait composée de $6n$ objets.

Nouvelle heuristique : IMA

4.1 Introduction

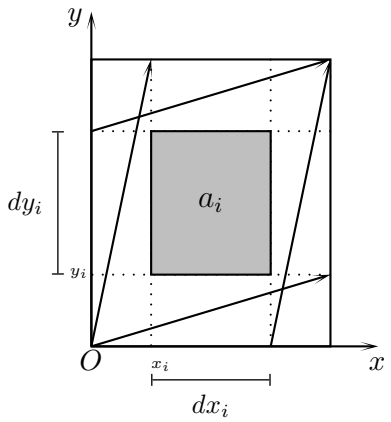
Comme nous l'avons vu au chapitre 1, plusieurs stratégies de rangement ont été proposées dans la littérature pour la résolution des problèmes de bin-packing. Il s'agit des stratégies Next Fit, First Fit, Best Fit, Worst Fit et Any Fit. Dans ce chapitre, nous proposons une nouvelle heuristique de résolution basée sur la stratégie "Best Fit" adaptée au problème en deux dimensions.

Plusieurs heuristiques ont été proposées dans la littérature pour le $2BP|R$ (*cf.* Section 1.6). La plupart de ces heuristiques consistent à affecter les objets aux bins dans un ordre prédéfini. Les critères de choix de placement des objets varient d'une méthode à l'autre. Dans la nouvelle heuristique proposée (IMA), aucun pré-ordre des objets n'est nécessaire. A chaque étape de rangement d'un objet, on cherche le "meilleur" couple : objet à ranger dans une orientation donnée et sa position, parmi tous les couples réalisables.

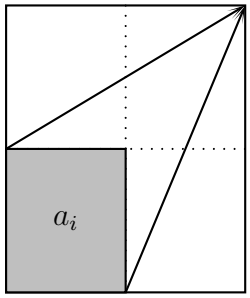
Le chapitre comporte quatre sections. Dans un premier temps (Section 4.2), nous introduisons la notion de *surfaces maximales*. Cette notion nous permet de considérer toutes les surfaces vides dans le bin sans redondance ni perte de surface. Nous présentons plusieurs propriétés propres à ces surfaces. Puis, nous abordons la notion des *rangements stables* et discutons du nombre maximal de surfaces présentes dans un rangement stable (Section 4.3). La section 4.4 est consacrée à la nouvelle heuristique IMA basée sur le choix du meilleur couple (objet à placer, surface maximale accueillante) à chaque étape du rangement d'un objet. Nous proposons l'algorithme de résolution correspondant et discutons du critère de choix du meilleur couple. Finalement, nous testons notre heuristique sur des benchmarks de la littérature (Section 4.5). Nous comparons l'heuristique IMA avec les méthodes de résolution approchée les plus récentes de la littérature et nous montrons son efficacité.

4.2 Placement des objets, surfaces maximales

La résolution d'une instance donnée $I = (A, B)$ d'un problème de bin-packing en deux dimensions, dans le cas non-orienté, consiste à placer les différents objets de



(a) Objet placé dans une position quelconque



(b) Objet placé dans le coin bas-gauche

Figure 4.1 – Les surfaces maximales résultantes du placement du premier objet dans le bin, chaque surface étant représentée par sa diagonale respective.

A dans les bins en associant à chaque objet a_i de A un bin dans lequel a_i est rangé, sa position (x_i, y_i) dans le bin et son orientation. Dans cette section, nous décrivons la représentation d'une solution. Le placement des objets est basé sur la notion de *surfaces maximales* introduite dans cette section. Nous discutons aussi de la mise à jour de la liste des surfaces maximales après chaque rangement d'un objet.

Pour chaque objet a_i placé dans un bin dans une orientation donnée, nous notons (x_i, y_i) sa position (les coordonnées de son coin bas-gauche) par rapport au coin bagauche du bin, le centre du repère orthogonal (x,y). Nous notons par dx_i et dy_i les projections respectives des arêtes de l'objet sur les axes Ox et Oy. Ainsi, $dx_i = w_i$ et $dy_i = h_i$ si a_i est placé dans son orientation initiale, ou $dx_i = h_i$ et $dy_i = w_i$ si a_i est tourné de 90 degrés (voir figure 4.1(a)).

Définition 4.1. *Etant donnée une liste de surfaces vacantes, une surface est dite maximale si elle n'est contenue en totalité dans aucune autre surface.*

Les surfaces maximales possèdent les deux propriétés suivantes. Ces deux propriétés sont des conséquences directes de la définition 4.1.

Propriété 4.1. *Chacun des quatre bords d'une surface maximale donnée dans un bin touche au moins un autre objet déjà placé ou bien une des quatre arêtes du bin.*

Propriété 4.2. *Deux surfaces maximales dans un même bin ayant leurs coins bas-gauche à la même position, ne peuvent avoir ni la même largeur ni la même hauteur. De plus, étant données deux surfaces maximales ma_1 et ma_2 qui ont leurs coins bas-gauche à la même position, si la hauteur (resp. largeur) de ma_1 est supérieure à la hauteur (resp. largeur) de ma_2, alors la largeur (resp. hauteur) de ma_1 est inférieure à la largeur (resp. hauteur) de ma_2.*

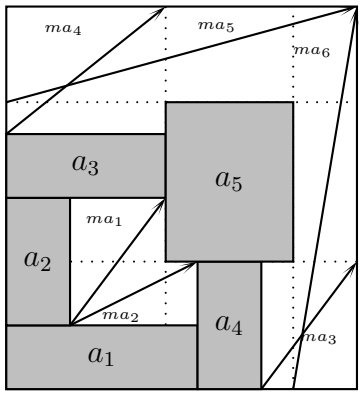


Figure 4.2 – Placement de cinq objets dans le bin, les six surfaces maximales résultantes sont représentées par leur diagonale

Lors du processus de rangement des objets, les surfaces résiduelles sont gérées par une liste de *surfaces maximales*. Initialement, la liste des surfaces maximales d'un bin ouvert vide est tout simplement constitué de la surface du bin. Le premier objet à rangé est placé dans une surface maximale correspondante au premier bin ouvert. Quand on place un objet dans une surface, cette surface est décomposée en quatre nouvelles surfaces au plus : la surface résiduelle à gauche, à droite, au dessus et en dessous de l'objet. Ce cas de figure est illustré dans la figure 4.1(a). Dans toutes les méthodes de résolution proposées dans la présente thèse, lors du rangement des objets, ils sont placés de telle sorte que les coins bas-gauches des objets placés coïncident avec les coins bas-gauches des surfaces qui les accueillent. Ainsi, le placement du premier objet dans la surface disponible (le bin) entraîne l'élimination de cette surface et la création de deux surfaces au plus (*cf.* figure 4.1(b)). A une étape donnée du packing un nouvel objet est placé dans une surface maximale qui peut le contenir. La surface dans laquelle l'objet a été placé est alors éliminée et la liste des surfaces disponibles est mise à jour. Seules les surfaces maximales sont maintenues.

Dans la figure 4.2 nous montrons un rangement intermédiaire où cinq objets $(a_1, ..., a_5)$ sont placés dans un bin. Les surfaces maximales $(ma_1, ..., ma_6)$ résultantes dans le bin sont représentées par leur diagonale.

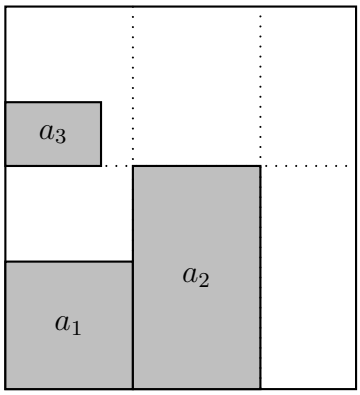


Figure 4.3 – Rangement non stable

4.3 Les rangements stables

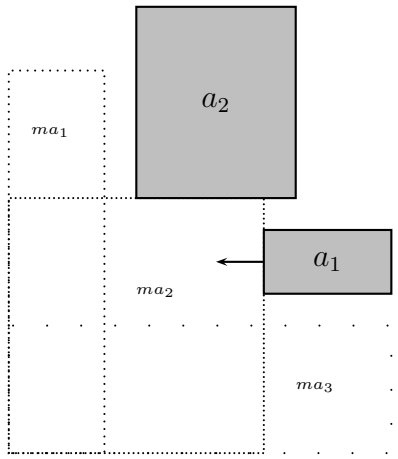
Lors du placement d'un objet dans une surface maximale donnée, deux cas se présentent. Dans le premier cas, l'objet est soutenu par ses bords bas et gauche, il est dit dans ce cas rangé dans une position bl-stable. Dans le second cas, au moins un de ses deux bords n'est pas soutenu. Un rangement des objets est dit *stable* si à chaque stade de rangement des objets, seules les positions bl-stables sont considérées. Un exemple de rangement non stable est illustré dans la figure 4.3. L'objet a_3 dans la figure n'est pas placé dans une position bl-stable car son arête du bas n'est soutenu ni par autre objet ni par une des arêtes du bin. Le rangement illustré dans la figure 4.2 présente un packing stable. Dans l'algorithme de résolution proposé dans la section 4.4, seuls les rangements stables sont considérés.

Lors d'un rangement stable, un nouvel objet est alors placé dans une surface maximale qui peut le contenir et qui préserve un rangement stable. La surface dans laquelle l'objet a été placé est alors éliminée et la liste des surfaces disponibles est mise à jour.

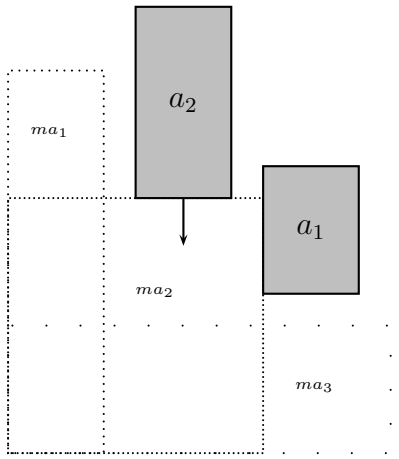
Nous allons maintenant montrer, en utilisant les propriétés 4.1 et 4.2 décrites dans la section précédente, la proposition suivante :

Proposition 4.1. *Le nombre de surfaces maximales à une étape quelconque d'un rangement stable est en* $O(n^2)$.

Démonstration. D'après la propriété 4.1, chaque coin bas-gauche d'une surface maximale est nécessairement issu de la projection de deux arêtes d'objets ou la projection d'une arête d'un objet sur une arête du bin. Nous déduisons donc, que le nombre de coins bas-gauches des surfaces maximales présentes dans un bin l est inférieur ou égal à $C^2_{n'} \leq n'^2$ où $n' = 4(n_l + 1)$, n_l étant le nombre d'objets placés dans le bin



(a) Premier cas, a_1 n'est pas placé dans une position bl-stable.



(b) Second cas, a_2 n'est pas placé dans une position bl-stable.

Figure 4.4 – Illustration de trois surfaces maximales associées au même coin bas-gauche.

l. Nous montrons maintenant que, dans le cas d'un rangement stable, on ne peut associer que deux surfaces maximales au plus à chaque coin bas-gauche.

Supposons que trois surfaces maximales ma_1, ma_2 et ma_3, dans un rangement stable, soient associées au même coin bas-gauche et montrons que c'est une contradiction. Soient a_1 et a_2 deux objets qui touchent respectivement les bords droit et haut de ma_2. La figure 4.4 illustre les deux cas de figures possibles considérant les objets a_1 et a_2 sans chevauchement. Les allures considérées des trois surfaces maximales sont déduites de la propriété 4.2. Nous pouvons facilement constater que, pour les deux cas de figures, le rangement n'est pas stable. Dans le premier cas (*cf.* figure 4.4(a)), la stabilité gauche de l'objet a_1 n'est pas respectée, et dans l'autre cas (*cf.* figure 4.4(b)) la stabilité bas de l'objet a_2 n'est pas respectée. Nous déduisons que le nombre de surfaces maximales dans un bin l est en $O(n_l^2)$. Soit $nbBin$, le nombre de bins utilisés pour ranger l'ensemble de tous les objets d'une instance donnée. Le nombre total des surfaces maximales est alors en $\sum_{l=1}^{nbBin} O(n_l^2)$ qui est en $O(n^2)$ car $\sum_{l=1}^{nbBin} n_l = n$. □

4.4 L'heuristique (IMA)

Dans cette section, nous proposons une nouvelle heuristique pour la résolution des instances $2BP|R$. Dans les méthodes heuristiques proposées dans la littérature, les auteurs procèdent à un tri des objets puis ils les placent par ordre dans les bins suivant différentes stratégies (*cf.* Section 1.6). Dans la nouvelle heuristique proposée aucun pré-ordre des objets n'est imposé. Le choix de l'objet à placer, dans une

étape donnée du rangement, dépend des caractéristiques des surfaces maximales disponibles et des critères considérés.

Ainsi, dans la nouvelle heuristique appelée IMA *(Item Maximal Area)*, à chaque étape du rangement d'un nouvel objet, un choix est fait sur le couple : (l'objet à placer dans une orientation donnée, la surface maximale qui va l'accueillir). A chaque couple (objet, surface) réalisable correspond un seul placement (le coin bas-gauche de l'objet coïncide avec celui de la surface maximale qui va le contenir). Le choix du couple est géré par un certain critère qui dépend des caractéristiques des objets et des surfaces considérées. Cette stratégie du rangement peut être considérée comme une stratégie Best-Fit adaptée au $2BP|R$.

L'heuristique IMA est donné dans l'algorithme 2. Etant donnée une instance $I = (A, B)$ à résoudre, nous considérons deux listes différentes. La première liste A' contient les objets non rangés. Initialement A' vaut A, l'ensemble de tous les objets constituant l'instance I. La deuxième liste Lma est la liste des surfaces maximales disponibles. Elle est initialement constituée d'une seule surface : la surface du premier bin. A chaque étape de placement d'un objet, un choix est fait pour choisir le couple (objet à ranger, la surface maximale acceuillante) des listes A' et Lma, ainsi que l'orientation de l'objet. Ce choix est déterminé à l'aide d'un critère heuristique qui prend en considération les caractéristiques des objets et des surfaces maximales. Soit simultanément $a_i \in A'$ un objet non rangé considéré dans une orientation déterminée, et $ma \in Lma$, une surface maximale disponible qui peut contenir a_i suivant l'orientation considérée et dans une position bl-stable. Soient w_{ma} et h_{ma} les projections des arêtes de ma sur les axes x et y successivement. Notons q_1, q_2, q_3 et q_4 quatre nombres réels tel que :

$$0 \leq q_k \leq 1; k = 1, ..., 4 \text{ et } \sum_{k=1,...,4} q_k = 1$$

Le critère peut alors être écrit de la façon suivante :

$$\begin{aligned} \mathcal{O}(a_i, ma) = & q_1(w_i h_i)/(w_{ma} h_{ma}) & + \\ & q_2(dx_i/w_{ma}) & + \\ & q_3(dy_i/h_{ma}) & + \\ & q_4(w_i^2 + h_i^2)/(w_{ma}^2 + h_{ma}^2) \end{aligned}$$

Le couple (objet à placer, surface maximale accueillante) est alors le couple de (A', Lma) qui maximise le critère cité ci dessus. Les valeurs données aux paramètres $q_1, ..., q_4$ influencent fortement la solution finale fournie. Par exemple, si la valeur donnée à q_1 est assez grande (proche de 1), le couple (a^*, ma^*) désigné serait celui qui présente des surfaces les plus proches. Les termes pondérés par q_1 et q_4 sont

indépendants de l'orientation des objets, tandis que l'orientation considérée des objets affecte les termes pondérés par q_2 et q_3.

Une fois le couple (a^*, ma^*) et l'orientation de a^* sont choisis, l'objet est placé dans ma^* et la liste Lma est mise à jour. La procédure est répétée jusqu'au rangement de tous les objets. Si le nombre de bins utilisés est inférieur à la solution record $nbBinMin$, celle ci est alors mise à jour. La procédure est répétée après un changement des valeurs des paramètres q_k suivant les pas $step1$, $step2$ et $step3$ déterminés expérimentalement.

Pour déterminer la complexité d'une itération de l'heuristique IMA donnée dans l'algorithme 2, notons C le nombre maximal de surfaces maximales présentes dans la liste Lma à une étape quelconque du rangement. Lors du placement d'un objet, le coût de la désignation du couple (a^*, ma^*) est en $O(nC)$. Placer l'objet a^* dans ma^* et mettre à jour la liste Lma est en $O(C^2)$. Par conséquent, la complexité globale pour ranger les n objets d'une instance I est en $O(n^2C + nC^2)$. Or, C est en $O(n^2)$ (*cf.* proposition 4.1), la complexité temporelle de l'heuristique IMA est alors en $O(n^5)$. Nous montrons dans la section 4.5 que, pour les jeux d'essais de la littérature, le nombre de surfaces maximales ne dépasse jamais n, ce qui explique le temps CPU avantageux de notre heuristique.

4.5 Résultats expérimentaux

Nous avons testé l'heuristique IMA sur les jeux d'essai de la littérature proposés dans [4] et [48] et décrits dans la section 2.4. Nous avons comparé notre heuristique avec les résultats les plus récents de la littérature. Nous rappelons que le benchmark utilisé est composé de dix classes d'objets générées aléatoirement et que chaque classe est composée en cinq groupes, chaque groupe ayant une taille d'instance différente.

Dans le tableau 4.1, nous testons l'heuristique IMA sur les instances du benchmark considéré. Nous montrons les résultats de deux versions : avec et sans prétraitements. Dans le premier cas, les prétraitements α, β et γ proposés dans le chapitre 2 sont appliqués l'un après l'autre tant que l'instance est modifiée, l'heuristique IMA est ensuite appliquée à l'instance transformée. Les résultats concernent le nombre de bins utilisés pour ranger l'ensemble des objets ($nbBin$), et le temps CPU correspondant en secondes (CPU). Nous notons MA_{max} le nombre maximum de surfaces maximales présentes en une étape quelconque du rangement, tandis que $MA_{average}$ présente la moyenne de cette valeur (MA_{max}) relevée à chaque accès à la liste des surfaces maximales. Chaque ligne du tableau présente la moyenne de ces valeurs sur les dix instances constituant un groupe d'une classe donnée sauf pour la valeur MA_{max} qui correspond à la valeur maximale. Le tableau 4.1 permet de constater que, pour les deux versions testées (avec et sans prétraitements), le nombre

de surfaces maximales ne dépasse en pratique jamais la taille n de l'instance quelle que soit l'instance du benchmark considéré. Cela explique la rapidité de l'heuristique proposée. Nous constatons aussi que la résolution du problème prétraité est plus rapide que la résolution du problème initial pour la plupart des instances testées. Notons aussi que la solution obtenue par la version avec prétraitement est meilleur que celle sans prétraitement pour un groupe d'instances dans la classe 3 ($n = 80$).

Dans le tableau 4.2, nous comparons l'heuristique proposée avec les méthodes de résolution approchée les plus récentes de la littérature : la méthode tabou proposée dans [45] notée $TS-TP$, l'heuristique itérative HBP proposée dans [6] et la méthode tabou $ATS - BP$ proposée dans [38]. Pour chaque méthode, le rapport G entre le nombre de bins obtenus par la méthode de résolution et la borne inférieure proposée dans [19] est utilisée pour mesurer les performances de $TS - TP$ et de $ATS - BP$ (résultats issus de [45] et [38]). Tandis que la borne inférieure proposée dans [6] est utilisée dans le rapport G concernant l'heuristique HBP, la même borne inférieure est utilisée, dans les résultats affichés pour IMA, appliquée aux objets prétraités. Les colonnes CPU représentent les temps CPU (en secondes) correspondant à chaque algorithme de résolution. Notons que l'algorithme $TS - TP$ a été exécuté sur un processeur "Silicon Graphics INDY R10000sc 195Mhz", HBP a été exécuté sur un processeur "Pentium 3 Intel 933 MHz", et IMA a été exécuté sur un processeur "Pentium 4 Intel 2.66 GHz". En se référant au tableau 4.2, nous constatons qu'avec des temps CPU très compétitifs, le rapport obtenu en utilisant l'heuristique IMA est meilleur que celui des trois autres méthodes pour toutes les classes de ce benchmark.

Pour pouvoir comparer les résultats obtenus par les différentes méthodes de résolution indépendamment de la borne inférieure utilisée, nous avons besoin des valeurs des nombres de bins utilisés dans chaque méthode. Ce résultat n'est pas communiqué dans les articles correspondants. Par contre, grace aux résultats communiqués dans le mémoire du doctorat de Harwig et aux résultats que Boschetti nous a envoyés, nous pouvons comparer les nombres de bins utilisés par chacune des trois méthodes de résolution suivantes : IMA, HBP et $ATS - BP$ (*cf.* tableau 4.3). Les résultats montrent que pour les classes 1 et 3 l'heuristique HBP fournit des résultats meilleurs en moyenne comparée à $ATS - BP$. Pour toutes les autres classes les résultats obtenus par $ATS - BP$ sont meilleurs ou équivalents aux ceux du HBP.

L'heuristique IMA que nous avons proposée fournit sur ces instances des résultats qui dominent les heuristiques de la littérature. La méthode tabou $ATS-BP$ de la littérature ne produit de meilleurs résultats que pour une unique classe (la classe 10). Les résultats obtenus par IMA sont meilleurs pour les autres classes, malgré le temps de calcul beaucoup plus important pour la méthode tabou.

4.6 Conclusion

Dans ce chapitre, nous avons proposé une nouvelle heuristique de résolution des instances $2BP|R$. Les positions d'insertions possibles des objets sont définies par des surfaces vides non redondantes appelées surfaces maximales. L'heuristique IMA choisit, de façon adéquate, à chaque étape de la procédure du rangement des objets, l'objet à placer, son orientation et la surface maximale dans laquelle il va être placé. Les résultats expérimentaux montrent l'efficacité de notre heuristique par rapport aux méthodes de résolution approchée proposées dans la littérature. Ce chapitre correspond au travail soumis à une revue internationale [22] et présenté dans un congrès [25].

Algorithme 2 : L'heuristique IMA

Données : $I = (A, B)$ (une instance $2BP|R$), $step1$, $step2$ et $step3$ (trois nombres réels).
nbBinMin := n ;
pour *(*$q_1 := 0$, $q_1 \leq 1$ *;* $q_1 := q_1 + step1$*)* **faire**
  pour *(*$q_2 := 0$, $q_2 \leq 1 - q_1$ *;* $q_2 := q_2 + step2$*)* **faire**
    pour *(*$q_3 := 0$, $q_3 \leq (1 - q_1 - q_2)$ *;* $q_3 := q_3 + step3$*)* **faire**
      $q_4 := 1 - (q_1 + q_2 + q_3)$;
      $A' := A$;
      Initialisation de la liste Lma des surfaces maximales disponibles. Initialement elle contient une seule surface maximale : un bin vide à la position $(0, 0)$, l'origine des axes x et y ;
      $nbBin := 1$;
      tant que $(|A'| \neq 0)$ **faire**
        rapMax := 0 ; **pour** $(i := 1; i \leq |A'|; i++)$ **faire**
          pour $(j := 1; j \leq |Lma|; j++)$ **faire**
            si (a_i *peut être placé dans* ma_j *sans chevauchement et* $\mathcal{O}(a_i, ma_j) > rapMax$) **alors**
              $rapMax = \mathcal{O}(a_i, ma_j)$; $a^* = a_i$; $ma^* = ma_j$; (les deux orientations de a_i sont testées)
        si $(rapMax > 0)$ **alors**
          placer l'objet a^* dans la surface maximale ma^*, et mettre à jour la liste Lma ;
          $A' = A' \setminus \{a^*\}$;
          sinon
            réinitialiser Lma avec un bin ; $nbBin := nbBin + 1$;
      si $(nbBin < nbBinMin)$ **alors**
        $nbBinMin := nbBin$;
retourner nbBinMin ;

Problème			avec prétraitement				sans prétraitement			
Classe	HxW	n	$nbBin$	CPU	MA_{max}	MA_{avg}	$nbBin$	CPU	MA_{max}	MA_{avg}
I	10x10	20	6.6	0,003	3.0	2.3	6.6	0,001	3.0	2.9
		40	12.9	0,067	4.0	2.7	12.9	0,078	4.0	2.9
		60	19.5	0,001	3.0	2.4	19.5	0,006	4.0	2.8
		80	27.0	0,110	4.0	2.4	27.0	0,123	4.0	2.9
		100	31.3	0,321	6.0	3.6	31.3	0,361	6.0	3.7
II	30x30	20	1.0	0,001	11.0	8.9	1.0	0,001	11.0	9.4
		40	1.9	0,003	10.0	7.5	1.9	0,001	10.0	8.3
		60	2.5	0,000	12.0	9.0	2.5	0,006	10.0	8.6
		80	3.1	0,015	11.0	8.8	3.1	0,007	14.0	8.6
		100	3.9	0,024	11.0	9.2	3.9	0,026	10.0	8.5
III	40x40	20	4.7	0,001	5.0	3.4	4.7	0,000	5.0	3.9
		40	9.4	0,126	7.0	3.9	9.4	0,143	7.0	4.5
		60	13.5	0,242	8.0	4.5	13.5	0,242	8.0	4.8
		80	18.4	0,359	6.0	4.3	18.5	0,509	6.0	4.9
		100	22.2	1,173	9.0	5.2	22.2	1,254	9.0	5.7
IV	100x100	20	1.0	0,000	17.0	10.2	1.0	0,001	17.0	10.3
		40	1.9	0,001	15.0	10.8	1.9	0,006	15.0	10.7
		60	2.5	0,432	18.0	11.4	2.5	0,459	18.0	10.9
		80	3.1	0,323	14.0	11.2	3.1	0,318	14.0	11.0
		100	3.7	0,032	16.0	10.7	3.7	0,040	16.0	10.9
V	100x100	20	5.9	0,000	5.0	2.6	5.9	0,000	5.0	3.9
		40	11.4	0,067	7.0	2.8	11.4	0,071	7.0	4.4
		60	17.4	0,334	6.0	3.3	17.4	0,429	6.0	4.7
		80	23.9	0,228	6.0	3.4	23.9	0,400	6.0	4.7
		100	27.9	1,154	7.0	4.7	27.9	1,298	7.0	5.5
VI	300x300	20	1.0	0,000	12.0	10.5	1.0	0,000	12.0	10.5
		40	1.7	0,304	25.0	15.4	1.7	0,312	25.0	15.4
		60	2.1	0,009	20.0	13.8	2.1	0,006	20.0	12.8
		80	3.0	0,012	14.0	12.0	3.0	0,015	15.0	12.4
		100	3.2	0,300	16.0	13.9	3.2	0,482	16.0	12.8
VII	100x100	20	5.2	0,065	5.0	4.0	5.2	0,079	5.0	4.3
		40	10.4	0,276	7.0	5.0	10.4	0,309	7.0	4.9
		60	14.7	0,537	7.0	5.6	14.7	0,615	7.0	5.5
		80	21.2	1,228	7.0	5.6	21.2	1,317	7.0	5.6
		100	25.3	1,789	7.0	5.3	25.3	1,928	7.0	5.3
VIII	100x100	20	5.3	0,057	6.0	4.0	5.3	0,063	5.0	4.0
		40	10.4	0,342	7.0	5.2	10.4	0,354	7.0	5.2
		60	15.0	0,632	7.0	5.4	15.0	0,704	7.0	5.4
		80	20.8	1,127	7.0	5.3	20.8	1,168	7.0	5.3
		100	25.7	1,812	9.0	5.8	25.7	1,904	9.0	5.8
IX	100x100	20	14.3	0,000	0.0	0.0	14.3	0,001	4.0	2.4
		40	27.5	0,000	1.0	0.6	27.5	0,003	4.0	2.5
		60	43.5	0,000	1.0	0.5	43.5	0,004	4.0	2.9
		80	57.3	0,000	1.0	0.8	57.3	0,012	4.0	3.2
		100	69.3	0,000	1.0	0.7	69.3	0,014	4.0	2.7
X	100x100	20	4.1	0,039	9.0	3.9	4.1	0,040	9.0	5.2
		40	7.3	0,120	6.0	4.5	7.3	0,146	7.0	5.4
		60	10.1	0,473	8.0	5.9	10.1	0,533	8.0	5.9
		80	12.8	0,786	8.0	6.8	12.8	0,876	9.0	6.8
		100	15.8	1,001	9.0	6.9	15.8	1,089	10.0	7.1

Tableau 4.1 – L'heuristique IMA.

Problème			IMA		$TS-TP$		HBP		$ATS-BP$	
Classe	WxH	n	G	CPU	G	CPU	G	CPU	G	CPU
I	10x10	20	1,0000	0,003	1.05	18.00	1.0000	0.005	1.03	0.13
		40	1,0157	0,067	1.04	30.02	1.0162	0.127	1.04	4.11
		60	1,0000	0,001	1.04	34.00	1.0000	0.028	1.04	4.27
		80	1,0037	0,110	1.06	48.08	1.0036	0.253	1.06	0.97
		100	1,0064	0,321	1.03	47.92	1.0093	1.011	1.03	1.20
		Average	1,0052	0,100	1.044	35.60	1.0058	0.285	1.040	2.14
II	30x30	20	1,0000	0,001	1.00	0.01	1.0000	0.005	1.00	0.13
		40	1,0000	0,003	1.10	0.01	1.0000	0.226	1.00	1.46
		60	1,0000	0,000	1.00	0.01	1.0000	0.021	1.00	1.14
		80	1,0000	0,015	1.03	6.10	1.0000	0.336	1.00	22.61
		100	1,0000	0,024	1.00	0.01	1.0000	0.054	1.00	4.25
		Average	1,0000	0,009	1.026	1.23	1.0000	0.128	1.000	5.92
III	40x40	20	1,0000	0,001	1.06	18.00	1.0000	0.017	1.02	11.35
		40	1,0330	0,126	1.09	42.17	1.0488	0.361	1.09	0.38
		60	1,0227	0,242	1.08	54.15	1.0241	0.599	1.08	64.85
		80	1,0166	0,359	1.07	60.07	1.0178	1.104	1.07	9.94
		100	1,0326	1,173	1.07	60.18	1.0354	2.817	1.06	68.85
		Average	1,0210	0,380	1.074	46.91	1.0252	0.980	1.064	30.94
IV	100x100	20	1,0000	0,000	1.00	0.01	1.0000	0.006	1.00	0.14
		40	1,0000	0,001	1.00	0.01	1.0000	0.027	1.00	0.25
		60	1,0870	0,432	1.10	0.09	1.1000	2.775	1.05	28.80
		80	1,0333	0,323	1.07	12.00	1.0667	4.081	1.07	36.16
		100	1,0000	0,032	1.03	6.00	1.0333	3.647	1.03	9.19
		Average	1,0241	0,158	1.040	3.62	1.0400	2.107	1.030	14.91
V	100x100	20	1,0000	0,000	1.04	12.01	1.0000	0.011	1.04	1.31
		40	1,0088	0,067	1.07	42.00	1.0254	0.330	1.06	41.21
		60	1,0296	0,334	1.06	45.23	1.0393	1.790	1.06	74.59
		80	1,0127	0,228	1.07	54.14	1.0213	2.238	1.06	143.01
		100	1,0257	1,154	1.07	60.12	1.0325	5.471	1.05	196.23
		Average	1,0154	0,357	1.062	42.70	1.0237	1.968	1.054	91.27
VI	300x300	20	1,0000	0,000	1.00	0.01	1.0000	0.017	1.00	0.17
		40	1,1333	0,304	1.40	0.03	1.2000	3.368	1.10	5.97
		60	1,0000	0,009	1.05	0.05	1.0000	0.207	1.00	25.92
		80	1,0000	0,012	1.00	0.01	1.0000	0.308	1.00	0.42
		100	1,0000	0,300	1.07	12.00	1.0667	11.089	1.07	7.64
		Average	1,0267	0,125	1.104	2.42	1.0533	2.998	1.034	8.02
VII	100x100	20	1,1064	0,065	1.11	30.00	1.1100	0.198	1.11	0.77
		40	1,0722	0,276	1.08	48.06	1.0833	1.142	1.07	99.28
		60	1,0500	0,537	1.06	59.45	1.0769	3.027	1.04	613.98
		80	1,0653	1,228	1.10	60.12	1.1008	5.356	1.08	259.78
		100	1,0586	1,789	1.08	60.36	1.0841	8.133	1.07	231.66
		Average	1,0705	0,779	1.086	51.60	1.0910	3.571	1.074	241.09
VIII	100x100	20	1,0816	0,057	1.10	30.01	1.0800	0.182	1.10	1.34
		40	1,0833	0,342	1.10	54.22	1.0929	1.280	1.08	77.91
		60	1,0638	0,632	1.07	56.17	1.0923	3.158	1.07	51.74
		80	1,0558	1,127	1.08	60.11	1.0816	5.439	1.07	529.40
		100	1,0620	1,812	1.09	60.14	1.0868	8.150	1.08	232.88
		Average	1,0693	0,794	1.088	52.13	1.0867	3.642	1.080	178.65
IX	100x100	20	1,0000	0,000	1.00	0.06	1.0000	0.017	1.00	0.26
		40	1,0000	0,000	1.01	18.85	1.0000	0.077	1.01	1.25
		60	1,0000	0,000	1.01	18.03	1.0000	0.199	1.01	2.54
		80	1,0000	0,000	1.01	30.51	1.0000	0.389	1.01	1.65
		100	1,0000	0,000	1.01	36.86	1.0000	0.568	1.01	52.69
		Average	1,0000	0,000	1.008	20.86	1.0000	0.250	1.008	11.68
X	100x100	20	1,0513	0,039	1.12	6.01	1.1000	0.125	1.12	0.36
		40	1,0429	0,120	1.06	24.01	1.0486	0.445	1.06	3.93
		60	1,0632	0,473	1.06	30.44	1.0672	2.094	1.06	235.17
		80	1,0492	0,786	1.05	39.04	1.0492	3.363	1.05	78.27
		100	1,0327	1,001	1.05	43.38	1.0455	5.540	1.04	269.68
		Average	1,0478	0,484	1.068	28.58	1.0621	2.313	1.066	117.48

Tableau 4.2 – Comparaison du rapport borne supérieure sur borne inférieure.

Problème			nombre de bins		
Classe	WxH	n	IMA	HBP	$ATS-BP$
I	10x10	20	6.6	6.6	6.6
		40	12.9	12.9	12.9
		60	19.5	19.5	19.5
		80	27.0	27.0	27.0
		100	31.3	31.3	31.4
		Average	19.46	19.46	19.48
II	30x30	20	1.0	1.0	1.0
		40	1.9	1.9	1.9
		60	2.5	2.5	2.5
		80	3.1	3.1	3.1
		100	3.9	3.9	3.9
		Average	2.48	2.48	2.48
III	40x40	20	4.7	4.7	4.7
		40	9.4	9.4	9.4
		60	13.5	13.5	13.6
		80	18.4	18.4	18.6
		100	22.2	22.2	22.3
		Average	13.64	13.64	13.72
IV	100x100	20	1.0	1.0	1.0
		40	1.9	1.9	1.9
		60	2.5	2.5	2.4
		80	3.1	3.2	3.2
		100	3.7	3.8	3.8
		Average	2.44	2.48	2.46
V	100x100	20	5.9	5.9	5.9
		40	11.4	11.5	11.4
		60	17.4	17.5	17.5
		80	23.9	24.0	23.9
		100	27.9	28.0	28.0
		Average	17.30	17.38	17.34
VI	300x300	20	1.0	1.0	1.0
		40	1.7	1.7	1.6
		60	2.1	2.1	2.1
		80	3.0	3.0	3.0
		100	3.2	3.4	3.4
		Average	2.20	2.24	2.22
VII	100x100	20	5.2	5.2	5.2
		40	10.4	10.5	10.4
		60	14.7	15.1	14.6
		80	21.2	21.8	21.3
		100	25.3	25.9	25.5
		Average	15.36	15.70	15.40
VIII	100x100	20	5.3	5.3	5.3
		40	10.4	10.5	10.4
		60	15.0	15.4	15.1
		80	20.8	21.3	20.8
		100	25.7	26.3	26.0
		Average	15.44	15.76	15.52
IX	100x100	20	14.3	14.3	14.3
		40	27.5	27.5	27.6
		60	43.5	43.5	43.5
		80	57.3	57.3	57.3
		100	69.3	69.3	69.3
		Average	42.38	42.38	42.40
X	100x100	20	4.1	4.1	4.1
		40	7.3	7.3	7.3
		60	10.1	10.0	9.9
		80	12.8	12.8	12.8
		100	15.8	16.0	15.9
		Average	10.02	10.04	10.00

Tableau 4.3 – Comparaison du nombre de bins moyen utilisés pour ranger l'ensemble de tous les objets.

Nouvel algorithme tabou

5.1 Introduction

Les méta-heuristiques sont des approches de résolution efficaces pour un très grand nombre de problèmes difficiles de l'optimisation combinatoire, elles permettent en général d'obtenir de très bonnes solutions. L'efficacité de la méthode de recherche tabou a été vérifiée pour la résolution du problème de bin-packing en particulier dans [45] et [38]. Dans ce chapitre, nous proposons un nouvel algorithme tabou dans lequel nous évitons d'explorer des solutions à *configurations équivalentes*.

Nous synthétisons dans la section 5.2 les points essentiels dans une méthode tabou en général. Nous proposons par la suite un nouvel algorithme tabou pour la résolution du problème $2BP|R$ (Section 5.3). Nous commençons par la description du voisinage considéré. Nous introduisons ensuite la notion de *configurations équivalentes*. Puis nous présentons l'heuristique de base utilisée pour évaluer les solutions, la fonction d'évaluation utilisée, et le schéma du nouvel algorithme tabou. Dans la section 5.4, nous exposons les résultats numériques obtenus par le nouvel algorithme et nous les comparons avec les résultats de la littérature. Nous comparons aussi quelques variantes de notre méthode entre elles.

5.2 Les points essentiels dans une méthode tabou

En général, les méta-heuristiques sont utilisées dans la résolution approchée des problèmes combinatoires dans le but de trouver une solution de bonne qualité en un temps raisonnable. Une méta-heuristique, contrairement aux algorithmes gloutons, remet en question les choix pris au cours de la résolution du problème.

Certaines méthodes s'appuient sur l'étude de populations de solutions, comme par exemple les algorithmes génétiques. Cette méthode consiste à générer une nouvelle population à partir d'une population courante, en profitant des caractéristiques jugées bonnes dans les individus de celle ci. D'autres méta-heuristiques adaptent le comportement de certaines sociétés comme dans le cas des colonies de fourmis qui communiquent entre elles indirectement par une modification légère de leur environnement. D'autres méthodes, partant d'un point initial (solution courante

initiale) explorent les solutions voisines et mettent à jour la solution courante suivant une certaine stratégie dépendante de la méthode utilisée. C'est le cas des méthodes tabou et le recuit simulé (voir [1] pour une introduction plus complète sur les méta-heuristiques).

Etant donné un problème combinatoire à résoudre, soit $\mathcal{E}$ l'ensemble des solutions réalisables. Nous supposons, sans perte de généralité, que le problème à résoudre consiste à minimiser une fonction f dépendante de la solution :

$$\min_{S \in \mathcal{E}} f(S)$$

Nous décrivons brièvement dans la suite les notions de base d'une méthode tabou.

Transformation élémentaire et voisinage. Une transformation est définie de la façon suivante :

Définition 5.1. *Une transformation ρ est une opération qui permet de modifier une solution S de $\mathcal{E}$ en une autre solution $S' = \rho(S)$ de $\mathcal{E}$.*

Une transformation est dite *élémentaire* (ou locale) si elle ne modifie que *faiblement* la structure. Une transformation locale transforme une solution en une autre solution dans laquelle on peut reconnaître des éléments, ou une structure particulière propre à la solution initiale.

Le voisinage d'une solution donnée peut être défini à l'aide des transformations locales :

Définition 5.2. *Soit un ensemble de transformations locales $\mathcal{L}$. Le voisinage $\mathcal{N}(S)$ d'une solution S est l'ensemble des solutions que l'on peut obtenir en appliquant une fonction $\rho \in \mathcal{L}$ à S.*

Le choix du voisinage (la transformation élémentaire à appliquer) est très important, il affecte fortement la qualité de la solution obtenue.

Point initial. Dans une méthode tabou, on doit partir d'une solution initiale. Le plus souvent, cette solution est donnée par une heuristique de résolution appropriée au problème à résoudre.

Mise à jour de la solution courante : échapper aux optima locaux. Un optimum local est défini de la façon suivante :

Définition 5.3. *Un optimum local $\widehat{S}$ est tel que $\forall S \in \mathcal{N}(\widehat{S})$, $f(\widehat{S}) \leq f(S)$.*

A chaque itération d'une méthode tabou, on passe d'une solution courante S_C à une nouvelle solution choisie dans son voisinage ($\mathcal{N}(S_C)$). La solution S^*, choisie

pour mettre à jour la solution courante S_C, est celle qui minimise la fonction d'évaluation parmi les voisins S_C :

$$\forall S \in \mathcal{N}(S_C), f(S^*) \leq f(S). \tag{5.1}$$

La solution courante est mise à jour : S^* devient la nouvelle solution courante. Supposons qu'à l'itération i la solution courante $S_C(i)$ était un optimum locale par rapport au voisinage et à la fonction d'évaluation considérés, la méthode tabou accepte alors de mettre à jour la solution courante en dégradant la solution : $S_C(i+1)$ est choisie dans ce cas comme la solution qui dégrade le moins la solution courante $S_C(i)$. Cette méthode permet d'échapper aux optima locaux et d'aller explorer d'autres solutions à la recherche d'un optimum global (*i.e.* la solution qui minimise f dans l'ensemble $\mathcal{E}$ de toutes les solutions réalisables). Remarquons qu'à l'itération $i+1$ la solution qui minimise f parmi les voisins de $S_C(i+1)$ pourrait être $S_C(i)$ puisque cette solution est un optimum local ce qui implique que $S_C(i+2)$ serait égale à $S_C(i)$. Dans ce cas, la méthode risque d'osciller entre les solutions $S_C(i)$ et $S_C(i+1)$. La méthode tabou est principalement conçue pour éviter ce genre de cyclage en rendant certains mouvements tabous *i.e.* non acceptables.

Notons que l'évaluation des voisins dépend de la façon dont le problème est codé. Dans le cas d'un codage direct, l'évaluation est immédiate. Dans le cas d'un codage indirect, une heuristique de résolution est le plus souvent utilisée pour évaluer les voisins.

Mouvements tabous. Lors de la mise à jour d'une solution courante qui dégrade la solution actuelle, l'inverse du mouvement élémentaire qui a été appliqué pour ce passage est sauvegardé dans une liste dite tabou : ce mouvement sera interdit pendant un certain nombre d'itérations. Ainsi à chaque itération, la solution courante est mise à jour en la substituant par la solution voisine qui minimise f parmi les voisins obtenus par des mouvements non tabous. Le nombre d'itérations durant lequel le critère tabou est maintenu sur les mouvements tabous dépend de la taille de la liste tabou qui est gérée comme une File (premier entré, premier sorti). La taille de la liste tabou affecte fortement la qualité des solutions obtenues, elle est déterminée expérimentalement.

A cause de l'interdiction de certains mouvements (les mouvements tabous), on risque parfois de ne pas considérer certaines solutions qui peuvent être plus intéressantes que la solution *record* (la meilleure solution rencontrée). Pour y remédier, on évalue tous les voisins d'une solution courante même ceux qui sont tabous (obtenus par des mouvements tabou) et on enlève, temporairement, le critère tabou sur un voisin tabou S si on le juge assez intéressant *i.e.* si $f(S)$ est inférieure à un certain niveau appelé niveau d'aspiration.

La liste des mouvements tabous n'est pas toujours suffisante pour s'éloigner d'un optimum local. Une opération appelée *diversification* peut être utilisée au sein

d'un algorithme tabou. Elle consiste à effectuer une transformation non-locale sur la solution courante. On obtient donc une nouvelle solution, qui a de grandes chances de ne plus être proche de l'optimum local, tout en conservant des propriétés de la solution précédente. Une autre méthode pus radicale, consiste à tirer aléatoirement une solution, complètement nouvelle, à partir de laquelle on relance la méthode tabou. On peut se référer à [35] pour une étude plus détaillée des méthodes tabou.

Nous proposons maintenant un nouvel algorithme tabou pour la résolution du problème $2BP$ dans le cas non-orienté.

5.3 Nouvel algorithme tabou

Dans cette section nous décrivons un nouvel algorithme tabou pour le $2BP|R$ basée sur la notion de *configurations équivalentes*. Cet algorithme est noté $TS-EC$ (Tabu Search algorithm avoiding Equivalent Configurations). Nous commençons par la description du voisinage considéré.

5.3.1 Voisinage

Une solution courante S_c du problème correspond à un ordre donné représenté par une liste σ_c telle que : $\sigma_c[i]$ désigne le i^{me} objet à placer. L'évaluation de la solution courante consiste à appliquer l'heuristique de base décrite dans la section 5.3.3.

Les transformations locales considérées sont des mouvements d'insertions. Un voisin de la liste σ_c résulte de l'insertion d'un objet $\sigma_c[i]$ à la position occupée par un autre objet $\sigma_c[j]$. Si $i > j$, cette insertion s'effectue en décalant à droite les objets $\sigma_c[k] : k = j, ..., i-1$, l'objet $\sigma_c[i]$ est alors placé à la position j. Si $i < j$, $\sigma[i]$ est placé après l'objet $\sigma[j]$ en décalant les objets $\sigma[k] : k = i+1, ..., j$, à gauche.

La taille du voisinage d'une solution courante est en $O(n^2)$. En effet, chaque voisin consiste à choisir un objet à insérer (parmi les n objets de l'instance considérée) et une position dans laquelle il sera inséré (parmi $n-1$ positions possibles). Le voisinage d'une solution courante S_c, correspondant à toutes les insertions possibles, est noté $\mathcal{N}(S_c)$.

Dans l'algorithme tabou TS-EC nous réduisons le voisinage considéré en interdisant la mise à jour de la solution courante par une autre solution qui lui soit "similaire". Dans la section suivante, nous introduisons la notion de *configurations équivalentes* pour définir les similarités entre les solutions par rapport aux objets placés par bin.

5.3.2 Notion de configurations équivalentes

Dans cette section, nous introduisons la notion de configurations équivalentes en se basant sur certaines similarités qu'on peut détecter entre les solutions.

Soit $I = (A, B)$ une instance $2BP$ donnée. Une configuration (rangement) $S = (B(S,1), ..., B(S,m))$ dans m (notée $nbBin(S)$) bins est une partition de A en m sous-ensembles $A_1, ..., A_m$, tel que chaque sous-ensemble d'objets A_j correspond à une solution de bin-packing dans un bin $B(S,j)$ sans chevauchement des objets. Pour chaque objet a_i, nous notons $B_S(a_i)$, le bin contenant a_i dans la configuration S.

Définition 5.4. *Deux configurations F et G sont dites équivalentes si et seulement si $\forall k = 1, ..., nbBin(F)$ il existe un entier $k' \in \{1, ..., nbBin(G)\}$ tel que $B(F,k) = B(G,k')$.*

Soit une instance $I = (A, B)$ telle que $B = (10, 12)$ et,

$$A = \{a_1 = (4,9), a_2 = (5,4), a_3 = (4,3), a_4 = (5,6),$$
$$a_5 = (6,8), a_6 = (3,9), a_7 = (11,5), a_8 = (4,5)\}$$

Deux solutions F et G de cette instance, sont représentées dans la figure 5.1. En examinant les deux configurations, nous constatons que $B(F,1) = B(G,2) = \{a_3, a_5, a_6\}$, $B(F,2) = B(G,1) = \{a_1, a_2, a_4\}$ et $B(F,3) = B(G,3) = \{a_7, a_8\}$. Les deux configurations sont alors équivalentes.

La caractéristique suivante des configurations équivalentes est déduite immédiatement de la définition 5.4.

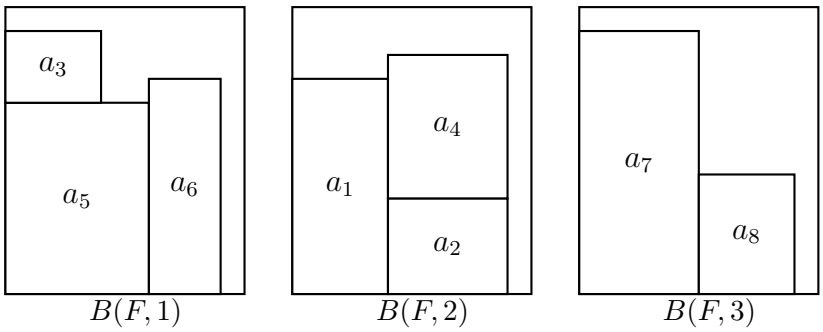
Proposition 5.1. *Deux configurations F et G sont dites équivalentes SSi $\forall (a_i, a_j) \in A^2$, $B_F(a_i) = B_F(a_j) \Leftrightarrow B_G(a_i) = B_G(a_j)$.*

En se servant de la proposition 5.1, la reconnaissance de deux configurations équivalentes peut être effectuée en $O(n^2)$. En effet, il suffit de vérifier pour chaque a_i et a_j dans A si :

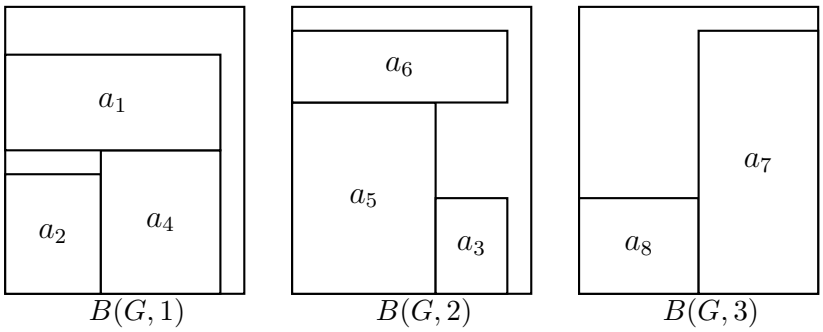
$$B_F(a_i) = B_F(a_j) \text{ ssi } B_G(a_i) = B_G(a_j)$$

Proposition 5.2. *L'identification de configurations équivalentes est en $O(n)$.*

Démonstration. Ayant deux configurations F et G, si $nbBin(F) \neq nbBin(G)$, il est clair, d'après la définition 5.4, que les configurations F et G ne sont pas équivalentes. Maintenant si $nbBin(F) = nbBin(G)$, nous proposons un algorithme d'identification des configurations équivalentes (Algorithme 3) en $O(n)$. La structure de données considérée est celle introduire au début de cette section : étant donné une configuration $S = (B(S,1), ..., B(S, nbBin(S)))$, $B(S,j) = \{a_{j_1}, ..., a_{j_t}\}$ est le sous-ensemble



(a) Configuration F.



(b) Configuration G.

Figure 5.1 – Exemple de deux configurations équivalentes F et G.

Algorithme 3 : Algorithme d'identification des configurations équivalentes

Données : F et G deux configurations telles que $nbBin(F) = nbBin(G)$
Résultat : $FLAG$: une variable qui vaut 1 si F et G sont des configurations équivalentes, sinon elle vaut 0.

$FLAG := 1$; $j := 1$;
tant que *($j \leq nbBin(F)$) et ($FLAG = 1$)* **faire**
  Soit $\{a_{j_1}, ..., a_{j_t}\}$ l'ensemble des objets rangés dans le bin $B(F, j)$;
  $k := 2$;
  tant que *($k \leq t$) et ($FLAG = 1$)* **faire**
    si $B_G(a_{j_k}) \neq B_G(a_{j_1})$ **alors**
      $FLAG := 0$;
      retourner $FLAG$;
    $k := k + 1$;
  $j := j + 1$;
retourner $FLAG$;

des objets placés dans le bin j et $B_S(a_i)$ représente le bin contenant a_i dans la configuration S. L'algorithme 3 est constitué d'une boucle principale et une boucle intérieure. La boucle principale est répétée $nbBin(F)$ fois au plus. A chaque itération j de la boucle principale, la boucle intérieure vérifie que les objets $\{a_{j_2}, ..., a_{j_t}\}$ sont placés dans le même bin qui contient a_{j_1} dans la configuration G (a_{j_1} est un représentant du bin $B(F, j)$). La boucle intérieure est répétée $|B(F, j))|$ fois au plus, puisque chaque objet placé dans le bin $B(F, j)$ est visité une seule fois au plus. A la fin de l'itération j de la boucle principale, deux cas se présentent. Dans le premier cas, $Flag = 0$ cela signifie qu'il existe au moins un objet a_{j_u} $(2 \leq u \leq t)$ tel que a_{j_1} et a_{j_u} sont placés dans le même bin dans la configuration F et ils sont placés dans des bins différents dans G. Les deux configurations sont par conséquent non équivalentes. Dans le deuxième cas, $Flag = 1$ ce qui signifie que tous les objets sont placés dans le même bin dans G et que les deux configurations sont équivalentes. La complexité temporelle de l'algorithme 3 est alors en : $\sum_{j=1}^{nbBin(F)} |B(F, j)| = n$. □

Dans la section suivante nous décrivons l'heuristique de base utilisée au sein du nouvel algorithme tabou.

5.3.3 Heuristique de base

Dans cette section nous décrivons l'heuristique de base utilisée dans l'évaluation des solutions. Cette heuristique consiste à placer les objets dans un ordre prédéfini.

Etant donné un ordre σ des objets, l'heuristique range les objets un par un de sorte à minimiser localement le nombre de bins utilisés. Un critère heuristique est utilisé pour choisir la surface maximale disponible la plus adéquate pour placer l'objet en cours. L'objet est placé dans la surface maximale retenue tel que son coin bas-gauche coïncide avec le coin bas-gauche de la surface. Si aucune surface ne peut contenir l'objet en cours, un nouveau bin est ajouté à la liste Lma des surfaces maximales.

Le critère considéré est le même que celui utilisé dans l'heuristique IMA (section 4.4). L'objet courant a_i est donc placé dans la surface maximale qui, parmi les surfaces qui peuvent le contenir, maximise un certain critère. Pour un objet a_i et une surface maximale ma donnés, ce critère s'écrit :

$$\begin{aligned} \mathcal{O}(a_i, ma) = & q_1(w_i h_i)/(w_{ma} h_{ma}) & + \\ & q_2(dx_i/w_{ma}) & + \\ & q_3(dy_i/h_{ma}) & + \\ & q_4(w_i^2 + h_i^2)/(w_{ma}^2 + h_{ma}^2) & \end{aligned}$$

où les paramètres q_1, q_2, q_3 et q_4 sont des réels tels que $0 \leq q_k \leq 1; k = 1, ..., 4$ et

Algorithme 4 : L'heuristique de base

Données : une instance $I = (A, B)$, un ordre σ des objets et quatre nombres réèls q_1, q_2, q_3 et q_4.
Initialiser la liste des surfaces maximales (Lma) avec un bin vide à la position $(0, 0)$, l'origine des axes x et y ;
$nbBin := 1$;
$i := 1$;
tant que $(i \neq n)$ **faire**
  rapMax $:= 0$;
  pour $(j := 1; j \leq |Lma|; j++)$ **faire**
    si ($\sigma[i]$ *peut être placé dans* ma_j *sans chevauchement et* $\mathcal{O}(a_i, ma_j) > rapMax$) **alors**
      $rapMax = \mathcal{O}(a_i, ma_j)$; $a^* = a_i$; $ma^* = ma_j$; (Les deux orientation de a_i sont testées)
  si $(rapMax > 0)$ **alors**
    placer l'objet a^* dans la surface maximale ma^*, et mettre à jour la liste Lma ;
    $i := i + 1$;
    sinon
      ajouter à Lma une nouvelle surface maximale correspondante à un nouveau bin ;
      $nbBin := nbBin + 1$;
retourner nbBin ;

$q_1 + q_2 + q_3 + q_4 = 1$, w_{ma} et h_{ma} sont les projections des dimensions de la surface ma sur les axes x et y.

Soit C le nombre maximum de surfaces maximales présentes dans la liste Lma à une étape quelconque du rangement. Chaque placement d'un objet nécessite un parcours de la liste Lma pour choisir la surface qui maximise le critère cité ci-dessus (*cf.* Algorithme 4). Placer l'objet a^* dans ma^* et mettre à jour la liste Lma est en $O(C^2)$. Le placement de tous les objets de l'ordre σ est alors en $O(n(C + C^2)) = O(nC^2)$. Comme on ne considère que des rangements stables, nous avons établi dans la section 4.3 que Le nombre de surfaces maximales (C) à une étape quelconque d'un rangement stable est en $O(n^2)$. L'heuristique de base considérée est donc en $O(n^5)$. Mais les résultats numériques (Section 5.4) montrent que, pour les benchmarks de la littérature, le nombre de surfaces maximales C ne dépasse jamais $2n$.

Dans la section suivante nous présentons la fonction d'évaluation adoptée.

5.3.4 Fonction d'évaluation

Le problème à résoudre étant de minimiser le nombre de bins qui peuvent contenir l'ensemble des objets d'une instance $2BP|R$ donnée, la fonction d'évaluation peut être définie comme le nombre de bins utilisés. Cependant, une telle fonction d'évaluation ne serait pas assez sélective pour pouvoir choisir la nouvelle solution courante à chaque itération de la méthode tabou. En effet, ayant une solution courante donnée, plusieurs solutions voisines auraient un nombre de bins utilisés identiques, et ceci indépendamment du voisinage considéré. Nous proposons alors une fonction d'évaluation contenant d'autres mesures discriminantes, pour trancher entre des solutions ayant le même nombre de bins.

A chaque itération tabou, la solution courante S_C doit être mise à jour en la remplaçant par la *meilleure* solution de son voisinage. Si plusieurs candidats produisent le même nombre de bins, nous favorisons la solution qui contient le bin avec la plus petite surface occupée (la somme des surfaces des objets rangés dans le bin). En cas d'égalité, nous considérons la solution qui a le nombre le plus grand d'objets placés dans le bin le moins rempli.

Etant donnée une solution S, soit $Area(B(S,j))$ (resp. $nbItem(B(S,j))$) la somme des surfaces des objets (resp. le nombre d'objets) rangés dans le bin j de S :

$$Area(B(S,j)) = \sum_{a_i \in B(S,j)} w_i * h_i$$

$$nbItem(B(S,j)) = |B(S,j)|$$

Nous notons aussi $MinArea(S_C)$, l'ensemble de bins dont la surface remplie est égale à $\min_{j=1}^{nbBin(S)} Area(B(S_C,j))$.

$$MinArea(S) = \{j^* : Area(B(S,j^*)) = \min_{j=1}^{nbBin(S)} Area(B(S,j))\}$$

Pour chaque voisin S_N de la solution courante S_C, la fonction d'évaluation $f(S_N)$ s'écrit de la façon suivante :

$$\begin{aligned} f(S_N) = M_1 * nbBin(S_N) + M_2 * & \min_{j=1}^{nbBin(S_N)} Area(B(S_N,j)) \\ & - \max_{j \in MinArea(S_N)} nbItem(B(S_N,j)) \end{aligned} \tag{5.2}$$

où

$$M_2 = n \text{ et } M_1 = nWH$$

Le voisin qui minimise la fonction d'évaluation décrite ci-dessus est utilisé pour la mise à jour de la solution courante. Si plusieurs voisins ont la même valeur du coût minimal, l'un d'eux est tiré aléatoirement suivant une loi uniforme pour mettre à jour la solution courante.

5.3.5 Schéma de l'algorithme tabou proposé (TS-EC)

Etant donnée une instance I à résoudre, notre méthode commence par chercher une solution courante initiale. La solution initiale est donnée par l'heuristique IMA proposée dans le chapitre 4. Rappelons que, dans chaque étape de placement d'un objet, l'heuristique IMA choisit le couple (objet à placer, la surface qui va l'accueillir) ainsi que l'orientation de l'objet. L'ordre initial des objets correspond donc à l'ordre dans lequel IMA a choisi de les placer dans la dernière itération améliorante de l'heuristique.

Nous considérons une liste tabou conçue comme une file de taille déterminée expérimentalement. Dans nos résultats expérimentaux la taille de la liste tabou vaut $min(10, n/4)$. Initialement, la liste est vide.

Ayant une solution courante S_C correspondant à un ordre donné σ_C des objets, nous explorons le voisinage de la solution courante. Tous les mouvements d'insertions possibles sont testés (*cf.* Section 5.3.1). Chaque voisin est évalué en appliquant l'heuristique de base décrite dans la section 5.3.3. Le choix des valeurs des paramètres (q_1, q_2, q_3, q_4) du critère utilisé dans l'heuristique de base pour choisir la surface la plus adaptée pour ranger l'objet courant est déterminé expérimentalement. Il consiste à alterner avec les valeurs suivantes :

- (0.25, 0.25, 0.25, 0.25).
- (0.7, 0.2, 0.1, 0).
- (0, 0.2, 0.1, 0.7).

La solution courante est mise à jour par la solution voisine qui minimise la fonction d'évaluation décrite dans la section 5.3.4.

L'objet, dont l'insertion a mené à la nouvelle solution courante (nouvel ordre courant des objets), est ajouté à la liste tabou. Toute insertion de cet objet est interdite tant qu'il figure dans la liste tabou. Le nombre d'itérations durant lesquelles l'objet est maintenu tabou est égal à la taille de la liste tabou.

Cette procédure est répétée tant que la solution trouvée ne correspond pas à une borne inférieure connue et que le temps de calcul n'a pas dépassé un temps limite (fixé dans nos résultats expérimentaux à 120 secondes).

Plusieurs mouvements d'insertion mènent en pratique à des configurations équivalentes à la configuration courante. Il suffit, par exemple, de modifier les positions des objets dans le bin sans modifier leurs affectations aux différents bins. Nous réduisons le voisinage considéré en excluant les configurations équivalentes à la solution courante, dans le but d'accélérer la procédure d'éloignement d'un optimum local. Les versions adoptant le voisinage complet et réduit sont comparées dans la section suivante.

Notons que nous acceptons de mettre à jour la solution courante par une solution voisine qui résulte de l'insertion d'un objet tabou si cette insertion améliore

strictement la solution record.

La complexité de l'heuristique de base est en $O(nC^2)$ (*cf.* Section 5.3.3), où C est le nombre maximum de surfaces maximales présentes à une étape donnée du rangement. La complexité d'une itération tabou est alors en $O(n^3C^2)$ étant donné que le voisinage considéré est en $O(n^2)$ (*cf.* Section 5.3.1) et que la complexité d'une itération tabou est égale à la complexité de l'heuristique base. Les résultats expérimentaux dans la section suivante montrent que le nombre de surfaces maximales (C) ne dépasse jamais $2n$.

5.4 Résultats expérimentaux

Dans cette section, nous rapportons les résultats des tests de notre algorithme tabou sur le benchmark proposé dans [4] et [48] (*cf.* Section 2.4 pour une description des instances).

Dans le tableau 5.1, nous comparons deux versions de l'algorithme tabou $TS - EC$ décrit dans la section 5.3.5 avec et sans prétraitements. Nous rappelons que le voisinage considéré dans $TS - EC$ est le voisinage réduit, les configurations équivalentes à la solution courante n'étant pas retenues comme candidates pour la mise à jour de la solution. Un temps limite de 120 secondes est choisi pour la résolution d'une instance. Un deuxième critère d'arrêt correspondant à une borne inférieure connue pour l'instance à résoudre est considéré. Nous montrons les résultats qui concernent le nombre de bins utilisés pour ranger l'ensemble des objets ($nbBin$), le temps CPU correspondant à l'exécution de la méthode en secondes (CPU), le nombre maximum de surfaces maximales présentes en une étape quelconque du rangement (MA_{max}) ainsi que le nombre moyen ($MA_{average}$). Les valeurs correspondent à la moyenne calculée pour chaque groupe de dix instances d'une classe. Pour la valeur MA_{max}, les résultats affichés correspondent à la valeur maximale pour chaque groupe d'instances. La ligne intitulée *Average* correspond à la moyenne calculée par classe. En se référant au tableau 5.1, nous constatons l'avantage de la version avec prétraitements, en effet cette version fournit des solutions meilleures que la version sans prétraitements pour trois classes du benchmark (les classes 1, 3 et 10). Les deux versions sont presque équivalentes en terme du temps de calcul. Remarquons que pour les deux versions, la valeur MA_{max} est en $O(n)$. La version avec prétraitement est retenue par la suite.

Dans le tableau 5.2, nous comparons la version $TS - EC$ avec celle qui permet la mise à jour d'une solution courante par une configuration équivalente (TS). Nous rapportons dans les résultats le nombre de configurations équivalentes (donc interdites) en moyenne par itération tabou de l'algorithme $TS - EC$. Le tableau 5.2 montre que l'algorithme $TS - EC$ est strictement meilleur que TS. En effet il utilise un nombre de bins plus petit pour les classes 1 et 7, et le même pour les autres

Problème			avec prétraitement				sans prétraitement			
Classe	HxW	n	$nbBin$	CPU	MA_{max}	MA_{avg}	$nbBin$	CPU	MA_{max}	MA_{avg}
I	10x10	20	6.6	0.009	3	2.3	6.6	0.003	3	2.8
		40	**12.8**	12.298	24	6.5	12.9	24.017	28	7.4
		60	19.5	0.005	4	2.5	19.5	0.014	3	2.7
		80	27.0	12.156	34	5.4	27.0	12.048	35	5.9
		100	31.3	25.225	44	11.3	31.3	24.778	43	11.2
		Average	**19.44**	9.938	44	5.600	19.46	12.172	43	6.000
II	30x30	20	1.0	0.000	10	8.7	1.0	0.000	10	8.6
		40	1.9	0.008	10	7.8	1.9	0.002	9	7.2
		60	2.5	0.005	10	8.2	2.5	0.008	10	8.0
		80	3.1	0.017	13	9.5	3.1	0.009	11	9.0
		100	3.9	0.022	11	8.9	3.9	0.025	12	9.5
		Average	2.48	0.010	13	8.620	2.48	0.008	12	8.460
III	40x40	20	4.7	0.003	5	3.4	4.7	0.008	5	3.9
		40	**9.3**	29.250	44	15.1	9.4	36.059	45	15.8
		60	13.5	36.514	62	20.7	13.5	36.472	62	21.1
		80	**18.4**	37.811	73	24.0	18.5	49.467	72	30.9
		100	22.2	92.511	88	57.9	22.2	91.227	88	58.2
		Average	**13.62**	39.217	88	24.220	13.66	42.646	88	25.980
IV	100x100	20	1.0	0.000	15	9.9	1.0	0.000	15	9.6
		40	1.9	0.003	14	10.2	1.9	0.002	14	9.9
		60	2.5	24.156	54	18.1	2.5	24.211	53	17.9
		80	3.1	12.967	67	16.2	3.1	13.139	67	16.2
		100	3.7	0.042	16	10.6	3.7	0.044	16	11.0
		Average	2.44	7.434	67	13.000	2.44	7.478	67	12.920
V	100x100	20	5.9	0.000	5	2.7	5.9	0.002	5	4.0
		40	11.4	12.058	48	6.9	11.4	12.086	48	8.4
		60	17.4	60.680	76	32.4	17.4	60.975	77	38.5
		80	23.9	36.194	86	19.3	23.9	37.842	100	31.7
		100	27.9	93.392	108	70.6	27.9	92.694	110	75.6
		Average	17.30	40.464	108	26.380	17.30	40.719	110	31.640
VI	300x300	20	1.0	0.003	14	11.9	1.0	0.000	14	11.9
		40	1.7	24.030	42	20.3	1.7	24.126	42	20.2
		60	2.1	0.009	19	12.6	2.1	0.013	18	12.6
		80	3.0	0.016	16	12.4	3.0	0.017	17	12.5
		100	3.2	0.477	17	14.0	3.2	0.487	17	13.8
		Average	2.20	4.906	42	14.240	2.20	4.928	42	14.200
VII	100x100	20	5.2	60.003	29	14.0	5.2	60.009	28	15.5
		40	10.3	76.517	52	34.5	10.3	82.167	54	37.2
		60	14.6	73.380	72	43.1	14.6	73.333	74	44.7
		80	21.2	124.581	96	90.2	21.2	124.495	98	92.1
		100	25.3	135.275	115	108.6	25.3	134.069	117	109.3
		Average	15.32	93.951	115	58.080	15.32	94.814	117	59.760
VIII	100x100	20	5.3	48.006	28	12.3	5.3	48.009	29	13.4
		40	10.4	96.255	51	40.4	10.4	96.388	53	41.3
		60	15.0	97.302	73	56.6	15.0	97.442	73	56.9
		80	20.8	112.964	96	79.5	20.8	114.280	97	81.6
		100	25.7	134.120	113	108.0	25.7	135.488	113	108.8
		Average	15.44	97.729	113	59.360	15.44	98.321	113	60.400
IX	100x100	20	14.3	0.000	0	0.0	14.3	0.003	4	2.4
		40	27.5	0.000	1	0.6	27.5	0.003	4	2.5
		60	43.5	0.000	1	0.5	43.5	0.008	4	2.9
		80	57.3	0.000	1	0.8	57.3	0.022	4	3.2
		100	69.3	0.000	1	0.7	69.3	0.023	4	2.7
		Average	42.38	0.000	1	0.520	42.38	0.012	4	2.740
X	100x100	20	4.1	24.003	25	7.3	4.1	24.006	25	8.4
		40	7.3	36.056	45	16.1	7.3	36.106	49	17.5
		60	**9.9**	52.063	65	37.3	10.0	62.348	66	40.0
		80	12.8	73.492	85	51.3	12.8	74.200	86	51.5
		100	15.8	67.216	102	51.3	15.8	63.956	100	51.2
		Average	**9.98**	50.566	102	32.660	10.00	52.123	100	33.720

Tableau 5.1 – L'algorithme tabou avec et sans prétraitements.

Problème			$TS-EC$			TS	
Classe	WxH	n	$Bins$	CPU	$nbCE$	$Bins$	CPU
I	10x10	20	6.6	0.009	0.00	6.6	0.003
		40	**12.8**	12.298	31.00	12.9	24.048
		60	19.5	0.005	0.00	19.5	0.002
		80	27.0	12.156	64.34	27.0	12.019
		100	31.3	25.225	103.09	31.3	25.019
		Average	**19.44**	9.938	39.68	19.46	12.218
II	30x30	20	1.0	0.000	0.00	1.0	0.002
		40	1.9	0.008	0.00	1.9	0.000
		60	2.5	0.005	0.00	2.5	0.008
		80	3.1	0.017	0.00	3.1	0.011
		100	3.9	0.022	0.00	3.9	0.020
		Average	2.48	0.010	0.00	2.48	0.008
III	40x40	20	4.7	0.003	0.00	4.7	0.002
		40	9.3	29.250	37.05	9.3	27.953
		60	13.5	36.514	64.25	13.5	36.497
		80	18.4	37.811	133.15	18.4	37.608
		100	22.2	92.511	278.55	22.2	92.258
		Average	13.62	39.217	102.60	13.62	38.863
IV	100x100	20	1.0	0.000	0.00	1.0	0.000
		40	1.9	0.003	0.00	1.9	0.002
		60	2.5	24.156	27.56	2.5	24.525
		80	3.1	12.967	19.22	3.1	13.144
		100	3.7	0.042	0.00	3.7	0.036
		Average	2.44	7.434	9.36	2.44	7.541
V	100x100	20	5.9	0.000	0.00	5.9	0.000
		40	11.4	12.058	11.49	11.4	12.048
		60	17.4	60.680	114.84	17.4	60.636
		80	23.9	36.194	75.83	23.9	36.880
		100	27.9	93.392	281.24	27.9	91.052
		Average	17.30	40.464	96.68	17.30	40.123
VI	300x300	20	1.0	0.003	0.00	1.0	0.000
		40	1.7	24.030	14.78	1.7	24.176
		60	2.1	0.009	0.00	2.1	0.008
		80	3.0	0.016	0.00	3.0	0.019
		100	3.2	0.477	0.00	3.2	0.479
		Average	2.20	4.906	2.96	2.20	4.936
VII	100x100	20	5.2	60.003	22.41	5.2	60.005
		40	**10.3**	76.517	109.41	10.4	84.220
		60	**14.6**	73.380	165.48	14.7	72.922
		80	21.2	124.581	358.55	21.2	125.608
		100	25.3	135.275	584.32	25.3	134.780
		Average	**15.32**	93.951	248.04	15.36	95.506
VIII	100x100	20	5.3	48.006	31.17	5.3	48.006
		40	10.4	96.255	142.36	10.4	96.155
		60	15.0	97.302	178.62	15.0	97.248
		80	20.8	112.964	297.66	20.8	112.953
		100	25.7	134.120	503.86	25.7	134.322
		Average	15.44	97.729	230.74	15.44	97.737
IX	100x100	20	14.3	0.000	0.00	14.3	0.000
		40	27.5	0.000	0.00	27.5	0.000
		60	43.5	0.000	0.00	43.5	0.002
		80	57.3	0.000	0.00	57.3	0.000
		100	69.3	0.000	0.00	69.3	0.000
		Average	42.38	0.000	0.00	42.38	0.000
X	100x100	20	4.1	24.003	7.44	4.1	24.003
		40	7.3	36.056	34.84	7.3	36.081
		60	9.9	52.063	98.25	9.9	60.334
		80	12.8	73.492	169.12	12.8	75.070
		100	15.8	67.216	189.43	15.8	65.783
		Average	9.98	50.566	99.82	9.98	52.254

Tableau 5.2 – Comparaison entre les deux versions.

Problème			IMA		$ATS-TP$		$TS-EC$			
Classe	WxH	n	*Bins*	*CPU*	*Bins*	*CPU*	*Bins*	*CPU*	*G*	*Opt*
I	10x10	20	6.6	0,003	6.6	0.13	6.6	0.009	1,000	10
		40	12.9	0,067	12.9	4.11	**12.8**	12.298	1,000	10
		60	19.5	0,001	19.5	4.27	19.5	0.005	1,000	10
		80	27.0	0,110	27.0	0.97	27.0	12.156	1,000	10
		100	**31.3**	0,321	31.4	1.20	**31.3**	25.225	1,000	10
		Average	19.46	0,100	19.48	2.14	**19.44**	9.938	1,000	10
II	30x30	20	1.0	0,001	1.0	0.13	1.0	0.000	1,000	10
		40	1.9	0,003	1.9	1.46	1.9	0.008	1,000	10
		60	2.5	0,000	2.5	1.14	2.5	0.005	1,000	10
		80	3.1	0,015	3.1	22.61	3.1	0.017	1,000	10
		100	3.9	0,024	3.9	4.25	3.9	0.022	1,000	10
		Average	2.48	0,009	2.48	5.92	2.48	0.010	1,000	10
III	40x40	20	4.7	0,001	4.7	11.35	4.7	0.003	1,000	10
		40	9.4	0,126	9.4	0.38	**9.3**	29.250	1,025	8
		60	**13.5**	0,242	13.6	64.85	**13.5**	36.514	1,024	7
		80	**18.4**	0,359	18.6	9.94	**18.4**	37.811	1.012	8
		100	**22.2**	1,173	22.3	68.85	**22.2**	92.511	1.035	3
		Average	13.64	0,380	13.72	30.94	**13.62**	39.217	1.019	7.2
IV	100x100	20	1.0	0,000	1.0	0.14	1.0	0.000	1,000	10
		40	1.9	0,001	1.9	0.25	1.9	0.003	1,000	10
		60	2.5	0,432	**2.4**	28.80	2.5	24.156	1,100	8
		80	**3.1**	0,323	3.2	36.16	**3.1**	12.967	1.033	9
		100	**3.7**	0,032	3.8	9.19	**3.7**	0.042	1,000	10
		Average	**2.44**	0,158	2.46	14.91	**2.44**	7.434	1,027	9,4
V	100x100	20	5.9	0,000	5.9	1.31	5.9	0.000	1,000	10
		40	11.4	0,067	11.4	41.21	11.4	12.058	1,014	9
		60	**17.4**	0,334	17.5	74.59	**17.4**	60.680	1,027	6
		80	23.9	0,228	23.9	143.01	23.9	36.194	1,013	7
		100	**27.9**	1,154	28.0	196.23	**27.9**	93.392	1,028	3
		Average	**17.30**	0,357	17.34	91.27	**17.30**	40.464	1,016	7
VI	300x300	20	1.0	0,000	1.0	0.17	1.0	0.003	1,000	10
		40	1.7	0,304	**1.6**	5.97	1.7	24.030	1,200	8
		60	2.1	0,009	2.1	25.92	2.1	0.009	1,000	10
		80	3.0	0,012	3.0	0.42	3.0	0.016	1,000	10
		100	**3.2**	0,300	3.4	7.64	**3.2**	0.477	1,000	10
		Average	**2.20**	0,125	2.22	8.02	**2.20**	4.906	1,040	9.6
VII	100x100	20	5.2	0,065	5.2	0.77	5.2	60.003	1,110	5
		40	10.4	0,276	10.4	99.28	**10.3**	76.517	1,042	6
		60	14.7	0,537	**14.6**	613.98	**14.6**	73.380	1,042	4
		80	**21.2**	1,228	21.3	259.78	**21.2**	124.581	1,061	0
		100	**25.3**	1,789	25.5	231.66	**25.3**	135.275	1,058	0
		Average	15.36	0,779	15.40	241.09	**15.32**	93.951	1,063	3
VIII	100x100	20	5.3	0,057	5.3	1.34	5.3	48.006	1,060	7
		40	10.4	0,342	10.4	77.91	10.4	96.255	1,073	3
		60	**15.0**	0,632	15.1	51.74	**15.0**	97.302	1,063	2
		80	20.8	1,127	20.8	529.40	20.8	112.964	1,057	1
		100	**25.7**	1,812	26.0	232.88	**25.7**	134.120	1,062	0
		Average	**15.44**	0,794	15.52	178.65	**15.44**	97.729	1,063	2.6
IX	100x100	20	14.3	0,000	14.3	0.26	14.3	0.000	1,000	10
		40	**27.5**	0,000	27.6	1.25	**27.5**	0.000	1,000	10
		60	43.5	0,000	43.5	2.54	43.5	0.000	1,000	10
		80	57.3	0,000	57.3	1.65	57.3	0.000	1,000	10
		100	69.3	0,000	69.3	52.69	69.3	0.000	1,000	10
		Average	**42.38**	0,000	42.40	11.68	**42.38**	0.000	1,000	10
X	100x100	20	4.1	0,039	4.1	0.36	4.1	24.003	1,100	8
		40	7.3	0,120	7.3	3.93	7.3	36.056	1,049	7
		60	10.1	0,473	**9.9**	235.17	**9.9**	52.063	1,047	6
		80	12.8	0,786	12.8	78.27	12.8	73.492	1,049	4
		100	**15.8**	1,001	15.9	269.68	**15.8**	67.216	1,034	5
		Average	10.02	0,484	10.00	117.48	**9.98**	50.566	1,056	6

Tableau 5.3 – L'algorithme tabou, comparaison avec les meilleurs résultats de la littérature.

classes. Le temps de calcul est équivalent pour les deux algorithmes.

Enfin, nous comparons l'algorithme tabou $TS - EC$ avec l'heuristique IMA proposée dans le chapitre 4, et la méthode tabou $ATS - BP$ [38]. Le nouvel algorithme tabou proposé réussit à améliorer les résultats de l'heuristique IMA pour les classes 1, 3, 7 et 10. L'algorithme $TS - EC$ comparé à $ATS - BP$ permet d'améliorer plusieurs instances des différentes classes du benchmark considéré. Les deux résultats supplémentaires montrés pour $TS - EC$ concernent le rapport entre le nombre de bins obtenus par l'algorithme et la borne inférieure L^R_{CJE} proposée dans le chapitre 3. Le nombre d'instances résolues d'une façon optimale est noté Opt. Avec le nouvel algorithme tabou et la nouvelle borne inférieure, nous résolvons d'une façon optimale 374 instances des 500.

5.5 Conclusion

Nous avons introduit dans ce chapitre la notion de configurations équivalentes et proposé un algorithme d'identification de telles configurations en $O(n)$. Cette notion nous a servi pour la proposition d'un nouvel algorithme tabou pour le problème $2BP|R$. Nous avons montré expérimentalement l'efficacité de la méthode proposée. Ces travaux ont donné lieu à une présentation dans une conférence internationale avec actes (Incom 2006) [24].

Conclusion

Nous avons traité dans cette thèse le problème de bin-packing en deux dimensions dans le cas non orienté ($2BP|R$). Ce problème appartient à la classe des problèmes NP-difficiles. Notre contribution concerne plus précisément les prétraitements des instances, les évaluations par défaut et la résolution approchée du problème. Nos méthodes ont été testées expérimentalement sur les benchmarks de la littérature [4, 48].

Nous avons proposé trois prétraitements (α, β et γ) sous forme de fonctions identiquement réalisables (des fonctions qui, appliquées à une instance donnée, ne changent pas sa valeur optimale). Le premier prétraitement consiste à évaluer les espaces perdus dans le bin. Il utilise une variante du problème de sub-set sum appliquée à chaque dimension des objets. Les deux autres prétraitements consistent à identifier des sous-ensembles particuliers dans l'instance et trouver leur rangement optimal. Les trois prétraitements proposés sont complémentaires, leur application itérative a permis de réduire la taille de nombreuses instances du benchmark de la littérature et d'améliorer de façon significative la borne continue. Nous avons introduit la notion d'objets à orientation dépendante qui nous a servi dans le prétraitement β. L'intérêt majeur de cette notion est de définir des sous-ensembles d'objets dont la possibilité de rotation est contrainte s'ils sont placés dans un même bin.

Nous avons aussi proposé une nouvelle méthode d'évaluation des bornes inférieures pour le $2BP|R$. Toute borne inférieure proposée pour le problème $2BP|O$ peut servir à l'évaluation des bornes inférieures pour le $2BP|R$. La nouvelle borne inférieure L^R_{CJE} utilise la borne de Carlier *et al.* [10] pour l'évaluation par défaut des instances $2BP|O$ considérées. Nous avons montré que L^R_{CJE} domine théoriquement les bornes de Dell'Amico *et al.* [19] et de Boschetti et Mingozzi [6]. La nouvelle borne testée expérimentalement, permet d'améliorer strictement les bornes inférieures de la littérature.

Nous avons également proposé une nouvelle méthode heuristique (IMA) pour la résolution des instances du problème étudié. Elle adapte une stratégie Best Fit pour le placement des objets. Nous avons testé notre heuristique avec les méthodes de résolution approchée de la littérature (l'heuristique HBP [6] et la méthode tabou $ATS - BP$ [38]). Les résultats obtenus en appliquant IMA dominent ceux de la

méthode HBP. Elle fournit de meilleurs résultats que la méthode tabou $ATS-BP$ pour plusieurs classes. La classe 10 est la seule classe pour laquelle $ATS-BP$ fournit en moyenne des résultats meilleurs que IMA. Les instances de cette classe feront l'objet d'une étude particulière.

Nous avons proposé un nouvel algorithme tabou. Nous avons introduit la notion de configurations équivalentes pour définir certaines similarités entre les solutions. Dans l'algorithme tabou proposé, nous interdisons la mise à jour de la solution courante par une autre solution qui lui soit équivalente. Notre méthode améliore les résultats de la littérature pour toutes les classes du benchmark considéré. Les méthodes proposées dans la présente thèse nous ont permis de résoudre d'une façon optimale 374 instances des 500 instances constituant le benchmark considéré.

Les perspectives de ce travail sont nombreuses. Nous les présentons en quatre points :

- Les bons résultats obtenus par les prétraitements proposés nous encouragent à approfondir davantage l'étude des prétraitements tenant compte de la possibilité de tourner les objets de 90 degrés.
- Nous travaillons sur une nouvelle définition des "configurations équivalentes" en cherchant à introduire de nouvelles formes de similarité entre les solutions. On peut ainsi, par exemple, chercher à définir les configurations équivalentes en terme de surfaces maximales présentes dans les bins.
- Etant donnés les résultats fort encourageants obtenus sur de nombreux points, nous envisageons aujourd'hui la conception d'une méthode exacte pour le $2BP|R$. Les prétraitements et les bornes proposés dans cette thèse pourraient jouer un rôle prédominant pour envisager de résoudre de nombreuses instances non résolues de la littérature.
- L'ensemble des travaux proposés dans cette thèse est adaptable au problème de bin-packing 3D. Il est aussi intéressant d'étudier la possibilité d'adapter les méthodes proposées à d'autres variantes du problème comme par exemple l'introduction de la contrainte guillotine ou d'autres types de contraintes fréquemment rencontrées, le problème de strip packing, etc.

Bibliographie

[1] E. Aarts et J. K. Lenstra. *Local Search in Combinatorial Optimization*. Series in Discrete Mathematics and Optimization. John Wiley and Sons, 1997.

[2] J. E. Beasley. An exact two-dimensional non-guillotine cutting tree search procedure. *Operations Research*, 33 :49–64, 1985.

[3] B. E. Bengtsson. Packing rectangular pieces - a heuristic approach. *The computer journal*, 25 :353–357, 1982.

[4] J. O. Berkey et P. Y. Wang. Two-dimensional finite bin-packing algorithms. *Journal of Operational Research Society*, 38 :423–429, 1987.

[5] M. A. Boschetti et A. Mingozzi. The two-dimensional finite bin packing problem. part I : New lower bounds for the oriented case. *4OR*, 1 :27–42, 2003.

[6] M. A. Boschetti et A. Mingozzi. The two-dimensional finite bin packing problem. part II : New lower and upper bounds. *4OR*, 1 :135–147, 2003.

[7] J.M. Bourjolly et V. Rebetz. An analysis of lower bounds procedures for the bin packing problem.

[8] E. K. Burke, G. Kendall, et G. Whitwell. A new placement heuristic for the orthogonal stock-cutting problem. *Operations Research*, 52(4) :655–671, 2004.

[9] A. Caprara, M. Locatelli, et M. Monaci. Bilinear packing by bilinear programming. *In Michael Jünger and Volker kaibel, editors, Integer Programming and Combinatorial Optimization, 11th International IPCO Conference, Berlin, Germany, June 8-10, 2005, volume 3509 of Lecture Notes in Computer Science, pages 377-391*, Springer, june 8-10 2005.

[10] J. Carlier, F. Clautiaux, et A. Moukrim. New reduction procedures and lower bounds for the two-dimensional bin-packing problem with fixed orientation. *Computers O. R.*, 2006.

[11] B. Chazelle. The bottom-left bin-packing heuristic : an efficient implementation. *IEEE Trans. Comput.*, C-32 :697–707, 1983.

[12] N. Christophides et C. Whitlock. An algorithm for two-dimensional cutting problems. *Operations Research*, 25 :30–44, 1977.

[13] F.R.K. Chung, M.R. Garey, et D.S. Johnson. On packing two-dimensional bins. *SIAM J. alg. Disc. Meth.*, 3(1) :66–76, 1982.

[14] F. Clautiaux, J. Carlier, et A. Moukrim. A new exact method for the orthogonal packing problem. *European Journal of Operational Research*, (to appear), 2006.

[15] F. Clautiaux, J. Carlier, et A. Moukrim. A new exact method for the two-dimensional bin-packing problem with fixed orientaion. *Operations Research Letters*, (to appear), 2006.

[16] F. Clautiaux, A. Jouglet, et J. El Hayek. A new lower bound for the non-oriented two-dimensional bin-packing problem. *Operations Research Letters*, to appear, 2006.

[17] E. G. Coffman, Jr., M. R. Garey, et D. S. Johnson. Approximation algorithms for bin packing - an updated survey. In G. Ausiello, M. Lucertini, et P. Serafini, editors, *Algorithms design for computer system design*, pages 49–106. Spring-Verlag, New York, 1984.

[18] V.-D. Cung, M. Hifi, et B. Le Cun. Constrained two-dimensional cutting problem : a best-first branch-and-bound exact algorithm. *International Transactions in Operational Research*, 7 :185–210.

[19] M. Dell' Amico, M. Martello, et D. Vigo. A lower bound for the non-oriented two-dimensional bin packing problems. *dam*, 118 :13–24, 2002.

[20] K. A. Dowsland. Some experiments with simulated annealing techniques for packing problems. *European Journal of Operational Research*, 68 :389–399, 1993.

[21] U Dyckhoff, H.and Finke. *Cutting and Packing in Production and Distribution : Typology and Bibliography.* (eds. Mueller, W.A. and Schuster, P.) Springer-Verlag, New York, 1992.

[22] J. El hayek, A. Moukrim, et S. Negre. New resolution algorithm and pretreatments for the two-dimensional bin-packing problem. *Computers and Operations Research (in minor revision)*, 2006.

[23] J. El hayek, A. Moukrim, et S. Negre. Prétraitements et résolution approchée pour le problème de bin-packing en deux dimensions, le cas non orienté. *Congrès de la ROADEF, LILLE*, février 2006.

[24] J. El hayek, A. Moukrim, et S. Negre. A tabu search method for the non-oriented two-dimensional bin-packing problem. *INCOM conference, SAINT-ETIENNE*, may 2006.

[25] J. El hayek, A. Moukrim, et Negre S. Le problème de bin-packing en deux dimensions : nouveaux prétraitements et heuristique de résolution pour le cas non orienté. *Conférence MOSIM, RABAT*, avril 2006.

[26] O. Faroe, D. Pisinger, et M. Zachariasen. Guided local search for the three-dimensional bin-packing problem. *INFORMS J. Comput.*, 15 :267–283, 2003.

[27] S. Fekete et J. Schepers. New classes of fast lower bounds for bin packing problems. *Mathematical Programming*, 91 :11–31, 2001.

[28] S. Fekete et J. Schepers. A combinatorial characterization of higher-dimensional orthogonal packing. *Mathematics of Operations Research*, 29 :353–368, 2004.

[29] S. Fekete et J. Schepers. A general framework for bounds for higher-dimensional orthogonal packing problems. *Mathematical Methods of Operations Research*, 60 :311–329, 2004.

[30] S. Fekete, J. Schepers, et J. van der Veen. An exact algorithm for higher-dimensional orthogonal packing. *To appear in Operations Research*, 2006.

[31] M. R. Garey et D. S. Johnson. *Computers and intractability, a guide to the theory of NP-completeness*. Freeman, New York, 1979.

[32] P. Gilmore et R. Gomory. A linear programming approach to the cutting stock problem. *Ops. Res.*, 9 :849–859, 1961.

[33] P. Gilmore et R. Gomory. A linear programming approach to the cutting stock problem - part II. *Ops. Res.*, 11 :863–888, 1963.

[34] P. Gilmore et R. Gomory. Multistage cutting stock problems of two and more dimensions. *Ops. Res.*, 13 :94–120, 1965.

[35] F. Glover et M. Laguna. Tabu search. *Kluwer Academic Publishers*, 1997.

[36] E. Hadjiconstantinou et N. Christophides. An exact algorithm for general, orthogonal, two-dimensional knapsack problem. *European Journal of Operational Research*, 83 :39–56, 1995.

[37] M. Haouari et A. Gharbi. Fast lifting procedures for the bin packing problem. *Discrete Optimization*, 2 :201–218, 2005.

[38] J. Harwig et J.W. Barnes. An adaptive tabu search approach for 2-dimensional orthogonal packing problems. *to appear in Military Operations Research*, 2006.

[39] Mhand Hifi. An improvement of viswanathan and bagchi's exact algorithm for constrained two-dimensional cutting stock. *Computers and Operations Research*, 24 :727–736, 1997.

[40] S. Jakobs. On genetic algorithms for the packing of polygons. *European Journal of Operational Research*, 88 :165–181, 1996.

[41] D. S. Johnson. Near optimal bin packing algorithms, 1973. Dissertation, Massachussetts Institute of Technology, Cambridge, Massachussetts.

[42] M. Labbé, G. Laporte, et H. Mercure. Capacitated vehicle routing on trees. *Ops. Res.*, 39(6) :16–22, 1991.

[43] Z. Li et V. Milenkovic. Compaction and separation algorithms for non-convex polygons and their applications. *European Journal of Operational Research*, 84 :539–561, 1995.

[44] A. Lodi, S. Martello, et D. Vigo. Neighborhood search algorithm for the guillotine non-oriented two-dimensional bin packing problem. *Meta-Heuristics : Advances and Trends in Local Search Paradigms of Optimization*, S. Voss, S. Martello, I.H. Osman, and C. Roucairol, (eds.), kluwer Academic Publishers, Boston :125–139, 1998.

[45] A. Lodi, S. Martello, et D. Vigo. Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems. *INFORMS J. Comput.*, 11 :345–357, 1999.

[46] A. Lodi, S. Martello, et D. Vigo. Recent advances on two-dimensional bin packing problems. *Discrete Applied Mathematics*, 123 :379–396, 2002.

[47] S. Martello et P. Toth. Lower bounds and reduction procedure for the bin packing problem. *Discrete Applied Mathematics*, 28 :59–70, 1990.

[48] S. Martello et D. Vigo. Exact solution of the two-dimensional finite bin packing problem. *Management Sci.*, 44 :388–399, 1998.

[49] S. Ben Messaoud. *Caractérisation, Modélisation et Algorithmes pour des Problèmes de Découpe Guillotine.* PhD thesis, Université de Technologie de Troyes, France, 2005.

[50] G. Scheithauer. Equivalence and dominance for problems of optimal packing of rectangles. *Ricerca Operativa*, 83 :3–34, 1997.

[51] G. Wäscher, H. Haussner, et H. Schumann. An improved typology of cutting and packing problems. *European Journal of Operational Research*, to appear, 2006.

Titre :

Le problème de bin-packing en deux-dimensions, le cas non-orienté : résolution approchée et bornes inférieures

Résumé :

Notre travail porte sur le problème de bin-packing qui consiste à déterminer le nombre minimum de grands rectangles (*bins*) nécessaires pour ranger un ensemble de petits rectangles (*objets*). Ce problème d'optimisation combinatoire est NP-difficile au sens fort. Nous proposons des prétraitements des objets permettant la valorisation des espaces perdus dans les bins et la diminution de la taille du problème à résoudre. Nous proposons une nouvelle méthode d'évaluation de bornes inférieures tenant compte de la possibilité de tourner les objets de 90 degrés. Nous procédons à une résolution approchée du problème grâce à deux nouvelles méthodes : une heuristique et un algorithme de recherche tabou.

Mots-clés :

optimisation combinatoire, bin-packing, méta-heuristiques, bornes inférieures.

Title:

The two-dimensional bin-packing problem, the non-oriented case : resolution algorithms and lower bounds.

Abstract:

We treat the bin-packing problem which involves packing a given set of rectangles (*items*) into a minimum number of larger identical rectangles called *bins*. This combinatorial problem is strongly NP-hard. We propose pretreatments that allow the valorization of the lost areas in the bins and the reduction of the size problem. We attempt the resolution of the problem by computing lower and upper bounds. We propose a new scheme of computing lower bounds taking into account the possibility of rotating the items by 90 degrees. We propose a new heuristic method and a new tabu search algorithm for computing upper bounds.

Key-words:

combinatorial optimization, bin-packing, meta-heuristics, lower bounds.