Bornes inférieures et méthodes exactes pour le problème de bin packing en deux dimensions avec orientation fixe

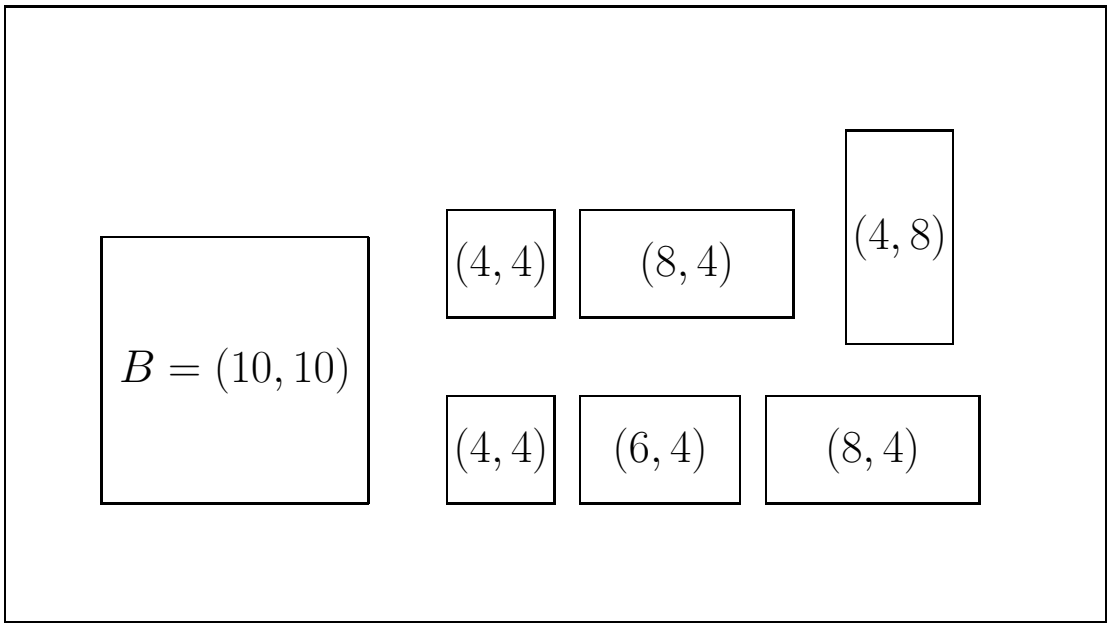
François Clautiaux

utc
Université de Technologie
Compiègne

par François Clautiaux

Bornes inférieures et méthodes exactes pour le problème de bin packing en deux dimensions avec orientation fixe

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



Soutenue le : 30 mars 2005
Spécialité : Technologie de l'information et des systèmes

Bornes inférieures et méthodes exactes pour le problème de bin packing en deux dimensions avec orientation fixe

Thèse soutenue le 30 mars 2005 devant le jury composé de :

M.	Chengbin Chu	PdU, Univ. de Technologie de Troyes	(Rapporteur)
M.	Gerd Finke	PdU, Univ. J. Fourrier, Grenoble	(Rapporteur)
M.	Edward Coffman	Prof., Univ. of Columbia, New York	
M.	Philippe Baptiste	CR, Prof. à l'école Polytechnique	
M.	Jacques Carlier	PdU, UTC, Compiègne	(Directeur de thèse)
M.	Aziz Moukrim	PdU, UTC, Compiègne	(Co-directeur de thèse)

Cette thèse est dédiée à Gaëlle pour de nombreuses raisons que je ne saurais lister ici.

Remerciements

Je souhaite tout d'abord remercier mes directeurs de thèse, Jacques Carlier et Aziz Moukrim. Ils ont toujours su encadrer mon travail tout en me laissant suffisamment de liberté. J'apprécie aussi tous leurs efforts pour m'aider à préparer l'avenir. Ces trois années passées à travailler ensemble me laisseront un formidable souvenir professionnel, mais aussi humain.

Je remercie aussi Chengbin Chu et Gerd Finke pour avoir accepté de rapporter ma thèse, ainsi que Philippe Baptiste et Edward Coffman qui ont bien voulu me faire l'honneur de faire partie du jury.

Je ne saurais trop remercier Stéphane Nègre sans qui ma carrière aurait sûrement pris une toute autre tournure. Tu m'as inspiré la vocation de chercheur, fait découvrir la théorie des graphes et la RO, mis le pied à l'étrier en DEA. Merci pour tout ; j'espère pouvoir rééquilibrer la balance un jour.

Cette thèse n'aurait peut-être pas eu lieu sans un précieux coup de pouce de Dritan Nace au moment opportun. Merci encore Dritan.

Merci aux chercheurs qui m'ont envoyé des fichiers de données, des articles ou des résultats numériques : Emmanuel Néron, Marco Boschetti, Anis Gharbi, Alberto Caprara, et Silvano Martello.

Ma thèse a aussi été l'occasion de discussions très enrichissantes avec de nombreux chercheurs et étudiants-chercheurs : Joseph Hayek, Antoine Jouglet, ainsi que David Savourey de l'UMR Heudiasyc, Emmanuel Néron, Fabrice Tercinet, et Christophe Lenté du laboratoire d'informatique de Tours.

Je souhaite aussi remercier Gaëlle Leveau, David Savourey et Marlyse Pelletier pour avoir relu ma thèse et permis d'en améliorer le style.

Je remercie tous ceux qui ont contribué à faire de ces trois dernières années les plus riches de ma courte existence : mes amis Antoine, David, Pierre-Alexandre, Marc, Julien, Astride et tous les autres. Je remercie aussi mes parents et mon frère Sylvain pour leur soutien.

Je tiens aussi à remercier les membres du laboratoire Heudiasyc et du LMAC, chercheurs, enseignants-chercheurs, ITA, IATOS et doctorants, qui ont grandement facilité mon intégration au sein du département de génie informatique.

Merci enfin à toi, lecteur, de t'intéresser à mon travail.

Table des matières

Liste des tableaux

Liste des figures

Introduction

Le secteur industriel est fréquemment confronté à des problèmes de découpe de matière première. Dans une optique de réduction des coûts et d'impact environnemental, l'objectif est de minimiser la quantité de matériau perdu dans les chutes. Dans le secteur des transports, le développement des activités de logistique provoque la multiplication de problèmes de conditionnement. La résolution de ces problèmes est un enjeu économique très important pour de nombreuses entreprises. Découpe et conditionnement sont des généralisations du problème classique de *bin-packing* en une dimension, qui a fait l'objet de nombreuses études dans la communauté de recherche opérationnelle. Le développement des applications industrielles pour ces généralisations en deux ou trois dimensions a conduit un nombre croissant de chercheurs à les étudier. Au cours des cinq dernières années, plusieurs thèses de doctorat ont eu pour objets des problèmes de découpe et de conditionnement de type *bin-packing*. Andrea Lodi [46], puis Michele Monaci [54], ont ainsi traité les problèmes classiques de bin-packing en une et deux dimensions. Plus récemment, Pierre Lemaire [44] a étudié le *rangement d'objets multi-boîtes* qui est une nouvelle variante des problèmes de bin-packing. Enfin, Saïd Ben Messaoud [3] a considéré des problèmes de découpe de type *guillotine*.

Dans cette thèse, nous nous intéressons à la résolution exacte d'un problème de bin-packing en deux dimensions que nous notons $2BP$: les articles sont rectangulaires, de dimensions entières, et la rotation n'est pas permise. L'objectif est de déterminer le nombre minimum de grands rectangles (ou *bins*) de même taille nécessaires pour placer un ensemble donné de rectangles (ou *articles*) sans chevauchement. Ce problème apparaît dans plusieurs applications industrielles, comme la découpe de tissu, de papier, de bois, d'acier ou de verre. L'objectif est de minimiser la quantité de matière première perdue dans les opérations de découpe. Interdire la rotation des articles peut être liée à l'existence de motifs sur les rectangles de matière première. De plus, les méthodes peuvent être adaptées à d'autres jeux de contraintes, comme l'obligation de coupes guillotine, ou la possibilité de changer l'orientation des articles. Elles sont également généralisables en trois dimensions.

Les heuristiques proposées dans la littérature construisent de bonnes solutions, mais les algorithmes existants ne permettent pas de prouver l'optimalité des solutions obtenues pour de nombreuses instances de référence. Nos principales contributions concernent les méthodes exactes. Deux chapitres sont consacrés à de nouvelles évaluations par défaut, qui s'appuient sur des classes de fonctions particulières pouvant être appliquées aux deux dimensions du problème. Une autre partie de notre travail est consacrée au problème de *faisabilité*, qui consiste à déterminer si un ensemble d'articles peut être rangé dans un bin unique. Pour ce problème, ainsi

que pour le problème d'optimisation $2BP$, nous proposons de nouveaux principes de séparation, ainsi que des bornes inférieures qui s'appliquent à des solutions partielles. Les performances de nos méthodes sont validées de manière théorique et pratique. Nous montrons tout d'abord que nos bornes inférieures dominent celles de la littérature, puis nous nous confrontons à la littérature sur des jeux d'essai de référence. Notre approche permet de fermer de nombreuses instances et d'améliorer l'efficacité de la résolution dans de nombreux cas.

Dans le Chapitre 1, nous décrivons le problème de bin-packing en deux dimensions, ainsi que ses variantes. Nous exposons brièvement les modèles existants et les différentes méthodes qui ont été employées pour résoudre $2BP$. Nous présentons les heuristiques proposées par Berkey et Wang [5], Lodi *et al.* [48], et Boschetti et Mingozzi [8], qui permettent de trouver de bonnes solutions rapidement. Nous décrivons aussi deux prétraitements de Martello et Vigo [53] et Boschetti et Mingozzi [7]. Nous développons particulièrement la partie concernant les bornes inférieures de la littérature, notamment les travaux de Fekete et Schepers [29, 31] et de Boschetti et Mingozzi [7, 8] qui sont fréquemment utilisés au cours de la thèse. Nous décrivons enfin les méthodes exactes les plus récentes, pour le $2BP$ [53] et plus particulièrement pour le problème de faisabilité [30, 53].

Dans le Chapitre 2, nous proposons tout d'abord des prétraitements qui réduisent la taille du problème. Ils reposent sur la détection de sous-ensembles d'articles pour lesquels une solution optimale peut être calculée. Le traitement consiste alors à supprimer ces articles du problème considéré, et à retenir le nombre de bins qui ont été nécessaires pour les ranger. Les méthodes qui permettent d'effectuer un tel traitement utilisent ce que nous nommons les *fonctions identiquement réalisables*. Nous comparons notre procédure de réduction à celle de Boschetti et Mingozzi [7] sur les jeux d'essai de la littérature : elle réduit la taille des instances de manière beaucoup plus importante. Nous consacrons aussi une grande partie du chapitre à de nouvelles bornes inférieures. Elles dominent les meilleures bornes de la littérature, proposées par Boschetti et Mingozzi [7, 8], de manière théorique et pratique. Nos méthodes reposent sur le concept de *fonctions dual-réalisables* (DFF), qui a été introduit par Johnson [42] et qui a déjà été utilisé pour des problèmes de bin packing en une [29, 50] et deux dimensions [31]. Nous nous intéressons à ces fonctions dans le cas discret, et introduisons le nouveau concept de *fonctions dual-réalisables dépendantes de la donnée* (DDFF), qui peuvent être appliquées à une instance particulière. L'idée est de calculer la surface totale des articles après application de telles fonctions sur les dimensions des articles. Nous proposons deux familles de DFF, ainsi qu'une DDFF, qui conduisent à des évaluations qui dominent théoriquement celles de la littérature. Nos méthodes sont validées numériquement sur les instances de référence [5, 53] : la dominance est confirmée, et les résultats sont améliorés pour plusieurs instances.

Le Chapitre 3 est consacré lui aussi aux bornes inférieures : nous comparons deux approches qui nous permettent d'améliorer les résultats du chapitre précédent en proposant de nouvelles DFF et DDFF. La première méthode de génération consiste à combiner itérativement les trois fonctions décrites dans le Chapitre 2. Nous montrons que l'ensemble des fonctions qui peuvent être obtenues de cette manière est dominé par un ensemble de fonctions de cardinalité polynomiale en fonction de la taille

du problème. La deuxième méthode repose sur un générateur de fonctions proposé par Carlier et Néron [12, 13]. Elle consiste à construire de manière énumérative un ensemble dominant de DFF dites maximales (MDFF). Nous utilisons les fonctions obtenues de cette manière et montrons qu'elles conduisent à de meilleures évaluations pour certains jeux d'essai. Il apparaît aussi que la composition de ces fonctions avec la DDFF du Chapitre 3 permet d'améliorer les résultats. Nous obtenons ainsi, à notre connaissance, les meilleures bornes inférieures polynomiales de la littérature pour $2BP$.

Le principal apport du Chapitre 4 est une méthode arborescente pour le problème de faisabilité, basée sur un nouveau principe de séparation et évaluation. On trouve deux approches très différentes dans la littérature : Martello et Vigo [53] cherchent à placer itérativement les articles dans le bin, tandis que Fekete et Schepers [30,32], utilisent une modélisation sous forme de graphes d'intervalles. Notre algorithme est plus proche de celui de Martello et Vigo : il consiste à résoudre dans un premier temps un problème relâché dans lequel on ne fixe que les abscisses des articles. Nous levons pour celà les contraintes d'intégrité liées à la hauteur des articles. Pour chaque solution trouvée à cette étape, nous cherchons une solution du problème initial. En pratique, lorsque le problème n'admet pas de solution, il est fréquent que le problème relâché n'en admette pas non plus. Pour améliorer l'efficacité de la méthode, nous proposons aussi des règles de dominance qui réduisent le nombre d'états explorés. Nous avons généré aléatoirement des instances difficiles du problème de faisabilité afin de valider notre algorithme : elle permet d'améliorer très sensiblement les résultats de Martello et Vigo [53] et se révèle très compétitive par rapport à la méthode de Fekete et Schepers [30].

Dans le Chapitre 5, nous proposons une méthode exacte pour la résolution du problème d'optimisation $2BP$ qui utilise toutes les techniques décrites dans les chapitres précédents. Contrairement à la séparation classique [53] qui affecte les articles itérativement dans chaque bin, nous cherchons à partionner l'ensemble des articles en deux ensembles. A chaque ensemble est associé une nouvelle instance de $2BP$ que nous traitons si possible de manière heuristique. Dans le cas contraire, la méthode exacte est appelée récursivement. L'algorithme utilise les bornes inférieures de manière systématique, il coupe ainsi de nombreuses branches dès les premiers niveaux. Nous adaptons aussi nos bornes pour les appliquer à une solution partielle. Ces bornes réduisent le nombre d'états énumérés dans plusieurs cas, mais leur coût algorithmique important réduit leur intérêt pratique. Notre méthode permet de résoudre plus d'instances que celle de la littérature. La qualité des bornes inférieures est primordiale dans ces résultats.

Les apports de la thèse se situent donc essentiellement dans le domaine des bornes inférieures, avec une étude des fonctions dual-réalisables. Les méthodes exactes proposées, ainsi que les procédures de réduction, permettent de traiter des problèmes qui ne l'étaient pas auparavant. La plupart des résultats que nous obtenons peuvent être généralisés à d'autres problèmes de bin-packing, comme celui où la rotation des articles est autorisée, ou adaptés à d'autres problèmes, comme le strip-packing, et le problème de découpe de type guillotine.

Des problèmes de bin-packing

1.1 Introduction

Le problème de bin-packing en deux dimensions ($2BP$) consiste à déter-miner le nombre minimum de grands rectangles identiques nécessaires pour ranger un ensemble de petits rectangles. Il a fait l'objet depuis quelques années d'une attention croissante pour deux raisons : outre son intérêt théorique, il a de nombreuses applications dans le domaine industriel, de la logistique, de l'informatique et même de l'édition. On retrouve le problème en deux dimensions essentiellement dans l'industrie du tissu, de l'acier, mais aussi dans le cadre de la découpe de bois, ou de verre. On peut citer plusieurs autres applications moins courantes : l'optimisation du placement de publicités dans un journal, ou l'allocation de tâches à un processeur reconfigurable.

Ce problème est NP-difficile car il généralise le problème classique de bin-packing en une dimension $1BP$. Ce dernier a été étudié par un grand nombre de chercheurs et est très bien traité dans le cas général par des heuristiques avec garantie de performance et des évaluations par défaut très efficaces. L'intérêt pour le problème en deux dimensions $2BP$ est plus récent car il se prête moins bien à des études théoriques. Néanmoins, depuis quelques années, un nombre croissant d'articles traitent de ce sujet et de ses variantes. La majeure partie des contributions concernent des méthodes heuristiques, mais on trouve aussi des bornes inférieures efficaces ainsi que des méthodes exactes très intéressantes.

Dans ce chapitre, nous présentons les résultats les plus récents pour le problème $2BP$ dans lequel les articles et le bin sont rectangulaires et d'orientation fixe. Les nombreuses variantes de $2BP$ dont nous citons quelques exemples ne sont pas traitées ici. Le problème qui nous intéresse possède des analogies très fortes avec des problèmes d'ordonnancement. Les deux types de problèmes différent essentiellement dans des contraintes d'intégrité supplémentaires pour $2BP$. Nous rappelons quelques résultats de la littérature obtenus pour le $1BP$, notamment les heuristiques et les bornes inférieures, qui peuvent être généralisées en deux dimensions. Nous présentons des modèles très différents pour les problèmes de bin-packing, qui vont de formulations linéaires à des modèles utilisant la théorie des graphes. Concernant les bornes inférieures, deux types d'approche sont possibles pour $2BP$: celles qui sont calculées globalement en fonction des caractéristiques des articles, et celles qui consistent à modifier les articles sur chaque dimension indépendamment. Comme nous le verrons dans la suite de ce document, il existe des liens très forts entre les deux approches. Les méthodes exactes de la littérature sont en général des méthodes

arborescentes en deux étapes : la première affecte les articles aux bins, et la deuxième vérifie s'il est possible de ranger l'ensemble d'articles obtenu dans ce bin. A notre connaissance, une seule séparation a été proposée pour la première étape, mais il existe en revanche deux types de méthodes pour la deuxième étape. L'une énumère itérativement les possibilités de placer des articles dans le bin, l'autre repose sur un modèle utilisant la théorie des graphes.

Dans un premier temps, nous définissons les données du problème (Section 1.2). Nous présentons quelques modèles courants dans la Section 1.3, puis les principaux résultats heuristiques (Section 1.4) avant d'étudier de manière plus précise les prétraitements dans la Section 1.5, les bornes inférieures dans la Section 1.6, et les méthodes exactes dans la Section 1.7.

1.2 Définitions, notations

Dans cette section, nous définissons les problèmes qui sont évoqués au cours de ce document, et plus particulièrement le problème de bin-packing en deux dimensions $2BP$. Nous décrivons aussi quelques problèmes qui sont évoqués dans ce document : le bin-packing en une dimension, quelques problèmes de conditionnement et de découpe, ainsi qu'un problème d'ordonnancement cumulatif proche de $2BP$.

1.2.1 Bin-packing en une et deux dimensions

Le problème de bin-packing en une dimension ($1BP$) consiste à minimiser le nombre de containers en une dimension (bins) nécessaires pour ranger une liste d'articles caractérisés par leur longueur. Ce problème est NP-complet [34]. Une instance de $1BP$, notée D, est une paire (A, C) où $A = a_1, \ldots, a_n$ est la liste des articles à ranger et C une valeur positive. Chaque article a_i possède une longueur c_i inférieure à C. On considére le cas où les données sont entières.

Le problème de bin-packing en deux dimensions ($2BP$) est une généralisation naturelle de $1BP$. Il s'agit de minimiser le nombre de grands rectangles (bins) identiques pour ranger une liste d'articles rectangulaires. Les articles doivent être rangés de telle manière que les côtés des rectangles soient parallèles à ceux du bin. On note $D = (A, B)$ une instance de $2BP$, $A = a_1, \ldots, a_n$ est la liste des articles à ranger, B représente le bin. Chaque article a_i a une largeur w_i et une hauteur h_i (w_i et h_i entiers). Le bin B possède lui aussi une largeur W et une hauteur H. Lorsque l'on range un article a_i dans un bin, on note x_i et y_i les coordonnées horizontale et verticale de son coin bas-gauche. Dans le problème qui nous intéresse, les articles ont une orientation fixe et doivent être rangés de telle manière que leurs côtés soient parallèles à ceux du bin. Dans les exemples, un article de largeur w_i et de hauteur h_i est noté (w_i, h_i). Un exemple simple de problème de bin-packing en deux dimensions est illustré par la Figure 1.1. Le bin B est de taille $(10, 10)$ et la liste A d'éléments à placer est constituée de six articles. $A = \{(4,4), (4,4), (6,4), (8,4), (8,4), (4,8)\}$. Une solution pour cette instance est représentée par la Figure 1.2.

On appelle *configuration* une liste de coordonnées pour les articles de A. Si ces coordonnées permettent d'éviter tout chevauchement entre deux articles ou d'un

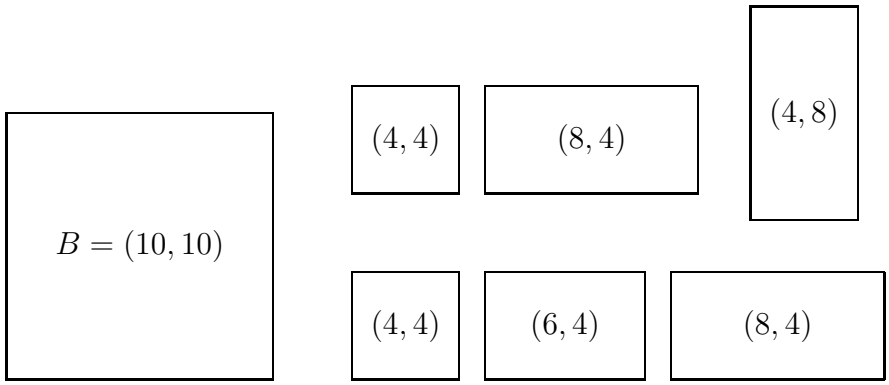


Figure 1.1 – Une instance de $2BP$

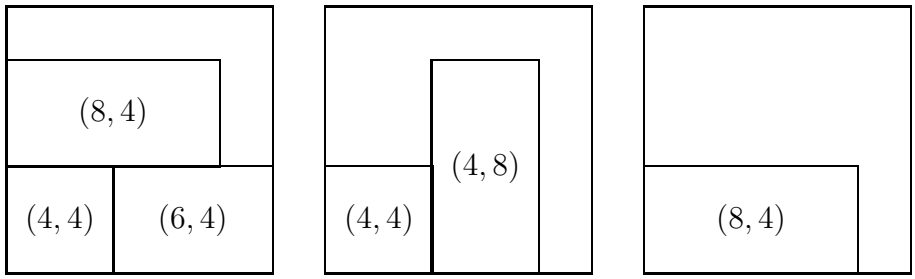


Figure 1.2 – Une solution pour notre instance de $2BP$

article avec le bin, on dit que c'est une configuration *réalisable*. Une des différences fondamentales entre $1BP$ et $2BP$ réside dans **le problème de *faisabilité*** : une fois qu'un ensemble A' est affecté à un bin B_p donné, existe-t-il une configuration réalisable pour les articles de A' dans un bin de type B ? Alors que ce problème est trivial en une dimension (il suffit de sommer la longueur des articles), il est NP-complet en deux dimensions. On a donc deux problèmes imbriqués à résoudre : un problème d'affectation, et plusieurs problèmes de faisabilité. Le problème de faisabilité apparait aussi dans d'autres problèmes de rangement/découpe, et a fait l'objet d'une attention particulière.

1.2.2 Problèmes de conditionnement et de découpe

Il existe un grand nombre de variantes pour les problèmes conditionnement et de découpe, en fonction de la dimension (1, 2, 3, ou plus), la connaissance a priori des articles (on-line ou off-line), de la forme des articles et des bins (carrés, rectangulaires, circulaires, convexes ou quelconques), de la possibilité de modifier l'orientation des articles, ou du type de machine utilisé dans le cas des problèmes de découpe (guillotine ou non). Le lecteur peut se référer à la bibliographie annotée de Dyckhoff [28] qui liste un grand nombre de ces problèmes et des articles qui en traitent. Dans cette section, nous nous concentrons sur les problèmes en deux dimensions où les articles et les bins sont rectangulaires. Pour une typologie précise des problèmes de packing et de découpe, le lecteur pourra se référer aux travaux de Dowsland [26], ainsi que de Wäscher *et al.* [57], qui ont enrichi ceux de Dyckhoff [27]. Nous présentons maintenant les problèmes qui sont évoqués dans la présente thèse, sans être traités spécifiquement.

Le problème subset-sum n'est pas à proprement parler un problème *packing*, mais il apparait parfois comme un sous-problème à résoudre pour obtenir des évaluations par défaut. La donnée est composée d'une liste de valeur L et d'une valeur Xů L'objectif est de sélectionner dans L une liste de valeurs dont la somme est la plus grande qui soit inférieure à X.

Le problème de strip-packing $2SP$ est une variante très étudiée du bin-packing en deux dimensions. On dispose d'une liste A d'articles et d'une container unique de largeur W et de hauteur infinie. L'objectif est de ranger tous les articles de A dans le container en minimisant la hauteur totale H^*.

Dans **le problème de sac à dos** $1KP$, on dispose d'un ensemble de types d'articles, d'un nombre d'occurences pour chaque type, ainsi que d'une taille de bin. Chaque article possède en outre un coût. L'objectif est de ranger dans le bin un ensemble d'articles qui maximise la somme des coûts associés. **Le problème de sac à dos en deux dimensions** $2KP$, est la généralisation en deux dimensions de $1KP$ où les articles et les bins sont rectangulaires.

Dans **le problème de rectangle packing** $2RP$, on dispose d'une liste A d'articles à ranger et d'une liste de types de bins $(W_1, H_1), (W_2, H_2), \ldots, (W_k, H_k)$. L'objectif est de trouver le bin de plus petite taille qui puisse accueillir la totalité des articles de A. Ce problème repose essentiellement sur des problèmes de faisabilité.

Une grande partie des méthodes proposées pour $2BP$ peuvent l'être aussi pour **le problème de bin-packing en trois dimensions** $3BP$. Dans [51], par exemple,

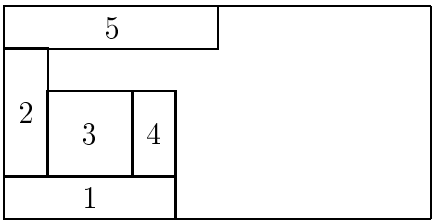


Figure 1.3 – Une solution pour un problème de faisabilité

Martello *et al.* généralisent en trois dimensions les travaux de Martello et Vigo [53] en deux dimensions). De même, Boschetti [6] a généralisé ses résultats obtenus sur les bornes inférieures en deux dimensions [7,8] au problème en trois dimensions. Pour les problèmes en trois dimensions, comme le **container loading**, on trouve souvent des contraintes supplémentaires comme l'équilibre, le voisinage, *etc.*. On parle aussi de $4BP$ (la quatrième dimension est le temps).

1.2.3 Lien avec un problème d'ordonnancement

Les problèmes de packing et d'ordonnancement partagent certaines propriétés (voir [33] par exemple). Le problème de faisabilité est fortement lié à un **problème d'ordonnancement cumulatif** que l'on note CuSP. Il s'agit d'ordonnancer une liste de tâches i de durée t_i et de demande b_i. La demande est la quantité de la ressource unique (dont B unités sont disponibles à chaque instant) nécessaire pour pouvoir exécuter la tâche i. On considère la version du problème où il n'y a ni dates de début et de fin, ni relations de précédence entre les tâches. Il est très proche du problème de faisabilité. En effet, on peut considérer le temps comme la première dimension d'un problème $2BP$ et la ressource comme la deuxième dimension. Néanmoins, les deux problèmes ne sont pas identiques : le problème d'ordonnancement est une relaxation pour le problème de faisabilité. En effet, deux portions d'une même tâche peuvent être ordonnancées sur deux parties de la ressource non contiguës. Or, cela est impossible pour le problème de faisaibilité, car on doit respecter l'intégrité des articles. La Figure 1.3 représente un problème de faisabilité, et la Figure 1.4 représente un ordonnancement cumulatif correspondant. Dans le CuSP, il suffit de fixer les dates de début de chaque tâche en respectant la contrainte de ressource : il n'est pas utile de savoir à quelle(s) partie(s) de la ressource on l'affecte. On notera que pour une solution au problème d'ordonnancement, on peut dériver un grand nombre de solutions pour le problème de faisabilité en énumérant toutes les ordonnées possibles pour les abscisses fixées dans le problème d'ordonnancement. Malgré cette différence, des méthodes peuvent être communes aux deux problèmes. Par exemple, Carlier et Néron [13] utilisent pour le CuSP des techniques pour obtenir des bornes inférieures analogues à celles utilisées par Fekete et Schepers [29,30] pour des problèmes de bin-packing.

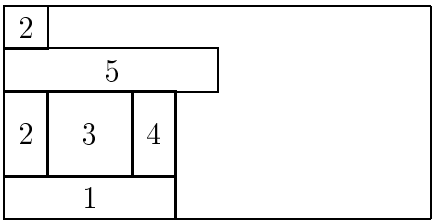


Figure 1.4 – Une solution pour un problème CuSP

1.3 Modèles

On ne connait pas de modèle linéaire efficace lorsque le nombre d'articles est grand. Nous proposons donc plusieurs modélisations pour $2BP$ et plus précisément pour le problème de faisabilité qui permettent d'appliquer des algorithmes exacts ou approchés. Le programme linéaire proposé s'applique à $2BP$, alors que les autres ne s'appliquent qu'au problème de faisabilité. Le modèle à base de graphes d'intervalles est avant tout dédié à des résolutions exactes, et la modélisation sous forme de permutation ne peut amener qu'à une heuristique. Le modèle utilisant des listes de coordonnées est le plus répandu : il est utilisé dans des résolutions exactes, mais aussi approchées.

1.3.1 Modèles linéaires pour $2BP$

Nous présentons maintenant un modèle linéaire très connu pour $2BP$ qui a été présenté par Gilmore et Gomory [37] comme une extension de leur approche pour le bin-packing en une dimension [35,36]. L'idée est d'énumérer tous les sous-ensembles S_k d'articles de A qui peuvent être rangés dans un bin (*i.e.* toutes les configurations réalisables pour la liste d'articles). Notons M la cardinalité de cet ensemble. Une solution est équivalente à une liste de configurations réalisables. Pour modéliser ces configurations, on crée pour chaque configuration S_k un vecteur colonne contenant n éléments $s_{k,i}$ $(i = 1, \ldots, n)$ qui prennent la valeur 1 lorsque l'article a_i appartient à la configuration k, 0 sinon. L'ensemble des configurations possibles est représentée par la matrice S qui est composée des vecteurs S_k pour $k = 1, \ldots, M$. On cherche à minimiser la fonction objectif suivante :

$$min \sum_{k=1}^{M} x_k \tag{1.1}$$

Sous les contraintes

$$\sum_{k=1}^{M} s_{ik} x_k = 1 \quad (i = 1, \ldots, n) \tag{1.2}$$

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

La faiblesse de cette modélisation réside dans le très grand nombre de colonnes. Gilmore et Gomory [37] ont présenté des méthodes pour générer dynamiquement les colonnes en utilisant une résolution pseudo-polynomiale de problèmes de sac-à-dos en une dimension $1KP$. On peut aussi considérer les configurations dépendant du *type* des articles, sans prendre en compte le nombre d'occurences.

1.3.2 Modèles pour le problème de faisabilité

Un grand nombre de méthodes exactes ou heuristiques considèrent une solution pour le problème de faisabilité comme une suite de coordonnées des articles dans le bin. Toutefois, il existe d'autres manières de modéliser le problème qui peuvent être mieux adaptées en fonction de la méthode utilisée.

1.3.2.1 Modélisation par une liste de coordonnées

Une solution est un vecteur contenant les coordonnées (x_i, y_i) de chaque article a_i. Les coordonnées doivent respecter les contraintes suivantes. Ce premier jeu de contraintes assure que les articles sont bien contenus dans le bin.

$$x_i \geq 0, \forall a_i \in A \tag{1.4}$$
$$y_i \geq 0, \forall a_i \in A \tag{1.5}$$
$$x_i + w_i \leq W, \forall a_i \in A \tag{1.6}$$
$$y_i + h_i \leq H, \forall a_i \in A \tag{1.7}$$

Pour chaque couple d'articles a_i et a_j de A, une des relations suivantes doit être vraie, sinon les articles se chevauchent. Pour chaque paire a_i et a_j, a_i est soit à gauche, soit à droite, soit au dessus, soit en dessous de a_j.

$$x_i + w_i \leq x_j \tag{1.8}$$
$$x_j + w_j \leq x_i \tag{1.9}$$
$$y_i + h_i \leq y_j \tag{1.10}$$
$$y_j + h_j \leq y_i \tag{1.11}$$

En règle générale, on s'assure que les conditions sont respectées pour un article donné au moment où l'on fixe ses coordonnées.

1.3.2.2 Graphes d'intervalles

Fekete et Schepers [30–32] ont proposé un nouveau modèle sous forme de graphes d'intervalles. Ils montrent que deux graphes d'intervalles peuvent être associés à une *classe de configurations* (*i.e.* un ensemble de configurations partageant certaines propriétés). L'intérêt de ce concept est qu'un grand nombre de configurations symétriques sont évitées car une seule est réellement énumérée pour chaque classe. Un graphe $G_d = (V, E_d)$ de taille n est associé à chaque dimension du problème, où n

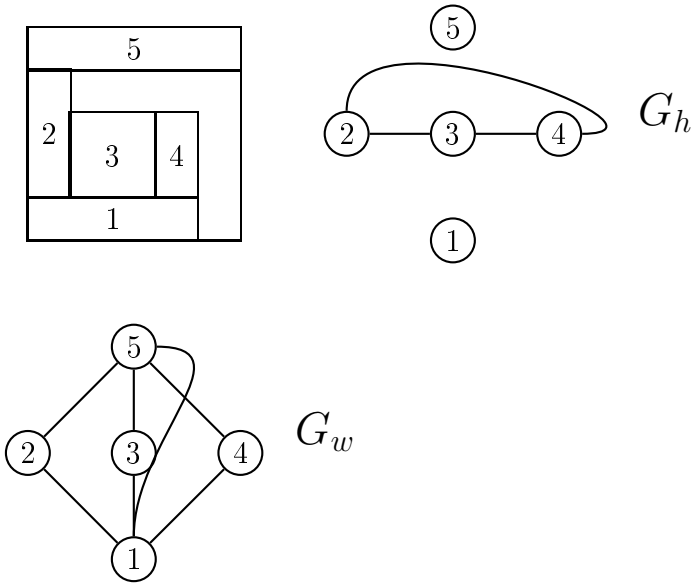


Figure 1.5 – Modélisation sous forme de graphes d'intervalles

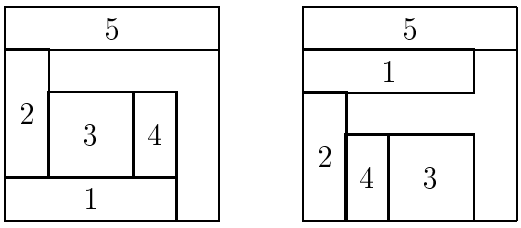
est le nombre d'articles dans le bin et $d \in \{w, h\}$ la dimension considérée. Dans les deux graphes, chaque sommet v_i est associé à un article a_i. Une arête est ajoutée dans G_w (respectivement dans G_h) entre deux sommets v_i et v_j si les projections des articles a_i et a_j sur l'axe horizontal (respectivement vertical) entrent en intersection (voir Figure 1.5). Les auteurs montrent qu'une classe de configurations associée à un couple de graphes d'intervalles G_w et G_h est faisable si les trois conditions suivantes sont respectées. Les deux premières permettent de vérifier que les articles sont bien contenus dans le bin, et la troisième contrainte permet de s'assurer qu'il n'y a pas de chevuachement entre deux articles.

1. chaque ensemble stable S_w de G_w est tel que $\sum_{v_i \in S_w} w_i \leq W$
2. chaque ensemble stable S_h de G_h est tel que $\sum_{v_i \in S_h} h_i \leq H$
3. $E_w \cap E_h = \emptyset$

Les algorithmes pour reconnaitre ces graphes sont polynomiaux (voir [39] pour plus de détails).

1.3.2.3 Permutation des articles

Certaines méthodes utilisent une permutation σ des articles pour calculer une configuration. A partir d'une permutation, on peut obtenir une configuration en appliquant un algorithme qui dépend de l'ordre d'examen des articles (voir Figure 1.6). La méthode la plus classique est l'algorithme *Bottom-Left* (BL) [15], qui consiste à placer à chaque étape le prochain article de la liste dans la position la plus en bas - à gauche du bin. Cet algorithme est décrit plus en détail dans la section concernant les heuristiques. Il a été montré [1, 10] qu'il existait des instances pour lesquelles aucun



$[1, 2, 5, 3, 4]$ $[2, 1, 5, 4, 3]$

Figure 1.6 – Modélisation à l'aide d'une permutation

ordre ne pouvait amener à la solution optimale en utilisant l'heuristique BL. Bien que cette stratégie ne soit pas optimale dans le cas général, elle permet de trouver de bonnes solutions dans de nombreux cas.

1.4 Méthodes heuristiques

Contrairement à $1BP$ pour lequel on trouve essentiellement des algorithmes avec garantie de performance (voir [22, 24, 25]), les méthodes proposées pour $2BP$ sont en général des heuristiques. Pour une synthèse très complète des travaux réalisés en une dimension, se référer aux travaux de Coffman *et al.* [23]. Le lecteur intéressé par les schémas d'approximation peut étudier les travaux de Sleator [56] et Zhang [58] pour $2BP$, et de Golan [38] pour le problème à orientation variable.

Au cours de la présente thèse, nous avons essentiellement travaillé sur des méthodes exactes pour $2BP$. Pour cette raison, nous n'utilisons que les heuristiques qui fournissent de bons résultats en pratique, même si elles n'ont pas de garantie de performance. Elles peuvent être divisées en deux catégories : les méthodes en une ou en deux phase(s). Nous commençons par décrire quelques approches en deux phases qui permettent l'adaptation de résultats obtenus pour le problème de strip-packing $2SP$. Nous présentons ensuite quelques méthodes classiques en une phase. Des synthèses complètes sur le sujet ont été réalisées par Lodi *et al.* [47, 49], nous y ajoutons les résultats obtenus plus récemment par Boschetti et Mingozzi [8].

Pour améliorer la compréhension, nous décrivons au préalable deux algorithmes classiques utilisés pour $1BP$, qui sont soit généralisés pour $2BP$, soit utilisés comme sous-routine dans des algorithmes dédiés au $2BP$. L'algorithme **First Fit Decreasing** fonctionne de la manière suivante : les articles a_i sont considérés dans l'ordre décroissant de leur longueur c_i. A chaque étape i, l'algorithme place l'article a_i dans le premier bin qui peut l'accueillir. L'algorithme **Best Fit Decreasing** fonctionne de manière similaire : la seule différence réside dans le choix du bin dans lequel l'article courant est placé. On utilise cette fois le bin le plus rempli qui puisse l'accueillir.

1.4.1 Méthodes en deux phases

Les méthodes en deux phases fonctionnent de la manière suivante. Dans un premier temps, on cherche une solution pour le problème de strip-packing $2SP$ suivant : l'ensemble des articles est l'ensemble original A, et la largeur du bin est

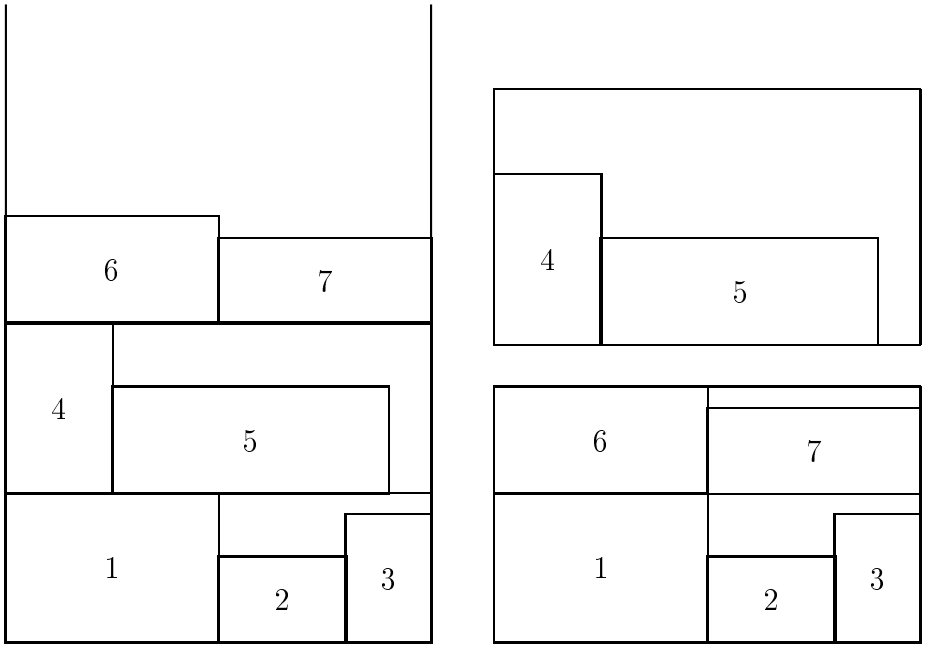


Figure 1.7 – Méthode heuristique en deux phases

égale à W. On détermine une solution pour le problème $2BP$ original en découpant la bande obtenue en bins de hauteur H. Les méthodes qui suivent diffèrent par les algorithmes utilisés dans les deux phases. La Figure 1.7 est une illustration du concept de méthode en deux phases. La partie gauche de la figure représente la solution pour le problème de strip-packing généré et la partie droite la solution pour $2BP$ obtenue en *rangeant* les bandes créées dans des bins. Dans la majorité des cas, le premier algorithme range les articles par *niveaux*. Lorsqu'un article a_i est rangé à gauche sur une nouvelle ordonnée y_i, on cherche à ranger les articles suivants dans la bande horizontale située entre y_i et $y_i + h_i$. Cette méthode permet de ne gérer que des bandes de largeur W et de hauteur variable dans la deuxième étape, et donc de pouvoir le traiter comme un $1BP$.

La méthode Hybrid Best Fit (HBF) a été proposée par Berkey et Wang en 1987 [5]. Au cours de la première phase, la stratégie *Best Fit Decreasing Height* est employée : un article est placé le plus à gauche au niveau dans lequel il va minimiser l'espace horizontal vacant. Si aucun niveau ne peut le contenir alors on en crée un nouveau. Au cours de la deuxième phase, c'est l'heuristique *Best Fit Decreasing* qui est utilisée. On range le niveau courant dans le bin qui peut le contenir pour lequel l'espace vertical vacant va être minimisé. On peut définir la méthode Hybrid First Fit (HFF) de manière équivalente en remplaçant *Best Fit* par *First Fit*.

Lodi *et al.* [48] ont proposé en 1998 **la méthode Floor-Ceiling** (FC). Au cours de la première phase, la stratégie est la même que dans la méthode précédente. Toutefois, lorsqu'un niveau est plein, on se permet de le *retourner*, échangeant le haut et le bas, la gauche et la droite, puis de recommencer à ranger des articles à gauche. Au cours de la deuxième phase, l'algorithme *Best Fit Decreasing* est utilisé, ou un algorithme exact pour $1BP$. Les mêmes auteurs proposent aussi **la méthode**

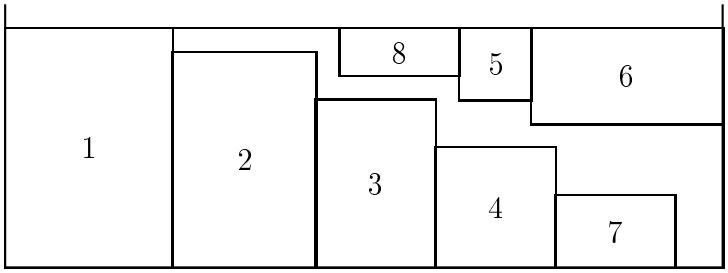


Figure 1.8 – Heuristique FC

Knapsack packing (KP). Lorsqu'un niveau est initialisé, on le remplit en entier avant de passer au suivant. Pour ce faire, on range le plus grand article restant et on remplit au maximum le niveau créé à l'aide de l'algorithme de résolution du sac-à-dos en une dimension $1KP$, qui a une complexité pseudo-polynomiale de $O(Wn)$.

1.4.2 Méthodes en une phase

Les méthodes en une phase consistent à ranger les articles itérativement dans les bins. Deux règles sont à définir : l'ordre dans lequel les articles sont examinés, le bin et la position dans laquelle on cherche à les placer en priorité.

Une des méthodes les plus naturelles consiste à généraliser l'algorithme classique dédié au $1BP$ *First Fit Decreasing*. Cette méthode fonctionne de la manière suivante : on considère les articles par ordre décroissant de surface, et on cherche toujours à placer l'article courant le plus en bas - à gauche possible dans le bin. On parle de l'algorithme **bottom-left decreasing**. Chazelle [15] en a proposé une implantation efficace.

Berkey et Wang [5] proposent eux aussi une généralisation de l'algorithme First-Fit qu'ils nomment **Finite First Fit** (FFF). Celle-ci utilise un rangement en niveau comme les algorithmes en deux phases. L'algorithme range les articles dans le premier niveau qui peut les contenir.

Alternate Directions (AD) [48] est proche de la méthode FC qui fonctionne en deux phases. Ce nouvel algorithme s'initialise avec $LB(A)$ bins, où $LB(A)$ est une borne inférieure pour le nombre de bins nécessaires au rangement des articles de A. Il commence par placer en bas de ces bins des articles selon la règle BFD (*Best Fit Decreasing*). Les articles restants sont rangés un par un en bandes allant alternativement de gauche à droite et de droite à gauche. Lorsqu'un article ne peut être placé dans aucune direction dans le bin courant, on l'affecte au prochain bin initialisé ou on construit un nouveau bin. Les trois algorithmes de Lodi *et al.* [48] (FC, KP et AD) retournent de meilleurs résultats que Berkey et Wang sur les instances de Berkey et Wang ainsi que sur celles de Martello et Vigo.

Boschetti et Mingozzi [8] proposent **l'heuristique HBP**. L'idée est de considérer les articles dans un ordre fixé au préalable. Initialement, on ne considère qu'un seul bin. Quand on ne peut plus placer aucun article dans le bin courant, on le *ferme*, et il ne sera plus reconsidéré. Dans ce cas, on *ouvre* un nouveau bin et on

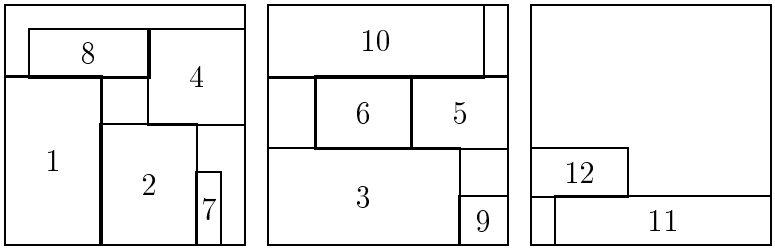


Figure 1.9 – Heuristique Alternate Directions

répète l'opération tant que tous les articles ne sont pas rangés. L'heuristique dépend fortement de l'ordre dans lequel les articles sont considérés. Pour cela, les auteurs proposent de relancer la méthode avec des ordres différents. A chaque itération, l'ordre est déterminé en fonction d'un paramètre lié aux articles, pondéré par une quantité aléatoire. Les critères retenus sont les suivants : la surface, la hauteur, la largeur et le périmètre. La part de l'aléatoire est déterminée au préalable.

1.5 Prétraitements

Le problème de bin-packing en deux dimensions $2BP$ est difficile à résoudre de manière exacte lorsque la taille du problème devient grande. Pour cette raison, il est intéressant de réduire la taille du problème en effectuant des rangements optimaux d'articles. Les prétraitements que nous décrivons ci-dessous ont un intérêt différent. Celui de Martello et Vigo [53] permet de réduire la taille de l'instance, tandis que celui de Boschetti et Mingozzi [7] permet d'augmenter la taille des articles et ainsi d'améliorer les évaluations.

1.5.1 Le prétraitement de Martello et Vigo

Martello et Vigo [53] proposent deux prétraitements qui peuvent aussi s'appliquer en cours de méthode quand on a placé un article de grande taille a_i dans un bin. L'idée est de trouver le *meilleur* ensemble d'articles que l'on peut placer avec a_i. Les auteurs proposent des méthodes élémentaires pour les cas où un ou deux articles au maximum peuvent être placés avec a_i.

Pour un article donné a_i, soit $Q^{a_i} = \{a_j \in A : h_i + h_j \leq H\} \cup \{a_j \in A : w_i + w_j \leq W\}$, l'ensemble des articles compatibles avec a_i. Si $Q^{a_i} = \emptyset$, le prétraitement effectué est immédiat, dans le cas contraire, les auteurs proposent de calculer une borne supérieure k pour le nombre d'articles qui peuvent être rangés avec a_i.

1. Si $Q^{a_i} = \emptyset$, alors on ne peut ranger aucun article avec a_i. Cet article forme donc un sous-problème trivial et peut être ranger seul dans un bin.
2. Si $Q^{a_i} \neq \emptyset$, on considère a_j le plus grand article de Q^{a_i}. Un prétraitement est appliqué dans deux cas de figure :

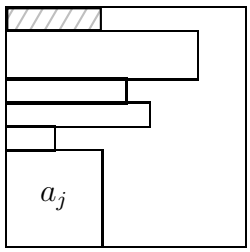


Figure 1.10 – Prétraitement de Boschetti et Mingozzi

(a) si $k = 1$, et que a_j a une plus grande hauteur et une plus grande largeur que tous les autres articles de Q^{a_i}

(b) si $k = 2$ et tous les couples de pièces de Q^{a_i} différentes de a_j peuvent être rangés dans une surface égale à celle de a_j.

Dans les deux cas, a_i et a_j sont placés dans le même bin et ne sont plus examinés par la suite. Cette méthode peut être généralisée à de plus grandes valeurs de k, mais le coût algorithmique devient rapidement trop élevé.

1.5.2 Le prétraitement de Boschetti et Mingozzi

Le prétraitement qui suit est proposé par Boschetti et Mingozzi [7]. Pour un article donné a_j, on peut calculer la largeur minimum obligatoirement perdue lorsque cet article est placé dans un bin. Si on note w_j^* cette valeur, w_j peut être modifiée de la manière suivante.

$$w_j \leftarrow w_j + (W - w_j^*) \tag{1.12}$$

La zone hachurée dans la Figure 1.10 représente le gain obtenu. La valeur w_j^* est la valeur optimale du problème *subset sum* suivant.

$$w_j^* = w_j + max \left\{ \sum_{a_i \in A - \{a_j\}} w_i \xi_i : \sum_{a_i \in A - \{a_j\}} w_i \xi_i \leq W - w_j, \xi_i \in \{0, 1\} \right\} \tag{1.13}$$

Une méthode pseudo-polynomiale classique peut calculer w_j^* en $O(W \times n)$. La procédure de réduction dépend de l'ordre dans lequel les articles a_j sont considérés. Un critère heuristique peut être utilisé pour choisir cet ordre. Par exemple, il est intéressant d'accroître la largeur des articles qui ont une grande hauteur afin d'obtenir une plus grande surface totale. Une procédure similaire est obtenue en considérant la hauteur à la place de la largeur. Lorsqu'un article a été traité, tous les articles qui faisaient partie de la solution du problème *subset-sum* ne peuvent pas être modifiés.

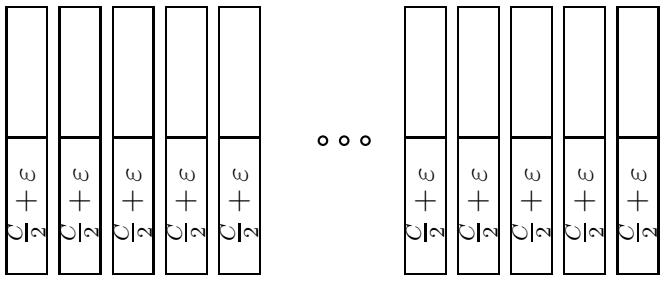


Figure 1.11 – Un exemple défavorable pour la borne continue L_0

1.6 Bornes inférieures

Dans cette section, nous commençons par rappeler des résultats existants pour le problème de bin-packing en une dimension $1BP$. Certains peuvent nous inspirer des méthodes pour $2BP$ et d'autres, qui utilisent les *fonctions dual-réalisables*, définies ultérieurement dans ce chapitre, peuvent être directement appliqués au $2BP$. De plus, certaines bornes pour le problème en deux dimensions utilisent des évaluations pour le bin-packing en une dimension. La formulation que nous livrons ici des bornes inférieures est modifiée par rapport à l'original pour plus de cohérence avec nos notations.

1.6.1 Bornes inférieures pour $1BP$

On peut distinguer deux familles d'évaluations pour $1BP$: celles qui reposent sur une analyse globale de l'instance, et celles qui sont uniquement obtenues à partir de la taille des articles.

1.6.1.1 La borne continue L_0

La borne continue est calculée en sommant les longueurs des articles et en divisant le résultat par la taille d'un bin. $L_0 = \lceil \frac{\sum_{a_i \in A} c_i}{C} \rceil$. Cette borne fournit de bons résultats lorsque les articles sont petits ou uniformément répartis sur $[0, C]$ [4], mais peut se révéler très mauvaise dans les autres cas. Martello et Vigo [53] ont montré qu'elle pouvait atteindre $\frac{1}{2}OPT(A)$ pour $1BP$ dans le pire des cas (voir Figure 1.11). Elle ne tient pas du tout compte des incompatibilités entre articles. L'objectif des évaluations par défaut est de calculer une approximation de la surface perdue de cette manière.

1.6.1.2 Borne de Martello et Vigo

Martello et Vigo [53] proposent une borne L_{1BP}^{MV} pour $1BP$. Ils introduisent une décomposition de l'ensemble A des articles en trois sous-ensembles dépendant d'un paramètre donné k : les *grands* articles (A_{gr}), les articles *moyens* (A_{moy}), et les *petits* articles (A_{pt}^s). Soit k un entier, $1 \leq k \leq \frac{1}{2}C$, $A_{gr} = \{a_i \in A : c_i > C - k\}$,

$A_{moy} = \{a_i \in A : C - k \geq c_i > \frac{1}{2}C\}$ et $A_{pt} = \{a_i \in A : \frac{1}{2}C \geq c_i \geq k\}$. Les articles les plus petits ne sont pas considérés. L_{1BP}^{MV} est égale au maximum entre les deux bornes inférieures L_α et L_β que nous décrivons ici.

La borne L_α a été initialement proposée par Martello *et al.* [52]. Elle repose sur l'observation suivante : il faut au moins autant de bins que de *grands* articles et d'articles *moyens* pour ranger les articles de A. S'il n'y a pas beaucoup d'articles moyens, la borne est égale à la somme du nombre de grands articles et de la borne continue calculée sur les articles restants.

$$L_\alpha = \max_{1 \leq k \leq \frac{1}{2}C} \left\{ |A_{gr} \cup A_{moy}| + max \left\{ 0, \left\lceil \frac{\sum_{a_i \in A_{moy} \cup A_{pt}} c_i}{C} - |A_{moy}| \right\rceil \right\} \right\} \quad (1.14)$$

Dans l'expression de la borne L_β, le numérateur de la partie droite est une évaluation par défaut du nombre d'articles de A_{pt} qui ne peuvent pas être rangés avec un article de A_{moy}. Le dénominateur est une évaluation par défaut de la place occupée par chaque article de A_{pt} dans un bin. Le résultat est donc une estimation du nombre de bins nécessaires pour ranger les petits articles supplémentaires.

$$L_\beta = \max_{1 \leq k \leq \frac{1}{2}C} \left\{ |A_{gr} \cup A_{moy}| + max \left\{ 0, \left\lceil \frac{|A_{pt}| - \sum_{a_i \in A_{moy}} \left\lfloor \frac{C - c_i}{k} \right\rfloor}{\left\lfloor \frac{C}{k} \right\rfloor} \right\rceil \right\} \right\} \quad (1.15)$$

Notons L_{1BP}^{MV} le maximum entre L_α et L_β.

1.6.1.3 Bornes de Fekete et Schepers

Fekete et Schepers [29] utilisent le concept de fonctions dual-réalisables (DFF) proposées par Johnson [42] et exploitées par Lueker [50] et Chao *et al.* [14] pour calculer des évaluations par défaut pour $1BP$. Ces fonctions ont la propriété suivante : pour tout ensemble de réels, si leur somme est inférieure à un, alors cette propriété reste vraie après transformation.

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

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

Une manière d'améliorer la borne continue L_0 est de modifier la taille des articles en utilisant une DFF. L'intérêt de la méthode est d'augmenter la taille de certains articles, en réduisant la taille d'autres articles de manière à obtenir une surface totale plus importante que dans l'instance originale. Les méthodes utilisant les DFF considèrent toujours un bin de longueur unitaire et des articles dont la longueur est un réel compris entre 0 et 1. Pour pouvoir utiliser ces fonctions, il faut donc projeter les dimensions dans l'intervalle $[0, 1]$. En utilisant des DFF sur un $1BP$, toute borne inférieure pour l'instance obtenue est aussi une borne inférieure pour l'instance initiale. Plusieurs articles traitent des DFF et de bornes inférieures pour

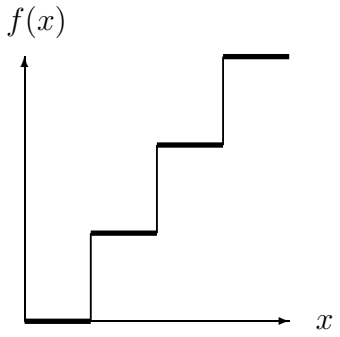


Figure 1.12 – La fonction $u^{(k)}$

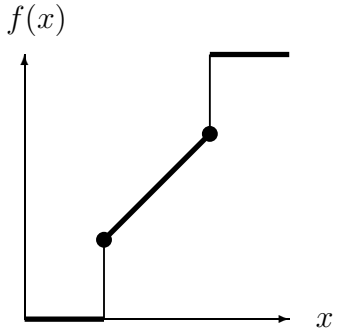


Figure 1.13 – La fonction $U^{(\varepsilon)}$

$1BP$ [15, 29, 50]. Fekete et Schepers [29] proposent plusieurs DFF et discutent l'efficacité des bornes obtenues.

Nous présentons maintenant les fonctions proposées par Fekete et Schepers [29] qui permettent de calculer de bonnes évaluations pour $1BP$. La première fonction est une fonction *en escalier* basée sur des techniques d'arrondis. Elle va réduire certaines valeurs pour en augmenter d'autres.

Théorème 1.1. *[29] Soit $k \in \mathbb{N}$. Alors*

$$u^{(k)} : [0,1] \to [0,1]$$
$$x \mapsto \begin{cases} x & \text{pour } x(k+1) \in \mathbb{Z} \\ \lfloor (k+1)x \rfloor \frac{1}{k} & \text{sinon.} \end{cases}$$

est une fonction dual-réalisable.

La deuxième fonction permet de calculer une borne équivalente à la borne L_α de Martello *et al.* [52]. L'idée est de négliger les petites valeurs et d'augmenter les grandes valeurs.

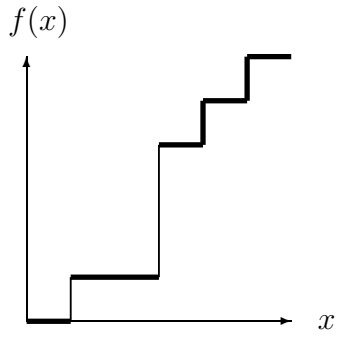


Figure 1.14 – La fonction $\phi^{(\varepsilon)}$

Théorème 1.2. *[29] Soit $\varepsilon \in [0, \frac{1}{2}]$. Alors*

$$U^{(\varepsilon)} : [0,1] \to [0,1]$$
$$x \mapsto \begin{cases} 1 & \text{pour } x > 1-\varepsilon \\ x, & \text{pour } \varepsilon \leq x \leq 1-\varepsilon \\ 0, & \text{pour } x < \varepsilon \end{cases}$$

est une fonction dual-réalisable.

La troisième fonction proposée ϕ est plus complexe. Elle s'appuie comme la première sur des techniques d'arrondis. Les petits articles sont ignorés, et ceux de taille inférieure à $\frac{1}{2}$ prennent comme valeur une constante. Pour calculer la nouvelle taille des grands articles, il faut prendre en compte le nombre d'articles de taille inférieure à $\frac{1}{2}$ que l'on peut ranger dans le même bin. Ce calcul est simplifié par le fait que tous ces articles sont de même taille.

Théorème 1.3. *[29] Soit $\varepsilon \in [0, \frac{1}{2}]$. Alors*

$$\phi^{(\varepsilon)} : [0,1] \to [0,1]$$
$$x \mapsto \begin{cases} 1 - \frac{\lfloor (1-x)\varepsilon^{-1} \rfloor}{\lfloor \varepsilon^{-1} \rfloor} & \text{pour } x > \frac{1}{2} \\ \frac{1}{\lfloor \varepsilon^{-1} \rfloor} & \text{pour } \varepsilon \leq x \leq \frac{1}{2} \\ 0, & \text{pour } x < \varepsilon \end{cases}$$

est une fonction dual-réalisable.

Les bornes proposées par Fekete et Schepers [29], notée L_{FS}^{1BP} combinent les bornes décrites ci-dessus. Les auteurs [29] montrent les deux résultats suivants.

Proposition 1.1. *[29]* $L_{FS}^{1BP}(A) \leq \frac{3}{4}OPT(A)$.

Ils améliorent donc la performance de la borne continue dans le pire des cas qui était égale à $\frac{1}{2}OPT(A)$. Les auteurs montrent aussi que si les articles sont tous trop grands pour être placés à trois dans un bin, leur borne est toujours égale à la valeur optimale d'une solution.

Proposition 1.2. *[29] Si tous les articles a_i de A sont de longueur strictement supérieure à $\frac{1}{3}C$, $L_{FS}^{1BP}(A) = OPT(A)$.*

1.6.1.4 Bornes de Labbé *et al.*

Labbé *et al.* [43] ont proposé une borne pour $1BP$, qui prend en compte les *grands* articles (*i.e.* les articles de taille supérieure à $\frac{C}{2}$) et les articles de taille *moyenne* (*i.e.* qui ne peuvent pas être rangés avec les grands articles). Bourjolly et Rebetz [9] montrent que cette borne domine celle de Martello *et al.* [52], que nous avons notée L_α.

Pour un paramètre donné $k \in [0, \frac{C}{3}]$, la borne proposée est calculée de la manière suivante : soit $A_{gr} = \{a_i \in A : c_i > C - k\}$ l'ensemble des grands articles, $A_{moy} = \{a_i \in A : \frac{C}{2} < c_i \leq C - k\}$. On pose $A_{med} = \{a_i \in A : \frac{C}{3} \leq c_i \leq \frac{C}{2}\}$ et A^+_{med} l'ensemble des articles de A_{med} qui ne peuvent être rangés avec aucun article de A_{gr}. Enfin, on pose $A' = \{a_i \in A : k \leq c_i \leq C - k\}$.

$$LB3 = \max_{0 \leq k \leq \frac{C}{3}} \left\{ |A_{gr}| + max \left\{ |A_{moy}| + \left\lceil \frac{|A^+_{med}|}{2} \right\rceil, \left\lceil \frac{\sum_{a_i \in A'} w_i}{C} \right\rceil \right\} \right\} \tag{1.16}$$

Das un cas de figure, cette borne est égale à L_α proposées par Martello *et al.* [52]. Dans l'autre cas, on considère un sous-ens emble d'articles qui ne peuvent pas être placés à trois par bin. Une évaluation par défaut pour le sous-problème obetenu est égale au nombre d'articles qui sont placés seuls dans leur bin, auquel on ajoute le nombre d'articles qui ne peuvent être placés que par deux dans leur bin divisé par deux.

1.6.1.5 Amélioration des bornes (Haouari et Gharbi)

Haouari et Gharbi [41] ont proposé une méthode d'amélioration itérative de la borne inférieure. Elle repose sur le lemme suivant.

Lemme 1.1. *Si une solution pour une instance de $1BP$, constituée de n articles, utilise exactement z bins, il existe au moins un ensemble de k bins ($1 \leq k \leq z$) qui contiennent au moins $k \left\lfloor \frac{n}{z} \right\rfloor + min(k, n - \left\lfloor \frac{n}{z} \right\rfloor)$ articles.*

L'idée est de calculer le nombre d'articles *moy* que l'on trouve en moyenne dans les bins et de vérifier si cette valeur est cohérente. Pour cela, il faut que l'on puisse placer les *moy* plus petits articles dans le premier bin, les $2 \times moy$ plus petits articles dans les deux premiers bins, *etc.* Considérons une instance de $1BP$ définie pour un ensemble A de n articles et $LB(A)$ une borne inférieure pour $OPT(A)$. Les auteurs définissent pour $k = 1, \ldots, z-1$

$$\lambda(k, LB(A), n) = k \left\lfloor \frac{n}{LB(A)} \right\rfloor + min(k, n - \left\lfloor \frac{n}{LB(A)} \right\rfloor LB(A)) \tag{1.17}$$

Soit $S_n^{LB(A),k} \in A$ l'ensemble des $\lambda(k, LB(A), n)$ plus petits articles de A. D'après le Lemme 1.1, on déduit la proposition suivante.

Proposition 1.3. *Soit $LB(A)$ une borne inférieure pour $OPT(A)$. Si l'on a la propriété que $LB(S_n^{LB(A),k}) > k$ pour une valeur de k, $k = 1, \ldots, z-1$, alors $LB(A)+1$ est une borne inférieure pour $OPT(A)$.*

Cette proposition permet de mettre au point un processus d'améliorations itératives des bornes inférieures. Les auteurs démontrent plusieurs propriétés qui permettent d'implanter cette méthode de manière efficace.

1.6.1.6 Synthèse pour $1BP$

En résumé, on peut trouver deux types d'évaluations pour $1BP$: celles qui s'appuient sur les dimensions des objets indépendamment de l'instance générale, et celles qui dépendent de l'instance générale (en raisonnant sur le nombre d'articles par boîte, par exemple). Typiquement, les bornes de Martello et Vigo, de Fekete et Schepers et de Labbé *et al.* n'utilisent que la taille des articles présents. L'amélioration itérative proposée par Haouari et Gharbi relèvent de la deuxième classe, puisqu'elle prend en compte le nombre d'articles dans chaque boîte.

1.6.2 Bornes pour $2BP$

Si la borne continue peut être utilisée en une dimension lorsque l'on cherche une évaluation rapide, elle est beaucoup moins efficace pour $2BP$. Dans le pire des cas, pour $2BP$, la borne continue L_0 peut atteindre un quart de l'optimium. Pour réaliser la faiblesse de cette borne, il suffit de considérer le cas où tous les articles sont de taille $(\frac{W}{2}+\varepsilon, \frac{H}{2}+\varepsilon)$ pour ε petit. Nous présentons maintenant trois approches permettant d'obtenir des bornes inférieures pour $2BP$. La première est proposée par Martello et Vigo [53]. Elle généralise les résultats que les auteurs ont obtenus en une dimension. Elle est dominée par celle de Fekete et Schepers [31], qui utilise les fonctions définies pour $1BP$. La dernière évaluation en date est dûe à Boschetti et Mingozzi [7] et domine les deux autres approches.

1.6.2.1 Bornes de Martello et Vigo

Martello et Vigo [53] exploitent le travail effectué pour le $1BP$ pour proposer des évaluations pour $2BP$. Les articles qui ne peuvent pas être placés côte-à-côte dans un bin forment un sous-problème équivalent à un $1BP$. Pour un entier k, $1 \leq k \leq \frac{1}{2}W$, on pose

$$A^w_{gr} = \{a_i \in A : w_i > W - k\} \tag{1.18}$$

$$A^w_{moy} = \{a_i \in A : W - k \geq w_i > \frac{1}{2}W\} \tag{1.19}$$

$$A^w_{pt} = \{a_i \in A : \frac{1}{2}W \geq w_i \geq k\} \tag{1.20}$$

On note L_1^W la valeur retournée par L^{MV}_{1BP} sur l'instance $1BP$ correspondant aux articles de $A^w_{gr} \cup A^w_{moy}$.

$$L_2^W(k) = L_1^W + max\left\{0, \left\lceil \frac{\sum_{a_i \in A^w_{moy} \cup A^w_{pt}} w_i h_i - (HL_1^W - \sum_{a_i \in A^w_{gr}} h_i)W}{WH} \right\rceil\right\} \tag{1.21}$$

La même évaluation peut être calculée en permutant le rôle des deux dimensions. Si l'on calcule le maximum entre ces deux bornes, on obtient L_2.

La borne L_3 est plus coûteuse mais peut améliorer les résultats de L_2. D'autres ensembles doivent être définis. Soit un couple d'entiers (k,l), $1 \leq l \leq \frac{1}{2}H$ et $1 \leq k \leq \frac{1}{2}W$.

$$A_{gr} = \{a_i \in A : h_i > H - l \text{ et } w_i > W - k\} \tag{1.22}$$

$$A_{moy} = \{a_i \in A \setminus A_{gr} : h_i > \frac{1}{2}H \text{ et } w_i > \frac{1}{2}W\} \tag{1.23}$$

$$A^s_{pt} = \{a_i \in A : \frac{1}{2}H \geq h_i \geq l \text{ et } \frac{1}{2}W \geq w_i \geq k\} \tag{1.24}$$

Pour calculer la nouvelle borne, il faut au préalable calculer la valeur $m(a_i,k,l)$ qui représente le nombre de pièces de taille (k,l) qui peuvent être rangées dans le bin contenant un article de type a_i.

$$m(a_i,k,l) = \lfloor\frac{H}{l}\rfloor\lfloor\frac{W-w_i}{k}\rfloor + \lfloor\frac{W}{k}\rfloor\lfloor\frac{H-h_i}{l}\rfloor - \lfloor\frac{H-h_i}{l}\rfloor\lfloor\frac{W-w_i}{k}\rfloor \tag{1.25}$$

$$L_3(k,l) = |A_{gr} \cup A_{moy}| + max\left\{0, \left\lceil \frac{|A^s_{pt}| - \sum_{a_i \in A_{moy}} m(a_i,k,l)}{\lfloor\frac{H}{l}\rfloor\lfloor\frac{W}{k}\rfloor} \right\rceil\right\} \tag{1.26}$$

L'évaluation finale que proposent Martello et Vigo est le maximum entre L_2 et L_3.

1.6.2.2 Bornes de Fekete et Schepers

Fekete et Schepers [30] généralisent leur méthode proposée pour $1BP$. Ils utilisent les mêmes fonctions dual-réalisables qu'ils ont appliquées au $1BP$ sur chaque dimension d'une instance $2BP$ indépendamment. Les Figures 1.15 et 1.16 illustrent l'application de deux fonctions à une instance simple. L'instance initiale est composée de cinq articles de taille $(\frac{2}{5},\frac{2}{5})$ et d'un bin unitaire. La borne continue est égale à $\lceil 5 \times \frac{4}{25}\rceil = 1$. En appliquant une DFF sur chaque dimension, on obtient l'instance de la Figure 1.16. La borne continue est égale à $\lceil 5 \times \frac{1}{4}\rceil = 2$.

Les auteurs montrent que l'instance obtenue est telle que toute borne inférieure pour le problème induit en est une pour le problème initial. L'idée de la démonstration est la suivante : lorsque l'on applique une fonction dual-réalisable sur une des deux dimensions du problème, toute configuration faisable initialement le reste. Ainsi, toute solution pour le problème initial est solution du problème modifié. La borne obtenue domine celle de Martello et Vigo [53]. Leur approche a l'avantage de pouvoir être appliquée quelle que soit la dimension. Pour $2BP$, la borne de Fekete et Schepers est égale à la valeur de l'expression résultante du théorème suivant.

Théorème 1.4. *Soit* $r,s \in]0,\frac{1}{2}]$. *Soit*

- $a_i^{(1)(r)} = u^{(1)}(w_j) \times U^{(r)}(h_j)$
- $a_i^{(2)(r)} = U^{(r)}(w_i) \times u^{(1)}(h_i)$

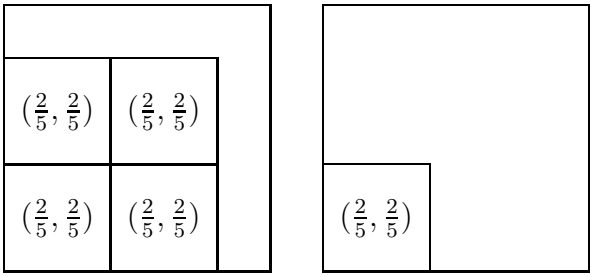


Figure 1.15 – Instance initiale de bin packing en deux dimensions

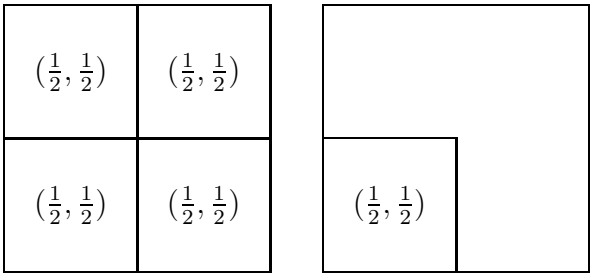


Figure 1.16 – Instance obtenue après application d'une DFF

- $a_i^{(3)(r)} = u^{(1)}(w_i) \times \phi^{(r)}(h_i)$
- $a_i^{(4)(r)} = \phi^{(r)}(w_i) \times u^{1}(h_i)$
- $a_i^{(5)(r)} = w_i \times U^{(r)}(h_i)$
- $a_i^{(6)(r)} = U^{(r)}(w_i) \times h_i$
- $a_i^{(7)(r,s)} = \phi^{(r)}(w_i) \times \phi^{(s)}(h_i)$

L'expression ci-dessous est une borne valide pour 2BP.

$$L_{FS} = max \left\{ \max_{\substack{0<r\leq\frac{1}{2} \\ k=1,\ldots,6}} \left\{ \left\lceil \sum_{a_i \in A} a_i^{(k)(r)} \right\rceil \right\}, \max_{\substack{0<s\leq\frac{1}{2} \\ 0<r\leq\frac{1}{2}}} \left\{ \left\lceil \sum_{a_i \in A} a_i^{(7)(r,s)} \right\rceil \right\} \right\} \tag{1.27}$$

1.6.2.3 Bornes de Boschetti et Mingozzi

Soit (k, l) un couple d'entiers tels que $1 \leq k \leq \frac{1}{2}W$ et $1 \leq l \leq \frac{1}{2}H$. Nous notons $A_{gr} = \{a_i \in A : h_i > H - l \text{ et } w_i > W - k\}$, $A_{ht} = \{a_i \in A \setminus A_{gr} : h_i > H - l \text{ et } w_i \geq k\}$, $A_{lg} = \{a_i \in A \setminus A_{gr} : h_i \geq l \text{ et } w_i > W - k\}$, $A_{pt} = \{a_i \in A \setminus A_{gr} \cup A_{ht} \cup A_{lg} : h_i \geq l \text{ et } w_i \geq k\}$. Boschetti et Mingozzi [7] proposent une borne inférieure notée L_2^{new} pour $2BP$: elle peut être décomposée en deux bornes distinctes. La première

repose sur la transformation du problème en $1BP$. La deuxième exploite le fait que deux *gros* articles de A_{gr} ne peuvent pas être rangés dans le même bin et qu'un article *haut* de A_{ht} ne peut pas être rangé avec un article *long* de A_{lg}.

$$L_2^{new} = \max_{\substack{1\leq k\leq \frac{1}{2}W \\ 1\leq l\leq \frac{1}{2}H}} \{|A_{gr}| + max\{L_2'(k,l), L_2''(k,l)\}\} \tag{1.28}$$

où les bornes $L_2'(k,l)$ et $L_2''(k,l)$ sont deux évaluations pour le nombre minimum de bins nécessaires au rangement des articles de A_{ht}, A_{lg}, et A_{pt}.

- Dans $L_2'(k,l)$, une instance D_{1BP} du problème $1BP$ est générée avec un bin de taille $C' = WH$: pour chaque article a_i de $A_{ht} \cup A_{lg} \cup A_{pt}$ un article est créé avec pour dimension c_i.

$$a_i \mapsto \begin{cases} w_i H \text{ si } a_i \in A_{ht} \\ W h_i \text{ si } a_i \in A_{lg} \\ w_i h_i \text{ si } a_i \in A_{pt} \end{cases}$$

  $L_2'(k,l)$ est obtenue en calculant une borne inférieure pour D_{1BP} en utilisant L_{1BP}^{MV} [53].

- Dans $L_2''(k,l)$, les articles de A_{pt} ne sont pas considérés. La borne exploite le fait qu'un article de A_{ht} ne peut pas être rangé dans le même bin qu'un article de A_{lg}. Ces deux ensembles induisent donc deux sous-problèmes indépendants. De plus, les articles hauts ne peuvent pas être placés les uns au-dessus des autres, ni les articles longs les uns à côté des autres. Ainsi, les deux sous-problèmes peuvent être vus comme des instances de $1BP$ et des bornes inférieures peuvent être calculées à l'aide de L_{1BP}^{MV}. Soit $L_2^H(k,l)$ une borne inférieure pour l'instance de $1BP$ associée à A_{ht}, et $L_2^W(k,l)$ une borne inférieure pour l'instance de $1BP$ associée à A_{lg}.

$$L_2''(k,l) = L_2^H(k,l) + L_2^W(k,l)$$

Théorème 1.5. *[7] L_2^{new} est une borne valide pour $2BP$.*

La borne L_3^{new} génère des problèmes type sac à dos $1KP$ pour déterminer le nombre de bins nécessaires au rangement des petits articles qui ne peuvent pas l'être avec les grands. Elle utilise une décomposition de l'ensemble des articles en trois ensembles qui dépendent de deux paramètres k et l : les grands articles (A_{gr}), les articles moyens (A_{moy}), et les petits articles (A_{pt}). La méthode exploite le fait que le nombre de grands et moyens articles est une borne inférieure pour $2BP$. Pour améliorer cette borne, L_3^{new} évalue le nombre de petits articles qui peuvent être rangés ensemble avec les articles moyens, puis une approximation du nombre de bins nécessaires pour ranger les articles restants est calculée. Soit k et l deux entiers tels que $1 \leq l \leq \frac{1}{2}H$ et $1 \leq k \leq \frac{1}{2}W$, $A_{gr} = \{a_i \in A : h_i > H - l \text{ et } w_i > W - k\}$, et $A_{moy} = \{a_i \in A \setminus A_{gr} : h_i > \frac{1}{2}H \text{ et } w_i > \frac{1}{2}W\}$, et $A_{pt}^3 = \{a_i \in A \setminus A_{gr} \cup A_{moy} : h_i \geq l \text{ et } w_i \geq k\}$. Plusieurs valeurs doivent être calculées :

- $M_W(W, A^3_{pt})$: le nombre maximum d'articles de A^3_{pt} qui peuvent être rangés côte à côte dans un bin ;
- $M_H(H, A^3_{pt})$: le nombre maximum d'articles de A^3_{pt} qui peuvent être rangés les uns au dessus des autres dans un bin ;
- $M_W(W - w_i, A^3_{pt})$: le nombre maximum d'articles de A^3_{pt} qui peuvent être rangés côte à côte dans un bin contenant l'article a_i de A_{moy} ;
- $M_H(H - h_i, A^3_{pt})$: le nombre maximum d'articles de A^3_{pt} qui peuvent être rangés les uns au-dessus des autres dans un bin contenant l'article a_i de A_{moy}.

Chaque valeur est la solution du problème de sac à dos en une dimension ($1KP$) défini dans la Section 1.2 et formalisé par les expressions des Equations 1.29 et 1.31 Il peut être résolu en temps linéaire si les articles sont triés par ordre croissant de largeur ou de hauteur selon le problème.

$$M_W(w^*, A^3_{pt}) = Max\{\sum_{a_i \in A^3_{pt}} \xi_i : \sum_{a_i \in A^3_{pt}} w_i \xi_i \leq w^*, \xi_i \in \{0, 1\}\} \tag{1.29}$$

$$M_H(h^*, A^3_{pt}) = Max\{\sum_{a_i \in A^3_{pt}} \xi_i : \sum_{a_i \in A^3_{pt}} h_i \xi_i \leq h^*, \xi_i \in \{0, 1\}\} \tag{1.30}$$

$$L_3^{new} = \max_{\substack{1 \leq k \leq \frac{1}{2}W \\ 1 \leq l \leq \frac{1}{2}H}} \left\{ |A_{gr} \cup A_{moy}| + max \left\{ 0, \left\lceil \frac{|A^3_{pt}| - \sum_{a_i \in A_{moy}} m'(a_i, A^3_{pt})}{M_W(W, A^3_{pt}) M_H(H, A^3_{pt})} \right\rceil \right\} \right\} \tag{1.31}$$

où $m'(a_i, A^3_{pt})$ est une borne supérieure pour le nombre d'articles de A^3_{pt} qui peuvent être rangés ensemble avec a_i :

$$\begin{aligned} m'(a_i, A^3_{pt}) = & M_W(W, A^3_{pt}) M_H(H - h_i, A^3_{pt}) \\ & + M_W(W - w_i, A^3_{pt}) M_H(H, A^3_{pt}) \\ & - M_W(W - w_j, A^3_{pt}) M_H(H - h_i, A^3_{pt}) \end{aligned} \tag{1.32}$$

La valeur $M_W(W, A^3_{pt}) \times M_H(H, A^3_{pt})$ est une borne supérieure pour le nombre d'articles de A^3_{pt} qui peuvent être rangés ensemble dans un bin.

Théorème 1.6. *[7] L_3^{new} est une borne inférieure pour $2BP$.*

La borne L_4^{new} utilise une autre méthode pour évaluer le nombre de petits articles qui ne peuvent pas être rangés avec des grands et le nombre de bins nécessaires pour les ranger. Pour cela, des techniques spéciales d'arrondis sont utilisées. Chaque article a_i de faible largeur (resp. hauteur) a sa largeur (resp. hauteur) réduite à une valeur calculée à partir de sa taille et d'un paramètre donné k (resp. l). La taille de tout grand article a_i est accrue de telle manière que le nombre de petits articles qui peuvent être rangés avec a_i soit le même. Soit $A_{gr} = \{a_i \in A : h_i > H - l \text{ et } w_i > W - k\}$, et $A_{moy} = \{a_i \in A \setminus A_{gr} : h_i > \frac{1}{2}H \text{ et } w_i > \frac{1}{2}W\}$. L'ensemble A^3_{pt} utilisé pour la borne L_3^{new} est décomposé ici en trois sous-ensembles. On a $A^t_{pt} = \{a_i \in$

$A : h_i > \frac{1}{2}H$ et $\frac{1}{2}W \geq w_i \geq k\}$, $A^w_{pt} = \{a_i \in A : \frac{1}{2}H \geq h_i \geq l$ et $w_i > \frac{1}{2}W\}$, et $A^s_{pt} = \{a_i \in A : \frac{1}{2}H \geq h_i \geq l$ et $\frac{1}{2}W \geq w_i \geq k\}$.

$$L_4^{new} = \max_{\substack{1\leq k\leq \frac{W}{2} \\ 1\leq l\leq \frac{H}{2}}} \left\{ |A_{gr} \cup A_{moy}| + max \left\{ 0, \left\lceil \frac{\sum_{a_i \in A\setminus A_{gr}} m''(a_i,k,l)}{\lfloor \frac{H}{l} \rfloor \lfloor \frac{W}{k} \rfloor} \right\rceil \right\} \right\} \tag{1.33}$$

où

- $m''(a_i,k,l) = \lfloor \frac{W-w_i}{k} \rfloor \lfloor \frac{H-h_i}{l} \rfloor - \lfloor \frac{W-w_i}{k} \rfloor \lfloor \frac{H}{l} \rfloor - \lfloor \frac{W}{k} \rfloor \lfloor \frac{H-h_i}{l} \rfloor$, si $a_i \in A_{moy}$
- $m''(a_i,k,l) = \lfloor \frac{w_i}{k} \rfloor \lfloor \frac{H}{k} \rfloor - \lfloor \frac{w_i}{k} \rfloor \lfloor \frac{H-h_i}{l} \rfloor$, si $a_i \in A^t_{pt}$
- $m''(a_i,k,l) = \lfloor \frac{W}{k} \rfloor \lfloor \frac{h_i}{l} \rfloor - \lfloor \frac{W-w_i}{k} \rfloor \lfloor \frac{h_i}{l} \rfloor$, si $a_i \in A^w_{pt}$
- $m''(a_i,k,l) = \lfloor \frac{w_i}{k} \rfloor \lfloor \frac{h_i}{l} \rfloor$, si $a_i \in A^s_{pt}$

Théorème 1.7. *[7] L_4^{new} est une borne inférieure pour $2BP$.*

1.6.2.4 Synthèse pour $2BP$

En résumé, on peut présenter la relation suivante : la borne de Boschetti et Mingozzi domine la borne de Fekete et Schepers qui domine celle de Martello et Vigo. Les bornes proposées pour le problème en deux dimensions peuvent, de la même manière que pour le problème en une dimension être séparées en deux classes distinctes : celles qui raisonnent sur l'instance dans son intégralité (nombre d'articles par bin, résolutions de problèmes type sac à dos) et celles qui n'utilisent que des calculs sur la taille des articles présents (à l'aide de DFF par exemple). Si les méthodes relevant de cette dernière catégorie sont diverses, nous voyons dans les chapitres consacrés aux bornes inférieures qu'il est possible d'unifier toutes ces méthodes.

1.7 Méthodes exactes

Il existe peu de méthode exacte possédant un intérêt pratique. La première a été proposée par Beasley [2]. Le problème de faisabilité a été traité par Hadjiconstantinou et Christophides [40] dans le cadre d'une méthode exacte pour le problème de sac à dos bidimensionnel. Christophides et Whitlock proposent une méthode pour le problème de découpe 2D sans guillotine [16] : il s'agit d'un schéma général de résolution de type procédure par séparation et évaluation mais aucun résultat numérique n'est présenté. Plus récemment, Martello et Vigo [53] ont proposé une méthode exacte pour $2BP$ et Fekete et Schepers [30] une méthode pour le problème de faisabilité. Ces deux dernières méthodes, ainsi qu'une méthode dédiée au cas des packings *parfaits* sont détaillées dans la suite de ce chapitre.

1.7.1 Schéma général de séparation

La méthode de résolution du problème $2BP$ peut être décomposée en deux étapes : le problème d'affectation des articles aux bins, et un problème de faisabilité

pour chaque bin utilisé. Dans ce paragraphe, nous décrivons la méthode générale d'affectation aux bins proposée par Martello et Vigo [53]. Cette méthode utilise une méthode de résolution du problème de faisabilité qui est exposée par la suite. A notre connaissance, Martello et Vigo [53] sont les seuls à avoir proposé une séparation pour cette étape.

Dans un premier temps, une borne supérieure z pour le nombre de bins utilisés est calculée. Au début de l'algorithme, z bins sont créés, mais non initialisés. Lorsqu'un bin B_p est initialisé, il est considéré actif tant qu'il est encore possible de lui affecter un nouvel article. Les articles sont triés par ordre décroissant de surface. A chaque niveau i, l'algorithme tente d'affecter l'article a_i à chaque bin actif, ou éventuellement à un nouveau bin. A chaque affectation, il faut tester si l'ensemble d'articles correspondant admet une configuration réalisable. Des méthodes heuristiques et exactes sont utilisées. On teste la possibilité de fermer un bin en tentant de lui affecter chaque article restant à tour de rôle et d'évaluer la faisabilité de la configuration obtenue. Lorsqu'un bin est fermé, les bornes inférieures pour le problème sont mises à jour : elles sont appliquées sur l'ensemble des articles non encore affectés ou affectés à des bins non fermés.

L'intérêt de cette méthode est qu'elle permet de trouver des solutions simples rapidement si le nombre d'articles par bin est faible. En revanche, si les articles sont de plus petits, les bornes inférieures peuvent tarder à être mises à jour et les coupes réalisées dans l'arbre de recherche peuvent être tardives.

1.7.2 Placement d'articles dans le bin

1.7.2.1 Cas général : la méthode de Martello et Vigo

Martello et Vigo [53] construisent une configuration article par article. A chaque étape, on dispose d'une liste d'articles déjà placés avec leurs coordonnées et d'une liste d'articles à placer. A une configuration donnée, on peut associer une liste finie de positions possibles pour un article. La séparation revient à énumérer le rangement de tous les articles non placés à ces coordonnées. Ce principe a le défaut de créer de nombreuses redondances qu'il faut gérer.

Les articles sont placés suivant la statégie *leftmost-downward*. Chaque article est placé de telle manière qu'il ne puisse être décalé vers la gauche ou vers le bas sans entrer en intersection avec un autre article ou le bord du bin (Figure 1.17). A chaque étape, on a une liste finie de k positions et une liste de l articles. Il y a deux choix à faire : l'article à placer et à quelles coordonnées. Cela mène à $k \times l$ nœuds fils. Cette méthode génère un grand nombre de redondances qui doivent être gérées pour obtenir un algorithme efficace. Scheithauer [55] énumère un certain nombre de répétitions possibles et des méthodes pour en éviter une partie.

1.7.2.2 Cas particulier des packings *parfaits*

Lesh *et al.* [45] ont proposé en 2004 une méthode dédiée aux packings *parfaits* ou configurations parfaites (*i.e.* lorsqu'il n'y a pas d'espace vide dans le bin, *cf.* Figure 1.18). Dans ce cas particulier, on obtient de manière immédiate le résultat suivant : il existe une permutation π^* des rectangles qui mène au packing parfait en utilisant

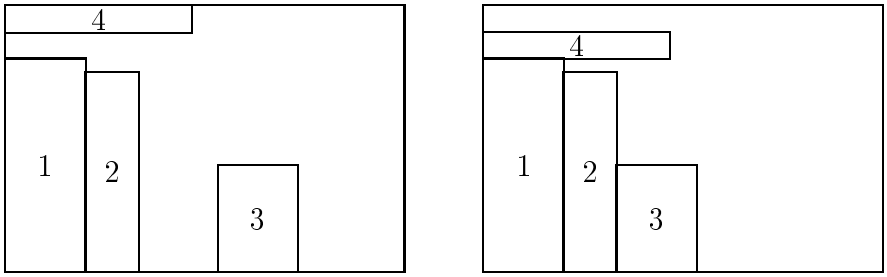


Figure 1.17 – La règle *leftmost-downward*

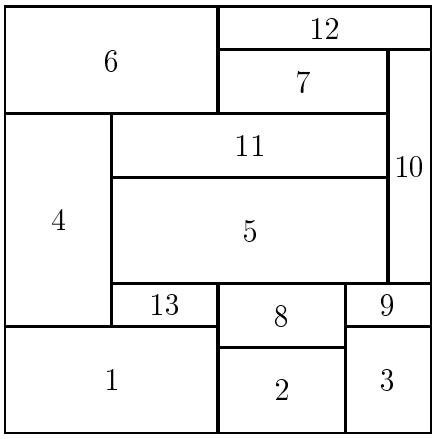


Figure 1.18 – Exemple de packing parfait

l'heuristique *bottom-left*. Les auteurs proposent une généralisation de leur travail pour le cas discret en ajoutant des carrés unitaires de manière à obtenir un packing parfait. Il est à noter que l'ajout de ces carrés unitaires augmente considérablement le nombre de combinaisons possibles et rend la méthode inefficace lorsque l'espace vide est important.

1.7.3 Algorithme utilisant un modèle *graphe*

Pour éviter l'exploration de solutions redondantes, Fekete et Schepers [32] utilisent le modèle décrit dans la Section 1.3 de ce chapitre. Ils proposent un algorithme énumératif [30] qui construit des graphes d'intervalles associés à des ensembles de configurations. Cette méthode réduit considérablement le nombre de répétitions dans l'arbre de recherche, par comparaison à la méthode classique. Néanmoins, le coût lié aux algorithmes associés aux graphes d'intervalles n'est pas négligeable.

A chaque dimension du problème est associé un graphe d'intervalles. L'idée de l'algorithme exact de Fekete et Schepers [30] est de construire pas à pas deux graphes d'intervalles qui vont conduire à des classes de configurations réalisables. Une liste d'arêtes existantes et une liste d'arêtes qui ne seront pas ajoutées est gardée en

mémoire. A chaque étape, la possibilité d'ajouter une arête à l'un ou l'autre des deux ensembles est testée. A chaque sommet de l'arborescence, des procédures sont lancées pour ajouter des arêtes au graphe. De plus, il faut vérifier à chaque étape si l'ensemble des graphes courants est déjà associé à une classe de configurations réalisables.

Trois procédures sont nécessaires pour répondre aux trois questions suivantes :

1. Les graphes courants forment-ils une classe réalisable ?
2. Peut-on ajouter des arêtes aux graphes courants ?
3. Quelles arêtes peut-on ajouter aux graphes courants ?

Chaque question trouve sa réponse dans l'identification de sous-graphes induits [30, 32].

Algorithme 1 : Algorithme de Fekete et Schepers appliqué au $2BP$

Données :
E_w^+ : l'ensemble des arêtes ajoutées pour la largeur;
E_w^- : l'ensemble des arêtes interdites pour la largeur;
E_h^+ : l'ensemble des arêtes ajoutées pour la hauteur;
E_h^- : l'ensemble des arêtes interdites pour la hauteur;
répéter
  enrichir les graphes entrés en paramètre;
  si *la configuration courante ne peut pas mener à une solution réalisable* **alors**
    retourner *ECHEC*;
jusqu'à *impossibilité* d'augmenter *les graphes*;
si *la configuration courante est réalisable* **alors**
  retourner *SUCCES*;
sinon
  Soit (a_i, a_j) un couple d'articles, et $d \in \{w, h\}$;
  Ajouter (i, j) à E_d^+ et relancer l'Algorithme 1 ;
  Ajouter (i, j) à E_d^- et relancer l'Algorithme 1;

1.8 Conclusion

Ces dernières années, les problèmes de bin-packing en deux dimensions ont suscité une attention très particulière. Les méthodes heuristiques mises au point permettent de trouver de très bonnes solutions dans la plupart des cas, mais il existe de nombreux jeux d'essai pour lesquels les bornes inférieure et supérieure ne sont pas égales. En l'état, il est très difficile de résoudre le problème $2BP$ à l'aide d'une méthode exacte pour des instances de taille moyenne.

Prétraitements et bornes inférieures

2.1 Introduction

Ce chapitre est consacré à de nouvelles bornes inférieures et à des procédures de réduction pour $2BP$[1]. Lorsque la taille des problèmes traités est grande, il devient difficile de les résoudre de manière exacte. Pour éviter d'avoir à traiter des instances de taille trop importante, il est intéressant de les réduire à l'aide de procédures de réduction qui peuvent être utilisées en prétraitement avant d'appliquer une méthode de résolution. Dans cette optique, nous proposons le concept de *Fonctions Identiquement Réalisables* qui peuvent être utilisées pour modifier la taille des articles sans modifier la valeur de la solution optimale du problème. Nous montrons que l'on peut réduire la taille de l'instance initiale en supprimant de petits articles et en augmentant la taille des grands articles.

La principale contribution de ce chapitre est une nouvelle borne inférieure pour $2BP$. Les bornes les plus récentes ont été introduites par Boschetti et Mingozzi [7]. Elles dominent les meilleurs résultats connus [31, 53]. Une borne élémentaire pour le bin est la borne continue : elle est égale au rapport entre la surface totale des articles et celle du bin. Plusieurs bornes [29, 31, 50] proposées pour $1BP$ ou $2BP$ reposent sur le calcul de cette borne pour une instance obtenue après prétraitement. Un tel traitement peut être obtenu en utilisant les *Fonctions Redondantes* (RF) proposées par Carlier et Néron [13] pour des problèmes d'ordonnancements cumulatifs. Les RF forment un sous-ensemble des *Fonctions Dual-Réalisables* (DFF) introduites par Johnson [42] et utilisées par Lueker [50], et Fekete et Schepers [29] pour obtenir des bornes inférieures pour le $1BP$. Fekete et Schepers [31] ont montré que cette approche pouvait être généralisée à $2BP$ en appliquant séparemment des fonctions dual-réalisables sur les deux dimensions de l'instance. Dans la suite, nous parlons de DFF discrètes pour faire référence aux fonctions redondantes. En effet, pour chaque RF, on peut associer une RF définie dans l'ensemble $[0, 1]$. Nous introduisons aussi un nouveau type de fonctions : les *DFF Dépendantes de la Donnée* (DDFF). Ces fonctions sont calculées pour une instance spécifique et se comportent sur cette instance comme si elles étaient des DFF. Nous proposons trois familles de fonctions et nous montrons qu'elles conduisent à des évaluations qui dominent strictement celles de Boschetti et Mingozzi [7]. Les bornes obtenues sont les meilleures connues pour $2BP$.

Nous reproduisons les résultats expérimentaux obtenus par nos procédures de réduction, ainsi que par nos bornes inférieures. Nous les avons testées sur des jeux

1. Ce chapitre correspond au travail réalisé dans [11] et présenté en congrès [19].

d'essai issus de la littérature [5, 53], ce qui nous permet de nous comparer avec les meilleurs résultats connus [8]. Nos procédures de réduction sont très efficaces pour plusieurs classes d'instances : un grand nombre d'articles sont supprimés pour de multiples jeux d'essai. Les résultats expérimentaux confirment que notre borne domine les autres pour toutes les instances et améliorent strictement les meilleurs résultats.

Dans la section 2.2, nous introduisons le concept de *Fonctions Identiquement Réalisables (IFF)* et nous proposons deux nouvelles procédures de réduction qui reposent sur ce concept. La section 2.3 est consacrée à nos nouvelles bornes inférieures. Nous montrons dans la section 2.4 qu'elles dominent celles proposées par Boschetti et Mingozzi [7]. Dans la section 2.5, nous comparons nos procédures de réduction et nos bornes inférieures avec les méthodes de la littérature. Les résultats sont strictement améliorés pour plusieurs instances.

2.2 Procédures de réduction

Dans une instance de $2BP$, les grands articles ont en général un rôle plus important que les petits. Si un article a une trop petite taille, il peut ne pas avoir d'impact sur la valeur de $OPT(A)$, et s'il est de très grande taille, il forme une sous-instance triviale qui peut être traitée séparément. En général, il y a peu de très grands articles dans l'instance initiale, mais leur nombre peut être augmenté lorsque des procédures de réduction sont employées. Cette section est consacrée à des méthodes qui vont accroître la taille des grands objets tout en réduisant celle des petits. Le nombre d'objets supprimés est un bon indicateur de la qualité de ces prétraitements.

Nous définissons le concept de *Fonctions Identiquement Réalisables* (IFF) qui va nous permettre de regrouper toutes les fonctions qui vont mener à des procédures de réduction. La procédure proposée par Boschetti et Mingozzi [7] entre dans ce cadre. Nous proposons par la suite deux nouvelles fonctions qui s'attaquent respectivement aux hauts et aux grands articles.

Définition 2.1. *Soit $D = (A, B)$ une instance de $2BP$. Une fonction f est une* Fonction Identiquement Réalisable associée à la donnée *D si et seulement si l'instance $f(D) = (\{a'_1, \ldots, a'_n\}, B)$ obtenue en appliquant f à tous les articles de A est telle que $OPT(D) = OPT(f(D))$.*

La procédure de réduction proposée par Boschetti et Mingozzi [7] et décrite en détail dans le Chapitre 1 peut être écrite sous la forme d'une IFF que nous notons u_w^j.

Proposition 2.1. *[7] Soit D une instance de $2BP$ et $a_j \in A$ un article.*

$$u_w^j : A \to A'$$
$$a_i \mapsto a'_i \text{ où } \begin{cases} w'_i = w_i + (W - W_j^*) \text{ et } h'_i = h_i & \text{si } i = j \\ w'_i = w_i \text{ et } h'_i = h_i & \text{sinon} \end{cases}$$

est une IFF associée à l'instance D.

De manière similaire, la fonction u_h^j est définie en inversant les rôles de W et H. Nous notons u l'application itérative de u_h^j et u_w^j en utilisant un critère heuristique pour le choix de a_j.

2.2.1 La nouvelle IFF v_1

Il existe des cas où tous les petits articles peuvent être rangés dans la surface disponible autour des grands objets. La méthode précédente ne prend pas cette considération en compte. Les deux fonctions que nous proposons dans cette section sont conçues pour faire face à ce cas.

Soit A_{ht} un ensemble d'articles *hauts*, et A_s l'ensemble des articles qui peuvent être rangés au-dessus d'un article de A_{ht}. Si tous les articles de A_s peuvent être rangés dans la surface au-dessus de A_{ht}, ces petits articles n'ont pas d'impact sur la valeur de $OPT(A)$.

Soit D une instance de $2BP$ et $1 \leq l \leq \frac{1}{2}H$ un entier. Nous définissons $A_{ht} = \{a_i \in A : h_i > H - l\}$, et $A_s = \{a_i \in A : h_i < l\}$. Notons $m = |A_{ht}|$. Pour chaque article a_i de A_{ht}, un bin B_i de taille $(H - h_i, w_i)$ est créé. Considérons le problème suivant :

> **Nom** : PB1
>
> **Donnée** : Une liste d'articles A_s, une liste de bins de tailles diverses $B_1, \ldots, B_m$.
>
> **Question** : Existe-t-il un rangement pour les articles de A_s dans les bins $B_1, \ldots B_m$?

Théorème 2.1. *Si PB1 a une solution, alors la fonction*

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

$$a_i \mapsto a_i' \text{ où } \begin{cases} w_i' = w_i \text{ et } h_i' = H \text{ if } h_i > H - l \\ w_i' = 0 \text{ et } h_i' = 0 \text{ if } h_i < l \\ w_i' = w_i \text{ et } h_i' = h_i \text{ sinon} \end{cases}$$

est une IFF associée à l'instance D.

Démonstration. Considérons l'instance initiale D et l'instance modifiée $v_1^l(D)$ obtenue après application de l'IFF v_1^l à D. Supposons qu'une configuration optimale est connue pour $v_1^l(D)$. A partir de cette configuration, une configuration optimale pour D peut être construite de la manière suivante : les articles de $A - A_s$ sont rangés de la même façon, et les articles de A_s sont rangés dans la surface disponible au-dessus des grands articles (cette surface est libre car les articles de $A - A_s$ ne peuvent pas être rangés au-dessus d'un article de A_{ht}). En utilisant une méthode similaire, une configuration optimale pour $v_1^l(D)$ peut être construite à partir d'une configuration pour D. Si, dans le rangement optimal obtenu pour D, les articles de A_s ne sont pas rangés au-dessus des articles de A_{ht}, un rangement similaire peut être construit en plaçant les petits articles au-dessus des grands (si nécessaire, les articles hauts pourront être décalés verticalement). A partir du rangement obtenu, un rangement peut être trouvé pour $v_1^l(D)$ en utilisant les mêmes coordonnées pour les articles restants. □

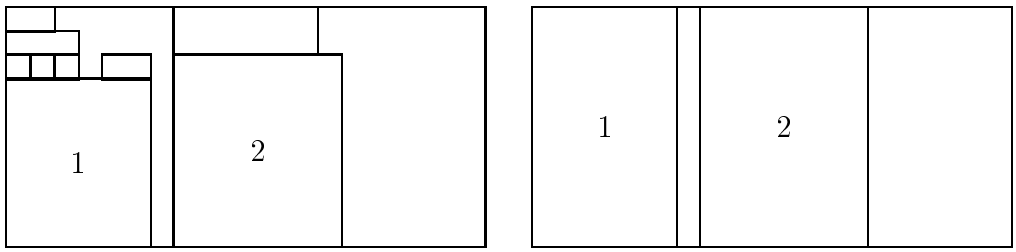


Figure 2.1 – La fonction v_1

Notons v_1^H l'application itérative de v_1^l. Cet algorithme peut être implanté en $O(H \times \alpha(n))$, où $\alpha(n)$ est la complexité de l'algortithme utilisé pour résoudre le problème de décision D'. En prenant en compte le fait que les valeurs intéressantes pour l sont égales à la hauteur d'un article, la complexité devient $O(n \times \alpha(n))$.

Nous définissons une fonction similaire v_1^W en remplaçant H par W. Lorsque la taille d'un article est augmentée sur une dimension, des problèmes de décision qui n'avaient pas été résolus auparavant peuvent devenir réalisables. Notons v_1 l'application itérative de v_1^W et v_1^H. La méthode s'arrête lorsque l'instance n'est plus modifiée. Les problèmes de décision générés sont NP-complets, mais une heuristique est suffisante pour obtenir de bons résultats (*cf.* Section 2.5).

Exemple 2.1. ***Exemple 1.*** *Considérons l'instance de 2BP suivante. Nous décrivons l'application de la fonction v_1^W.*

- $A = \{(8,8),(8,7),(2,6),(1,3),(2,3),(1,3),(2,2),(10,2),(10,2),(10,2)\}$
- $B = (10,10)$

$$\begin{array}{ll} \textit{Pour } l = 3, & A_{ht} = \{(8,8),(8,7),(10,2),(10,2),(10,2)\}. \\ & A_s = \{(2,6),(1,3),(2,3),(1,3),(2,2)\} \\ & D' = (\{(2,6),(1,3),(2,3),(1,3),(2,2)\},\{(2,8),(2,7)\}) \end{array}$$

Il existe une solution pour D' (ranger $(2,6)$ dans le bin de taille $(2,7)$ et les autres articles dans le bin de taille $(2,8)$). L'instance initiale peut donc être modifiée.

$$A' = \{(10,8),(10,7),(10,2),(10,2),(10,2)\}$$

2.2.2 La nouvelle IFF v_2

La procédure de réduction issue de la fonction v_2 prend en compte toute la surface autour des grands articles. Considérons un ensemble A_{gr} d'articles qui occupent plus de la moitié du bin en hauteur et en largeur, et un ensemble A_{pt} d'articles qui peuvent être rangés avec un article de A_{gr}. Si tous les articles de A_{pt} peuvent être rangés avec les articles de A_{gr}, ces articles n'ont pas d'impact sur la valeur de OPT.

Soit D une instance de $2BP$, $1 \leq l \leq \frac{1}{2}H$ et $1 \leq k \leq \frac{1}{2}W$ deux entiers. Notons $A_{gr} = \{a_i \in A : w_i > W - k \text{ et } h_i > H - l\}$, ainsi que $A_{pt} = \{a_i \in A : w_i < k \text{ ou } h_i < l\}$. Soit PB2 le problème suivant :

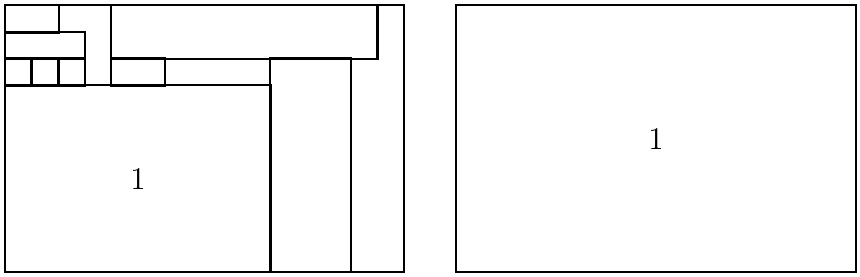


Figure 2.2 – La fonction v_2

Nom : PB2

Donnée : Une liste d'articles $A_{pt} \cup A_{gr}$, une taille de bin unique B.

Question : Existe-t-il un rangement pour les articles de $A_{pt} \cup A_{gr}$ dans $|A_{gr}|$ bins de taille B ?

Théorème 2.2. *Si PB2 a une solution, alors la fonction*

$$v_2^{k,l} : A \to A'$$

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

est une IFF associée à l'instance D.

Démonstration. Considérons l'instance initiale D et l'instance modifiée $v_2^{k,l}(D)$. Supposons qu'une configuration soit connue pour $v_2^{k,l}(D)$. A partir de cette configuration, une configuration pour D peut être construite de la manière suivante : les articles de $A - A_{pt}$ sont rangés de la même manière, et les articles de A_{pt} sont rangés dans l'espace disponible autour des grands articles (espace disponible car aucun article de $A - A_{pt}$ ne peut être rangé avec un article de A_{gr}). En utilisant une méthode similaire, une configuration pour $v_2^{k,l}(D)$ peut être obtenue à partir de D. Si, dans la configuration obtenue pour D, les articles de A_{pt} ne sont pas rangés avec un article de A_{gr}, une configuration équivalente peut être construite en rangeant les petits articles dans les mêmes bins que les grands articles (en déplaçant éventuellement les grands articles horizontalement ou verticalement). A partir de la nouvelle configuration obtenue, une configuration pour $v_2^{k,l}(D)$ peut être construite en utilisant les mêmes coordonnées pour les articles restants. □

Exemple 2.2. ***Exemple 2.*** *Considérons l'instance suivante.*

- $B = (10, 10)$
- $A = \{(9, 9), (7, 5), (6, 8), (6, 2), (10, 1), (1, 1), (1, 1)\}$

Pour $k = 2$ *et* $l = 2$, $A_{gr} = \{(9,9)\}$
$A_{pt} = \{(10,1),(1,1),(1,1)\}$
Tous les articles de A_{pt} *peuvent être rangés autour de* $(9,9)$.

$$OPT(D) = OPT(\{(10,10),(7,5),(6,8),(6,2)\},(10,10))$$

La valeur optimale pour l'instance de $2BP$ générée D' n'est généralement pas connue, mais une simple heuristique permet de résoudre de nombreux cas de figure. Notons v_2 l'application itérative de $v_2^{k,l}$. Si $\alpha(n)$ est la complexité de l'heuristique utilisée pour trouver une solution pour l'instance de $2BP$ générée, l'algorithme peut être implanté en $O(HW \times \alpha(n))$. La complexité peut être réduite en utilisant l'observation suivante : les seules valeurs intéressantes pour k et l sont celles qui correspondent à la taille d'un article de A. Par conséquent, la complexité de la méthode est $O(n^2 \times \alpha(n))$.

2.3 Nouvelles bornes inférieures

Nous avons présenté dans l'état de l'art des méthodes pour obtenir des bornes inférieures qui utilisent le concept de fonction dual-réalisable (DFF). En utilisant des DFF sur les deux dimensions d'une instance de $2BP$, toute borne inférieure pour l'instance obtenue est aussi une borne inférieure pour l'instance initiale [31].

2.3.1 Les Fonctions Redondantes ou DFF discrètes

Carlier et Néron [12, 13] introduisent le concept de *Fonction Redondante* afin d'obtenir des évaluations par défaut pour des problèmes d'ordonnancement. Ces fonctions forment un sous-ensemble des DFF dans le cas discret.

Définition 2.2. *Une Fonction Redondante (RF), ou DFF discrète* f *est une application discrète de* $[0,X]$ *dans* $[0,X']$ *(*X *et* X' *entiers) telle que*

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

Dans le reste du document, nous parlons de DFF pour faire référence à ces fonctions discrètes. Dans ce chapitre, nous présentons de nouvelles fonctions, inspirées du travail de Boschetti et Mingozzi [7], qui vont pouvoir être appliquées à des instances de grande taille. Les bornes obtenues dominent strictement celles proposées par Boschetti et Mingozzi [7].

2.3.2 DFF dépendantes de la donnée (DDFF)

Les DFF sont définies indépendamment de l'instance. Une autre classe de fonctions peut être définie, qui dépend de la taille des articles dans l'instance considérée.

Définition 2.3. *Soit* $I = \{1,\ldots,n\}$, $c_1, c_2, \ldots, c_n$ *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*

$$\forall I_1 \subset I, \sum_{i \in I_1} c_i \leq C \Rightarrow \sum_{i \in I_1} f(c_i) \leq f(C) = C'$$

Pour une instance donnée D, une DDFF dépendant de D peut être utilisée comme une DFF. Cette classe de fonctions prend en compte le nombre d'occurences de chaque valeur, et peut ainsi conduire à des évaluations qui améliorent strictement celles obtenues avec les DFF. Nous donnons maintenant un exemple d'une DDFF g qui est meilleure que n'importe quelle DFF pour une instance spécifique. Soit $5, 6, 7$ une liste de valeurs et $C = 10$. $g(5) = 10$, $g(6) = 10$, $g(7) = 10$, $g(10) = 10$. g n'est pas une RF car $20 = g(5) + g(5) > f(10) = 10$.

Théorème 2.3. *Soit* D *une instance de* $2BP$ *et* D' *l'instance obtenue après application de* f *(une DFF ou une DDFF) sur la largeur (resp. la hauteur) de* D*. Toute borne inférieure pour* D' *est aussi une borne inférieure pour* D*.*

Démonstration. Clairement, toute configuration réalisable pour D est réalisable pour D' (voir [31]). Ainsi, l'ensemble de configurations réalisables associé à D' contient l'ensemble des configurations associées à D. □

2.3.3 Trois familles de DFF et DDFF

Nous décrivons maintenant trois familles de fonctions f_0, f_1 et f_2. Ces familles sont inspirées des bornes proposées par Boschetti et Mingozzi [7], mais peuvent être appliquées sur chaque dimension séparément. f_0 et f_2 sont des DFF, f_1 est une DDFF. Nous montrons dans la section suivante que les familles f_0, f_1 et f_2 peuvent être combinées pour obtenir des bornes qui dominent strictement celles de Boschetti et Mingozzi [7]. Dans la suite, C est une valeur entière et k un entier compris entre 1 et $\frac{C}{2}$.

2.3.3.1 La fonction classique f_0

Nous introduisons dans ce paragraphe une DFF directement dérivée d'une DFF classique, déjà proposée par Fekete et Schepers [29], que nous présentons dans le Chapitre 1. Elle repose sur l'observation suivante : si aucun article ne peut être rangé avec un article a_i, la taille de a_i peut être augmentée jusqu'à la taille du bin. D'une manière analogue, pour une valeur donnée k ($1 \leq k \leq \frac{C}{2}$), la taille de tous les articles de taille strictement supérieure à $C - k$ peut être augmentée si tout article plus petit que k est supprimé. Nous notons f_0^k la famille de DFF correspondante.

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

2.3.3.2 La fonction f_1

Soit f_1^k une nouvelle fonction définie pour un paramètre donné k, $1 \leq k \leq \frac{1}{2}C$, et une liste d'entiers $c_1, c_2, \ldots, c_n$ ($I = \{1, \ldots, n\}$). Nous introduisons l'ensemble $J = \{i \in I : \frac{1}{2}C \geq c_i \geq k\}$ et $M_C(X, J)$ la valeur optimale pour le problème de sac à dos en une dimension ($1KP$) induit par l'ensemble J et la taille X. Cette valeur est égale au nombre maximum d'articles qui peuvent être contenus ensemble dans un container de taille X. Le problème peut être résolu en temps linéaire si les articles sont triés par ordre croissant de taille. Cette fonction a été utilisée implicitement par Boschetti et Mingozzi [7].

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

Théorème 2.4. *La fonction f_1^k est une DDFF pour la Donnée $c_1, \ldots, c_n$ et la valeur C.*

Démonstration. Supposons que f_1^k ne soit pas une DDFF. Par conséquent, il existe un sous-ensemble I_1 d'entiers tel que

$$\sum_{i \in I_1} c_i \leq C \text{ et } \sum_{i \in I_1} f_1^k(c_i) > M_C(C, J) \tag{2.1}$$

Nous considérons différents cas qui dépendent de l'existence d'un élément $j \in I_1$ tel que $c_j > \frac{1}{2}C$. Les articles de taille inférieure à k ne sont pas considérés.

1. Il existe $j \in I_1$ tel que $c_j > \frac{1}{2}C$ (par conséquent, on a $i \in J$ pour $i \neq j$).

$$f_1^k(c_j) + \sum_{i \in I_1 - \{j\}} f_1^k(c_i) > M_C(C, J) \tag{2.2}$$

$$M_C(C, J) - M_C(C - c_j, J) + \sum_{i \in I_1 - \{j\}} 1 > M_C(C, J) \tag{2.3}$$

   Comme $\sum_{i \in I_1} c_i \leq C$, $M_C(C - c_j, J) \geq |I_1 - \{j\}|$. Donc,

$$\sum_{i \in I_1 - \{j\}} 1 = |I_1 - \{j\}| > M_C(C - c_j, J) \geq |I_1 - \{inditb\}| \tag{2.4}$$

   On obtient une contradiction.

2. Pour tout $i \in I_1$, $\frac{1}{2}C \geq c_i \geq k$ ($I_1 \subseteq J$). Notez bien que dans ce cas, $M_C(C, J) \geq |I_1|$.

$$|I_1| = \sum_{i \in I_1} 1 > M_C(C, J) \geq |I_1| \tag{2.5}$$

   On obtient une contradiction.

□

Il est important de préciser que f_1 n'est pas une DFF car $M_C(C, J)$ est dépendant des valeurs $c_1, \ldots, c_n$. Appliquer f_1 sur une autre occurence de problème ne permet pas automatiquement d'obtenir des bornes inférieures.

2.3.3.3 La fonction f_2

La fonction f_2 repose sur des techniques d'arrondis. C'est une amélioration de celle qui est utilisée implicitement par Boschetti et Mingozzi [7] dans leur bornes inférieures.

Soit f_2^k la fonction définie de la manière suivante :

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

Théorème 2.5. *La fonction f_2^k est une DFF.*

Démonstration. Supposons que la fonction f_2^k ne soit pas une DFF. Dans ce cas, il existe une liste d'entiers $c_1, \ldots, c_n$ ($I = \{1, \ldots, n\}$) telle que

$$\sum_{i \in I} c_i \leq C \text{ et } \sum_{i \in I} f_2^k(c_i) > 2 \left\lfloor \frac{C}{k} \right\rfloor \tag{2.6}$$

Les cas considérés dans cette preuve dépendent des valeurs présentes dans la liste.

1. Il existe une valeur $j \in I$ telle que $c_j > \frac{1}{2}C$.

$$2 \left\lfloor \frac{C}{k} \right\rfloor - 2 \left\lfloor \frac{C - c_j}{k} \right\rfloor + 2 \sum_{i \in I - \{j\}} \left\lfloor \frac{c_i}{k} \right\rfloor > 2 \left\lfloor \frac{C}{k} \right\rfloor \tag{2.7}$$

$$\sum_{i \in I - \{j\}} \left\lfloor \frac{c_i}{k} \right\rfloor > \left\lfloor \frac{C - c_j}{k} \right\rfloor \tag{2.8}$$

   Rappelons que $\sum_{i \in I - \{j\}} c_i \leq C - c_j$. Dans ce cas,

$$\left\lfloor \frac{C - c_j}{k} \right\rfloor < \sum_{i \in I - \{j\}} \left\lfloor \frac{c_i}{k} \right\rfloor \leq \left\lfloor \sum_{i \in I - \{j\}} \frac{c_i}{k} \right\rfloor \leq \left\lfloor \frac{C - c_j}{k} \right\rfloor \tag{2.9}$$

   On obtient une contradiction.

2. Il existe deux valeurs $j, l \in I$ telles que $c_j = c_l = \frac{C}{2}$. La contradiction est immédiate.

3. Il n'y a qu'une valeur $j \in I$ telle que $c_j = \frac{C}{2}$.

$$\left\lfloor \frac{C}{k} \right\rfloor + 2 \sum_{i \in I - \{j\}} \left\lfloor \frac{c_i}{k} \right\rfloor > 2 \left\lfloor \frac{C}{k} \right\rfloor \tag{2.10}$$

$$2 \sum_{i \in I - \{j\}} \left\lfloor \frac{c_i}{k} \right\rfloor > \left\lfloor \frac{C}{k} \right\rfloor \tag{2.11}$$

Rappelons que $\sum_{i \in I - \{j\}} c_i \leq \frac{C}{2}$. Dans ce cas, $2 \sum_{i \in I - \{j\}} \frac{c_i}{k} \leq \frac{C}{k}$.

$$\left\lfloor \frac{C}{k} \right\rfloor < 2 \sum_{i \in I - \{j\}} \left\lfloor \frac{c_i}{k} \right\rfloor \leq \left\lfloor 2 \sum_{i \in I - \{j\}} \frac{c_i}{k} \right\rfloor \leq \left\lfloor \frac{C}{k} \right\rfloor \tag{2.12}$$

On obtient une contradiction.

4. Pour chaque valeur $i \in I$, $c_i < \frac{1}{2}C$.

$$\sum_{i \in I} c_i \leq C \text{ et } 2 \sum_{i \in I} \left\lfloor \frac{c_i}{k} \right\rfloor > 2 \left\lfloor \frac{C}{k} \right\rfloor \tag{2.13}$$

$$\sum_{i \in I} \left\lfloor \frac{c_i}{k} \right\rfloor \leq \left\lfloor \frac{\sum_{i \in I} c_i}{k} \right\rfloor \leq \left\lfloor \frac{C}{k} \right\rfloor \text{ et } \sum_{i \in I} \left\lfloor \frac{c_i}{k} \right\rfloor > \left\lfloor \frac{C}{k} \right\rfloor \tag{2.14}$$

On obtient une contradiction.

□

2.3.4 Une nouvelles borne inférieure pour $2BP$

2.3.4.1 La nouvelle borne L_{DFF}

Nous introduisons dans un premier temps des bornes dédiées au $1BP$. Pour simplifier les notations, nous notons DFF nos bornes, même si elles utilisent aussi une DDFF. Pour $u = 0, 1, 2$ nous définissons $L_{f_u}^{1BP}$:

$$L_{f_u} = \max_{1 \leq k \leq \frac{C}{2}} \left\{ \left\lceil \frac{\sum_{a_i \in A} f_u^k(c_i)}{f_u^k(C)} \right\rceil \right\} \tag{2.15}$$

$$L_{1BP}^{DFF} = \max_{u \in \{0,1,2\}} \left\{ L_{f_u}^{1BP}(c_i) \right\} \tag{2.16}$$

Nous introduisons trois bornes analogues pour $2BP$. Pour $u = 0, 1, 2$ et $v = 0, 1, 2$ nous définissons $L_{(f_u, f_v)}$:

$$L_{(f_u, f_v)} = \max_{1 \leq k \leq \frac{W}{2}, 1 \leq l \leq \frac{H}{2}} \left\{ \left\lceil \frac{\sum_{a_i \in A} f_u^k(w_i) f_v^l(h_i)}{f_u^k(W) f_v^l(H)} \right\rceil \right\} \tag{2.17}$$

Soit L_{DFF} la plus grande borne continue parmi les instances obtenues après avoir appliqué une DFF sur chaque dimension.

$$L^{DFF} = \max_{u\in\{0,1,2\},v\in\{0,1,2\}} \left\{L_{(f_u,f_v)}\right\} \tag{2.18}$$

Une conséquence immédiate des Théorèmes 2.4 et 2.5 est le fait que L^{DFF} est une borne valide pour $2BP$.

2.3.4.2 Améliorer la borne L_2^{new} de Boschetti et Mingozzi [7]

Soit (k,l) un couple d'entiers tels que $1 \leq k \leq \frac{1}{2}W$ et $1 \leq l \leq \frac{1}{2}H$. Nous notons $A_{gr} = \{a_i \in A : h_i > H-l \text{ et } w_i > W-k\}$, $A_{ht} = \{a_i \in A \setminus A_{gr} : h_i > H-l \text{ et } w_i \geq k\}$, $A_{lg} = \{a_i \in A \setminus A_{gr} : h_i \geq l \text{ et } w_i > W-k\}$, $A_{pt} = \{a_i \in A \setminus A_{gr} \cup A_{ht} \cup A_{lg} : h_i \geq l \text{ et } w_i \geq k\}$.

Boschetti et Mingozzi [7] proposent une borne inférieure notée L_2^{new} pour $2BP$: elle peut être décomposée en deux bornes distinctes. La première repose sur la transformation du problème en $1BP$. La deuxième exploite le fait que deux *gros* objets de A_{gr} ne peuvent pas être rangés dans le même bin et qu'un *haut* article de A_{ht} ne peut pas être rangé avec un article *long* de A_{lg}.

$$L_2^{new} = \max_{1\leq l\leq\frac{1}{2}H,1\leq k\leq\frac{1}{2}W} \{|A_{gr}| + max\{L_2'(k,l), L_2''(k,l)\}\} \tag{2.19}$$

où les bornes $L_2'(k,l)$ et $L_2''(k,l)$ sont deux évaluations pour le nombre minimum de bins nécessaires pour ranger les articles de A_{ht}, A_{lg}, et A_{pt}. Ces bornes sont détaillées dans le premier chapitre de la présente thèse.

Dans les deux bornes, une borne inférieure L_{1BP}^{MV} proposée par Martello et Vigo [53] pour $1BP$ est utilisée. Soit L_2^{2CM} la borne obtenue à partir de L_2^{new} en remplaçant la borne L_{1BP}^{MV} par L_{1BP}^{DFF}. Nous utilisons cette borne car elle ne peut pas toujours être atteinte en utilisant uniquement des DFF.

Proposition 2.2. *L_2^{2CM} est une borne valide pour $2BP$.*

Démonstration. Boschetti et Mingozzi [7] ont montré que leur borne L_2^{new} est une borne valide pour $2BP$. La seule différence entre L_2^{new} et L_2^{2CM} est la borne pour $1BP$ utilisée. L_{1BP}^{DFF} est valide, donc L_2^{2CM} l'est aussi. □

Nous proposons donc une nouvelle borne inférieure.

$$L^{2CM} = max\{L^{DFF}, L_2^{2CM}\}$$

2.4 Relations de dominance

Dans cette section, nous montrons que la borne L^{2CM} domine L^{BM} proposée par Boschetti et Mingozzi [7]. Dans un premier temps, nous montrons que L_{1BP}^{2CM} domine L_{1BP}^{MV} proposé par Martello et Vigo [53].

2.4.1 La borne pour $1BP$ de Martello et Vigo [53]

Martello et Vigo [53] proposent une borne L_{1BP}^{MV} pour $1BP$. Nous donnons ici la formulation brute de cette borne. Le lecteur peut se référer au premier chapitre de la présente thèse pour une présentation plus complète. Soit k un entier, $1 \leq k \leq \frac{1}{2}C$, $A_{gr} = \{a_i \in A : c_i > C - k\}$, $A_{moy} = \{a_i \in A : C - k \geq c_i > \frac{1}{2}C\}$ et $A_{pt} = \{a_i \in A : \frac{1}{2}C \geq c_i \geq k\}$.

$$L_\alpha = \max_{1 \leq k \leq \frac{1}{2}C} \left\{ |A_{gr} \cup A_{moy}| + max \left\{ 0, \left\lceil \frac{\sum_{a_i \in A_{moy} \cup A_{pt}} c_i}{C} - |A_{moy}| \right\rceil \right\} \right\} \tag{2.20}$$

$$L_\beta = \max_{1 \leq k \leq \frac{1}{2}C} \left\{ |A_{gr} \cup A_{moy}| + max \left\{ 0, \left\lceil \frac{|A_{pt}| - \sum_{a_i \in A_{moy}} \left\lfloor \frac{C - c_i}{k} \right\rfloor}{\left\lfloor \frac{C}{k} \right\rfloor} \right\rceil \right\} \right\} \tag{2.21}$$

Proposition 2.3. *La borne L_{1BP}^{DFF} domine la borne L_α proposée par Martello et Vigo [53].*

Démonstration. Pour k donné,

$$L_\alpha(k) = |A_{gr}| + max \left\{ |A_{moy}|, \left\lceil \frac{\sum_{a_i \in A_{moy} \cup A_{pt}} c_i}{C} \right\rceil \right\} \tag{2.22}$$

Deux cas sont possibles :

1. Si $\frac{\sum_{a_i \in A_{moy} \cup A_{pt}} c_i}{C} \leq |A_{moy}|$, $L_\alpha(k) = |A_{gr} \cup A_{moy}|$.
   Dans ce cas, pour chaque k, $L_\alpha(k)$ exploite le fait que le nombre d'articles grands ou moyens est une borne inférieure pour $2BP$. Considérons la DFF $f_0^{\frac{1}{2}C}$ définie précédemment. La borne continue pour l'instance modifiée est la suivante :

$$L_{f_0^{\frac{1}{2}C}} = \left\lceil \frac{\sum_{a_i \in A_{gr} \cup A_{moy}} C + \sum_{\{a_i \in A : c_i = \frac{1}{2}C\}} \frac{1}{2}C}{C} \right\rceil \tag{2.23}$$

$$L_{f_0^{\frac{1}{2}C}} = |A_{gr} \cup A_{moy}| + \frac{1}{2} \left| \left\{ a_i \in A : c_i = \frac{1}{2}C \right\} \right| \tag{2.24}$$

   Dans ce cas particulier, $L_\alpha(k)$ est dominée par L_{f_0} et donc par L_{1BP}^{DFF}.

2. Si $\frac{\sum_{a_i \in A_{moy} \cup A_{pt}} c_i}{C} > |A_{moy}|$, $L_\alpha(k) = |A_{gr}| + \left\lceil \frac{\sum_{a_i \in A_{moy} \cup A_{pt}} c_i}{C} \right\rceil$.
   Dans ce cas, $L_\alpha(k)$ est égale à la borne continue après application de la DFF f_0^k. Elle est donc dominée par L_{1BP}^{DFF}.

□

Proposition 2.4. *La borne L_{1BP}^{DFF} domine la borne L_β proposée par Martello et Vigo [53].*

Démonstration. Pour k fixé,

$$L_\beta(k) = |A_{gr} \cup A_{moy}| + max\left\{0, \left\lceil \frac{|A_{pt}| - \sum_{a_i \in A_{moy}} \left\lfloor \frac{C-c_i}{k} \right\rfloor}{\left\lfloor \frac{C}{k} \right\rfloor} \right\rceil\right\} \tag{2.25}$$

Pour le cas où $L_\beta(k) = |A_{gr} \cup A_{moy}|$, se référer à la preuve de la Proposition 2.3.
Si $|A_{pt}| > \sum_{a_i \in A_{moy}} \left\lfloor \frac{C-c_i}{k} \right\rfloor$, $L_\beta(k) = |A_{gr} \cup A_{moy}| + \left\lceil \frac{|A_{pt}| - \sum_{a_i \in A_{moy}} \left\lfloor \frac{C-c_i}{k} \right\rfloor}{\left\lfloor \frac{C}{k} \right\rfloor} \right\rceil$

La méthode utilisée pour évaluer le nombre d'articles de A_{pt} qui peuvent être rangés ensemble dans le même bin est équivalente à une modification de leur taille suivant un paramètre k. La taille de chaque grand article a_i est augmentée de telle manière que le même nombre de petits objets peut être rangé avec a_i.

Nous montrons maintenant qu'une meilleure évaluation peut être obtenue en appliquant la famille de DFF f_2^k. La borne continue obtenue sur l'instance modifiée par cette fonction est la suivante :

$$L_{f_2^k} = \left\lceil \frac{\sum_{a_i \in A_{gr} \cup A_{moy} \cup A_{pt}} f_2^k(c_i)}{2\left\lfloor \frac{C}{k} \right\rfloor} \right\rceil \tag{2.26}$$

Pour simplifier la preuve, nous prenons en compte le fait que f_2^k est une fonction croissante. Ainsi, nous considérons que $f_2^k(x) \geq 2\lfloor \frac{x}{k} \rfloor$ si $x = \frac{C}{2}$.

$$L_{f_2^k} \geq \left\lceil \frac{2\sum_{a_i \in A_{gr}} \left\lfloor \frac{C}{k} \right\rfloor + 2\sum_{a_i \in A_{pt}} \left\lfloor \frac{c_i}{k} \right\rfloor + 2\sum_{a_i \in A_{moy}} \left(\left\lfloor \frac{C}{k} \right\rfloor - \left\lfloor \frac{C-c_i}{k} \right\rfloor \right)}{2\left\lfloor \frac{C}{k} \right\rfloor} \right\rceil \tag{2.27}$$

$$L_{f_2^k} \geq |A_{gr} \cup A_{moy}| + \left\lceil \frac{\sum_{a_i \in A_{pt}} \left\lfloor \frac{c_i}{k} \right\rfloor - \sum_{a_i \in A_{moy}} \left\lfloor \frac{C-c_i}{k} \right\rfloor}{\left\lfloor \frac{C}{k} \right\rfloor} \right\rceil \tag{2.28}$$

Pour $a_i \in A_{pt}$, on a $c_i \geq k$, et donc $\lfloor \frac{c_i}{k} \rfloor \geq 1$.

$$L_{f_2^k} \geq |A_{gr} \cup A_{moy}| + \left\lceil \frac{|A_{pt}| - \sum_{a_i \in A_{moy}} \left\lfloor \frac{C-c_i}{k} \right\rfloor}{\left\lfloor \frac{C}{k} \right\rfloor} \right\rceil = L_\beta(k) \tag{2.29}$$

$$L_{1BP}^{DFF} \geq L_{f_2^k} \geq L_\beta \tag{2.30}$$

□

Rappelons que $L_{1BP}^{MV} = max\{L_\alpha, L_\beta\}$. D'après les Propositions 2.3 et 2.4, nous obtenons le résultat suivant :

Théorème 2.6. *La borne L_{1BP}^{DFF} domine L_{1BP}^{MV} proposé par Martello et Vigo [53].*

2.4.2 Les bornes L_1^{new} et L_2^{new} de Boschetti et Mingozzi [7]

La seule différence entre L_2^{new} et L_2^{2CM} est la borne inférieure pour $1BP$ utilisée. D'après le Théorème 2.6, L_{1BP}^{DFF} domine L_{1BP}^{MV}, donc L_2^{2CM} domine L_2^{new}. Le résultat suivant est valide.

Corollaire 2.1. *La borne L_2^{2CM} domine la borne L_2^{new} proposée par Boschetti et Mingozzi [7].*

Comme Boschetti et Mingozzi [7] ont montré que la borne L_1^{new} est dominée par L_2^{new}, L_2^{2CM} la domine aussi.

2.4.3 La borne L_3^{new} de Boschetti et Mingozzi [7]

La borne L_3^{new} génère des problèmes type sac à dos afin de déterminer le nombre de bins nécessaires pour ranger les petits objets qui ne peuvent pas l'être avec les grands (*cf.* Chapitre 1).

Soit k et l deux entiers tels que $1 \leq l \leq \frac{1}{2}H$ et $1 \leq k \leq \frac{1}{2}W$, $A_{gr} = \{a_i \in A : h_i > H - l \text{ et } w_i > W - k\}$, et $A_{moy} = \{a_i \in A \setminus A_{gr} : h_i > \frac{1}{2}H \text{ et } w_i > \frac{1}{2}W\}$, et $A_{pt} = \{a_i \in A \setminus A_{gr} \cup A_{moy} : h_i \geq l \text{ et } w_i \geq k\}$. Plusieurs valeurs doivent être calculées :

- $M_W(W, A_{pt})$: le nombre maximum d'articles de A_{pt} qui peuvent être rangés côte à côte dans un bin ;
- $M_H(H, A_{pt})$: le nombre maximum d'articles de A_{pt} qui peuvent être rangés les uns au-dessus des autres dans un bin ;
- $M_W(W - w_i, A_{pt})$: le nombre maximum d'articles de A_{pt} qui peuvent être rangés côte à côte dans un bin contenant l'article a_i de A_{moy} ;
- $M_H(H - h_i, A_{pt})$: le nombre maximum d'articles de A_{pt} qui peuvent être rangés les uns au-dessus des autres dans un bin contenant l'article a_i de A_{moy}.

$$L_3^{new} = \max_{\substack{1 \leq k \leq \frac{1}{2}W \\ 1 \leq l \leq \frac{1}{2}H}} \left\{ |A_{gr} \cup A_{moy}| + max \left\{ 0, \left\lceil \frac{|A_{pt}| - \sum_{a_i \in A_{moy}} m'(a_i, A_{pt})}{M_W(W, A_{pt}) M_H(H, A_{pt})} \right\rceil \right\} \right\} \tag{2.31}$$

où $m'(a_i, A_{pt})$ est une borne supérieure pour le nombre d'articles de A_{pt} qui peuvent être rangés ensemble avec a_i :

$m'(a_i, A_{pt}) = M_W(W, A_{pt}) M_H(H - h_i, A_{pt}) + M_W(W - w_i, A_{pt}) M_H(H, A_{pt}) - M_W(W - w_j, A_{pt}) M_H(H - h_i, A_{pt})$

Notez que $M_W(W, A_{pt}) M_H(H, A_{pt})$ est une borne supérieure pour le nombre d'articles de A_{pt} qui peuvent être rangés ensemble dans un bin.

Théorème 2.7. *L^{2CM} domine la borne L_3^{new} proposée par Boschetti et Mingozzi [7].*

Démonstration.

$$L_3^{new}(l, k) = |A_{gr} \cup A_{moy}| + max \left\{ 0, \left\lceil \frac{|A_{pt}| - \sum_{a_i \in A_{moy}} m'(a_i, A_{pt})}{M_W(W, A_{pt}) M_H(H, A_{pt})} \right\rceil \right\} \tag{2.32}$$

Si $L_3^{new}(l,k) = |A_{gr} \cup A_{moy}|$, L_3^{new} est dominée par L^{DFF} car elle est dominée par la borne continue après application de $f_0^{\frac{1}{2}W}$ et $f_0^{\frac{1}{2}H}$. Nous nous intéressons à l'autre cas.

$$L_3^{new}(l,k) = |A_{gr} \cup A_{moy}| + \left\lceil \frac{|A_{pt}| - \sum_{a_i \in A_{moy}} m'(a_i, A_{pt})}{M_W(W, A_{pt}) M_H(H, A_{pt})} \right\rceil \tag{2.33}$$

Nous montrons maintenant que f_1 conduit à une borne qui est égale à L_3^{new}. Pour simplifier l'écriture, nous introduisons $W' = M_W(W, A_{pt})$ et $H' = M_H(H, A_{pt})$.

$$L_{f_1^{k,l}} = \left\lceil \sum_{a_i \in A} \frac{f_1^k(w_i) \times f_1^l(h_i)}{W'H'} \right\rceil \tag{2.34}$$

En remplaçant $f_1^k(w_i)$ et $f_1^l(h_i)$ par leur valeur, on obtient :

$$\begin{aligned} L_{f_1^{k,l}} = & \left\lceil \frac{\sum_{a_i \in A_{gr}} W'H' + \sum_{a_i \in A_{pt}} 1}{W'H'} \right. \\ & \left. + \frac{\sum\limits_{a_i \in A_{moy}} (W' - M_W(W - w_i, A_{pt}))(H' - M_H(H - h_i, A_{pt}))}{W'H'} \right\rceil \end{aligned} \tag{2.35}$$

$$\begin{aligned} L_{f_1^{p,q}} = & |A_{gr}| + \\ & \left\lceil \frac{|A_{pt}| + \sum\limits_{a_i \in A_{moy}} (W' - M_W(W - w_i, A_{pt}))(H' - M_H(H - h_i, A_{pt}))}{W'H'} \right\rceil \end{aligned} \tag{2.36}$$

$$\begin{aligned} L_{f_1^{p,q}} = & |A_{gr} \cup A_{moy}| + \left\lceil \sum_{a_i \in A_{moy}} \Big(- \frac{H' M_W(W - w_i, A_{pt})}{W'H'} - \frac{W' M_H(H - h_i, A_{pt})}{W'H'} \right. \\ & \left. + \frac{M_H(H - h_i, A_{pt}) M_W(W - w_i, A_{pt})}{W'H'} \Big) + \frac{|A_{pt}|}{W'H'} \right\rceil = L_3^{new}(k,l) \end{aligned} \tag{2.37}$$

$$L^{2CM} \geq L_{f_1^{k,l}} = L_3^{new} \tag{2.38}$$

□

2.4.4 La borne L_4^{new} de Boschetti et Mingozzi [7]

La borne L_4^{new} utilise une autre méthode pour évaluer le nombre de petits articles qui ne peuvent pas être rangés avec des grands et le nombre de bins nécessaires pour les ranger. Pour cela, des techniques spéciales d'arrondis sont utilisées (*cf.* Chapitre 1).

Soit $A_{gr} = \{a_i \in A : h_i > H - l \text{ et } w_i > W - k\}$, $A_{moy} = \{a_i \in A \setminus A_{gr} : h_i > \frac{1}{2}H \text{ et } w_i > \frac{1}{2}W\}$, $A^t_{pt} = \{a_i \in A : h_i > \frac{1}{2}H \text{ et } \frac{1}{2}W \geq w_i \geq k\}$, $A^w_{pt} = \{a_i \in A : \frac{1}{2}H \geq h_i \geq l \text{ et } w_i > \frac{1}{2}W\}$, $A^s_{pt} = \{a_i \in A : \frac{1}{2}H \geq h_i \geq l \text{ et } \frac{1}{2}W \geq w_i \geq k\}$.

$$L_4^{new} = \max_{\substack{1\leq k\leq \frac{W}{2}\\ 1\leq l\leq \frac{H}{2}}} \left\{ |A_{gr} \cup A_{moy}| + max\left\{0, \left\lceil \frac{\sum_{a_i \in A\setminus \widehat{A}_1} m''(a_i,k,l)}{\lfloor \frac{H}{l} \rfloor \lfloor \frac{W}{k} \rfloor} \right\rceil \right\} \right\} \tag{2.39}$$

où

- $m''(a_i,k,l) = \lfloor \frac{W-w_i}{k} \rfloor \lfloor \frac{H-h_i}{l} \rfloor - \lfloor \frac{W-w_i}{k} \rfloor \lfloor \frac{H}{l} \rfloor - \lfloor \frac{W}{k} \rfloor \lfloor \frac{H-h_i}{l} \rfloor$, if $a_i \in A_{moy}$
- $m''(a_i,k,l) = \lfloor \frac{w_i}{k} \rfloor \lfloor \frac{H}{l} \rfloor - \lfloor \frac{w_i}{k} \rfloor \lfloor \frac{H-h_i}{l} \rfloor$, if $a_i \in A^t_{pt}$
- $m''(a_i,k,l) = \lfloor \frac{W}{k} \rfloor \lfloor \frac{h_i}{l} \rfloor - \lfloor \frac{W-w_i}{k} \rfloor \lfloor \frac{h_i}{l} \rfloor$, if $a_i \in A^w_{pt}$
- $m''(a_i,k,l) = \lfloor \frac{w_i}{k} \rfloor \lfloor \frac{h_i}{l} \rfloor$, if $a_i \in A^s_{pt}$

Théorème 2.8. *L^{2CM} domine la borne L_4^{new} proposée par Boschetti et Mingozzi [7].*

Démonstration. Nous ne considérons que le cas où la borne est différente de $|A_{gr} \cup A_{moy}|$. Nous montrons maintenant que f_2 conduit à une meilleure borne.

$$L_{f_2^{k,l}} = \left\lceil \sum_{a_i \in A} \frac{f_2^k(w_i) \times f_2^l(h_i)}{2\lfloor \frac{W}{k} \rfloor 2 \lfloor \frac{H}{l} \rfloor} \right\rceil \tag{2.40}$$

Afin de simplifier les notations, nous prenons en compte le fait que f_2 est une fonction croissante. On considère donc que $f_2^k(x) \geq 2\lfloor \frac{x}{k} \rfloor$ si $x = \frac{C}{2}$. En remplaçant f_2^l et f_2^k par leurs valeurs, on obtient

$$\begin{aligned} L_{f_2^{k,l}} \geq & |A_{gr}| + \left\lceil \sum_{a_i \in A_{moy}} \frac{\left(2\lfloor \frac{H}{l} \rfloor - 2\lfloor \frac{H-h_i}{l} \rfloor\right)\left(2\lfloor \frac{W}{k} \rfloor - 2\lfloor \frac{W-w_i}{k} \rfloor\right)}{4\lfloor \frac{W}{k} \rfloor \lfloor \frac{H}{l} \rfloor} \right. \\ & + \sum_{a_i \in A^t_{pt}} \frac{\left(2\lfloor \frac{H}{l} \rfloor - 2\lfloor \frac{H-h_i}{l} \rfloor\right) 2\lfloor \frac{w_i}{k} \rfloor}{4\lfloor \frac{W}{k} \rfloor \lfloor \frac{H}{l} \rfloor} \\ & \left. + \sum_{a_i \in A^w_{pt}} \frac{2\lfloor \frac{h_i}{l} \rfloor \left(2\lfloor \frac{W}{k} \rfloor - 2\lfloor \frac{W-w_i}{k} \rfloor\right)}{4\lfloor \frac{W}{k} \rfloor \lfloor \frac{H}{l} \rfloor} + \sum_{a_i \in A^s_{pt}} \frac{4\lfloor \frac{h_i}{l} \rfloor \lfloor \frac{w_i}{k} \rfloor}{4\lfloor \frac{W}{k} \rfloor \lfloor \frac{H}{l} \rfloor} \right\rceil \end{aligned} \tag{2.41}$$

$$\begin{aligned} L_{f_2^{k,l}} \geq |A_{gr} \cup A_{moy}| + & \left\lceil \sum_{a_i \in A_{moy}} \frac{-\lfloor \frac{H}{l} \rfloor \lfloor \frac{W-w_i}{k} \rfloor - \lfloor \frac{W}{k} \rfloor \lfloor \frac{H-h_i}{l} \rfloor + \lfloor \frac{W-w_i}{k} \rfloor \lfloor \frac{H-h_i}{l} \rfloor}{\lfloor \frac{H}{l} \rfloor \lfloor \frac{W}{k} \rfloor} \right. \\ & + \sum_{a_i \in A^t_{pt}} \frac{\lfloor \frac{H}{l} \rfloor \lfloor \frac{w_i}{k} \rfloor - \lfloor \frac{H-h_i}{l} \rfloor \lfloor \frac{w_i}{k} \rfloor}{\lfloor \frac{H}{l} \rfloor \lfloor \frac{W}{k} \rfloor} \\ & \left. + \sum_{a_i \in \widehat{A^w_{pt}}} \frac{\lfloor \frac{W}{k} \rfloor \lfloor \frac{h_i}{l} \rfloor - \lfloor \frac{W-w_i}{k} \rfloor \lfloor \frac{h_i}{l} \rfloor}{\lfloor \frac{H}{l} \rfloor \lfloor \frac{W}{k} \rfloor} + \sum_{a_i \in \widehat{A^s_{pt}}} \frac{\lfloor \frac{h_i}{l} \rfloor \lfloor \frac{w_i}{k} \rfloor}{\lfloor \frac{H}{l} \rfloor \lfloor \frac{W}{k} \rfloor} \right\rceil \end{aligned} \tag{2.42}$$

$$L_{f_2^{k,l}} \geq |A_{gr} \cup A_{moy}| + \left\lceil \sum_{a_i \in A \setminus A_{gr}} \frac{m''(a_i,k,l)}{\lfloor \frac{H}{l} \rfloor \lfloor \frac{W}{k} \rfloor} \right\rceil = L_4^{new}(k,l) \tag{2.43}$$

$$L^{2CM} \geq L_{f_2^{k,l}} \geq L_4^{new} \tag{2.44}$$

□

2.5 Résultats expérimentaux

Nous avons testé nos procédures de réduction et nos bornes inférieures sur des jeux d'essai de la littérature [5, 53]. Le jeu d'essai est composé de dix classes d'instances générées aléatoirement. Chaque classe contient cinq groupes de dix instances chacun [2]. Les résultats des méthodes proposées par Martello et Vigo [53] , Fekete et Schepers [31], et Boschetti et Mingozzi [7] sont connus pour ces jeux d'essai [8].

2.5.1 Procédures de réduction

Nous avons testé nos procédures de réduction et nous les avons comparées avec la méthode proposée par Boschetti et Mingozzi [7] (Tableau 2.1). L'heuristique utilisée pour le problème de décision à résoudre dans v_1 et v_2 repose sur l'algorithme classique *Bottom Left Decreasing* (*cf.* Chapitre 1). L'ensemble des articles est trié par ordre décroissant de surface. Les surfaces libres sont considérées de la gauche vers la droite, l'article considéré à chaque étape est rangé dans la première surface libre. Une meilleure heuristique permettrait d'améliorer les résultats, mais nous avons préféré utiliser une méthode rapide.

Les critères de comparaison pour les procédures de réduction sont le nombre d'articles restants après avoir supprimé les articles de taille (W, H) et la surface totale des articles obtenus. Nous reproduisons quatre résultats pour chaque critère : un pour chaque procédure testée, et un obtenu par l'application successive des trois procédures de réduction (*i.e.* chaque fonction est appliquée à tour de rôle tant que l'instance est modifiée. Ce dernier résultat est reproduit dans la colonne *all*.

Dans la plupart des cas, la fonction u est moins efficace pour supprimer des articles, mais elle est parfois utile pour améliorer les résultats des deux autres procédures. La fonction v_1 est la meilleure en moyenne pour réduire la taille de l'instance. La seule classe pour laquelle il y a de meilleurs résultats (obtenus avec v_2) est la classe IX. Pour les classes I, III et $VIII$, v_1 est la seule méthode qui supprime des articles dont la taille n'est pas initialement égale au bin. La fonction v_2 est souvent moins efficace que v_1, mais elle est très utile lorsque les articles sont grands, comme dans la classe IX.

Pour la surface totale, la fonction u est en moyenne aussi bonne que la fonction v_2. Encore une fois, la fonction v_1 est la meilleure méthode pour ce critère. Elle ne retourne un résultat inférieur par rapport à v_2 que pour la classe IX. Dans la plupart des cas, v_1 retourne les meilleurs résultats. Notez que les résultats de la

2. Ces données sont disponibles à l'URL suivante : *http ://www.or.deis.unibo.it/ORinstances*

colonne *all* sont meilleurs que le maximum entre les trois autres résultats obtenus. Il paraît donc intéressant d'utiliser les trois procédures itérativement pour obtenir la meilleure procédure de réduction. Toutefois, utiliser v_1 est suffisant pour la plupart des instances.

2.5.2 Bornes inférieures

Nous avons testé nos bornes inférieures sur les mêmes jeux d'essai que les procédures de réduction (Tableau 2.2). Deux critères sont reproduits pour comparer les bornes :

1. La valeur U/L est le ratio entre la borne supérieure proposée par Boschetti et Mingozzi [8] et la borne inférieure testée [3].
2. OPT est le nombre d'instances résolues par les bornes ($UB = LB$) parmi les dix instances considérées.

Notre borne inférieure est notée L^{2CM}, et la borne de Boschetti et Mingozzi L^{BM}. Dans le Tableau 2.2, les colonnes BM et $2CM$ contiennent les résultats pour ces deux bornes. Pour toutes les classes, la dominance théorique est confirmée. L^{2CM} est toujours meilleure que L^{BM}. Les résultats montrent que nos bornes inférieures ne sont pas équivalentes aux bornes proposées par [7]. L'amélioration est stricte pour plusieurs classes. Ces améliorations sont dues à la possibilité d'utiliser une famille de fonction différente pour la hauteur et pour la largeur.

2.6 Conclusion

Nous avons proposé des procédures de réduction qui réduisent la taille de l'instance initiale. Elles peuvent être combinées et réduisent considérablement la taille du problème pour plusieurs instances d'un jeu d'essai de la littérature. Nous avons aussi proposé des bornes inférieures pour $2BP$ et montré qu'elles dominent les meilleurs résultats connus. La dominance n'est pas seulement théorique car les résultats obtenus expérimentalement sont toujours meilleurs. L'amélioration est stricte pour plusieurs instances. Plusieurs méthodes peuvent être proposées pour améliorer les évaluations : utiliser des compositions des (D)DFF que nous avons proposées dans ce chapitre, mais aussi utiliser le générateur de fonctions dual-réalisable proposé par Carlier et Néron [13] pour obtenir de nouvelles fonctions.

3. Les valeurs de bornes supérieures et inférieures nous ont aimablement été envoyées par Marco Boschetti.

Problème			% Articles restants				Surface totale			
Class	$H \times W$	n	u	v_1	v_2	all	u	v_1	v_2	all
I	10×10	20	98.50	80.50	98.50	76.50	6.40	6.60	6.40	6.70
		40	98.50	68.50	98.50	63.00	12.00	12.90	12.00	13.10
		60	99.00	77.83	99.00	70.50	18.50	19.40	18.50	19.60
		80	99.00	77.00	99.00	71.88	25.30	26.50	25.30	26.60
		100	99.00	84.40	99.00	84.20	30.50	31.10	30.50	31.10
II	30×30	20	100.00	100.00	100.00	10.00	1.00	1.00	1.00	1.00
		40	100.00	100.00	100.00	99.50	1.90	1.90	1.90	1.90
		60	100.00	100.00	100.00	99.83	2.50	2.50	2.50	2.50
		80	100.00	100.00	100.00	99.88	3.10	3.10	3.10	3.10
		100	100.00	100.00	100.00	99.10	3.90	3.90	3.90	3.90
III	40×40	20	100.00	91.00	100.00	91.00	4.50	4.50	4.40	4.60
		40	100.00	85.25	100.00	81.00	8.20	8.60	8.20	8.70
		60	100.00	94.67	100.00	92.00	12.50	12.50	12.50	12.50
		80	100.00	90.25	100.00	90.25	17.30	17.60	17.30	17.70
		100	100.00	98.50	100.00	98.50	20.50	20.60	20.50	20.60
IV	100×100	20	100.00	100.00	100.00	100.00	1.00	1.00	1.00	1.00
		40	100.00	100.00	100.00	100.00	1.90	1.90	1.90	1.90
		60	100.00	100.00	100.00	100.00	2.30	2.30	2.30	2.30
		80	100.00	100.00	100.00	100.00	3.00	3.00	3.00	3.00
		100	100.00	100.00	100.00	100.00	3.70	3.70	3.70	3.70
V	100×100	20	99.50	83.50	98.50	72.00	5.40	5.60	5.40	6.00
		40	100.00	88.50	93.25	74.75	10.20	10.50	10.30	11.20
		60	99.83	89.83	100.00	86.67	15.70	16.10	15.70	16.20
		80	99.88	85.88	99.88	82.25	21.50	22.30	21.50	22.50
		100	100.00	91.30	99.60	91.00	26.00	26.30	25.90	26.30
VI	300×300	20	100.00	100.00	100.00	100.00	1.00	1.00	1.00	1.00
		40	100.00	100.00	100.00	100.00	1.50	1.50	1.50	1.50
		60	100.00	100.00	100.00	100.00	2.10	2.10	2.10	2.10
		80	100.00	100.00	100.00	100.00	3.00	3.00	3.00	3.00
		100	100.00	100.00	100.00	100.00	3.20	3.20	3.20	3.20
VII	100×100	20	100.00	96.00	98.50	96.00	5.10	4.90	4.70	5.30
		40	99.75	98.50	99.75	97.75	9.90	9.90	9.70	10.10
		60	100.00	97.67	99.67	97.50	14.00	14.30	14.00	14.30
		80	99.88	96.88	100.00	96.88	20.10	20.10	19.70	20.70
		100	100.00	96.50	100.00	96.50	23.90	24.50	23.80	24.60
$VIII$	100×100	20	100.00	97.50	100.00	95.50	5.10	4.90	4.80	5.40
		40	100.00	97.25	100.00	97.25	10.00	10.00	9.60	10.20
		60	100.00	97.50	100.00	97.50	14.30	14.40	14.10	14.70
		80	100.00	97.62	100.00	97.62	19.90	19.90	19.50	20.40
		100	100.00	94.50	100.00	94.10	24.10	25.30	24.10	25.60
IX	100×100	20	93.00	46.00	13.00	0.00	11.20	12.10	14.10	14.30
		40	98.00	39.75	20.25	3.75	19.70	24.00	26.50	27.80
		60	98.67	29.50	6.50	0.50	28.90	38.70	42.90	43.70
		80	99.38	27.50	6.88	1.62	37.70	52.60	56.10	57.70
		100	99.90	15.80	5.20	1.60	45.40	65.60	67.90	69.40
X	100×100	20	100.00	76.00	97.00	67.00	3.80	3.80	3.80	3.90
		40	100.00	90.50	100.00	90.50	6.90	7.00	6.90	7.00
		60	100.00	94.00	100.00	93.00	9.50	9.60	9.40	9.60
		80	100.00	95.37	100.00	94.00	12.20	12.30	12.20	12.30
		100	100.00	97.30	100.00	97.30	15.30	15.30	15.30	15.30

Tableau 2.1 – Procédures de réduction

Problème			Ratio U/L		Opt	
Class	$H \times W$	n	BM	2CM	BM	2CM
I	10 × 10	20	1.0125	1.0125	9	9
		40	1.0265	1.0265	7	7
		60	1.0227	1.0171	6	7
		80	1.0042	1.0000	9	10
		100	1.0032	1.0032	9	9
		Avg	1.0138	1.0119	8.0	8.4
II	30 × 30	20	1.0000	1.0000	10	10
		40	1.0000	1.0000	10	10
		60	1.0000	1.0000	10	10
		80	1.0000	1.0000	10	10
		100	1.0000	1.0000	10	10
		Avg	1.0000	1.0000	10	10
III	40 × 40	20	1.0367	1.0367	8	8
		40	1.0411	1.0411	7	7
		60	1.0315	1.0244	6	7
		80	1.0219	1.0160	6	7
		100	1.0251	1.0251	5	5
		Avg	1.0313	1.0286	6.4	6.8
IV	100 × 100	20	1.0000	1.0000	10	10
		40	1.0000	1.0000	10	10
		60	1.1000	1.1000	8	8
		80	1.1000	1.1000	7	7
		100	1.0333	1.0333	9	9
		Avg	1.0467	1.0467	8.8	8.8
V	100 × 100	20	1.0000	1.0000	10	10
		40	1.0271	1.0271	7	7
		60	1.0110	1.0110	8	8
		80	1.0295	1.0295	4	4
		100	1.0308	1.0308	3	3
		Avg	1.0197	1.0197	6.4	6.4
VI	300 ×300	20	1.0000	1.0000	10	10
		40	1.3000	1.3000	7	7
		60	1.0000	1.0000	10	10
		80	1.0000	1.0000	10	10
		100	1.0667	1.0667	8	8
		Avg	1.0733	1.0733	9.0	9.0
VII	100 × 100	20	1.0200	1.0000	9	10
		40	1.0202	1.0111	8	9
		60	1.0257	1.0186	6	7
		80	1.0367	1.0233	2	5
		100	1.0188	1.0108	5	7
		Avg	1.0243	1.0128	6.0	7.6
VIII	100 × 100	20	1.0000	1.0000	10	10
		40	1.0091	1.0091	9	9
		60	1.0184	1.0121	7	8
		80	1.0139	1.0139	7	7
		10	1.0217	1.0217	4	4
		Avg	1.0126	1.0114	7.4	7.6
IX	100 × 100	20	1.0000	1.0000	10	10
		40	1.0000	1.0000	10	10
		60	1.0000	1.0000	10	10
		80	1.0000	1.0000	10	10
		100	1.0000	1.0000	10	10
		Avg	1.0000	1.0000	10.0	10.0
X	100 × 100	20	1.0950	1.0700	7	8
		40	1.0343	1.0343	8	8
		60	1.0533	1.0450	5	6
		80	1.0563	1.0563	3	3
		100	1.0584	1.0584	1	1
		Avg	1.0595	1.0528	4.8	5.2

Tableau 2.2 – Bornes inférieures

Améliorations des bornes inférieures

3.1 Introduction

Dans le Chapitre 2, nous avons proposé des bornes inférieures pour $2BP$ qui dominent les résultats de la littérature. Dans ce chapitre, nous montrons comment elles peuvent être améliorées en proposant de nouvelles (D)DFF. Pour ce faire, nous comparons deux méthodes pour générer de telles fonctions. La première repose sur la composition itérative des fonctions f_0^k, f_1^k et f_2^k proposées dans le Chapitre 2. Nous montrons que le nombre de fonctions intéressantes pour une instance donnée est borné par une fonction polynomiale du nombre d'articles. La deuxième méthode utilise le générateur de fonctions de Carlier et Néron [12,13]. Les auteurs introduisent le concept de DFF Maximales (MDFF), et proposent un algorithme [13] pour calculer un sous-ensemble de ces fonctions. Nous testons l'efficacité de cette méthode et nous la comparons à la première. Nous montrons aussi que les deux DFF proposées dans le Chapitre 2 sont des MDFF et que les bornes décrites dans ce chapitre dominent celles proposées par Labbé *et al.* [43]. Nos méthodes sont testées expérimentalemenet sur les jeux d'essai de la littérature pour $1BP$ [29] et $2BP$ (*cf.* Chapitre 2). Nous analysons l'impact de la composition des DFF obtenues avec la fonction f_1^k, et avec la méthode L_2^{BM} proposée par Boschetti et Mingozzi [7]. Nous améliorons les résultats connus pour plusieurs instances de $1BP$ et de $2BP$.

Dans la Section 3.2, nous étudions les compositions de (D)DFF, puis nous présentons le concept de MDFF dans la section 3.3, ainsi qu'un générateur de MDFF. Dans la Section 3.4, nous montrons comment elles peuvent être utilisées pour obtenir des bornes pour $1BP$ et $2BP$, qui dominent celles obtenues précédemment. La Section 3.5 est dédiée aux résultats numériques.

3.2 Composition de DFF et de DDFF

Afin d'améliorer les résultats, les DFF et les DDFF peuvent être composées pour obtenir de nouvelles fonctions. Par définition, une composition de DFF est une DFF, et une composition de (D)DFF est une (D)DFF.

Proposition 3.1. *Une composition de (D)DFF est une (D)DFF.*

Les (D)DFF modifient à la fois la taille du bin et celle des articles, il faut donc être veiller à appliquer des fonctions avec des paramètres adaptés à la nouvelle taille

du bin. L'ensemble de toutes les compositions itératives possibles des fonctions f_0^k, f_1^l et f_2^m est infini, il est donc intéressant de déterminer les compositions inutiles. Nous montrons ainsi que le nombre de compositions dominantes est polynomial en fonction du nombre d'articles. Pour comparer les fonctions définies pour des valeurs différentes de C', nous introduisons la notion de dominance entre fonctions.

Définition 3.1. *Soit f et f' deux fonctions dual-réalisables. f est* dominée *par f' ($f \leq f'$) si $\frac{f(c_i)}{f(C)} \leq \frac{f'(c_i)}{f'(C)}$ pour tout $c_i \leq C$.*

Si de plus, il existe une valeur j telle que $\frac{f(c_j)}{f(C)} < \frac{f'(c_j)}{f'(C)}$, on dit que f est strictement dominée *par f' ($f < f'$).*

Nous étudions maintenant la composition de chaque famille de fonctions f_0^k, f_1^l ou f_2^m deux à deux et nous montrons à chaque fois qu'elle est dominée par une autre fonction, ou qu'elle peut conduire à de meilleures bornes. Par exemple, pour $k \geq l$, il est évident que $f_0^k \circ f_0^l$ et $f_0^l \circ f_0^k$ sont équivalentes à f_0^k. Une fois tous les couples de fonctions étudiés, nous exploitons les résultats obtenus pour proposer une famille dominante de compositions dont le nombre est de l'ordre de n^2. Le résultat principal que nous obtenons est le suivant : **toutes les combinaisons itératives de fonctions sont dominées par une fonction de la forme décrite dans l'Equation (3.1).**

$$f_1^1 \circ f_2^l \circ f_0^k \text{ ou } f_2^l \circ f_0^k \text{ pour } 1 \leq k \leq \frac{C}{2} \text{ et } 1 \leq l \leq \frac{C}{2} \tag{3.1}$$

Pour cela, nous montrons que seule f_2^l est utile lorsque l'on considère une instance tranformée par f_0^k (Propositions 3.2, 3.3, 3.4, et 3.5). De plus, aucune fonction ne devrait être appliquée après f_1^k (Propositions 3.6, 3.7, et 3.8). Nous montrons aussi que seule la fonction f_1^l devrait être appliquée après la fonction f_2^k (Propositions 3.9, et 3.10).

3.2.1 Appliquer une fonction après f_0^k

Composer deux fonctions de la famille f_0^k n'a pas d'intérêt, et appliquer f_1^l après f_0^k conduit à une fonction strictement égale à f_1^m ou f_1^k. La seule fonction qu'il est intéressant d'appliquer après f_0^k est la fonction f_2^l. Le premier résultat est immédiat.

Proposition 3.2. *Pour $1 \leq k \leq \frac{C}{2}$ et $1 \leq l \leq \frac{C}{2}$, $f_0^k \circ f_0^l = f_0^m$, où $m = max\{k, l\}$.*

Proposition 3.3. *Pour $1 \leq l \leq k \leq \frac{C}{2}$, $f_1^l \circ f_0^k = f_1^k$.*

Démonstration. Lorsque la fonction f_0^k est appliquée, on modifie l'ensemble I des articles à considérer pour les valeurs dépendantes de la donnée. Pour simplifier la démonstration, nous décomposons l'ensemble I en trois parties : $I_1 = \{i \in I : c_i > \frac{C}{2}\}$, $I_2 = \{i \in I : \frac{C}{2} \geq c_i \geq k\}$, $I_3 = \{i \in I : k > c_i \geq 1\}$. Après application de la fonction f_0^k, tous les articles de I_3 ont une taille égale à 0. Posons $I^k = \{i \in I : c_i \geq k\} = I_1 \cup I_2$ le nouvel ensemble à considérer après application de f_0^k. On peut définir le nouvel ensemble dépendant de la donnée $J_0^l(k) = \{i \in I^k : \frac{C}{2} \geq c_i \geq l\} = \{i \in I_1 \cup I_2 : \frac{C}{2} \geq c_i \geq l\}$. Or I_1 ne contient aucun article de taille inférieure à $\frac{C}{2}$

et I_2 ne contient pas d'articles dont la taille est comprise entre k et l. L'ensemble considéré est donc le suivant : $J_0^l(k) = \{i \in I : \frac{C}{2} \geq c_i \geq k\} = J^k$. C'est le même ensemble que celui utilisé dans la fonction f_1^k. C'est sur cette propriété que repose la démonstration.

1. Si $x < k$, $f_1^l \circ f_0^k(x) = f_1^l(0) = 0$ et $f_1^k(x) = 0$.
2. Si $\frac{1}{2}C \geq x \geq k$, la valeur de $f_1^l(x)$ ne dépend pas de la valeur dépendante de la donnée. Comme $f_0^k(x) = x$, on a $f_1^l \circ f_0^k(x) = f_1^l(x) = 1$ et $f_1^k(x) = 1$.
3. Si $C - k \geq x > \frac{1}{2}C$, on a $f_0^k(x) = x$. On obtient $f_1^l \circ f_0^k(x) = M_C(C, J_0^l(k)) - M_C(C - f_0^k(x), J_0^l(k))$. On s'appuie maintenant sur le fait que l'ensemble dépendant de la donnée obtenu est égal à J^k. On obtient ainsi la relation suivante : $f_1^l \circ f_0^k(x) = M_C(C, J^k) - M_C(C - x, J^k) = f_1^k(x)$.
4. Si $x > C - k$, comme aucun article de taille inférieure à k n'est considéré, on a $M_C(C, J^k) - M_C(C - x, J^k) = M_C(C, J^k)$. De plus, on a $f_0^k(x) = C$. On obtient donc l'expression suivante : $f_1^l \circ f_0^k(x) = M_C(C, J^k) - M_C(C - C, J^k) = f_1^k(x)$.

□

Le corollaire suivant est immédiatement déduit de la Proposition 3.3. Il nous permet de n'utiliser que f_1^1 comme représentant de la famille f_1^k si la fonction est appliquée après f_0^k.

Corollaire 3.1. *Pour $1 \leq k \leq \frac{C}{2}$, la fonction f_1^k est égale à la fonction $f_1^1 \circ f_0^k$.*

Proposition 3.4. *Si $1 \leq k < l \leq \frac{C}{2}$, $f_1^l \circ f_0^k = f_1^l$*

Démonstration. D'après le Corollaire 3.1 et en utilisant la Proposition 3.2, on obtient $f_1^l \circ f_0^k = (f_1^1 \circ f_0^l) \circ f_0^k = f_1^1 \circ (f_0^l \circ f_0^k) = f_1^1 \circ f_0^l$. En utilisant le Corollaire 3.1, nous obtenons $f_1^l \circ f_0^k = f_1^1 \circ f_0^l = f_1^l$.

□

Nous reproduisons maintenant le résultat de la composition de f_2^l et f_0^k pour $1 \leq k \leq \frac{C}{2}$ et $1 \leq l \leq \frac{C}{2}$, puis nous montrons que pour certaines valeurs d'indices, la fonction obtenue est égale à f_2^l.

$$f_2^l \circ f_0^k(x) = \begin{cases} f_2^l(C) & \text{si } x > C - k \\ f_2^l(x) & \text{si } C - k \geq x \geq k \\ 0 & \text{sinon} \end{cases} \tag{3.2}$$

$$f_2^l \circ f_0^k(x) = \begin{cases} 2\lfloor \frac{C}{l} \rfloor & \text{si } x > C - k \\ 2(\lfloor \frac{C}{l} \rfloor - \lfloor \frac{C-x}{l} \rfloor) & \text{si } C - k \geq x > \frac{1}{2}C \\ \lfloor \frac{C}{l} \rfloor & \text{si } x = \frac{C}{2} \\ 2\lfloor \frac{x}{l} \rfloor & \text{si } \frac{1}{2}C > x \geq k \\ 0 & \text{sinon} \end{cases} \tag{3.3}$$

Nous montrons maintenant que pour certaines valeurs de k et l, il n'est pas intéressant de composer f_2^l et f_0^k, car f_2^l réduit déjà les petits articles à la valeur 0.

Proposition 3.5. *Si $1 \leq k \leq l \leq \frac{C}{2}$, $f_2^l \circ f_0^k = f_2^l$.*

Démonstration. Trois cas sont possibles.

1. Si $x < k$, on a $x < l$ et donc $\lfloor \frac{x}{l} \rfloor = 0$. On a donc $f_2^l \circ f_0^k(x) = f_2^l(0) = 0$ et $f_2^l(x) = 2\lfloor \frac{x}{l} \rfloor = 0$.
2. Si $C - k \geq x \geq k$, $f_2^l \circ f_0^k(x) = f_2^l(x)$ (voir Equation 3.3).
3. Si $x > C - k$, nous avons $C - x < k \leq l$ et donc $\lfloor \frac{C-x}{l} \rfloor = 0$. On a ainsi $f_2^l \circ f_0^k(x) = 2\lfloor \frac{C}{l} \rfloor$ et $f_2^l(x) = 2\lfloor \frac{C}{l} \rfloor - 2\lfloor \frac{C-x}{l} \rfloor = 2\lfloor \frac{C}{l} \rfloor$.

□

Si $k > l$, la fonction obtenue n'est dominée par aucune des trois fonctions f_0^m, f_1^m ou f_2^m. $f_2^l \circ f_0^k$ peut alors conduire à de meilleures bornes que f_0^m et f_2^q. Les études expérimentales montrent que c'est le cas.

3.2.2 Appliquer une fonction après f_1^k

Nous montrons maintenant qu'il n'est jamais intéressant d'appliquer une fonction f_0^l, f_1^m ou f_2^q à une instance modifiée par f_1^k. Les fonctions f_0^1 et f_2^1 sont égales à la fonction identité. Donc $f_0^1 \circ f_1^k = f_2^1 \circ f_1^k = f_1^k$. Pour d'autres valeurs de paramètres, nous montrons que les fonctions obtenues sont dominées par $f_0^{\frac{C}{2}}$, car f_1^k réduit tous les articles de taille inférieure à $\frac{1}{2}C$ à la valeur 1 (qui est réduite à 0 par la fonction suivante), ou à la taille 0. Pour simplifier les preuves, nous introduisons le lemme suivant.

Lemme 3.1. *Soit g une (D)DFF définie de $[0, C]$ dans $[0, C']$. Si $g(x) = 0$ pour $x \leq \frac{1}{2}C$, g est dominée par $f_0^{\frac{1}{2}C}$.*

Démonstration. Nous montrons que pour toute valeur de x, $\frac{g(x)}{g(C)}$ est inférieur à $\frac{f_0^{\frac{1}{2}C}(x)}{f_0^{\frac{1}{2}C}(C)}$.

1. Si $x < \frac{C}{2}$, $g(x) = 0$ et donc $\frac{g(x)}{g(C)} = 0 = \frac{f_0^{\frac{1}{2}C}(x)}{f_0^{\frac{1}{2}C}(C)}$.
2. Si $x = \frac{C}{2}$, $\frac{g(x)}{g(C)} = 0 < \frac{1}{2} = \frac{f_0^{\frac{1}{2}C}(x)}{f_0^{\frac{1}{2}C}(C)}$.
3. Si $x > \frac{C}{2}$, on a $\frac{g(x)}{g(C)} \leq 1 = \frac{f_0^{\frac{1}{2}C}(x)}{f_0^{\frac{1}{2}C}(C)}$.

□

Proposition 3.6. *Pour $1 < l \leq \frac{C}{2}$ et $1 \leq k \leq \frac{f_1^k(C)}{2}$, $f_0^l \circ f_1^k \leq f_0^{\frac{1}{2}C}$.*

Démonstration. Notons que $f_0^l \circ f_1^k$ est bien définie. Rappelons que pour tout l supérieur à un, $f_0^l(1) = 0$. Si $x \leq \frac{C}{2}$, $f_1^k(x) \leq 1$ et donc $f_0^l \circ f_1^k(x) = 0$. Le résultat est donc immédiatement montré grâce au Lemme 3.1.

□

Lorsque la valeur $M_C(C, J^k)$ est égale à un, il n'y a donc qu'un article de taille c^* comprise entre $\frac{C}{2}$ et la valeur k. Dans ce cas, la fonction $f_0^{c^*}$, ou la fonction $f_0^{\frac{C}{2}}$ permet de trouver la solution optimale du problème de bin-packing associé. Nous ne considérons donc dans la suite que le cas où $M_C(C, J^k)$ est supérieur ou égal à deux.

Proposition 3.7. *Pour* $1 \leq l \leq \frac{f_1^k(C)}{2}$ *et* $1 \leq k \leq \frac{C}{2}$, *si* $M_C(C, J^k) > 1$ *alors* $f_1^l \circ f_1^k \leq f_0^{\frac{C}{2}}$.

Démonstration. Si $x \leq \frac{1}{2}C$, $f_1^k(x) \leq 1 \leq \frac{M_C(C,J^k)}{2}$ et donc $f_1^l \circ f_1^k(x) = 0$. En utilisant le Lemme 3.1, nous montrons que $f_1^l \circ f_1^k < f_0^{\frac{1}{2}C}$.

□

Proposition 3.8. *Pour* $1 < l \leq \frac{C}{2}$, $1 \leq k \leq \frac{C}{2}$, $f_2^l \circ f_1^k \leq f_0^{\frac{1}{2}C}$.

Démonstration. Si $x \leq \frac{C}{2}$, $f_1^k(x) \leq 1$ et donc $f_2^l(f_1^k(x)) \leq 2\lfloor \frac{1}{l} \rfloor = 0$. En utilisant le Lemme 3.1, nous montrons le résultat souhaité. □

Il n'est jamais intéressant d'appliquer une fonction des familles f_0^k, f_1^l et f_2^m après f_1^q. Donc, f_1^q devrait apparaître une fois au plus dans toute liste de compositions itératives de fonctions.

3.2.3 Appliquer une fonction après f_2^k

Les trois résultats suivants montrent que seule la fonction f_1^l est intéressante à appliquer après f_2^k. En effet, $f_0^l \circ f_2^k$ est égale à une fonction $f_2^k \circ f_0^{\frac{kl}{2}}$ et une application successive de f_2^k et f_2^l revient à l'application d'une unique fonction $f_2^{\frac{kl}{2}}$. Lorsqu'une fonction de type f_0^k, f_1^k ou f_2^k est employée, on n'utilise pour le paramètre k que les valeurs qui correspondent à la taille d'un article. Lorsque la fonction f_2^k est employée, tous les articles ont une longueur paire dans l'instance tranformée, sauf ceux qui ont pour taille $\frac{C}{2}$. Dans la suite, on considère donc que le paramètre employé dans une fonction après application de f_2^k est pair.

Proposition 3.9. *Pour* $1 \leq k \leq \frac{C}{2}$, $2 \leq l \leq \lfloor \frac{C}{k} \rfloor$, $f_0^l \circ f_2^k = f_2^k \circ f_0^{\frac{kl}{2}}$.

Démonstration. Vérifions tout d'abord que la fonction $f_0^{\frac{kl}{2}}$ est valide. Pour cela, il faut que $1 \leq \frac{kl}{2} \leq \frac{C}{2}$. On a par hypothèse $1 \leq k$ et $2 \leq l$, on a donc $1 \leq \frac{kl}{2}$. De même, on a $l \leq \lfloor \frac{C}{k} \rfloor$, ce qui nous donne $l \leq \frac{C}{k}$ et donc $\frac{kl}{2} \leq \frac{C}{2}$.

Vérifions maintenant que la fonction f_0^k peut bien être appliquée à l'instance obtenue après application de $f_0^{\frac{kl}{2}}$. Pour cela, il reste à montrer que $k \leq \lfloor \frac{2C}{kl} \rfloor$. On a $l \leq 2C$ et donc $1 \leq \frac{2C}{l}$ et $k \leq \frac{2C}{kl}$. Comme k est entier, on obtient le résultat souhaité. Calculons maintenant $f_0^l \circ f_2^k(x)$.

$$f_0^l \circ f_2^k(x) = \begin{cases} f_0^l(2(\lfloor \frac{C}{k} \rfloor - \lfloor \frac{C-x}{k} \rfloor)) & \text{si } x > \frac{C}{2} \\ f_0^l(\lfloor \frac{C}{k} \rfloor) & \text{si } x = \frac{C}{2} \\ f_0^l(2\lfloor \frac{x}{k} \rfloor) & \text{si } x < \frac{C}{2} \end{cases} \tag{3.4}$$

Pour servir la démonstration, nous décomposons les valeurs possibles en fonction du paramètre $\frac{kl}{2}$.

$$f_0^l \circ f_2^k(x) = \begin{cases} f_0^l(2(\lfloor \frac{C}{k} \rfloor - \lfloor \frac{C-x}{k} \rfloor)) & \text{si } x > C - \frac{kl}{2} \\ f_0^l(2(\lfloor \frac{C}{k} \rfloor - \lfloor \frac{C-x}{k} \rfloor)) & \text{si } \frac{C}{2} < x \leq C - \frac{kl}{2} \\ f_0^l(\lfloor \frac{C}{k} \rfloor) & \text{si } x = \frac{C}{2} \\ f_0^l(2\lfloor \frac{x}{k} \rfloor) & \text{si } \frac{kl}{2} \leq x < \frac{C}{2} \\ f_0^l(2\lfloor \frac{x}{k} \rfloor) & \text{si } 1 \leq x < \frac{kl}{2} \end{cases} \tag{3.5}$$

Nous calculons maintenant la valeur obtenue pour chacun des cinq cas énumérés dans l'Equation (3.5). Nous rappelons que les valeurs considérées sont toutes entières, ce qui nous permet d'effectuer des simplifications sur les parties entières.

1. $x > C - \frac{kl}{2}$, on a $C - x < \frac{kl}{2}$ et donc $2\lfloor \frac{C-x}{k} \rfloor < l$. On obtient donc $2(\lfloor \frac{C}{k} \rfloor - \lfloor \frac{C-x}{k} \rfloor) > 2\lfloor \frac{C}{k} \rfloor - l$. On en déduit que $f_0^l \circ f_2^k(x) = 2\lfloor \frac{C}{k} \rfloor$.
2. $\frac{C}{2} < x \leq C - \frac{kl}{2}$. On a $C - x \leq \frac{kl}{2}$ et donc $2\lfloor \frac{C-x}{k} \rfloor \geq l$. Ainsi, $2(\lfloor \frac{C}{k} \rfloor - \lfloor \frac{C-x}{k} \rfloor) \leq 2\lfloor \frac{C}{k} \rfloor - l$. On en déduit que $f_0^l \circ f_2^k(x) = 2\lfloor \frac{C}{k} \rfloor - 2\lfloor \frac{C-x}{k} \rfloor$.
3. $x = \frac{C}{2}$, on obtient directement $f_0^l \circ f_2^k(\frac{C}{2}) = \lfloor \frac{C}{k} \rfloor$.
4. $\frac{kl}{2} \leq x < \frac{C}{2}$, on a $x \geq \frac{kl}{2}$ et donc $2\lfloor \frac{x}{k} \rfloor \geq l$.
5. $1 \leq x < \frac{kl}{2}$, on a $x < \frac{kl}{2}$. Ainsi $2\lfloor \frac{x}{k} \rfloor < l$.

On obtient donc la formulation suivante.

$$f_0^l \circ f_2^k(x) = \begin{cases} 2\lfloor \frac{C}{k} \rfloor & \text{si } x > C - \frac{kl}{2} \\ 2(\lfloor \frac{C}{k} \rfloor - \lfloor \frac{C-x}{k} \rfloor) & \text{si } C - \frac{kl}{2} \geq x > \frac{C}{2} \\ \lfloor \frac{C}{k} \rfloor & \text{si } x = \frac{C}{2} \\ 2\lfloor \frac{x}{k} \rfloor & \text{si } \frac{C}{2} \geq x \geq \frac{kl}{2} \\ 0 & \text{sinon} \end{cases} \tag{3.6}$$

On peut calculer $f_2^k \circ f_0^{\frac{kl}{2}}(x)$ en remplaçant dans l'Equation (3.3) le terme k par $\frac{kl}{2}$ et le terme l par k.

$$f_2^k \circ f_0^{\frac{kl}{2}}(x) = \begin{cases} 2\lfloor \frac{C}{k} \rfloor & \text{si } x > C - \frac{kl}{2} \\ 2(\lfloor \frac{C}{k} \rfloor - \lfloor \frac{C-x}{k} \rfloor) & \text{si } C - \frac{kl}{2} \geq x > \frac{C}{2} \\ \lfloor \frac{C}{k} \rfloor & \text{si } x = \frac{C}{2} \\ 2\lfloor \frac{x}{k} \rfloor & \text{si } \frac{C}{2} \geq x \geq \frac{kl}{2} \\ 0 & \text{sinon} \end{cases} \tag{3.7}$$

Les deux fonctions sont donc égales.

□

Nous étudions maintenant la fonction obtenue en composant f_1^l et f_2^k pour $1 \leq k \leq \frac{C}{2}$ et $1 \leq l \leq \frac{C}{2}$.

$$f_1^l \circ f_2^k(x) = \begin{cases} f_1^l(2(\lfloor \frac{C}{k} \rfloor - \lfloor \frac{C-x}{k} \rfloor)) & \text{si } x > \frac{C}{2} \\ f_1^l(\lfloor \frac{C}{k} \rfloor) & \text{si } x = \frac{C}{2} \\ f_1^l(2\lfloor \frac{x}{k} \rfloor) & \text{si } x < \frac{C}{2} \end{cases} \tag{3.8}$$

Nous utilisons des notations similaires à celles de la Proposition 3.3 pour le nouvel ensemble dépendant de la donnée. Soit $J_2^k(l)$ l'ensemble à considérer pour la fonction f_1^l après application de f_2^k. $J_2^k(l) = \{i \in I : 2\lfloor \frac{c_i}{k} \rfloor \geq l\} = \{i \in I : c_i \geq \frac{lk}{2}\}$.

$$f_1^l \circ f_2^k(x) = \begin{cases} M_C(2\lfloor \frac{C}{k} \rfloor, J_2^k(l)) - M_C(2(\lfloor \frac{C}{k} \rfloor - \lfloor \frac{C-x}{k} \rfloor), J_2^k(l)) & \text{si } x > \frac{C}{2} \\ 1 & \text{si } \frac{lk}{2} \leq x \leq \frac{C}{2} \\ 0 & \text{sinon} \end{cases} \tag{3.9}$$

$f_1^l \circ f_2^k$ peut conduire à de meilleures bornes que f_1^m et f_2^q. Les résultats expérimentaux montrent que c'est le cas.

Proposition 3.10. *Pour* $1 \leq k \leq \frac{C}{2}$ *et* $1 \leq l \leq \lfloor \frac{C}{k} \rfloor$, $f_2^l \circ f_2^k = f_2^{\frac{kl}{2}}$.

Démonstration. Nous rappelons ici que l est une valeur paire.

$$f_2^l \circ f_2^k(x) = \begin{cases} f_2^l(2(\lfloor \frac{C}{k} \rfloor - \lfloor \frac{C-x}{k} \rfloor)) & \text{si } x > \frac{C}{2} \\ f_2^l(\lfloor \frac{C}{k} \rfloor) & \text{si } x = \frac{C}{2} \\ f_2^l(2\lfloor \frac{x}{k} \rfloor) & \text{si } x < \frac{C}{2} \end{cases} \tag{3.10}$$

$$f_2^l \circ f_2^k(x) = \begin{cases} 2(\lfloor \frac{2\lfloor \frac{C}{k} \rfloor}{l} \rfloor - \lfloor \frac{2\lfloor \frac{C}{k} \rfloor - 2(\lfloor \frac{C}{k} \rfloor - \lfloor \frac{C-x}{k} \rfloor)}{l} \rfloor) & \text{si } x > \frac{C}{2} \\ \lfloor \frac{2\lfloor \frac{C}{k} \rfloor}{l} \rfloor & \text{si } x = \frac{C}{2} \\ 2\lfloor \frac{2\lfloor \frac{x}{k} \rfloor}{l} \rfloor & \text{si } x < \frac{C}{2} \end{cases} \tag{3.11}$$

$$f_2^l \circ f_2^k(x) = \begin{cases} 2(\left\lfloor \frac{2\lfloor \frac{C}{k} \rfloor}{l} \right\rfloor - \left\lfloor \frac{2\lfloor \frac{C-x}{k} \rfloor}{l} \right\rfloor) & \text{si } x > \frac{C}{2} \\ \lfloor \frac{2C}{kl} \rfloor & \text{si } x = \frac{C}{2} \\ 2\lfloor \frac{2x}{kl} \rfloor & \text{si } x < \frac{C}{2} \end{cases} \tag{3.12}$$

$$f_2^l \circ f_2^k(x) = \begin{cases} 2(\left\lfloor \frac{C}{\frac{kl}{2}} \right\rfloor - \left\lfloor \frac{C-x}{\frac{kl}{2}} \right\rfloor) & \text{si } x > \frac{C}{2} \\ \left\lfloor \frac{C}{\frac{kl}{2}} \right\rfloor & \text{si } x = \frac{C}{2} \\ 2\left\lfloor \frac{x}{\frac{kl}{2}} \right\rfloor & \text{si } x < \frac{C}{2} \end{cases} \tag{3.13}$$

$$f_2^l \circ f_2^k(x) = f_2^{\frac{kl}{2}} \tag{3.14}$$

□

3.2.4 Un ensemble dominant de compositions

Toute composition itérative des fonctions f_0^k, f_1^l et f_2^m est de la forme suivante.

$$f_{r_1}^{m_1} \circ f_{r_2}^{m_2} \circ \ldots \circ f_{r_p}^{m_p} \text{ pour } p \in \mathbb{N}^*, r_k \in \{0,1,2\}, \text{ et } k = 1, \ldots p \tag{3.15}$$

Copnsidérons d'abord le fait qu'il n'est jamais intéressant d'appliquer une fonction f_0^k, f_1^k ou f_2^k sur une instance transformée par f_1^l.

Lemme 3.2. *Toute composition itérative des fonctions f_0^k, f_1^l, f_2^m est dominée par une fonction d'une des deux formes suivantes.*

$$f_{r_1}^{m_1} \circ f_{r_2}^{m_2} \circ \ldots \circ f_{r_p}^{m_p} \text{ pour } p \in \mathbb{N}^*, r_k \in \{0,2\} \text{ pour } k = 1, \ldots p \tag{3.16}$$

$$f_1^q \circ f_{r_1}^{m_1} \circ f_{r_2}^{m_2} \circ \ldots \circ f_{r_p}^{m_p} \text{ pour } p \in \mathbb{N}^*, r_k \in \{0,2\} \text{ pour } k = 1, \ldots p \tag{3.17}$$

Démonstration. D'après les Propositions 3.6, 3.7 et 3.8, on déduit que l'expression de l'Equation (3.18) est dominée par celle de l'Equation (3.19).

$$f_{r_1}^{m_1} \circ f_{r_2}^{m_2} \circ \ldots \circ f_{r_{q-2}}^{m_{q-2}} \circ f_{r_{q-1}}^{m_{q-1}} \circ f_1^k \circ f_{r_{q+1}}^{m_{q+1}} \circ f_{r_p}^{m_p} \tag{3.18}$$

$$f_{r_1}^{m_1} \circ f_{r_2}^{m_2} \circ \ldots \circ f_{r_{q-2}}^{m_{q-2}} \circ f_0^{\frac{C}{2}} \tag{3.19}$$

Il n'est donc pas intéressant d'appliquer une fonction après f_1^k dans toute suite de compositions, car on obtient une suite de fonctions égale à une fonction de type $f_0^{\frac{C}{2}}$. On en déduit que f_1^k est soit la dernière fonction utilisée, soit aucune fonction de ce type n'est utilisée dans la composition. Le Corollaire 3.1 nous permet de n'utiliser que la fonction f_1^1. □

Nous montrons maintenant que toute application itérative de $f_2^{k_m}$ et de $f_0^{l_m}$ est égale à une application de $f_2^k \circ f_0^l$.

Lemme 3.3. *Pour toute application itérative de $f_2^{k_m}$ et de $f_0^{l_q}$, il existe deux entiers k et l tels que la fonction $f_2^k \circ f_0^l$ conduit à un résultat équivalent.*

Démonstration. Toute suite de compositions de fonctions $f_0^{l_m}$ et $f_2^{k_m}$ peut être simplifiée en utilisant récursivement les opérations suivantes.

1. Les termes $f_0^{l_m} \circ f_0^{l_{m+1}}$ peuvent être simplifiés en $f_0^{max\{l_m, l_{m+1}\}}$ (Proposition 3.2).
2. Les termes $f_2^{k_m} \circ f_2^{k_{m+1}}$ sont simplifiés en $f_2^{\frac{k_m k_{m+1}}{2}}$ (Proposition 3.10).
3. Les termes $f_0^{l_m} \circ f_2^{k_m}$ peuvent être transformés en $f_2^{k_m} \circ f_0^{\frac{k_m l_m}{2}}$ (Proposition 3.9).

On peut donc regrouper tous les termes $f_2^{k_q}$ et tous les termes $f_0^{l_q}$ en effectuant des permutations, puis on peut simplifier les deux parties en regroupant toutes les fonctions $f_2^{k_q}$ et toutes les fonctions $f_0^{l_q}$ en une seule fonction. On obtient ainsi une fonction de la forme $f_2^k \circ f_0^l$, qui est strictement égale à la fonction initiale.

□

Donc, toute composition des trois (D)DFF proposées est dominée par une fonction de la forme donnée dans l'Equation (3.20).

$$(f_1^1 \circ f_2^l \circ f_0^k) \text{ ou } (f_2^l \circ f_0^k) \text{ pour } k, l = 1, \ldots, \frac{C}{2} \tag{3.20}$$

A partir des Lemmes 3.3 et 3.2, on déduit directement le Théorème 3.1.

Théorème 3.1. *Toute composition des trois (D)DFF f_0^m, f_1^q et f_2^r est dominée par l'expression de l'Equation (3.20). La cardinalité de l'ensemble dominant de fonctions est bornée par $min(2n^2, \frac{1}{2}C^2)$.*

3.2.5 Réduction de l'ensemble dominant de fonctions

Nous montrons maintenant que de nombreuses valeurs du paramètre k ne sont pas utiles pour la fonction f_2^k si elle est composée avec la fonction f_0^l. En effet, à partir de la valeur $\frac{C}{4}$, les fonctions obtenues sont égales à une application de f_0^l près.

Proposition 3.11. *Soit k et l deux entiers, tels que $\frac{C}{4} < k < l < \frac{C}{2}$ et $\lfloor \frac{C}{k} \rfloor = \lfloor \frac{C}{l} \rfloor$, $f_2^l = f_2^k \circ f_0^l$.*

Démonstration. Nous montrons que pour toutes les valeurs de x, les deux fonctions sont égales.

1. Si $x < l$, $f_2^l(x) = 2\lfloor \frac{x}{l} \rfloor = 0 = f_2^k(0) = f_2^k \circ f_0^l(x)$.
2. Si $l \leq x < \frac{C}{2}$, $f_2^l(x) = 2\lfloor \frac{x}{l} \rfloor$ et $f_2^k \circ f_0^l(x) = f_2^k(x) = 2\lfloor \frac{x}{k} \rfloor$. Il nous reste à montrer que $\lfloor \frac{x}{l} \rfloor = \lfloor \frac{x}{k} \rfloor$. Nous montrons que ces deux valeurs sont égales à 1. Par hypothèse, $\frac{C}{4} < k < l \leq x$ nous avons donc $\frac{4}{C} > \frac{1}{k} > \frac{1}{l} \geq \frac{1}{x}$, et $\frac{4x}{C} > \frac{x}{k} > \frac{x}{l} \geq 1$.

   $$\lfloor \frac{4x}{C} \rfloor \geq \lfloor \frac{x}{k} \rfloor \geq \lfloor \frac{x}{l} \rfloor \geq 1 \tag{3.21}$$

   Comme $\frac{C}{2} > x$, $4x < 2C$, et $2 > \frac{4x}{C}$ et donc

   $$1 \geq \lfloor \frac{4x}{C} \rfloor \tag{3.22}$$

   Le résultat est directement déduit des Equations (3.21) et (3.22).
3. Si $x = \frac{C}{2}$, $f_2^l(\frac{C}{2}) = \lfloor \frac{C}{l} \rfloor = \lfloor \frac{C}{k} \rfloor = f_2^k(\frac{C}{2}) = f_2^k \circ f_0^l(\frac{C}{2})$ car $f_0^l(\frac{C}{2}) = \frac{C}{2}$.
4. Si $\frac{1}{2}C < x \leq C - l$, $f_2^l(x) = 2(\lfloor \frac{C}{l} \rfloor - \lfloor \frac{C-x}{l} \rfloor)$.
   Par hyptohèse, nous avons $l \leq C - x < \frac{1}{2}C$, et nous avons montré dans un cas précédent que pour une telle valeur, $\lfloor \frac{C-x}{l} \rfloor = \lfloor \frac{C-x}{k} \rfloor$. Nous avons donc $f_2^l(x) = 2(\lfloor \frac{C}{l} \rfloor - \lfloor \frac{C-x}{l} \rfloor) = 2(\lfloor \frac{C}{k} \rfloor - \lfloor \frac{C-x}{k} \rfloor)$ et $f_2^k \circ f_0^l(x) = f_2^k(x) = 2(\lfloor \frac{C}{k} \rfloor - \lfloor \frac{C-x}{k} \rfloor)$.
5. Si $C - l < x$, $\lfloor \frac{C-x}{l} \rfloor = 0$ et donc $f_2^l(x) = 2\lfloor \frac{C}{l} \rfloor = 2\lfloor \frac{C}{k} \rfloor$ et $f_2^k \circ f_0^l(x) = f_2^k(C) = 2\lfloor \frac{C}{k} \rfloor$.

□

Cette proposition a un grand intérêt car elle divise par deux le nombre de valeurs de k à tester si nous nous assurons que f_0^l est appliquée avant f_2^k. Pour $k = 1$ ou $k = \frac{1}{2}C$, il n'est pas intéressant d'utiliser f_2^k, car elle est dans ce cas égale à f_0^k. De plus, en utilisant la Proposition 3.5, on ne teste que les combinaisons $f_2^l \circ f_0^k$ pour lesquelles $k > l$. Nous avons donc dans l'Equation (3.20)

1. $k \in [1, \ldots \frac{1}{2}C]$
2. $l \in ([1, k-1] - [\lfloor \frac{C}{4} \rfloor + 1, \lfloor \frac{C}{2} \rfloor]) \cup \{\lfloor \frac{C}{3} \rfloor + 1\}$

3.3 DFF Maximales (MDFF) et générateur de MDFF

De nombreuses DFF ne sont pas utiles pour obtenir des bornes inférieures. Pour cette raison, Carlier et Néron proposent le concept de DFF Maximales (MDFF) [13]. Nous en donnons maintenant une définition légèrement remaniée.

Définition 3.2. *Une MDFF f est une DFF qui n'est strictement dominée par aucune DFF, i.e. il n'existe pas de DFF f' avec $f' > f$.*

Les MDFF forment un ensemble dominant de DFF. Plusieurs propriétés caractérisent ces fonctions [13].

Proposition 3.12. *[13] Une DFF f est une MDFF si et seulement si elle respecte les quatre conditions suivantes.*

1. *f est croissante*
2. $\forall x \in [0, X], f(x) \geq max\{f(y) + f(z) : y + z = x, y \in \mathbb{N}, z \in \mathbb{N}\}$
3. $\forall x, y \in [0, X], [x + y = X] \implies [f(x) + f(y) = f(X)]$
4. $f(0) = 0$

Dans la suite de cette section, x, y et z sont des entiers positifs appartenant à l'intervalle $[0, C]$.

3.3.1 f_0^k et f_2^k sont des MDFF

Une manière d'améliorer les fonctions f_0^k et f_2^k est de trouver une MDFF qui les domine. Cette méthode ne peut conduire à de meilleurs résultats car f_0^k et f_2^k sont déjà maximales.

Proposition 3.13. *Pour $k = 1, \ldots, \frac{C}{2}$, la fonction f_0^k est une MDFF.*

Démonstration. Nous montrons que f_0^k vérifie les quatre propriétés de la Proposition 3.12.

1. f_0^k est croissante : immédiat.
2. $f_0^k(x) \geq max\{f_0^k(y) + f_0^k(z)/y + z = x, y \in \mathbb{N}, z \in \mathbb{N}\}$.
   (a) Si $x < k$, le résultat est immédiat.
   (b) Si $k \leq x \leq C - k$, $f_0^k(x) = x$, $f_0^k(y) \leq y$ et $f_0^k(z) \leq z$.
   (c) Si $x > C - k$, $f_0^k(x) = C$. Comme f_0^k est une DFF, si $y + z \leq C$, $f_0^k(y) + f_0^k(z) \leq f_0^k(C) = f_0^k(x)$.
3. $[x + y = C] \implies [f_0^k(x) + f_0^k(y) = f_0^k(C)]$. Si $x = y$, le résultat est immédiat. Nous supposons sans perte de généralité que $x > y$.
   (a) Si $x > C - k$, $y < k$: $f_0^k(x) + f_0^k(y) = C + 0 = C = f_0^k(C)$.
   (b) Si $x \leq C - k$, $y \geq k$: $f_0^k(x) + f_0^k(y) = x + y = C = f_0^k(C)$.
4. $f_0^k(0) = 0$ pour toutes les valeurs de k.

□

Proposition 3.14. *Pour $k = 1, \ldots, \frac{C}{2}$, la fonction f_2^k est une MDFF.*

Démonstration. La preuve suit une démarche similaire à la proposition précédente.

1. Pour montrer que f_2^k est croissante, il suffit de montrer que $x < \frac{C}{2}$ implique que $f_2^k(x) < \lfloor \frac{C}{2} \rfloor$ et que $x > \frac{C}{2}$ implique que $f_2^k(x) > \lfloor \frac{C}{2} \rfloor$.
   Si $x < \frac{C}{2}$, $\frac{x}{k} \leq \frac{C}{2k}$ et $2\lfloor \frac{x}{k} \rfloor \leq \frac{C}{k}$, on obtient directement le résultat souhaité.
   Si $x > \frac{C}{2}$, $\frac{C-x}{k} < \frac{C}{2k}$, on a $2\lfloor \frac{C}{k} \rfloor < \lfloor \frac{C-x}{k} \rfloor$. On en déduit que $2\lfloor \frac{C-x}{k} \rfloor - 2\lfloor \frac{C}{k} \rfloor > \lfloor \frac{C-x}{k} \rfloor$.
2. $f_2^k(x) \geq max\{f_2^k(y) + f_2^k(z)/y + z = x, y \in \mathbb{N}, z \in \mathbb{N}\}$.
   Plusieurs cas peuvent apparaître, en fonction de la valeur x.
   (a) Si $x < \frac{C}{2}$, nous montrons que pour tous les entiers y, z tels que $x = y + z$, on a $2\lfloor \frac{x}{k} \rfloor \geq 2\lfloor \frac{y}{k} \rfloor + 2\lfloor \frac{z}{k} \rfloor$. Comme $x = y + z$, $\lfloor \frac{x}{k} \rfloor = \lfloor \frac{y}{k} + \frac{z}{k} \rfloor \geq \lfloor \frac{y}{k} \rfloor + \lfloor \frac{z}{k} \rfloor$.
   (b) Si $x = \frac{C}{2}$, la propriété est directement vérifiée.
   (c) Si $x > \frac{C}{2}$, nous supposons sans perte de généralité que $y > z$.
      i. Si $y > \frac{C}{2}$, nous montrons que $\lfloor \frac{C}{k} \rfloor - \lfloor \frac{C-x}{k} \rfloor \geq \lfloor \frac{C}{k} \rfloor - \lfloor \frac{C-y}{k} \rfloor + \lfloor \frac{z}{k} \rfloor$.

$$\lfloor \frac{C-y-z}{k} \rfloor \leq \lfloor \frac{C-y}{k} \rfloor - \lfloor \frac{z}{k} \rfloor \tag{3.23}$$

      Par hypothèse, $x = y + z$, nous avons donc

$$\lfloor \frac{C-x}{k} \rfloor \leq \lfloor \frac{C-y}{k} \rfloor - \lfloor \frac{z}{k} \rfloor \tag{3.24}$$

$$\lfloor \frac{C}{k} \rfloor - \lfloor \frac{C-x}{k} \rfloor \geq \lfloor \frac{C}{k} \rfloor - \lfloor \frac{C-y}{k} \rfloor + \lfloor \frac{z}{k} \rfloor \tag{3.25}$$

      ii. Si $y < \frac{C}{2}$, nous montrons que $\lfloor \frac{C}{k} \rfloor - \lfloor \frac{C-x}{k} \rfloor \geq \lfloor \frac{y}{k} \rfloor + \lfloor \frac{z}{k} \rfloor$.

$$\lfloor \frac{C}{k} \rfloor - \lfloor \frac{C-y-z}{k} \rfloor \geq \lfloor \frac{y}{k} \rfloor + \lfloor \frac{z}{k} \rfloor \tag{3.26}$$

      En remplaçant $y + z$ par x dans l'Equation (3.26), nous obtenons le résultat désiré.
3. $[x + y = C] \implies [f_2^k(x) + f_2^k(y) = f_2^k(C)]$.
   Si $x = y$, le résultat est immédiat. Nous supposons sans perte de généralité que $x > y$.

$$f_2^k(x) + f_2^k(y) = 2\lfloor \frac{C}{k} \rfloor - 2\lfloor \frac{C-x}{k} \rfloor + 2\lfloor \frac{y}{k} \rfloor \tag{3.27}$$

   En remplaçant y par $C - x$, nous obtenons

$$f_2^k(x) + f_2^k(y) = 2\lfloor \frac{C}{k} \rfloor - 2\lfloor \frac{C-x}{k} \rfloor + 2\lfloor \frac{C-x}{k} \rfloor \tag{3.28}$$

$$f_2^k(x) + f_2^k(y) = 2\lfloor \frac{C}{k} \rfloor = f_2^k(C) \tag{3.29}$$

4. $f_2^k(0) = \lfloor \frac{0}{k} \rfloor = 0$ pour toutes les valeurs de k.

□

3.3.2 Une méthode de génération pour les *MDFF non dominées*

Même lorsqu'on ne considère que les DFF Maximales, le nombre de fonctions à considérer est grand. Dans le cas discret, Carlier et Néron [13] introduisent la notion de fonctions *non dominées*. Ces fonctions conduisent à des bornes inférieures qui ne sont obtenues par aucune autre MDFF.

Définition 3.3. *Une DFF f est une* MDFF non dominée *s'il existe une instance $D = (A, C)$ pour laquelle il n'existe pas de DFF g telle que*

$$\frac{\sum_{a_i \in A} g(c_i)}{g(C)} > \frac{\sum_{a_i \in A} f(c_i)}{f(C)} \tag{3.30}$$

Les auteurs [13] proposent une méthode pour générer **toutes** les MDFF non dominées pour une valeur donnée de X en énumérant toutes les fonctions de $[0, X] \rightarrow [0, X']$ satisfaisant les quatre propriétés des MDFF de la Proposition 3.12. Les MDFF non dominées sont sélectionnées par un programme linéaire et rangées dans un fichier. Le nombre de fonctions générées peut être grand, la méthode ne peut donc être appliquée qu'à de petites valeurs. Pour les grandes valeurs, un sous-ensemble de fonctions doit être calculé. Seules les petites valeurs de X' sont testées et des méthodes sont employées pour une génération efficace de l'ensemble des fonctions (voir [13] pour plus de détails).

3.4 (D)DFF et bornes inférieures pour $1BP$ et $2BP$

Pour $1BP$, nous proposons de nouvelles bornes inférieures qui utilisent les fonctions décrites dans la section précédente, et nous prouvons qu'elles dominent la borne de Labbé *et al.* [43]. Pour $2BP$, nous proposons de nouvelles bornes et montrons que l'utilisation de l'idée d'amélioration des bornes de L_2^{new} proposée par Boschetti et Mingozzi [7] ne peut améliorer le résulat que d'une unité.

3.4.1 Bornes inférieures pour $1BP$

3.4.1.1 Nouvelles bornes inférieures

Nous introduisons dans un premier temps des bornes dédiées au $1BP$. Soit ${\mathcal{F}^{\mathcal{X}}}_{DFF^a}$ l'ensemble des f_0^k et f_2^l associées à la taille X.

$$L_{DFF^a}^{1BP} = \max_{f \in {\mathcal{F}^{\mathcal{C}}}_{DFF^a}} \left\{ \left\lceil \frac{\sum_{a_i \in A} f(c_i)}{f(C)} \right\rceil \right\} \tag{3.31}$$

Nous introduisons une borne qui utilise les fonctions obtenues en composant f_0^k et f_2^l. Soit ${\mathcal{F}^{\mathcal{X}}}_{CDFF}$ l'ensemble dominant de compositions des fonctions f_0^k et f_2^l associé à la taille X. Une borne similaire L_{CDFF}^{1BP} peut être définie.

Soit ${\mathcal{F}^{\mathcal{X}}}_{CDDFF^a}$ l'ensemble dominant des compositions de fonctions f_0^k, f_1^l et f_2^m. Nous définissons aussi la borne $L_{CDDFF^a}^{1BP}$.

Les DFF générées par la méthode de Carlier et Néron [13] sont rangées dans un fichier. Ainsi, ces fonctions n'ont pas à être recalculées pour chaque nouveau problème, il suffit de lire dans le fichier correspondant. L'intérêt de cette approche est que la complexité de la borne inférieure est linéaire en fonction du nombre d'articles, même si un grand nombre de fonctions peut rendre la méthode difficile à appliquer pour certaines instances. En revanche, pour un très grand nombre d'articles, la méthode devient très efficace en terme de temps de calcul, comparée à celles dépendant de n. Dans la section consacrée aux résultats expérimentaux, le nombre de fonctions est rapporté pour quelques valeurs, et l'efficacité de la méthode est montrée numériquement. Soit ${\mathcal{F}^{\mathcal{X}}}_{DFF^e}$ l'ensemble de DFF^e associé à la taille X. Une borne $L^{1BP}_{DFF^e}$ peut être définie de manière analogue à $L^{1BP}_{DFF^a}$. On peut améliorer la borne obtenue après application d'une DFF générée par énumération en appliquant la DDFF f^1_1 à l'instance obtenue. La borne correspondante est notée $L^{1BP}_{CDDFF^e}$.

3.4.1.2 Relation de dominance

Nous montrons maintenant que nos bornes dominent $LB3$ proposée par Labbé *et al.* [43] (*cf.* Chapitre 1 pour une description de cette borne).

Proposition 3.15. *$L^{1BP}_{CDDFF^a}$ domine $LB3$ proposée par Labbé et al. [43].*

Démonstration. Pour une valeur donnée de k,

$$LB3(k) = |A_{gr}| + max\left\{|A_{moy}| + \left\lceil\frac{|A^+_{pt}|}{2}\right\rceil, \left\lceil\frac{\sum_{a_i\in A'} w_i}{C}\right\rceil\right\} \tag{3.32}$$

Si $|A_{moy}| + \lceil\frac{|A^+_{pt}|}{2}\rceil \leq \left\lceil\frac{\sum_{a_i\in A'} w_i}{C}\right\rceil$, $LB3(k)$ est égale à la borne continue après application de la fonction f^k_0.

Si $|A_{moy}| + \lceil\frac{|A^+_{pt}|}{2}\rceil > \left\lceil\frac{\sum_{a_i\in A'} w_i}{C}\right\rceil$, $LB3(k) = |A_{gr} \cup A_{moy}| + \left\lceil\frac{|A^+_{pt}|}{2}\right\rceil$.

Il existe un paramètre p tel que $c_i \geq p$ implique $a_i \in A^+_{pt}$ et $c_i < p$ implique $a_i \notin A^+_{pt}$. Calculons maintenant la borne continue L_2 obtenue après application de la fonction f^p_2 à l'instance.

$$L_2 = \left\lceil\frac{\sum_{a_i\in A} f^p_2(c_i)}{f^p_2(C)}\right\rceil \tag{3.33}$$

Comme $p > \frac{C}{3}$, les articles de A_{pt} qui ne sont pas dans A^+_{pt} et les articles de taille inférieure à $\frac{C}{3}$ ne sont pas considérés.

$$L_2 = \left\lceil\frac{\sum_{a_i\in A_{gr}\cup A_{moy}} f^p_2(c_i) + \sum_{a_i\in A^+_{pt}} f^p_2(c_i)}{f^p_2(C)}\right\rceil \tag{3.34}$$

$$L_2 = \left\lceil\frac{\sum_{a_i\in A_{gr}\cup A_{moy}} M_C(C, A^+_{pt}) - M_C(C - c_i, A^+_{pt}) + \sum_{a_i\in A^+_{pt}} 1}{M_C(C, A^+_{pt})}\right\rceil \tag{3.35}$$

Comme aucun article de A_{pt}^{+} ne peut être rangé avec un article de $A_{gr} \cup A_{moy}$, $M_C(C - c_i, A_{pt}^{+}) = 0$ pour tous les articles de $A_{gr} \cup A_{moy}$.

$$L_2 = \left\lceil \frac{\sum_{a_i \in A_{gr} \cup A_{moy}} M_C(C, A_{pt}^{+}) + \sum_{a_i \in A_{pt}^{+}} 1}{M_C(C, A_{moy})} \right\rceil \tag{3.36}$$

Comme $p > \frac{C}{3}$, $M_C(C, A_{pt}^{+}) \leq 2$.

$$L_2 \geq |A_{gr} \cup A_{moy}| + \left\lceil \frac{|A_{pt}^{+}|}{2} \right\rceil \tag{3.37}$$

Ceci conclut la preuve.

□

3.4.2 Bornes inférieures pour $2BP$

Nous utilisons comme dans le Chapitre 2 le schéma général proposé par Fekete et Schepers [31] pour appliquer des fonctions dual-réalisables à une instance de $2BP$.

3.4.2.1 Nouvelles bornes inférieures

De manière analogue aux bornes proposées pour le $1BP$, nous proposons plusieurs bornes pour $2BP$. Nous donnons la formulation de $L_{DFF^a}^{2BP}$, les autres sont similaires.

$$L_{DFF^a}^{2BP} = \max_{f \in \mathcal{F}_{DFF^a}^{W}, g \in \mathcal{F}_{DFF^a}^{H}} \left\{ \left\lceil \frac{\sum_{a_i \in A} f(w_i) g(h_i)}{f(W) g(H)} \right\rceil \right\} \tag{3.38}$$

Pour chaque borne proposée pour $1BP$, une borne pour $2BP$ peut être dérivée ($L_{DFF^e}^{2BP}$, L_{CDFF}^{2BP}, et $L_{CDDFF^a}^{2BP}$).

3.4.2.2 La borne inférieure L_2^{BM}

Boschetti et Mingozzi [7] proposent une borne inférieure L_2^{new} pour $2BP$. Cette borne est décrite dans le Chapitre 1. Cette borne utilise comme sous-routine la borne L_{1BP}^{MV} de Martello et Vigo [53] pour $1BP$. Nous avons défini dans un précédent article [11] la borne L_2^{2CM} qui consiste à remplacer dans cette borne L_{1BP}^{MV} par $L_{1BP}^{DFF^a}$. Cette borne est valide et permet d'améliorer les résultats. Nous montrons maintenant que si l'on utilise les combinaisons de (D)DRF, l'amélioration par rapport à $L_{2BP}^{CDDFF^a}$ ne peut dépasser une unité.

Proposition 3.16. $L_2^{2CM} \leq L^{CDDFF^a} + 1$

Démonstration.

$$L_2^{2CM} = |A_{gr}| + LB_{1BP}^{DFF^a}(A_{lg}) + LB_{1BP}^{DFF^a}(A_{ht}) \tag{3.39}$$

$$L_2^{2CM} = |A_{gr}| + \max_{f \in \mathcal{F}^W_{CDDFF^a}} \left\{ \left\lceil \sum_{a_i \in A_{ht}} \frac{f(w_i)}{f(W)} \right\rceil \right\} + \max_{g \in \mathcal{F}^H_{CDDFF^a}} \left\{ \left\lceil \sum_{a_i \in A_{lg}} \frac{g(h_i)}{g(H)} \right\rceil \right\} \tag{3.40}$$

$$L^{CDDFF^a} \geq \max_{\substack{g \in \mathcal{F}^H_{CDDFF^a} \\ f \in \mathcal{F}^W_{CDDFF^a}}} \left\{ \left\lceil \sum_{a_i \in A} \frac{f \circ f_0^k(w_i) g \circ f_0^l(h_i)}{f \circ f_0^k(W) g \circ f_0^l(H)} \right\rceil \right\} \tag{3.41}$$

$$L^{CDDFF^a} \geq \max_{\substack{g \in \mathcal{F}^H_{CDDFF^a} \\ f \in \mathcal{F}^W_{CDDFF^a}}} \left\{ \left\lceil \sum_{a_i \in A_{gr}} \frac{f(W)g(H)}{f(W)g(H)} + \sum_{a_i \in A_{ht}} \frac{f(w_i)g(H)}{f(W)g(H)} + \sum_{a_i \in A_{lg}} \frac{f(W)g(h_i)}{f(W)g(H)} \right\rceil \right\} \tag{3.42}$$

$$L^{CDDFF^a} \geq |A_{gr}| + \max_{\substack{g \in \mathcal{F}^H_{CDDFF^a} \\ f \in \mathcal{F}^W_{CDDFF^a}}} \left\{ \left\lceil \sum_{a_i \in A_{ht}} \frac{f(w_i)}{f(W)} + \sum_{a_i \in A_{lg}} \frac{g(h_i)}{g(H)} \right\rceil \right\} \tag{3.43}$$

$$L^{CDDFF^a} \geq |A_{gr}| + \left\lceil \max_{f \in \mathcal{F}^W_{CDDFF^a}} \left\{ \sum_{a_i \in A_{ht}} \frac{f(w_i)}{f(W)} \right\} + \max_{g \in \mathcal{F}^H_{CDDFF^a}} \left\{ \sum_{a_i \in A_{lg}} \frac{g(h_i)}{g(H)} \right\} \right\rceil \tag{3.44}$$

A partir des Equations (3.40) et (3.44), le résultat est directement déduit. □

La borne L_2^{BM} peut être calculée sans coût supplémentaire. Après application de la fonction f_0^k sur chaque dimension, les articles de A_{gr} sont de taille (W, H), les articles de A_{ht} sont de taille (w_i, H) et les articles de A_{lg} sont de taille (W, h_i). On pose pour un k donné $\tilde{f_2^k}(x) = max\{f_2^k(x), f_1^1 \circ f_2^k(x)\}$. L'expression suivante domine L_2^{BM}.

$$L_{2CM} = \max_{\substack{l=1,\dots,\frac{W}{2}, q=1,\dots,\frac{H}{2} \\ k=1,\dots,\frac{W}{2}, m=1,\dots,\frac{H}{2}}} \left\{ max\left\{ \left\lceil \frac{\sum_{a_i \in A} \tilde{f_2^l} \circ f_0^k(w_i) \tilde{f_2^q} \circ f_0^m(h_i)}{\tilde{f_2^l} \circ f_0^k(W) \tilde{f_2^q} \circ f_0^m(H)} \right\rceil, |A_{gr}| + \left\lceil \sum_{a_i \in A_{ht}} \frac{\tilde{f_2^l} \circ f_0^k(w_i)}{\tilde{f_2^l} \circ f_0^k(W)} \right\rceil + \left\lceil \sum_{a_i \in A_{lg}} \frac{\tilde{f_2^q} \circ f_0^m(h_i)}{\tilde{f_2^q} \circ f_0^m(H)} \right\rceil \right\} \right\} \tag{3.45}$$

La complexité de la méthode utilisant les compositions est plus élevée que celle de L_{2CM} qui utilisait la technique de Boschetti et Mingozzi ($O(n^5)$ contre $O(n^4)$ pour cette dernière). Cette évaluation ne pourra donc être utilisée que dans les premiers niveaux d'une méthode exacte.

3.5 Résultats expérimentaux

3.5.1 Nombre de DFF générées énumérativement

Le Tableau 3.1 présente le nombre de fonctions générées par la méthode de Carlier et Néron [13]. La première colonne indique le nombre de MDFF générées par la méthode heuristique, la seconde indique le nombre total de fonctions, s'il est connu. Le symbole $*$ signifie que le nombre exact de fonctions n'est pas connu. Dans ce cas, la valeur rapportée est une borne inférieure pour ce nombre.

Tableau 3.1 – Nombre de MDFF pour chaque taille de bin des jeux d'essai $2BP$

Valeur de X	Taille du sous-ensemble	Nombre total de MRF
10	8	8
30	28	261*
40	33	770*
100	87*	1637*
300	221*	*

Clairement, le nombre total de fonctions générées est souvent trop grand pour que la méthode soit applicable en pratique. Cependant, la méthode heuristique produit des sous-ensembles de taille raisonnable. En utilisant cette méthode, les bornes peuvent être calculées pour des jeux d'essai de très grande taille, car la méthode correspondante a une complexité de $O(n)$, même si l'on applique la fonction f_1^1 après chaque fonction.

3.5.2 Résultats pour $1BP$

Nous comparons nos résultats avec ceux obtenus par Haouari et Gharbi [41], qui sont les meilleurs résultats connus pour $1BP$. Ils utilisent une procédure d'amélioration itérative des bornes pour augmenter la valeur des bornes proposées par Fekete et Schepers [29]. Pour comparer les résultats, nous utilisons les jeux d'essai générés par Haouari et Gharbi suivant la méthode proposée par Fekete et Schepers [29]. Pour ces jeux d'essai, chaque instance possède 100 articles, une taille de bin égale à 100. Les tailles des articles sont générées aléatoirement suivant une répartition discrète et uniforme dépendant du paramètre K. Les tailles des articles sont dans l'intervalle $[20, K]$ où K est égal à $30, 35, 40, 50, 60, 70$ ou 80. Pour chaque valeur de K, 100 instances sont générées. Dans le Tableau 3.5.2, nous rapportons les résultats obtenus par les méthodes décrites ci-dessous.

1. $FS + HG$: Borne de Fekete et Schepers [29] et amélioration [41]
2. DFF^a : f_0^k ou f_2^k
3. DFF^e : une fonction du sous-ensemble généré énumérativement
4. $DDFF^a$: f_0^k ou f_1^k ou f_2^k
5. $CDFF$: composition de f_0^k et f_2^l
6. $CDDFF^a$: composition de f_0^k, f_1^1 et f_2^m

7. $CDDFF^e$: composition de chaque fonction générée énumérativement avec f_1^1

Si l'on note $BEST$ la meilleure valeur obtenue par une borne inférieure pour une instance donnée, les résultats rapportés sont les suivants :

1. RATIO : Valeur moyenne de $100(BEST - LB)/BEST$. Elle représente la distance moyenne à la meilleure valeur connue.
2. ONLY : nombre d'instances parmi 100 pour lesquelles LB est la seule valeur égale à $BEST$
3. MAX : nombre d'instances parmi 100 pour lesquelles LB est égale à $BEST$
4. LESS : nombre d'instances parmi 100 pour lesquelles LB est plus petite que $BEST$

Il apparaît que les résultats sont améliorés pour plusieurs instances comparé à la méthode de Haouari et Gharbi. Toutes les améliorations sont obtenues à l'aide de L^{DFF^e}. Utiliser des combinaisons de fonctions n'est jamais intéressant sur ces données. Les fonctions générées énumérativement se comportent très bien lorsque les articles sont grands, tandis que la procédure d'amélioration des bornes appliquée à celles de Fekete et Schepers est meilleure lorsque les articles sont petits.

LB	Res	$[20,80]$	$[20,70]$	$[20,60]$	$[20,50]$	$[20,40]$	$[20,35]$	$[20,30]$
DFF^a	RATIO	20.82	31.71	4.82	37.93	0.00	38.19	11.11
	ONLY	0	0	0	0	0	0	0
	MAX	89	85	98	86	100	89	97
	LESS	11	15	2	14	0	11	3
$DDFF^a$	RATIO	20.82	31.71	4.82	37.93	0.00	38.19	11.11
	ONLY	0	0	0	0	0	0	0
	MAX	89	85	98	86	100	89	97
	LESS	11	15	2	14	0	11	3
DFF^e	RATIO	2.00	0.00	0.00	37.93	0.00	38.19	11.11
	ONLY	10	15	2	0	0	0	0
	MAX	99	100	100	86	100	89	97
	LESS	1	0	0	14	0	11	3
$CDDFF^a$	RATIO	20.82	31.71	4.82	37.93	0.00	38.19	11.11
	ONLY	0	0	0	0	0	0	0
	MAX	89	85	98	86	100	89	97
	LESS	11	15	2	14	0	11	3
$FS + HG$	RATIO	20.86	59.48	4.82	0.00	0.00	0.00	0.00
	ONLY	0	0	0	14	0	11	3
	MAX	89	73	98	100	100	100	100
	LESS	11	27	2	0	0	0	0

Tableau 3.2 – Résultats de bornes inférieures pour la Classe I

3.5.3 Résultats pour $2BP$

Dans le Chapitre 2, nous rapportons les résultats obtenus en appliquant f_0^k, f_1^k et f_2^k aux jeux d'essai de la littérature [5, 53]. Nous comparons maintenant les résultats obtenus avec chaque famille de fonctions proposées dans ce chapitre avec les résultats

précédents. Dans le Tableau 3.5.3, nous analysons l'interêt d'appliquer la fonction f_1^1 et les compositions de fonctions générées analytiquement. Pour chaque ensemble de dix jeux d'essai, nous rapportons le nombre de fois où la borne inférieure L est égale à la borne supérieure U de Boschetti et Mingozzi [8], et le ratio U/L. Dans le Tableau 3.5.3, nous comparons les résultats obtenus par les deux méthodes de génération de DFF (analytique ou énumérative). Pour chaque manière de générer des DFF, nous rapportons les résultats obtenus en appliquant la fonction f_1^1 en post-traitement.

Plusieurs observations peuvent être effectuées. Tout d'abord, les compositions des deux DFF f_0^k et f_2^l ne conduisent pas à une amélioration substantielle des résultats. Seul le résultat obtenu pour une unique instance est amélioré (Classe IX). En revanche, l'impact de f_1^1 est important, car les résultats obtenus sans f_1^1 sont souvent moins bons (Classes I, III, V, VII, $XIII$, IX, X). Il apparaît aussi que combiner les (D)DFF conduit à de meilleures bornes (Classes III, V, $VIII$ et IX).

Toutes les fonctions utiles pour ces jeux d'essai qui sont issues des familles f_0^k et f_2^k sont incluses dans l'ensemble des fonctions générées énumérativement. Inversement, bien que les DFF^a et les DFF^e conduisent souvent à la même évaluation, les fonctions f_0^k et f_2^k ne couvrent pas l'intégralité des fonctions intéressantes (voir Classes $VIII$ et IX). Lorsque DFF^e sont utilisées, l'usage de f_1^1 est toujours intéressant et conduit à de meilleurs résultats pour plusieurs instances.

Dans le Tableau 3.5.3, nous analysons les performances de nos bornes lorsqu'elles sont combinées à la méthode L_2^{BM} proposée par Boschetti et Mingozzi [7]. Nous notons BM la borne proposée par Boschetti et Mingozzi [7] pour $2BP$, qui est déjà dominée par les bornes du Chapitre 2. L'intérêt de combiner notre méthode avec L_2^{BM} est moins grand lorsque $CDFF$ ou DFF^e sont utilisées, car les résultats sont améliorés pour un faible nombre d'instances. Lorsque la taille des instances est grande, le temps de calcul de $CDDFF^a$ devient très important. Nous ne rapportons donc pas les résultats pour toutes les classes considérées. Même si les bornes étaient bonnes, les résultats que nous proposons améliorent encore leur qualité.

3.6 Conclusion

Les bornes proposées dans le Chapitre 2 étaient bonnes, mais les deux méthodes de génération de (D)DFF que nous proposons dans ce chapitre permettent d'obtenir de meilleurs résultats. Le concept de MDFF est très utile, car le sous-ensemble de fonctions générées conduit à des évaluations qui ne sont pas atteintes par les autres méthodes. Le nombre de fonctions est toujours grand, et la méthode obtenue ne peut être utilisée dans le cadre d'une méthode par séparation et évaluation, mais peut l'être en revanche à la racine d'une telle méthode. Les MDFF analytiques sont en moindre nombre, et même si elles conduisent parfois à des résultats un peu moins bons, elles restent intéressantes à utiliser au cours d'une méthode arborescente. Nous comptons mettre au point des méthodes heuristiques pour générer des ensembles de MDFF de moindre cardinalité. Une autre manière d'améliorer les résultats est de proposer de nouvelles (D)DFF.

Problème			Ratio U/L				Opt			
Class	$H \times W$	n	DFF^a	$DDFF^a$	$CDFF$	$CDDFF^a$	DFF^a	$DDFF^a$	$CDFF$	$CDDFF^a$
I	10 × 10	20	1.0325	1.0125	1.0325	1.0125	8	9	8	9
		40	1.0320	1.0265	1.0320	1.0265	6	7	6	7
		60	1.0171	1.0171	1.0171	1.0171	7	7	7	7
		80	1.0000	1.0000	1.0000	1.0000	10	10	10	10
		100	1.0032	1.0032	1.0032	1.0032	9	9	9	9
		Avg	1.0170	1.0119	1.0170	1.0119	8.0	8.4	8.0	8.4
II	30 × 30	20	1.0000	1.0000	1.0000	1.0000	10	10	10	10
		40	1.0000	1.0000	1.0000	1.0000	10	10	10	10
		60	1.0000	1.0000	1.0000	1.0000	10	10	10	10
		80	1.0000	1.0000	1.0000	1.0000	10	10	10	10
		100	1.0000	1.0000	1.0000	1.0000	10	10	10	10
		Avg	1.0000	1.0000	1.0000	1.0000	10.0	10.0	10.0	10.0
III	40 × 40	20	1.0817	1.0367	1.0817	1.0200	6	8	6	9
		40	1.0411	1.0411	1.0411	1.0411	7	7	7	7
		60	1.0302	1.0302	1.0302	1.0244	6	6	6	7
		80	1.0160	1.0160	1.0160	1.0160	7	7	7	7
		100	1.0290	1.0290	1.0290	1.0204	4	4	4	6
		Avg	1.0396	1.0306	1.0396	1.0244	6.0	6.4	6.0	7.2
IV	100 × 100	20	1.0000	1.0000	1.0000	1.0000	10	10	10	10
		40	1.0000	1.0000	1.0000	1.0000	10	10	10	10
		60	1.1000	1.1000	1.1000	1.1000	8	8	8	8
		80	1.1000	1.1000	1.1000	1.1000	7	7	7	7
		100	1.0333	1.0333	1.0333	1.0333	9	9	9	9
		Avg	1.0467	1.0467	1.0467	1.0467	8.8	8.8	8.8	8.8
V	100 × 100	20	1.0861	1.0250	1.0861	1.0000	5	9	5	10
		40	1.0271	1.0271	1.0271	1.0271	7	7	7	7
		60	1.0173	1.0110	1.0173	1.0110	7	8	7	8
		80	1.0295	1.0295	1.0295	1.0295	4	4	4	4
		100	1.0308	1.0308	1.0308	1.0308	3	3	3	3
		Avg	1.0382	1.0247	1.0382	1.0197	5.2	6.2	5.2	6.4
VI	300 ×300	20	1.0000	1.0000	1.0000	1.0000	10	10	10	10
		40	1.3000	1.3000	1.3000	1.3000	7	7	7	7
		60	1.0000	1.0000	1.0000	1.0000	10	10	10	10
		80	1.0000	1.0000	1.0000	1.0000	10	10	10	10
		100	1.0667	1.0667	1.0667	1.0667	8	8	8	8
		Avg	1.0733	1.0733	1.0733	1.0733	9.0	9.0	9.0	9.0
VII	100 ×100	20	1.0367	1.0000	1.0367	1.0000	8	10	8	10
		40	1.0202	1.0111	1.0202	1.0111	8	9	8	9
		60	1.0257	1.0186	1.0257	1.0186	6	7	6	7
		80	1.0367	1.0233	1.0367	1.0233	2	5	2	5
		100	1.0188	1.0108	1.0188	1.0108	5	7	5	7
		Avg	1.0276	1.0128	1.0276	1.0128	5.8	7.6	5.8	7.6
$VIII$	100 × 100	20	1.0750	1.0417	1.0750	1.0250	6	8	6	9
		40	1.0091	1.0091	1.0091	1.0091	9	9	9	9
		60	1.0184	1.0121	1.0184	1.0121	7	8	7	8
		80	1.0139	1.0139	1.0139	1.0139	7	7	7	7
		100	1.0217	1.0217	1.0217	1.0217	4	4	4	4
		Avg	1.0276	1.0197	1.0276	1.0164	6.6	7.2	6.6	7.4
IX	100 × 100	20	1.0215	1.0138	1.0138	1.0000	7	8	8	10
		40	1.0140	1.0000	1.0140	1.0000	6	10	6	10
		60	1.0069	1.0022	1.0069	1.0022	7	9	7	9
		80	1.0086	1.0016	1.0086	1.0016	6	9	6	9
		100	1.0029	1.0014	1.0029	1.0014	8	9	8	9
		Avg	1.0108	1.0038	1.0093	1.0010	6.8	9.0	7.0	9.4
X	100 × 100	20	1.1450	1.0700	1.1450	1.0700	6	8	6	8
		40	1.0343	1.0343	1.0343	1.0343	8	8	8	8
		60	1.0533	1.0450	1.0533	1.0450	5	6	5	6
		80	1.0563	1.0563	1.0563	1.0563	3	3	3	3
		100	1.0584	1.0584	1.0584	1.0584	1	1	1	1
		Avg	1.0695	1.0528	1.0695	1.0528	4.6	5.2	4.6	5.2

Tableau 3.3 – Résultats pour la composition de (D)DFF analytiques

Problème			Ratio U/L				Opt			
Class	$H \times W$	n	DFF^a	DFF^e	$CDDFF^a$	$CDDFF^e$	DFF^a	DFF^e	$CDDFF^a$	$CDDFF^e$
I	10 × 10	20	1.0325	1.0325	1.0125	1.0125	8	8	9	9
		40	1.0320	1.0320	1.0265	1.0265	6	6	7	7
		60	1.0171	1.0171	1.0171	1.0171	7	7	7	7
		80	1.0000	1.0000	1.0000	1.0000	10	10	10	10
		100	1.0032	1.0032	1.0032	1.0032	9	9	9	9
		Avg	1.0170	1.0170	1.0119	1.0119	8.0	8.0	8.4	8.4
II	30 × 30	20	1.0000	1.0000	1.0000	1.0000	10	10	10	10
		40	1.0000	1.0000	1.0000	1.0000	10	10	10	10
		60	1.0000	1.0000	1.0000	1.0000	10	10	10	10
		80	1.0000	1.0000	1.0000	1.0000	10	10	10	10
		100	1.0000	1.0000	1.0000	1.0000	10	10	10	10
		Avg	1.0000	1.0000	1.0000	1.0000	10.0	10.0	10.0	10
III	40 × 40	20	1.0817	1.0817	1.0200	1.0200	6	6	9	9
		40	1.0411	1.0411	1.0411	1.0411	7	7	7	7
		60	1.0302	1.0219	1.0244	1.0160	6	7	7	8
		80	1.0160	1.0160	1.0160	1.0160	7	7	7	7
		100	1.0290	1.0187	1.0204	1.0148	4	6	6	7
		Avg	1.0396	1.0359	1.0244	1.0216	6.0	6.6	7.2	7.6
IV	100 × 100	20	1.0000	1.0000	1.0000	1.0000	10	10	10	10
		40	1.0000	1.0000	1.0000	1.0000	10	10	10	10
		60	1.1000	1.1000	1.1000	1.1000	8	8	8	8
		80	1.1000	1.1000	1.1000	1.1000	7	7	7	7
		100	1.0333	1.0333	1.0333	1.0333	9	9	9	9
		Avg	1.0467	1.0467	1.0467	1.0467	8.8	8.8	8.8	8.8
V	100 × 100	20	1.0861	1.0861	1.0000	1.0000	5	5	10	10
		40	1.0271	1.0271	1.0271	1.0271	7	7	7	7
		60	1.0173	1.0173	1.0110	1.0110	7	7	8	8
		80	1.0295	1.0249	1.0295	1.0249	4	5	4	5
		100	1.0308	1.0308	1.0308	1.0308	3	3	3	3
		Avg	1.0382	1.0373	1.0197	1.0188	5.2	5.4	6.4	6.6
VI	300 ×300	20	1.0000	1.0000	1.0000	1.0000	10	10	10	10
		40	1.3000	1.3000	1.3000	1.3000	7	7	7	7
		60	1.0000	1.0000	1.0000	1.0000	10	10	10	10
		80	1.0000	1.0000	1.0000	1.0000	10	10	10	10
		100	1.0667	1.0667	1.0667	1.0667	8	8	8	8
		Avg	1.0733	1.0733	1.0733	1.0733	9.0	9.0	9.0	9.0
VII	100 ×100	20	1.0367	1.0367	1.0000	1.0000	8	8	10	10
		40	1.0202	1.0202	1.0111	1.0111	8	8	9	9
		60	1.0257	1.0257	1.0186	1.0186	6	6	7	7
		80	1.0367	1.0367	1.0233	1.0233	2	2	5	5
		100	1.0188	1.0188	1.0108	1.0108	5	5	7	7
		Avg	1.0276	1.0276	1.0128	1.0128	5.8	5.8	7.6	7.6
$VIII$	100 × 100	20	1.0750	1.0583	1.0250	1.0250	6	7	9	9
		40	1.0091	1.0091	1.0091	1.0091	9	9	9	9
		60	1.0184	1.0184	1.0121	1.0121	7	7	8	8
		80	1.0139	1.0139	1.0139	1.0139	7	7	7	7
		100	1.0217	1.0217	1.0217	1.0217	4	4	4	4
		Avg	1.0276	1.0243	1.0164	1.0164	6.6	6.8	7.4	7.4
IX	100 × 100	20	1.0215	1.0000	1.0000	1.0000	7	10	10	10
		40	1.0140	1.0108	1.0000	1.0000	6	7	10	10
		60	1.0069	1.0048	1.0022	1.0000	7	8	9	10
		80	1.0086	1.0053	1.0016	1.0000	6	7	9	10
		100	1.0029	1.0029	1.0014	1.0014	8	8	9	9
		Avg	1.0108	1.0047	1.0010	1.0003	6.8	8.0	9.4	9.8
X	100 × 100	20	1.1450	1.0950	1.0700	1.0200	6	7	8	9
		40	1.0343	1.0143	1.0343	1.0143	8	9	8	9
		60	1.0533	1.0561	1.0450	1.0561	5	5	6	5
		80	1.0563	1.0486	1.0563	1.0486	3	4	3	4
		100	1.0584	1.0584	1.0584	1.0584	1	1	1	1
		Avg	1.0695	1.0545	1.0528	1.0395	4.6	5.2	5.2	5.6

Tableau 3.4 – Comparaison entre les deux manières de générer des (D)DFF

Problème			Ratio U/L				Opt			
Class	$H \times W$	n	BM	$DDFF^a$	DFF^e	$CDDFF^a$	BM	$DDFF^a$	DFF^e	$CDDFF^a$
I	10 × 10	20	1.0125	1.0125	1.0125	1.0125	9	9	9	9
		40	1.0265	1.0265	1.0265	1.0265	7	7	7	7
		60	1.0227	1.0171	1.0171	1.0171	6	7	7	7
		80	1.0042	1.0000	1.0000	1.0000	9	10	10	10
		100	1.0032	1.0032	1.0032	1.0032	9	9	9	9
		Avg	1.0138	1.0119	1.0119	1.0119	8.0	8.4	8.4	8.4
II	30 × 30	20	1.0000	1.0000	1.0000	1.0000	10	10	10	10
		40	1.0000	1.0000	1.0000	1.0000	10	10	10	10
		60	1.0000	1.0000	1.0000	1.0000	10	10	10	10
		80	1.0000	1.0000	1.0000	1.0000	10	10	10	10
		100	1.0000	1.0000	1.0000	1.0000	10	10	10	10
		Avg	1.0000	1.0000	1.0000	1.0000	10.0	10.0	10.0	10.0
III	40 × 40	20	1.0367	1.0367	1.0367	1.0200	8	8	8	9
		40	1.0411	1.0411	1.0411	1.0411	7	7	7	7
		60	1.0315	1.0244	1.0244	1.0244	6	7	7	7
		80	1.0219	1.0160	1.0160	1.0160	6	7	7	7
		100	1.0251	1.0251	1.0251	1.0204	5	5	5	6
		Avg	1.0313	1.0286	1.0286	1.0244	6.4	6.8	6.8	7.2
IV	100 × 100	20	1.0000	1.0000	1.0000	1.0000	10	10	10	10
		40	1.0000	1.0000	1.0000	1.0000	10	10	10	10
		60	1.1000	1.1000	1.1000	1.1000	8	8	8	8
		80	1.1000	1.1000	1.1000	1.1000	7	7	7	7
		100	1.0333	1.0333	1.0333	1.0333	9	9	9	9
		Avg	1.0467	1.0467	1.0467	1.0467	8.8	8.8	8.8	8.8
V	100 × 100	20	1.0000	1.0000	1.0000	-	10	10	10	-
		40	1.0271	1.0271	1.0271	-	7	7	7	-
		60	1.0110	1.0110	1.0110	-	8	8	8	-
		80	1.0295	1.0295	1.0295	-	4	4	4	-
		100	1.0308	1.0308	1.0308	-	3	3	3	-
		Avg	1.0197	1.0197	1.0197	-	6.4	6.4	6.4	-
VI	300 ×300	20	1.0000	1.0000	1.0000	-	10	10	10	-
		40	1.3000	1.3000	1.3000	-	7	7	7	-
		60	1.0000	1.0000	1.0000	-	10	10	10	-
		80	1.0000	1.0000	1.0000	-	10	10	10	-
		100	1.0667	1.0667	1.0667	-	8	8	8	-
		Avg	1.0733	1.0733	1.0733	-	9.0	9.0	9.0	-
VII	100 ×100	20	1.0200	1.0000	1.0367	-	9	10	8	-
		40	1.0202	1.0111	1.0202	-	8	9	8	-
		60	1.0257	1.0186	1.0257	-	6	7	6	-
		80	1.0367	1.0233	1.0367	-	2	5	2	-
		100	1.0188	1.0108	1.0188	-	5	7	5	-
		Avg	1.0243	1.0128	1.0276	-	6.0	7.6	5.8	-
$VIII$	100 ×100	20	1.0000	1.0000	1.0000	-	10	10	10	-
		40	1.0091	1.0091	1.0091	-	9	9	9	-
		60	1.0184	1.0121	1.0184	-	7	8	7	-
		80	1.0139	1.0139	1.0139	-	7	7	7	-
		10	1.0217	1.0217	1.0217	-	4	4	4	-
		Avg	1.0126	1.0114	1.0126	-	7.4	7.6	7.4	-
IX	100 ×100	20	1.0000	1.0000	1.0000	-	10	10	10	-
		40	1.0000	1.0000	1.0000	-	10	10	10	-
		60	1.0000	1.0000	1.0000	-	10	10	10	-
		80	1.0000	1.0000	1.0000	-	10	10	10	-
		100	1.0000	1.0000	1.0000	-	10	10	10	-
		Avg	1.0000	1.0000	1.0000	-	10.0	10.0	10.0	-
X	100 × 100	20	1.0950	1.0700	1.0950	-	7	8	7	-
		40	1.0343	1.0343	1.0343	-	8	8	8	-
		60	1.0533	1.0450	1.0644	-	5	6	4	-
		80	1.0563	1.0563	1.0640	-	3	3	2	-
		100	1.0584	1.0584	1.0584	-	1	1	1	-
		Avg	1.0595	1.0528	1.0632	-	4.8	5.2	4.4	-

Tableau 3.5 – Résultats pour les bornes combinées avec L_2^{BM}

Le problème de faisabilité

4.1 Introduction

Dans ce chapitre, nous nous intéressons au problème de *faisabilité*[1]. Il s'agit de déterminer si un sous-ensemble d'articles A peut être contenu dans un unique bin B. Ce problème trouve en lui-même des applications pratiques dans le monde industriel, mais représente aussi un sous-problème crucial pour $2BP$, ainsi que pour les problèmes de *knapsack* $2D$ ou de *rectangle packing* (*cf.* Chapitre 1).

Martello et Vigo [53] ont proposé une méthode exacte pour ce problème dans un article consacré au problème $2BP$. Parmi les méthodes exactes existantes [30, 40, 53], seuls Fekete et Schepers [30] n'utilisent pas le schéma classique de placement d'articles en bas à gauche dans le bin, mais une modélisation sous forme de graphes d'intervalles. Ces méthodes sont détaillées dans le premier chapitre de la présente thèse.

Nous montrons dans ce chapitre que les bornes inférieures et les procédures de réduction proposées dans le Chapitre 2 peuvent être adaptées au problème de faisabilité. Nous présentons aussi une méthode exacte dédiée à ce problème. Il s'agit d'une méthode arborescente en deux phases. Dans la première étape (étape *extérieure*), on procède à une énumération de toutes les solutions d'un problème relâché. Dans cette phase, seule l'abscisse des articles est fixée, ce qui permet de regrouper un grand nombre de configurations. Pour toutes les configurations trouvées de cette manière, on lance une deuxième méthode arborescente (*étape intérieure*) qui détermine s'il existe une solution au problème initial avec les abscisses fixées dans la première étape. C'est dans cette phase que l'ordonnée des articles est fixée. Lorsque le problème n'admet pas de solution, il est fréquent que le problème relâché n'en n'ait pas non plus. Dans ce cas, l'étape intérieure n'est jamais atteinte, et l'aspect combinatoire s'en trouve considérablement réduit. Dans le cas général, lorsque l'on effectue une coupe dans l'arbre de recherche de la première phase, c'est tout un ensemble de configurations non valides pour le *bin-packing* qui n'est pas développé.

A chaque étape de l'énumération, la qualité des bornes inférieures est améliorée en les appliquant à des instances tranformées. Les nouvelles instances auxquelles on applique nos évaluations sont déduites du placement des premiers articles dans le bin. Il s'agit *d'agréger* les articles qui sont placés les uns à côté des autres pour obtenir des articles de plus grande taille avec lesquels les bornes vont avoir un meilleur comportement.

1. Le travail présenté dans ce chapitre correspond à l'article [17]. Il a aussi été présenté en congrès [18].

Nous avons généré plusieurs instances de problèmes afin de tester l'efficacité de notre approche. Pour créer des problèmes difficiles, nous prenons en compte le nombre d'articles, la taille du bin et la différence entre la surface du bin et la surface globale des articles. Nous comparons notre algorithme avec celui proposé par Martello et Vigo [53] à l'aide de nouveaux jeux d'essai générés aléatoirement. Les résultats expérimentaux confirment l'intérêt de notre méthode.

Dans la section 4.2, nous montrons comment les prétraitements proposés pour le problème d'optimisation peuvent être améliorés pour le problème de faisabilité. La section 4.3 décrit le processus d'amélioration de nos bornes inférieures. Dans la section 4.4, nous proposons un nouveau principe de sépara-tion que nous intégrons dans une méthode de recherche par séparation et évaluation (pse). La section 4.5 est dédiée à la gestion des redondances qui apparaissent au cours de l'énumération. Dans la section 4.6, nous montrons l'intérêt pratique de notre méthode à l'aide de résultats numériques.

4.2 Procédures de réduction

Nous présentons maintenant des procédures de réduction pour le problème de faisabilité. Ces méthodes peuvent être utilisées comme prétraitement, ou au cours d'une méthode énumérative. Elles permettent de réduire la taille de l'instance dans de nombreux cas. Certaines sont directement dérivées des méthodes décrites dans le Chapitre 2 pour $2BP$, d'autres ne s'appliquent qu'au problème de faisabilité.

4.2.1 Grands articles

La première procédure de réduction s'appuie sur une constatation évidente. Si la hauteur d'un article est égale à celle du bin, alors il est optimal de le placer sur un côté du bin. Le prétraitement que nous proposons est le suivant : à chaque étape de l'algorithme, s'il existe dans A un article a_i tel que $h_i = H$ (resp. $w_i = W$), on le supprime de A et on retranche sa largeur (resp. sa hauteur) au bin. On s'arrête lorsqu'il n'existe plus de tels articles. Bien entendu, il n'est pas nécessaire que l'article a_i ait sa hauteur strictement égale à H. S'il n'existe pas d'article a_j tel que $h_i + h_j \leq H$, le traitement peut être appliqué.

Lorsque l'on réduit la taille du bin, il est possible qu'il ne puisse plus contenir certains articles. Dans ce cas, on déduit directement que le problème n'est pas réalisable. Plus généralement, ce traitement permet de supprimer des articles, surtout si l'on a appliqué au préalable un traitement sur la taille des articles (des DFF, par exemple). En outre, cette méthode peut être actualisée au cours de la méthode arborescente.

4.2.2 Cas particulier du cadre

Le cas particulier que nous traitons ici concerne non plus un article mais un ensemble de quatre articles qui ne peuvent être placés qu'en forme de *cadre* autour du bin (*cf.* Figure 4.1). On s'intéresse à une instance (A, B) du problème de faisabilité, avec quatre articles a_i, a_j, a_k, et a_l de A tels que

- $h_i + h_k = H$
- $h_j + h_l = H$
- $w_i + w_l = W$
- $w_j + w_k = W$

Proposition 4.1. *Il existe une configuration réalisable pour l'instance (A, B), si et seulement s'il existe une configuration réalisable pour (A, B) telle que*

- a_i *est placé en* $(0, 0)$
- a_l *en* $(w_i, 0)$
- a_k *en* $(0, h_i)$
- a_j *en* $(w_k, H - h_j)$

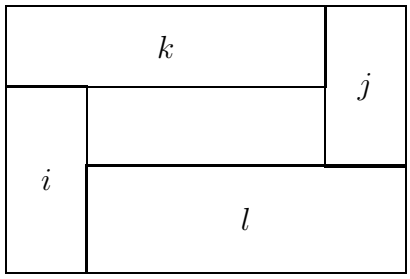


Figure 4.1 – Configuration du cadre

Le traitement proposé est directement déduit de la proposition. Si l'on trouve un tel quadruplet, on place les articles *en cadre* autour du bin, ce qui revient à supprimer ces articles de A et réduire B en conséquence. La méthode obtenue est plus coûteuse que la précédente, mais elle est utile car ce cas de figure est particulièrement défavorable aussi bien aux heuristiques qu'aux algorithmes énumératifs en raison de son caractère très contraint. Il est à noter que comme pour les grands articles, on peut généraliser cette méthode lorsque les quatre articles ne forment pas un cadre parfait mais ne peuvent pas être placés différemment.

4.2.3 Un prétraitement pour le problème de décision

Dans le Chapitre 2, nous proposons un prétraitement que nous notons v_1. Il repose sur la constatation suivante : si l'on peut ranger l'ensemble des articles de faible hauteur au-dessus des articles dont la hauteur est grande, alors ce rangement est optimal. Soit D une instance de $2BP$ et un entier l, $1 \leq l \leq \frac{1}{2}H$. Nous définissons $A_{ht} = \{a_i \in A : h_i > H - l\}$, et $A_s = \{a_i \in A : h_i < l\}$. Nous notons $m = |A_{ht}|$. Pour chaque article a_i de A_{ht}, un bin B_i de taille $(H - h_i, w_i)$ est créé. Considérons $D' = (A_s, \{B_1, \ldots B_m\})$ le problème suivant : *¨Existe-t-il une configuration réalisable pour A_s dans les bins $B_1, \ldots B_m$?¨* .

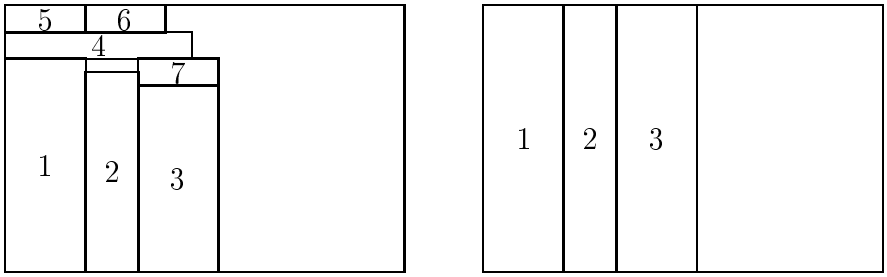


Figure 4.2 – Procédure v_1^*

Proposition 4.2. *Si D' a une solution réalisable alors la fonction*

$$v_1^{l,H} : [0,W] \times [0,H] \to [0,W] \times [0,H]$$
$$a_i \mapsto \begin{cases} (w_i, H) \text{ si } h_i > H - l \\ (0,0) \text{ si } h_i < l \\ (w_i, h_i) \text{ sinon} \end{cases}$$

est une IFF pour D.

Le traitement que nous proposons dans ce cas de figure est le suivant : on augmente la taille des grands articles jusqu'à la valeur H et on supprime les petits articles. Les grands articles ainsi obtenus peuvent être supprimés comme nous l'avons expliqué plus haut.

Cependant, ce traitement est inefficace lorsqu'il existe un article de faible hauteur et trop large pour être positionné au-dessus d'un seul grand article. On peut l'améliorer en exploitant le fait que tous les articles doivent être placés dans le même bin. Dans ce cas, un article de faible hauteur mais de grande largeur peut être rangé au-dessus de plusieurs articles hauts. Nous notons v_1^* le prétraitement obtenu. La figure 4.2 illustre le fait que v_1^* peut réduire le problème alors que v_1 ne le permet pas.

Nous utilisons un algorithme glouton pour résoudre le problème de décision spécifié. Pour une valeur donnée de l, les ensembles A_{ht} et A_s sont calculés. Les articles de A_{ht} sont rangés dans le coin bas gauche du bin par ordre décroissant de taille, puis nous essayons de ranger les petits articles au-dessus d'eux en utilisant la règle *bottom-left decreasing* décrite dans le Chapitre 1.

4.3 Bornes inférieures

Dans le chapitre précédent, nous avons présenté de nouvelles évaluations par défaut pour $2BP$. Elles s'appliquent aussi au problème de faisabilité : si la valeur de la borne inférieure dépasse un, alors il n'existe pas de solution pour l'instance considérée. Dans cette section, nous montrons comment les bornes inférieures peuvent être améliorées lorsqu'une configuration partielle pour le problème est connue.

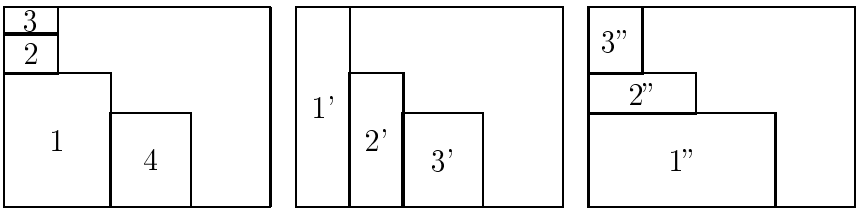


Figure 4.3 – Générer de nouvelles instances pour améliorer les bornes

Considérons maintenant une étape d'une méthode énumérative où l'abscisse des articles est connue. Les rangements d'articles déjà effectués ajoutent de l'information qui permet d'améliorer les bornes inférieures et les procédures de réduction. Une manière immédiate d'exploiter les placements d'articles est de vérifier que pour chaque article non placé, il existe une surface vacante qui peut l'accueillir. La nouvelle méthode que nous proposons repose sur une considération géométrique. Considérons le polygone ψ formé par l'ensemble A_1 des articles déjà placés dans le bin. On peut remplacer les articles de A_1 par ceux d'un autre ensemble A'_1 tel que les articles de A'_1 peuvent être rangés dans ψ de manière valide, on obtient la propriété suivante :

Proposition 4.3. *S'il n'existe pas de configuration réalisable pour $A - A_1 + A'_1$ dans B, alors il n'existe pas de configuration réalisable pour A dans B telle que les articles de A_1 forment le polygone ψ.*

Démonstration. Si l'instance originale D admet une solution réalisable telle que A_1 est rangé de manière à ce qu'il soit inclus dans le polygone ψ, une solution pour l'instance modifiée D' peut être calculée en rangeant les articles de A'_1 de telle sorte qu'ils soient inclus dans le même polygone. Les autres articles sont rangés de la même façon dans les deux configurations. □

On découpe le polygone en un nouvel ensemble d'articles en maximisant la hauteur des articles générés, ce qui produit une première instance. On obtient une seconde instance en maximisant la largeur (Figure 4.3).

La limite inhérente aux instances créées précédemment réside dans la liberté supplémentaire que l'on laisse dans le placement des articles, ce qui peut rendre le problème plus facile. Par exemple, si deux articles superposés ont une largeur assez faible pour pouvoir être placés côte à côte, une configuration initialement non réalisable peut le devenir. Pour éviter ce cas de figure, nous opérons une deuxième modification sur les articles. On force deux articles posés l'un au-dessus de l'autre à rester dans cette configuration après modification. Pour cela, on ajoute à ces articles une longueur égale à celle du bin et on double la longueur du bin. Si les articles placés n'occupent pas toute la hauteur du bin, on peut ajouter un article factice (dummy item) de longueur W et de hauteur égale à l'espace vacant au-dessus des

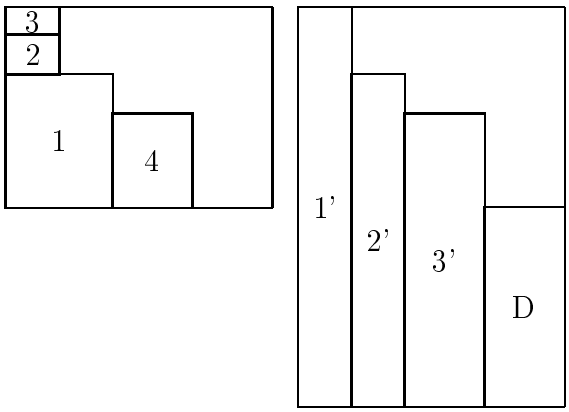


Figure 4.4 – Améliorer les nouvelles instances

articles placés (Figure 4.4). Le même traitement peut être effectué sur la longueur. L'instance obtenue est noté D''.

Proposition 4.4. *Si la nouvelle instance D'' n'admet pas de solution réalisable, il n'y a pas de solution réalisable pour l'instance initiale D qui inclut la configuration courante.*

Démonstration. S'il y a une solution pour D qui inclut le rangement des articles de A_1 dans ψ, il existe une solution pour D'. Elle peut être calculée en rangeant les articles modifiés dans la surface couverte par ψ et les autres articles de manière similaire à la solution trouvée. Donc, s'il y a une solution pour D qui inclut le rangement de A_1 dans le polygone ψ alors il y a une configuration réalisable possible pour l'instance modifiée. □

On peut à nouveau utiliser les prétraitements décrits dans la section 4.2, ainsi que nos bornes inférieures. Le nombre de grands articles est augmenté, le problème s'en retrouve ainsi considérablement réduit. De plus, le traitement v_1^* peut fournir de meilleurs résultats car le nombre de petits articles restants est plus faible.

4.4 Méthodes exactes

Nous proposons dans cette section deux nouvelles méthodes exactes pour le problème de faisabilité. Le premier algorithme est une amélioration de la méthode classique de placement des articles dans le bin. Le deuxième repose sur une méthode en deux étapes. Le principe est le suivant : dans un premier temps, on relâche des contraintes du problème initial, puis dans un deuxième temps on énumère toutes les solutions du problème obtenu. Pour chaque solution trouvée, on teste à l'aide d'une méthode exacte si elle peut conduire à une solution du problème initial.

Plusieurs méthodes de la littérature [40, 53] reposent sur la construction itérative d'une configuration réalisable. Les configurations énumérées sont celles qui respectent la stratégie *leftmost downward*. Les positions dominantes pour chaque

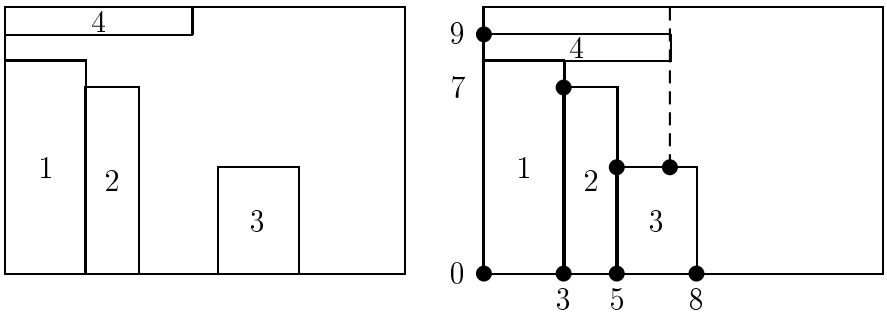


Figure 4.5 – La règle *leftmost downward*

article a_i sont celles pour lesquelles a_i a son côté gauche adjacent au bin, ou au côté droit d'un autre article, et son côté inférieur adjacent au bin ou à un autre article. Dans la Figure 4.5, la configuration gauche est dominée par la configuration droite. A chaque étape, il existe une liste finie de k positions et une liste de l items non encore placés. Il y a deux choix à faire : celui de l'article, et celui de sa position. Ces choix conduisent à $k \times l$ fils dans l'arbre de décision. Cette méthode génère un très grand nombre de redondances qui doivent être gérées si on veut obtenir un algorithme efficace. En effet, une même configuration peut être atteinte plusieurs fois dans l'arbre. Un exemple est donné dans la Figure 4.5 : la configuration de droite peut être obtenue en suivant deux ordres différents.

$$\pi_1 = [(a_1, (0,0)), (a_2, (3,0)), (a_3, (5,0)), (a_4, (0,8))]$$

$$\pi_2 = [(a_1, (0,0)), (a_4, (0,8)), (a_2, (3,0)), (a_3, (5,0))]$$

Comme Martello et Vigo [53] ont publié les meilleurs résultats pour $2BP$ en suivant ce modèle, nous notons l'algorithme obtenu MV.

4.4.1 Améliorer la méthode classique : $LMAO$

Nous proposons une amélioration de la méthode classique qui évite un grand nombre de répétitions au cours de la pse. A chaque nœud de l'arbre de recherche, notre méthode génère au plus $n_k + 1$ fils, qui correspondent au rangement dans une position unique des n_k articles restants, ainsi qu'à la possibilité de ne rien placer à cette position. Dans ce dernier cas, la coordonnée correspondante ne sera plus jamais utilisée.

Nous décrivons maintenant notre méthode en détail. Nous définissons le concept de positions *actives* et *inactives*. Initialement, toutes les positions sont actives. La position active la plus en bas à gauche est notée c. A chaque nœud de l'arbre de recherche, l'algorithme teste la possibilité de ranger chaque article à la position c. Dans ce cas, la possibilité de ne rien ranger à cet endroit est aussi testée : on ne reviendra plus jamais sur cette position. Cette position est dite *inactive*. Nous dénommons la méthode obtenue *LMAO (L'actif le plus en bas à gauche uniquement)*.

Quand un article a_i est rangé à la position $c = (x, y)$, la surface située à la gauche de a_i est supprimée. En effet, la possibilité de ranger un article dans cette position est énumérée dans une autre branche de l'arbre. En pratique, un article factice est créé et ajouté à gauche de a_i : une nouvelle position active est créée à la position $y_i + h_i$. Lorsque cet article factice est ajouté, la surface totale des articles peut devenir supérieure à celle du bin et conduire à une coupe dans l'arbre de recherche.

Définition 4.1. *Un ordre LMAO est une séquence de choix qui conduisent à une solution en suivant l'algorithme LMAO. Il est composé du choix du prochain article a_i à ranger dans la position dominante c : (a_i, c) et du choix de ne ranger aucun article dans la position dominante : $\overline{c}$.*

La méthode *LMAO* a l'avantage de réduire le nombre d'ordres sur les articles qui conduisent à la même solution. Cet ordre est unique pour une solution donnée : par exemple, la configuration de la Figure 4.5 ne peut être atteinte que par l'ordre qui suit :

$$\pi = [(a_1, (0, 0)), (a_4, (0, 8)), \overline{(0, 9)}, (a_2, (3, 0)), \overline{(3, 7)}, (a_3, (5, 0))]$$

Proposition 4.5. *Pour une configuration dominante P, il n'y a qu'un seul ordre LMAO π qui mène à P.*

Démonstration. Soit π_1, π_2 deux ordres *LMAO* distincts et k le premier indice pour lequel $\pi_1(k) \neq \pi_2(k)$. Comme les deux configurations partielles obtenues avant l'étape k sont les mêmes, deux cas sont possibles.

1. $\pi_1(k) = (a_i, c)$ et $\pi_2(k) = (a_j, c)$, $j \neq i$. Dans ce cas, l'article a_i ne peut avoir la même position dans les deux configurations obtenues car a_j est positionné à cette coordonnée si l'on suit l'ordre *LMAO* π_2.
2. $\pi_1(k) = \overline{c}$ et $\pi_2(k) = (a_j, c)$. Dans ce cas, la position c ne sera pas utilisée suivant l'ordre π_1 et l'article a_j ne peut donc pas être placé en position c.

Les configurations obtenues ne peuvent pas être les mêmes.

□

Théorème 4.1. *LMAO est une méthode valide pour le problème de faisabilité.*

Démonstration. Nous montrons que pour toute configuration P énumérée par *MV*, il existe un ordre *LMAO* qui mène à P. Comme *MV* est valide, le théorème est directement démontré.

Soit P une configuration obtenue par *MV*. Nous allons construire π_1 un nouvel ordre *MV* à partir de P en sélectionnant itérativement l'article qui est rangé dans la position la plus en bas à gauche à l'étape courante. $\pi_1 = [(a_1, c_1), (a_2, c_2), \ldots, (a_n, c_n)]$. A partir de π_1, il est possible d'obtenir un ordre *LMAO* π_2 qui mène à P. Pour chaque choix (a_i, c_k) dans π_1 fait avec *MV*, nous proposons une séquence équivalente de décisions pour *LMAO*.

1. c_k est le coin le plus en bas à gauche. Le choix *LMAO* consiste aussi à ranger l'article a_i dans cette position.

2. c_k n'est pas la position la plus en bas à gauche c_l. Par construction de π_1, il n'existe pas d'article positionné en c_k dans$P$. Le choix $LMAO$ consiste à rendre inactif le coin bas gauche courant tant qu'il est différent de celui utilisé par MV. Quand une position est déclarée inactive, cela signifie qu'elle n'est pas utilisée dans P, sans quoi elle serait égale à c_l. Ainsi, après plusieurs itérations, comme les deux configurations courantes sont les mêmes, la position c_k est atteinte.

□

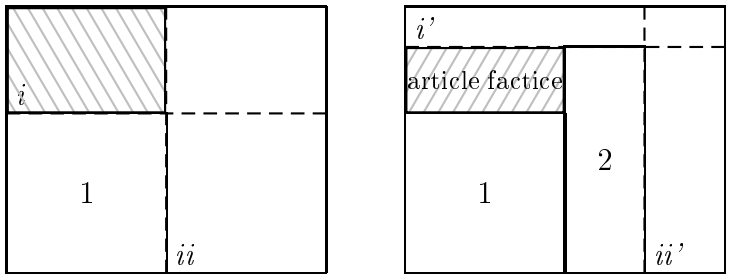


Figure 4.6 – Méthode LMAO

La Figure 4.6 illustre les deux principes de $LMAO$. La situation gauche apparaît si aucun article n'est placé dans la position active dominante i : c'est la position ii qui devient dominante. La situation droite apparaît si l'article 2 est rangé en position ii. Un article factice est inséré dans la surface située à gauche de l'article 2. Une nouvelle position active i' est créée.

$LMAO$ et MV énumèrent le même ensemble de configurations. Cependant, de nombreuses redondances sont évitées car toute configuration ne peut être atteinte que par un seul ordre $LMAO$. Quand une position c devient inactive, nous plaçons directement un article factice de hauteur 1 à la position c. En effet, comme la taille des articles est entière, quel que soit l'article qui va entrer en intersection avec c, la surface perdue sera au moins de hauteur 1.

4.4.2 Algorithme en deux phases : $TSBP$

L'objectif de notre approche est d'agréger dans un premier temps un grand nombre de configurations afin de montrer qu'un ensemble complet de configurations ne sont pas réalisables. A cette fin, nous proposons une méthode en deux phases composée de deux pse. La *pse extérieure* détermine les abscisses des articles. Les contraintes liées aux ordonnées des articles sont levées (*i.e.* plusieurs bandes horizontales d'un même article peuvent ne pas être contiguës). Pour chaque solution trouvée dans la première étape, une *pse intérieure* est lancée. Cette nouvelle méthode détermine les ordonnées des articles en respectant les abscisses déterminées précédemment. Chaque configuration de la pse extérieure correspond à un ensemble de configurations pour $2BP$. Ainsi, lorsqu'une coupe est réalisée dans l'arbre de

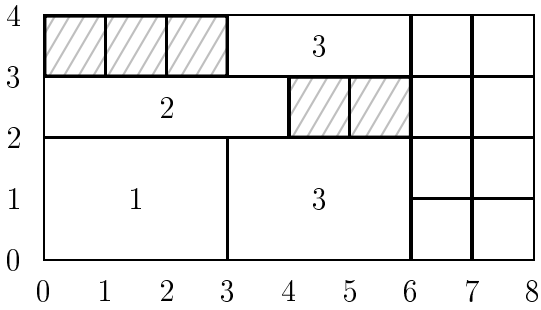


Figure 4.7 – PSE extérieure

recherche, un grand nombre de configurations non réalisables ne sont pas énumérées. Nous notons la méthode obtenue *TSBP (Procédure de séparation en deux étapes)*. Nous décrivons maintenant précisément les deux étapes de l'algorithme.

La pse extérieure détermine les abscisses des articles de la gauche vers la droite. Durant cette étape, les contraintes liées à la hauteur des articles ne sont pas prises en compte. Plusieurs bandes horizontales d'un même article peuvent être placées à des ordonnées non contiguës. Plusieurs contraintes sont conservées : la hauteur totale des articles rangés à une abscisse donnée doit être inférieure à la hauteur H du bin, et les bandes horizontales d'un même article doivent être rangées à la même abscisse (*cf.* Figure 4.7).

Pour chaque abscisse $x < W$, nous notons $K(x)$ la hauteur des bandes horizontales placées à cette position. Ces bandes correspondent à des articles placés en position x, ou en position $x_i < x$, $x_i + w_i > x$.

Avant la première étape, aucun article n'est rangé dans le bin. $K(x) = 0$ pour tout x. Lorsqu'un article a_i est rangé à l'abscisse 0, la valeur $K(x)$ est mise à jour pour $x = 0, \ldots, w_i - 1$ ($K(x) \leftarrow K(x) + h_i$).

A une étape donnée de la pse, l'algorithme énumère tous les ensembles a_j d'articles qui peuvent être placés à l'abscisse courante x. La valeur x est la plus petite pour laquelle $K(x) < H$. L'ensemble A_k doit respecter la contrainte suivante : $\sum_{a_i \in A_k} h_i \leq H - K(x)$. Par construction, les $K(x)$ sont décroissantes, la contrainte est donc satisfaite pour toutes les abscisses supérieures à x. Si pour une abscisse donnée x, nous avons $K(x) < H$ après avoir sélectionné a_j, un article factice de taille $(1, H - K(x))$ est créé de telle manière que $K(x) = H$.

En pratique, toutes les valeurs de $K(x)$ ne sont pas intéressantes. Les seules abscisses testées sont celles qui correspondent à un changement dans l'ensemble des articles présents (une abscisse x est intéressante s'il existe un article a_i tel que $x = x_i + w_i$). Lorsqu'un article factice est ajouté, sa longueur est égale à la différence entre la position courante et la première coordonnée dominante.

Dans la Figure 4.7, l'abscisse courante est 6. La première surface hachurée est un article factice qui a été ajouté pour remplir la configuration entre l'abscisse 0 et l'abscisse 3. Un tel article n'est pas nécessaire entre la position 3 et la position 4 car la somme des hauteurs des articles présents pour cette position est égale à celle du

bin. La représentation choisie est arbitraire : l'article 1 n'est placé ni au-dessus, ni au-dessous de l'article 2. Si à une étape donnée de l'algorithme un article dépasse l'extrémité droite du bin, ou si la surface totale des articles (articles factices compris) dépasse celle du bin, l'exploration de la branche courante est stoppée.

A chaque feuille de l'arbre de recherche, chaque article a une abscisse fixe. La liste des abscisses obtenues correspond à un ensemble de configurations, réalisables ou non. Une méthode est lancée pour déterminer s'il existe une configuration réalisable sous les contraintes ajoutées durant la pse extérieure.

Lemme 4.1. *La pse extérieure énumère toutes les configurations réalisables du problème relâché.*

Démonstration. Une configuration réalisable pour le problème relâché est associée à une liste d'articles rangés pour chaque abscisse. A chaque étape de l'algorithme, toutes les possibilités sont testées pour l'ensemble a_j des articles à ranger à l'abscisse courante x. Par conséquent, toute les configurations réalisables sont énumérées par l'algorithme. □

Théorème 4.2. *Si la méthode employée pour la phase intérieure est valide, $TSBP$ est valide.*

Démonstration. Supposons qu'une solution s est trouvée en utilisant uniquement la méthode intérieure. Si s est réalisable pour $2BP$, alors les abscisses correspondantes sont valides pour le problème relâché. D'après le Lemme 4.1, la configuration correspondante est énumérée par la pse extérieure et la méthode intérieure est lancée. □

Pour la pse intérieure, nous utilisons la méthode $LMAO$. Elle est modifiée de manière à ne rendre candidats au placement à la position (x, y) que les articles dont l'abscisse est fixée à x dans l'étape précédente.

4.5 Répétitions dans l'arbre extérieur

De par la séparation utilisée, la méthode en deux phases évite de nombreuses redondances, mais des méthodes peuvent être appliquées pour en réduire encore le nombre. Pour la méthode classique MV, Scheithauer [55] énumère plusieurs classes de redondances et propose des méthodes pour éviter ces répétitions. Pour la pse extérieure de $TSBP$, détecter les redondances est plus difficile. Nous proposons donc des relations de dominance pour nos configurations partielles. Deux principales redondances peuvent être évitées dans la recherche arborescente : une équivalence de *blocs* et une *pseudo-symétrie*. Dans les deux cas, nous proposons une classe de configurations dominantes qui seront effectivement énumérées durant la pse.

4.5.1 Equivalence de bloc

Nous considérons le cas où une configuration est composée de deux *blocs* indépendants.

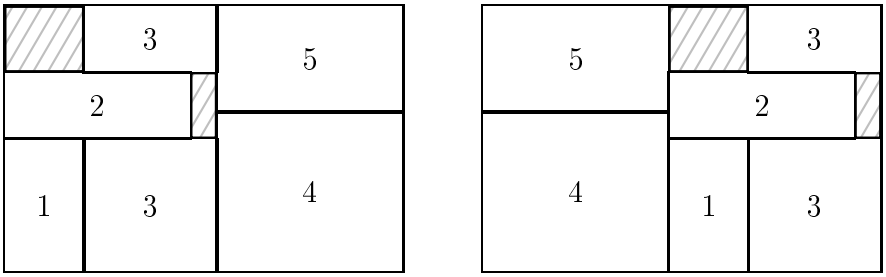


Figure 4.8 – Equivalence de blocs dans TSBP

Définition 4.2. *Une configuration pour l'ensemble d'articles A est dite composée de deux* blocs *indépendants s'il existe deux ensembles A_1 et A_2, $A = A_1 \cup A_2$, tels qu'aucune partie d'un élément de A_1 n'a la même abscisse qu'un élément de A_2. Pour deux blocs donnés A_1 et A_2, il suffit de vérifier la relation suivante.*

$$\exists A_1, A_2 \text{ tels que } A_1 \cup A_2 = A, \forall a_i \in A_1, \forall a_j \in A_2, x_i + w_i \leq x_j \text{ ou } x_j + w_j \leq x_i$$

La Figure 4.8 illustre l'équivalence entre deux configurations composées par les mêmes blocs. Dans ce cas, les deux blocs sont indépendants : échanger leur position conduit à une configuration équivalente. Nous n'énumérons qu'une seule des deux configurations. Pour cette raison, nous sélectionnons *a priori* un article a^*. Si cet article est dans le bloc de gauche, l'énumération continue. Dans le cas contraire, l'exploration du sous-arbre courant est stoppée. Cette proposition peut être généralisée si une configuration est composée de plus de deux blocs. Dans ce cas, une autre méthode doit être utilisée pour éviter d'explorer des solution équivalentes.

Afin de choisir une configuration parmi deux configurations équivalentes, nous introduisons une règle de dominance lexicographique.

Définition 4.3. *Soit a_i et a_j deux articles. $lbl(a_i) \succ lbl(a_j)$ si et seulement si une des propriétés suivantes est valide.*

1. $h_i > h_j$
2. $h_i = h_j$ *et* $w_i > w_j$

Définition 4.4. *Soit P_1 et P_2 deux blocs composés respectivement des ensembles A_1 et A_2. Soit a_k^l le l^{ieme} élément de A_k. Sans perte de généralité, nous supposons que les A_k^l sont triés par ordre décroissant de labels. $lbl(A_1) \succ lbl(A_2)$ si et seulement si*

1. $a_1^k = a_2^k$ *pour* $k = 1, \ldots, |A_1|$ *et* $|A_1| > |A_2|$
2. *il existe k tel que $lbl(a_1^k) \succ lbl(a_2^k)$ et pas de $m < k$ tel que $lbl(a_2^m) \succ lbl(a_1^k)$.*

Notez bien qu'avec cette définition, $lbl(A_k) \succ lbl(\emptyset)$ pour tout A_k.

Proposition 4.6. *Soit (A, B) un problème de faisabilité, et P une configuration composée de deux (ou plus) blocs indépendants $P_1, P_2, \ldots, P_l$. Si P est réalisable alors il existe une configuration réalisable équivalente P' composée des mêmes blocs triés par ordre décroissant de labels $lbl(P_k)$.*

Démonstration. Immédiat. □

Si deux blocs consécutifs ne sont pas triés par ordre décroissant de labels, le sous-arbre courant est élagué. Une méthode simple pour s'assurer que les blocs sont rangés par ordre décroissant de label à la fin d'un bloc P_1 est de vérifier qu'il n'existe pas d'article non rangé a_i tel que $lbl(a_i) \succ lbl(a_j)$ pour tous les articles a_j de P_1. Pour la configuration gauche de la Figure 4.8, dès que les abscisses de 1, 2 et 3 sont sélectionnées, l'énumération s'arrête. En effet, 4 étant le plus grand article, les blocs suivants ne pourront pas être rangés par ordre décroissant de $lbl(P_k)$.

4.5.2 Pseudo-symétrie

Dans la méthode classique, si deux configurations sont symétriques par rapport à l'axe des y, une seule est énumérée. Si cette configuration ne respecte pas la règle *leftmost downward*, elle n'est pas énumérée non plus.

Dans la pse extérieure, il n'existe pas réellement de symétrie axiale, car les ordonnées ne sont pas fixes. Une manière d'éviter l'exploration de configurations symétriques est de sélectionner deux articles a_1^* et a_2^* avant l'énumération, et de n'explorer que les solutions où $x_1^* \leq x_2^*$. Si a_2^* est rangé avant a_1^*, une coupe est réalisée. Comme nous nous intéressons à une pseudo-symétrie, un problème peut intervenir. Supposons que deux configurations pseudo-symétriques soient telles que $x_1^* > x_2^*$. Cela peut arriver à cause de la translation qui est appliquée après la symétrie. Dans ce cas, aucune des deux configurations n'est énumérée. Nous devons donc nous assurer que ce cas ne peut pas arriver. La méthode n'est appliquée que si $h_1^* + h_2^* > H$. En effet, si les articles sont translatés, leurs positions respectives ne peuvent être modifiées sans intersection des deux articles.

Proposition 4.7. *Soit (A, B) un problème de réalisabilité et a_1^*, a_2^* deux articles de A, $h_1^* + h_2^* > H$. S'il existe une configuration réalisable dominante pour A dans B, il existe une configuration réalisable dominante pour A dans B telle que $x_1^* \leq x_2^*$.*

Démonstration. Supposons qu'il existe une configuration réalisable P pour A dans B. Si $x_1^* \leq x_2^*$, la proposition est vraie. Si $x_1^* > x_2^*$, une configuration P' (éventuellement dominée) peut être obtenue en appliquant une symétrie axiale à P. Dans P', on a $x_1^* < x_2^*$. La configuration dominante correspondante P'' peut être obtenue en translatant les articles vers la gauche, puis vers le bas. Comme $h_1^* + h_2^* > H$, la translation ne peut pas changer la position relative de a_1^* et a_2^*. Donc P'' est une configuration réalisable et dominante telle que $x_1^* \leq x_2^*$. □

Notez bien que si a_1^* et a_2^* ne sont pas choisis tels que $lbl(a_2^*) \succ lbl(a_1^*)$, cette méthode peut être utilisée de concert avec la méthode de gestion des redondances de blocs. De plus, la pseudo-symétrie peut aussi apparaître au sein d'un même bloc. Dans ce cas, une procédure légèrement différente peut être appliquée. Les articles de référence a_i et a_j sont les deux articles de plus haut label lexicographique du bloc. La configuration retenue est celle où $x_i \leq x_j$.

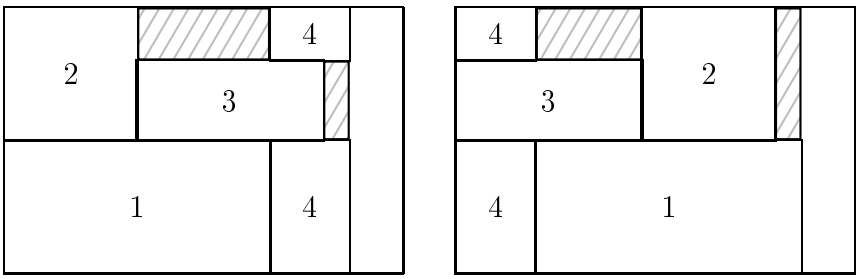


Figure 4.9 – Pseudo-symétrie dans TSBP

4.6 Résultats expérimentaux

Il existe de nombreuses instances dans la littérature qui permettent de tester les méthodes de découpe dans un rectangle unique. En règle générale, elles sont obtenues en partant d'un rectangle complet qui est récursivement découpé afin d'obtenir un ensemble de rectangles. On modifie alors un ou plusieurs rectangles lorsque l'on cherche à obtenir des instances non réalisables. Nous proposons ici de nouveaux jeux d'essai dédiés à ce problème. Certains n'ont pas de solution réalisable. Ces jeux de données sont générés dans des bins de taille $(20, 20)$ selon deux critères : le nombre d'articles n et la différence relative ε entre la surface du bin et la surface totale des articles. Par exemple, lorsque ε vaut 4, les articles occupent 96% de la surface du bin. Les instances conservées sont celles pour lesquelles nos méthodes approchées ne permettent pas de déterminer si elles sont réalisables ou non. La colonne *réal.* indique si l'instance est réalisable (F), ou non réalisable (N).

Les résultats obtenus montrent que $TSBP$ est beaucoup plus efficace que la méthode de Martello et Vigo [53] et est compétitive par rapport à la méthode de Fekete et Schepers [30]. Elle méthode permet de résoudre certains problèmes qui ne le sont pas par les deux autres méthodes. Nous avons étudié l'efficacité de $TSBP$ pour un grand nombre d'instances générées aléatoirement suivant deux paramètres : le nombre d'articles et la surface totale des articles. Nous avons aussi testé nos procédures de réduction, les méthodes de gestion des redondances, et nos bornes inférieures intégrées dans le processus énumératif. Les deux premières méthodes réduisent le temps de calcul, mais les bornes inférieures peuvent parfois se révéler trop coûteuse pour être efficaces. Tous les programmes ont été implantés en C sur un PC (Pentium IV 2,6 GHz). Pour chaque instance de problème, la limite de temps est fixée à 15 minutes.

4.6.1 Résultats obtenus par les méthodes énumératives

Le rapport entre la surface du bin et la surface totale des articles est un paramètre important pour la difficulté de l'instance. Nous notons ce paramètre ε, $\sum_{a_i \in A} w_i h_i = (1 - \frac{\varepsilon}{100})WH$. Si les articles couvrent totalement la surface du bin, une configuration non réalisable pourra être détectée plus rapidement à l'aide des bornes inférieures,

alors que si les articles ont une faible surface, une heuristique trouvera plus facilement une solution.

Afin de comparer l'efficacité de nos méthodes avec celle de Martello et Vigo et Fekete et Schepers, nous avons généré un ensemble d'instances *difficiles* avec des bins de taille $(20, 20)$. La difficulté d'une instance est déterminée par le paramètre ε et le nombre d'articles n. Nous analysons dans la suite de ce chapitre l'impact de ces paramètres sur la difficulté de l'instance. Nos bornes inférieures permettent de détecter un très grand nombre de configurations non réalisables. Nous n'avons gardé que les instances qui ne sont pas fermées directement à l'aide des bornes inférieures.

Pour chaque instance, nous rapportons les résultats pour les trois méthodes : MV pour la méthode de Martello et Vigo, $LMAO$ et $TSBP$ pour les méthodes exactes présentées dans ce chapitre (Tableau 4.1). Pour cette comparaison, les procédures de réduction sont lancées à la racine de l'énumération. En revanche, aucune méthode d'optimisation n'est lancée en cours de recherche. Pour chaque instance, nous précisons sa faisabilité si elle est connue (F et N, respectivement faisable et infaisable). Le symbole $-$ signifie que la méthode n'a pas été capable de déterminer la faisabilité de l'instance dans les 15 minutes autorisées.

$TSBP$ est beaucoup plus efficace que les autres méthodes, en particulier pour les instances non réalisables. En pratique, la pse extérieure est rarement lancée. Pour deux instances réalisables, $LMAO$ est meilleure que $TSBP$ (pour ces instances, une solution est trouvée dans les premières branches explorées de l'arbre). Cela s'explique par le fait que la méthode en deux temps doit lancer deux pse au lieu d'une seule. Pour les instances réalisables plus difficiles, $TSBP$ est plus efficace. Les résultats indiquent que pour être efficace, la méthode classique MV doit absolument être améliorée à l'aide de procédures de réduction des redondances. L'intérêt de cette comparaison est de montrer que pour $LMAO$ et $TSBP$, de nombreuses redondances sont évitées du fait même de la séparation.

4.6.2 Améliorer $TSBP$

Comme nous ne fixons que les abscisses dans la première étape de $TSBP$, le nombre de nœuds développés peut être différent si le role des abscisses et des ordonnées est échangé. Cette opération correspond à la rotation du problème d'un angle de $\frac{\pi}{2}$. L'instance obtenue est réalisable si et seulement si l'instance initiale l'était. Le problème consiste à déterminer quelle orientation va minimiser le temps de calcul. Nous utilisons la méthode suivante : pour les deux orientations du problème, nous calculons le nombre de nœuds générés dans le premier niveau de l'arbre. Cette valeur est égale au nombre d'ensembles a_j qui peuvent être rangés à l'abscisse 0. Cette règle n'est pas optimale, mais dans la plupart des cas, elle permet de trouver la meilleure orientation. En résumé, MV et $LMAO$ s'attaquent au problème initial, tandis que $TSBP$ résout dans plusieurs cas le problème obtenu après rotation du problème initial. Le temps nécessaire à la décision est inclus dans le temps de calcul de $TSBP$.

Notre méthode peut ne pas fonctionner efficacement lorsqu'il y a un sous-ensemble non réalisable de petite taille. Nous appliquons donc la méthode suivante : nous résolvons l'instance associée aux plus grands articles, et nous ajoutons les articles

Instance			Nœuds			Temps CPU		
ε	n	feas.	*MV*	*LMAO*	*TSBP*	*MV*	*LMAO*	*TSBP*
00	10	N	498940	914	386	6	0	0
00	15	N	44108432	6972444	45016	-	87	0
00	23	N	56882064	67408024	40569046	-	-	254
00	23	X	42608212	68155560	154997737	-	-	-
02	17	F	52758236	71765888	1706527	-	-	8
02	20	F	43113104	58100032	70690	-	-	0
02	22	F	1736591	58108864	108	28	-	0
02	20	N	53653556	73548088	171891273	-	-	-
03	10	N	4389743	198654	17122	55	2	0
03	15	N	54482588	79027208	22580963	-	-	90
03	16	N	46469712	68238856	24091255	-	-	106
03	17	N	41706968	73210184	147246675	-	-	720
03	18	N	43368960	5001083	2621993	-	67	12
04	15	F	48638400	66949008	4249	-	-	0
04	17	F	65952336	29908064	15937	-	377	0
04	19	F	57670816	5423174	25300	-	58	0
04	20	F	58401756	33	195	-	0	0
04	15	N	54766908	71919648	3340753	-	-	12
04	17	N	62933956	77090992	80888040	-	-	379
04	18	N	61351128	71299008	76571868	-	-	364
05	15	F	68788024	49042968	82098	-	523	0
05	18	F	59957500	73878872	4249532	-	-	29
05	20	F	23184128	68833	24836	254	1	0
05	15	N	63948032	84383344	13465274	-	-	50
05	17	N	51738224	78107640	44862814	-	-	209
05	15	X	63457040	80207048	211846864	-	-	-
07	15	F	65526928	20388888	822	-	219	0
07	10	N	5377856	424144	83795	63	3	0
07	15	N	57912524	83727840	12420952	-	-	53
07	15	X	60273792	79483544	221997738	-	-	-
08	15	F	67319104	21165528	50689	-	218	0
08	15	N	60148728	91504040	38186062	-	-	148
10	10	N	16777620	741677	27644	200	6	0
10	15	N	68136960	92775504	29091982	-	-	104
10	15	X	54992568	77875552	259205325	-	-	-
13	10	N	37400760	2606968	164016	411	22	0
13	15	N	87402296	4595959	451458	-	35	1
13	15	N	44129192	86488752	259296268	-	-	-
15	10	N	84001872	7191894	110200	-	59	0
15	15	N	70194208	88272600	6287316	-	-	22
20	15	F	41817	263837	81	1	3	0
20	15	X	50799084	72909992	7781	-	-	0

Tableau 4.1 – Comparaison entre les trois méthodes énumératives

un par un tant que l'instance obtenue est réalisable. Dans de nombreux cas, des ensembles de petite taille sont suffisants pour que l'instance ne soit pas réalisable. L'inconvénient de cette méthode est qu'elle augmente le temps de calcul pour les instances réalisables. Pour chaque instance, nous lançons cette méthode durant 60 secondes avant de chercher à résoudre le problème complet.

4.6.3 Analyse des jeux d'essai

Nous avons généré aléatoirement des jeux d'essai suivant le paramètre ε. Dans un premier temps, des valeurs dont la somme est égale à $(1 - \frac{\varepsilon}{100})WH$ sont tirées aléatoirement. Ces valeurs sont les surfaces des articles que l'on va construire dans un deuxième temps. On factorise ces valeurs pour obtenir la hauteur et la longueur de chaque article.

Le Tableau 4.2, la Figure 4.10 et la Figure 4.11 illustrent le fait que la valeur de ε influe sur la difficulté de l'instance. Pour $\varepsilon = 0\%$, aucune instance réalisable n'a été générée, tandis que pour $\varepsilon = 50\%$ et $\varepsilon = 75\%$, aucune instance non réalisable n'a été générée. Ces cas apparaissent dans le Tableau 4.2 comme *n.s.i* (pas d'instance de ce type). Pour chaque instance, le bin est de taille $(20, 20)$ (surface : 400) et le nombre d'articles n est égal à 15. Pour chaque valeur de ε, nous rapportons le nombre de nœuds développés en moyenne afin de déterminer la solution pour les 50 premières instances générées. La méthode utilisée est $TSBP$ après procédure de réduction et bornes inférieures. Aucune méthode d'optimisation n'est utilisée au cours de $TSBP$.

	Non réalisable		Réalisable		Réalisabilité		
ε	nodes	CPU time	nodes	CPU time	N	F	X
0	20102.34	0.22	n.s.i.	n.s.i	50	0	0
1%	448220.88	8.29	14104.00	0.00	49	1	0
2%	1782186.38	37.08	378055.94	7.08	37	13	0
3%	3310969.25	69.79	1929031.00	38.64	28	22	0
4%	7156912.00	147.28	865186.56	16.04	25	24	1
5%	4581232.00	96.27	1352387.50	25.42	11	38	1
7,5%	2192336.50	38.33	2566824.25	43.70	9	37	4
10%	0.00	0.00	503232.06	4.66	7	41	2
12,5%	0.00	0.00	62060.40	0.62	3	45	2
25%	0.00	0.00	216222.69	1.51	3	47	0
50%	n.s.i	n.s.i.	218606.52	1.41	0	49	1
75%	n.s.i	n.s.i.	580315.25	3.60	0	50	0

Tableau 4.2 – Difficulté des instances pour $n = 15$ avec différentes valeurs de ε

Les instances les plus difficiles sont générées pour $\varepsilon = 3\%$ ou $\varepsilon = 5\%$ pour les configurations non réalisables. Pour les configurations réalisables, les résultats sont moins nets, mais les instances les plus difficiles correspondent à des valeurs de ε comprises entre 3% et 7.5%. Quand $\varepsilon = 0$, les bornes inférieures fonctionnent extrêmement bien, et peu de pses sont réellement lancées. Il apparaît aussi que malgré un temps de calcul moyen inférieur pour les valeurs proches de 10%, le

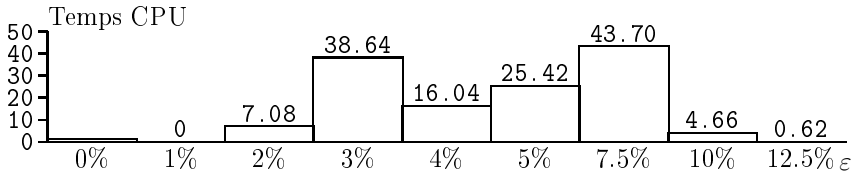


Figure 4.10 – Difficulté des instances réalisables pour $n = 15$

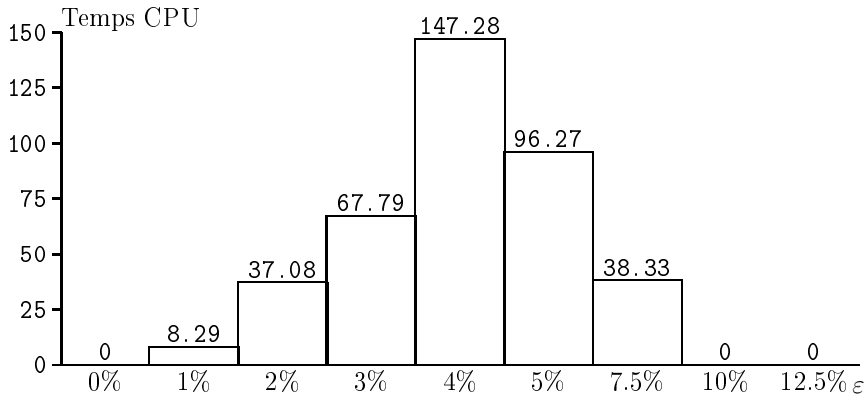


Figure 4.11 – Difficulté des instances non réalisables pour $n = 15$

nombre d'instances non résolues est plus grand. Pour de grandes valeurs de ε, le problème devient facile.

4.6.4 Optimisation de la méthode

Nous avons testé plusieurs techniques pour réduire le temps de calcul de la méthode exacte : les procédures de réduction décrites dans la Section 4.2 (rp), les méthodes pour éviter les redondances (rh), et nos bornes inférieures (lb). Lorsque les méthodes pour tester les redondances sont lancées, on vérifie aussi si chaque article restant peu être rangé dans les surfaces restantes. Les résultats de ces expérimentations sont résumés dans le Tableau 4.3. Les procédures de réduction sont utilisées à la racine en guise de prétraitement, et incluses dans les bornes inférieures. Lorsqu'une instance est modifiée à l'aide d'une (D)DFF, les procédures de réduction sont appliquées à l'instance obtenue.

Le temps de calcul est largement réduit pour plusieurs jeux d'essai. Les procédures de gestion des redondances dans l'arbre de recherche réduisent le nombre de nœuds énumérés pour un coût algorithmique faible, et réduisent donc le temps de calcul pour la plupart des instances. Le gain peut être très important pour certaines instances. Les bornes inférieures sont utiles pour réduire le nombre de nœuds, mais leur coût algorithmique peut se révéler important. Elles peuvent donc augmenter le temps nécessaire pour trouver une solution. Toutefois, le temps de calcul peut être réduit pour plusieurs instances, ce qui justifie l'usage de ces bornes.

Instance			Nœuds				Temps CPU			
ε	n	feas.	init	rp	rp+rh	rp+rh+lb	init	rp	rp+rh	rp+rh+lb
00	10	N	1	1	1	1	0	0	0	0
00	15	N	273008	181697	110943	96920	0	1	1	2
00	23	N	-	39240877	9342562	5968406	-	258	66	86
00	23	N	-	-	13383832	9057985	-	-	118	289
02	17	F	2969891	1814574	1171696	784796	23	9	6	12
02	20	F	608735	609161	487753	487230	10	14	12	12
02	22	F	175426	175426	175380	174943	1	4	4	4
02	20	N	1106	1	1	1	0	1	1	1
03	10	N	3100	3100	1222	417	0	0	0	0
03	15	N	5119	5119	4169	3707	0	1	1	1
03	16	N	4251766	4251766	1771831	1592400	15	17	9	32
03	17	N	242635	508053	343290	313007	0	3	2	4
03	18	F	3467378	3418946	2719210	2605815	16	18	17	22
04	15	F	8178	9053	6698	3949	0	1	1	1
04	17	F	2328156	2328156	2103026	1942682	24	27	25	26
04	19	F	2114390	2114390	1465752	1075159	8	11	8	7
04	20	F	6673	6673	6163	5876	0	3	3	3
04	15	N	63207	63207	43256	42844	0	1	1	1
04	17	N	13889	1	1	1	0	1	1	1
04	18	N	-	7315303	1689391	434824	-	36	11	7
05	15	F	503436	503436	382690	334434	2	3	2	3
05	18	F	62890632	62804148	40122815	20245458	300	293	220	126
05	20	F	39429	39429	39429	39387	0	2	2	2
05	15	N	61379	1	1	1	0	0	0	0
05	17	N	5393	2158	1429	993	0	1	1	1
05	15	N	158	1	1	1	0	0	0	0
07	15	F	232534	232464	129878	90219	0	1	1	1
07	10	N	5363	5206	1903	1758	0	0	0	0
07	15	N	1	1	1	1	0	0	0	0
07	15	N	1180	1180	677	651	0	1	1	1
08	15	F	32853635	32901221	27666296	22658934	130	128	130	117
08	15	N	303	414	267	261	0	1	1	1
10	10	N	153	1	1	1	0	0	0	0
10	15	N	1085	1	1	1	0	0	0	0
10	15	N	33545	33545	17618	17603	0	1	1	1
13	10	N	3846	3846	1889	1468	0	0	0	0
13	15	N	4233	93	91	91	0	0	0	0
13	15	N	262	1	1	1	0	0	0	0
15	10	N	621	621	331	331	0	0	0	0
15	15	N	4293	3396	1117	1117	0	0	0	0
20	15	F	867	747	747	747	0	1	1	1
20	15	F	4607934	4355764	4355764	4355492	34	34	36	44

Tableau 4.3 – Elaguer l'arbre de recherche

4.6.5 Comparaison avec la méthode de Fekete et Schepers

Dans le Tableau 4.4, nous comparons nos résultats avec ceux obtenus avec la méthode de Fekete et Schepers [30] [2]. Les deux méthodes se révèlent tout à fait complémentaires : elles se comportent en effet différemment en fonction du type de l'instance. La méthode utilisant le modèle de théorie des graphes est tenue en échec dans les 15 minutes accordées pour trois instances. Aucun résultat n'a été trouvé après une heure pour $\varepsilon = 0$ et $n = 23$, et 1200 seconds ont été nécessaires pour trouver une solution pour $\varepsilon = 2$, $n = 20$. Toutefois, la méthode de Fekete et Schepers conserve un grand intérêt, puisqu'elle est meilleure pour plusieurs instances ($\varepsilon = 7$, $n = 15$ et $\varepsilon = 13$, $n = 15$, par exemple). Le temps de calcul en chaque nœud est très grand, mais la taille de l'arborescence est plus faible dans de nombreux cas. Il apparait que le modèle utilisant la théorie des graphes se comporte bien lorsqu'un sous-ensemble d'articles de petite taille est suffisant pour que l'instance ne soit pas réalisable. Cela s'explique par le fait que la séparation de Fekete et Schepers considère des couples d'articles et peut ainsi concentrer la recherche sur les articles les plus contraints, alors que notre méthode doit traiter tous les articles, même les plus petits. Dans les cas où tous les articles sont nécessaires pour rendre l'instance non réalisable ($n = 23$ par exemple) notre modèle est très compétitif.

4.7 Conclusion

Dans ce chapitre, nous avons montré comment les bornes et les méthodes de réduction proposées dans le Chapitre 2 pouvaient être adaptées au problème de faisabilité. La contribution principale de ce chapitre est une nouvelle méthode exacte en deux phases pour ce problème. Les résultats obtenus par notre méthode sur des instances générées aléatoirement montrent l'intérêt de notre approche. En pratique, cette méthode permet de réduire considérablement le nombre de nœuds visités en comparaison des méthodes classiques de placement d'articles et se révèle compétitive par rapport à celle de Fekete et Schepers. Des méthodes pour réduire le temps de calcul sont discutées : la gestion des redondances est très efficace, mais l'utilisation des bornes inférieures peut parfois se révéler trop coûteuse.

2. Ces résultats ont été envoyés par J. Van der Ween et S. Fekete. Ils ont été obtenus par leurs soins à l'aide d'un PC équipé d'un processeur Pentium IV 3 Ghz.

Instance			Nœuds		Temps CPU	
ε	n	feas.	Graphe	TSBP	Graphe	TSBP
00	10	N	1	1	0	0
00	15	N	127	96920	0	2
00	23	N	-	5968406	-	**86**
00	23	N	-	9057985	-	**289**
02	17	F	28631	784796	**7**	12
02	20	F	-	487230	-	**12**
02	22	F	190617	174943	167	**4**
02	20	N	1	1	0	1
03	10	N	1	417	0	0
03	15	N	13	3707	0	1
03	16	N	9891	1592400	**2**	32
03	17	N	431	313007	0	4
03	18	F	574	2605815	**0**	22
04	15	F	933	3949	0	1
04	17	F	20270	1942682	**13**	26
04	19	F	786057	1075159	560	**7**
04	20	F	22796	5876	22	**3**
04	15	N	35	42844	0	1
04	17	N	1	1	0	1
04	18	N	24593	434824	10	**7**
05	15	F	1410	334434	0	3
05	18	F	262	20245458	**0**	126
05	20	F	547708	39387	491	**2**
05	15	N	1	1	0	0
05	17	N	1	993	0	1
05	15	N	18369	1	2	0
07	15	F	92	90219	0	1
07	10	N	17	1758	0	0
07	15	N	1	1	0	0
07	15	N	61	651	0	1
08	15	F	433	22658934	**0**	117
08	15	N	1	261	0	1
10	10	N	5	1	0	0
10	15	N	77	1	0	0
10	15	N	7	17603	0	1
13	10	N	17	1468	0	0
13	15	N	1	91	0	0
13	15	N	1	1	0	0
15	10	N	25	331	0	0
15	15	N	1	1117	0	0
20	15	F	325	747	0	1
20	15	F	36	4355492	**0**	44

Tableau 4.4 – Comparaison avec la méthode de Fekete et Schepers

Méthode exacte pour $2BP$

5.1 Introduction

Dans les chapitres précédents, nous avons proposé des prétraitements, des bornes inférieures et une méthode exacte pour le problème de faisabilité. Nous intégrons toute ces méthodes dans un algorithme exact pour $2BP$[1]. Il repose sur la décomposition itérative de l'ensemble des articles en deux sous-ensembles disjoints. Lorsqu'une bipartition est calculée, on cherche à résoudre les deux problèmes $2BP$ associés. Dans de nombreux cas, les bornes inférieure et supérieure permettent de résoudre ces problèmes. Pour une meilleure efficacité, l'initialisation des deux ensembles d'articles est importante. Nous proposons cinq manières d'initialiser les deux sous-ensembles, qui correspondent chacune à une propriété possible de l'ensemble des articles. Nous introduisons aussi une façon d'appliquer nos bornes inférieures à une solution partielle. L'idée principale est d'appliquer des bornes différentes sur les deux sous-ensembles. La prise en compte des articles qui ne sont pas encore affectés à un sous-ensemble constitue le principal apport de la méthode. Nos algorithmes sont testés à l'aide des jeux d'essai classiques de la littérature [5, 53].

Dans la Section 5.2, nous décrivons notre nouvelle méthode exacte. La Section 5.3 est consacrée aux bornes inférieures, et nos résultats expérimentaux sont présentés dans la Section 5.4.

5.2 Une nouvelles méthode exacte pour $2BP$

La méthode employée par Martello et Vigo [53] est décrite dans le Chapitre 1. Elle considére les articles par ordre décroissant de surfaces. A chaque étape i, l'affectation de l'article a_i à un bin contenant déjà des articles est testé, ainsi que la possibilité d'ouvrir un nouveau bin. Cette méthode construit des solutions rapidement lorsqu'elles sont simples. Néanmoins, elle ne permet pas un usage intensif des bornes inférieures, et elle génère un grand nombre de problèmes de faisabilité.

5.2.1 Décomposition du problème

La méthode de résolution que nous utilisons pour $2BP$ est une recherche dichotomique de la valeur d'une solution optimale. En pratique, les bornes sont proches, et

1. Le travail présenté dans ce chapitre correspond à l'article [21]. Il a aussi été présenté en congrès [20].

un ou deux tests sont souvent suffisants. Le cœur de l'algorithme est la méthode qui, pour un z donné, détermine s'il existe une solution qui utilise z bins pour l'instance considérée. Dans la suite, z fera toujours référence à cette valeur. L'idée consiste à calculer plusieurs bipartitions de l'ensemble A des articles, et à résoudre les deux sous-problèmes générés. Soient (A_1, A_2) une partition de A. La validité de notre méthode repose sur la proposition suivante.

Proposition 5.1. *Si pour deux entiers z_1 et z_2 tels que $z_1 + z_2 = z$, il n'existe pas de partition (A_1, A_2) de A telle que $OPT(A_1) = z_1$ et $OPT(A_2) = z_2$, il n'y a pas de solution pour A qui utilise z bins.*

A chaque étape de l'énumération (*cf.* Algorithme 2), on dispose de deux sous-ensembles A_1 et A_2, et d'un ensemble A_3 d'articles non-affectés. La méthode consiste à déterminer, pour chaque article de A_3, s'il est affecté à l'ensemble A_1 ou à l'ensemble A_2. Quand une partition est calculée, une solution pour chaque sous-problème est cherchée heuristiquement, ou de manière exacte si nécessaire.

Pour que la méthode soit la plus efficace possible, il convient d'initialiser les deux sous-ensembles A_1 et A_2 de telle manière que leur cardinalité soit la plus grande possible. Il faut pour cela détecter des sous-ensembles d'articles qui ne peuvent pas être affectés au même bin. De plus, si l'on veut connaître la valeur z_1, il faut déterminer le nombre de bins nécessaires pour ranger les articles de A_1. Dans le pire des cas, A_1 est initialisé avec un unique article, et A_2 avec l'ensemble vide. Dans ce cas, on fixe $z_1 = 1$, et la méthode obtenue équivaut à tester l'affectation de chaque article dans le premier bin. Soit z_1 le nombre de bins nécessaires pour ranger les articles de A_1 et z_2 le nombre de bins nécessaires pour ranger les articles de A_2. Si $z_1 = 1$ ou $z_2 = 1$, notre méthode dédiée au problème de faisabilité est utilisée. Dans le cas contraire, notre algorithme de résolution de $2BP$ est récursivement lancé sur les nouveaux problèmes générés. L'exploration d'un nœud de l'arbre est décrite dans l'Algorithme 2. Les problèmes associés aux ensembles A_1 et A_2 sont *NP-complets*, et la méthode exacte doit être utilisée récursivement. Nous proposons maintenant plusieurs initialisations possibles, en fonction des propriétés de l'ensemble des articles.

5.2.2 Initialisation de la méthode

La première initialisation est utilisée lorsque le nombre de grands articles est égal à z. S'il y a des grands articles, on utilise la deuxième initialisation. La troisième est utilisée lorsqu'il y a suffisamment d'articles longs et larges. La quatrième s'appuie sur un couple d'articles qui ne peuvent pas être rangés ensemble, et la cinquième initialisation est utilisée lorsque les autres méthodes ne peuvent être employées. Dans tous les cas, on cherche à favoriser l'obtention rapide de bornes inférieures. Les articles sont initialement triés par ordre décroissant de surface, et chaque initialisation possible est testée. L'initialisation s'arrête dès qu'une condition a été vérifiée.

5.2.2.1 Le nombre de grands articles est égal à z

Soit $A_{gr} = \{a_i \in A : w_i > \frac{W}{2}, h_i > \frac{H}{2}\}$ l'ensemble des grands articles. Si $|A_{gr}| = z$, nous prenons en compte le fait que $|A_{gr}|$ bins sont nécessaires pour ranger

Algorithme 2 : Un nœud de la recherche arborescente

Données :
A_1 : l'ensemble des articles affectés aux z_1 bins;
A_2 : l'ensemble des articles affectés aux z_2 bins;
A_3 : l'ensemble des articles non encore affectés;
z : nombre maximum de bins à utiliser;
// initialisation
si $A_1 = \emptyset$ **alors**
  Initialiser les ensembles A_1, A_2 et A_3 ($A_1 \cup A_2 \cup A_3 = A$) à l'aide de l'Algorithme 3;
  Calculer z_1 et z_2 ($z_1 + z_2 = z$);
si $A_3 = \emptyset$ **alors**
  // une partition est calculée
  si $z_1 \neq 1$ **alors**
    Résoudre le problème associé à A_1 en utilisant
    z_1 bins à l'aide de l'Algorithme 2;
    Si le problème n'a pas de solution, retourner FAUX;
  sinon
    Résoudre le problème de faisabilité associé à A_1;
    Si le problème n'a pas de solution, retourner FAUX;
  si $z_2 \neq 1$ **alors**
    Résoudre le problème associé à A_2 en utilisant z_2 bins à l'aide de l'Algorithme 2;
    Si le problème n'a pas de solution, retourner FAUX;
  sinon
    Résoudre le problème de faisabilité associé à A_2;
    Si le problème n'a pas de solution, retourner FAUX;
  retourner VRAI;
sinon
  // test des ensembles A_1 et A_2
  si $LB(A_1) > z_1$, *ou* $LB(A_2) > z_2$ **alors**
    retourner FAUX;
  pour chaque $a_i \in A_3$ **faire**
    si $LB(A_1 \cup \{a_i\}) > z_1$ **alors**
      $A_2 \leftarrow A_2 \cup \{a_i\}$;
    si $LB(A_2 \cup \{a_i\}) > z_2$ **alors**
      $A_1 \leftarrow A_1 \cup \{a_i\}$;
  // séparation
  Soit a_i le plus grand article de A_3;
  Affecter a_i à A_1, et utiliser l'Algorithme 2 pour les ensembles obtenus;
  Affecter a_i à A_2, et utiliser l'Algorithme 2 pour les ensembles obtenus;

ces articles. L'ensemble des grands articles est trié par surfaces décroissantes, et décomposé en deux sous-ensembles en fonction de leur indice. Les $\lceil \frac{|A_{gr}|}{2} \rceil$ premiers sont affectés à l'ensemble A_1, alors que les autres grands articles sont affectés à l'ensemble A_2. $z_1 = \lceil \frac{|A_{gr}|}{2} \rceil$ et $z_2 = z - z_1$. Quand cette initialisation est utilisée, elle peut l'être récursivement tant que l'ensemble considéré a une taille supérieure à un, et qu'aucune déduction ne peut être effectuée à l'aide des bornes inférieure et supérieure. Cette méthode de décomposition est choisie de manière à réduire la taille du plus grand problème restant. Une autre solution consiste à fixer une valeur arbitraire pour z_1. Le coût de cette initialisation est $O(n)$ si les articles sont triés de manière adéquate.

5.2.2.2 Grands articles

La deuxième méthode d'initialisation est appliquée lorsque l'ensemble des grands articles A_{gr} a sa cardinalité comprise entre $z-1$ et 1. Deux articles de A_{gr} ne peuvent pas être affectés au même bin. Ainsi, on peut décomposer les bins en deux sous-ensembles : ceux qui contiennent les articles de A_{gr}, et les $z - |A_{gr}|$ bins restants. A_1 est initialisé avec les articles de A_{gr}, et A_2 avec les articles qui ne peuvent être rangés avec aucun grand article. On a donc $z_1 = |A_{gr}|$ et $z_2 = z - z_1$. Initialement, A_2 peut être l'ensemble vide. Contrairement au cas précédent, où nous essayons d'équilibrer les deux sous-ensembles, nous générons deux sous-ensembles dont la taille peut différer. L'objectif est d'obtenir des ensembles plus contraints : A_1 est en effet plus contraint, car la surface totale des articles est très importante dès l'initialisation. Au cours de la méthode arborescente, de nombreux articles sont directement ajoutés à l'ensemble A_2 car il ne reste plus assez de place pour eux. Le coût de l'initialisation est $O(n^2)$.

5.2.2.3 Longs et hauts articles

Nous proposons maintenant une initialisation des ensembles A_1 et A_2 lorsque A ne contient aucun grand article. Soit (k, l) un couple d'entiers tels que $1 \leq l \leq \frac{W}{2}$ et $1 \leq k \leq \frac{H}{2}$. Nous définissons $A_{ht} = \{a_i \in A : h_i > H - k, w_i \geq l\}$ et $A_{lg} = \{a_i \in A : h_i \geq k, w_i > W - l\}$. Comme il n'y a plus de grands articles, $A_{ht} \cap A_{lg} = \emptyset$. Clairement, aucun article a_i de A_{ht} ne peut être rangé avec un article a_j de A_{lg}. Ainsi, si $LB()$ est une borne inférieure pour $2BP$, $LB(A_{ht}) + LB(A_{lg})$ est une borne inférieure valide pour A. Si $LB(A_{ht}) + LB(A_{lg}) = z$, on peut utiliser l'initialisation qui suit. A_1 est initialisé avec A_{ht} ($z_1 = LB(A_{ht})$), et A_2 avec A_{lg} ($z_2 = LB(A_{lg})$), et une décomposition peut être calculée. Le coût de cette initialisation est plus important que les autres : le nombre de valeurs intéressantes pour le couple (k, l) est borné par n^2, et le coût des bornes inférieures utilisées (celles du Chapitre 2 utilisant les fonctions f_0, f_1 et f_2) est $O(n^2)$. La complexité résultant de cette étape est $O(n^4)$. Si $LB(A_{ht}) + LB(A_{lg}) < z$, nous ne savons pas si les hauts (resp. longs) articles sont rangés dans les $LB(A_{ht})$ (resp. $LB(A_{lg})$) premiers bins ou s'ils vont être répartis dans plus de bins. Dans ce cas, nous n'utilisons pas la présente initialisation.

5.2.2.4 Deux articles incompatibles

Lorsqu'il existe deux articles a_i et a_j de A qui ne peuvent pas être affectés au même bin, deux bins sont nécessaires pour les ranger. L'ensemble A_1 est donc initialisé avec a_i et a_j, et l'ensemble A_2 avec l'ensemble vide (s'il n'y a plus de grands articles, aucun article ne peut être à la fois incompatible avec a_i et a_j). La valeur z_1 est fixée à 2, et z_2 à $z-2$. Le coût de cette étape est $O(n^2)$. Cette initialisation a été testée, mais elle conduit à de moins bons résultats que la suivante.

5.2.2.5 Lorsque les cas précédents ne sont pas vérifiés

Lorsque l'on n'est dans aucun des quatre cas présentés ci-dessus, nous utilisons la méthode suivante. A_1 est initialisé avec un article a_i de A, et A_2 avec l'ensemble vide. Nous donnons à z_1 la valeur 1 et à z_2 la valeur $z-1$. Lorsque notre méthode de décomposition est appliquée après cette initialisation, elle est équivalente au fait de choisir pour chaque article a_j si on l'affecte au premier bin ou non. L'avantage de cette approche repose dans l'usage plus fréquent des bornes inférieures et des prétraitements. En effet, un bin est fermé plus rapidement qu'il ne le serait en employant la méthode de Martello et Vigo [53]. Le coût de cette étape est $O(n)$.

5.2.3 Méthodes heuristiques

Lorsqu'une décomposition de l'ensemble A des articles est déterminée, des bornes inférieures et des heuristiques sont lancées pour évaluer le nombre de bins nécessaires pour les deux sous-problèmes créés. Les bornes inférieures sont celles qui sont décrites dans la Section 5.3. L'heuristique utilisée est une variante de l'algorithme *Bottom-Left Decreasing* où plusieurs ordres initiaux sont testés pour les articles : surface, hauteur, largeur, diagonale décroissantes ou un ordre aléatoire. Cette heuristique a été choisie pour son faible temps de calcul.

Un défaut de notre méthode réside dans le fait qu'elle doit gérer des petits articles qui peuvent être *superflus*, *i.e.* des articles qui ne changent pas la valeur de la solution optimale. Nous considérons donc, dans un premier temps, le problème correspondant à un sous-ensemble d'articles. S'il n'y a pas de solution pour ce problème, la méthode s'arrête. Si l'on trouve une solution, une heuristique est utilisée pour déterminer si elle peut amener à une solution pour le problème initial. Si l'heuristique ne parvient pas à ajouter les petits articles restants, on ajoute ces articles à l'instance et l'algorithme exact est relancé. Le point délicat est de choisir l'ensemble d'articles *intéressants*. Une manière de faire consiste à fixer a priori un seuil pour la surface des articles : si la surface d'un article est inférieure à ce seuil, il n'est pas considéré. Cette méthode ne prend pas en compte les propriétés de l'instance. Nous utilisons donc une autre méthode. A l'aide d'une heuristique simple, nous rangeons itérativement les articles par ordre décroissant de taille, tant qu'il est possible de le faire, en utilisant z bins. Lorsqu'on ne peut plus ajouter d'articles sans ouvrir un nouveau bin, l'heuristique s'arrête. A la fin de l'algortihme, un sous-ensemble *intéressant* est obtenu. Comme l'heuristique employée est simple, nous ajoutons deux articles supplémentaires dans la liste pour augmenter nos chances d'obtenir une instance non-réalisable, si cela est possible.

Algorithme 3 : Initialisation de l'ensemble des articles

Données :
A : l'ensemble des articles;
z : nombre de bins à utiliser;
Calculer A_{gr} l'ensemble des *grands* articles;
si $A_{gr} \neq \emptyset$ **alors**
  // il y a des grands articles
  si $|A_{gr}| = z$ **alors**
    Partitionner A_{gr} en deux sous-ensembles A^1_{gr} et A^2_{gr} de même taille;
    $A_1 \leftarrow A^1_{gr}$; $A_2 \leftarrow A^2_{gr}$;
    $z_1 \leftarrow |A^1_{gr}|$; $z_2 \leftarrow |A^2_{gr}|$;
  sinon
    Calculer l'ensemble Q_{large} des articles qui ne peuvent être rangés avec aucun article de A_{gr};
    $A_1 \leftarrow A_{gr}$; $A_2 \leftarrow Q_{large}$;
    $z_1 \leftarrow |A_{gr}|$; $z_2 \leftarrow z - z_1$;
sinon
  // il n'y a pas de grands articles
  Calculer A_{ht} et A_{lg};
  si $LB(A_{ht}) + LB(A_{lg}) = z$ **alors**
    $A_1 \leftarrow A_{ht}$; $A_2 \leftarrow A_{lg}$;
    $z_1 \leftarrow |A_{ht}|$; $z_2 \leftarrow |A_{lg}|$;
  sinon
    // rechercher deux articles non compatibles
    si $\exists a_i, a_j, w_i + w_j > W$ and $h_i + h_j > H$ **alors**
      $A_1 \leftarrow \{a_i, a_j\}$; $A_2 \leftarrow \emptyset$;
      $z_1 \leftarrow 2$; $z_2 \leftarrow z - 2$;
    sinon
      Soit a_i le plus grand article de A;
      $A_1 \leftarrow \{a_i\}$; $A_2 \leftarrow \emptyset$;
      $z_1 \leftarrow 1$; $z_2 \leftarrow z - 1$;

5.3 Bornes inférieures

Lorsqu'une partition d'un sous-ensemble d'articles est connu, les bornes inférieures décrites dans les Chapitres 2 et 3 peuvent être mises à jour. Nous gardons les mêmes notations que ci-dessus pour A_1, A_2 et A_3. Une première approche consiste à calculer de manière indépendante et à sommer les deux bornes inférieures obtenues pour A_1 et A_2. Elle retourne de bons résultats, mais peut être améliorée en prenant en compte les articles de A_3. Dans la suite de cette section, nous considérons uniquement le cas où $LB(A_1) + LB(A_2) \leq z$.

5.3.0.1 Une nouvelle borne inférieure

Dans un premier temps, nous définissons nos bornes pour $1BP$, avant de généraliser pour $2BP$. Soit C la taille du bin et A la liste d'articles à ranger. Une partition partielle est connue pour A (*i.e.* nous savons que chaque article de A_1 ne peut être rangé avec aucun article de A_2). Soit $A_3 = A \backslash (A_1 \cup A_2)$ l'ensemble d'articles restants. Pour simplifier les notations, nous notons $\widehat{f}(c_i) = \frac{f(c_i)}{f(C)}$ et $\widehat{g}(c_i) = \frac{g(c_i)}{g(C)}$. Pour deux DFF f et g, soit $LPART(f,g)$ la borne définie comme suit :

$$LPART(f,g) = \left\lceil \sum_{a_i \in A_1} \widehat{f}(c_i) + \sum_{a_i \in A_2} \widehat{g}(c_i) + \sum_{a_i \in A_3} min\left\{\widehat{f}(c_i), \widehat{g}(c_i)\right\} \right\rceil \tag{5.1}$$

Théorème 5.1. *$LPART(f,g)$ est une borne valide pour le nombre de bins en une dimension nécessaires pour ranger les articles de A lorsque les articles de A_1 ne peuvent être rangés avec les articles de A_2.*

Démonstration. Comme nous l'avons montré dans la proposition 5.1, la valeur optimale du problème peut être calculée en générant toutes les décompositions possibles et en déterminant la valeur optimale d'une solution pour les deux ensembles obtenus. Soit $LOPT(A_1, A_2)$ la valeur optimale d'une solution si deux articles de A_1 ne sont rangés avec aucun article de A_2. On a donc :

$$LOPT(A_1,A_2) \geq \min_{A'_3 \cup A''_3 = A_3} \left\{ \left\lceil \sum_{a_i \in A_1 \cup A'_3} \widehat{f}(c_i) \right\rceil + \left\lceil \sum_{a_i \in A_2 \cup A''_3} \widehat{g}(c_i) \right\rceil \right\} \tag{5.2}$$

$$LOPT(A_1,A_2) \geq \min_{A'_3 \cup A''_3 = A_3} \left\{ \left\lceil \sum_{a_i \in A_1 \cup A'_3} \widehat{f}(c_i) + \sum_{a_i \in A_2 \cup A''_3} \widehat{g}(c_i) \right\rceil \right\} \tag{5.3}$$

$$LOPT(A_1,A_2) \geq \min_{A'_3 \cup A''_3 = A_3} \left\{ \left\lceil \sum_{a_i \in A_1} \widehat{f}(c_i) + \sum_{a_i \in A_2} \widehat{g}(c_i) + \sum_{a_i \in A'_3 \cup A''_3} min\left\{\widehat{f}(c_i), \widehat{g}(c_i)\right\} \right\rceil \right\} \tag{5.4}$$

L'expression ci-dessus ne dépend pas de la décomposition de A_3. L'équation peut être simplifiée.

$$LOPT(A_1, A_2) \geq \left\lceil \sum_{a_i \in A_1} \widehat{f}(c_i) + \sum_{a_i \in A_2} \widehat{g}(c_i) + \sum_{a_i \in A_3} min\left\{\widehat{f}(c_i), \widehat{g}(c_i)\right\} \right\rceil \tag{5.5}$$

□

Un résultat similaire peut être obtenu avec la DDFF f_1, mais les articles de A_3 doivent être pris en compte lorsque les valeurs dépendantes de la donnée sont calculées. Si cela n'est pas fait, la fonction obtenue n'est pas une DDFF. L'ensemble J dans l'expression $M_C(X, J)$ doit inclure les articles de A_3 puisqu'ils peuvent être rangés avec les articles de A_1 ou A_2.

En appliquant la fonction $LPART(f, g)$ sur chaque dimension, la valeur obtenue peut être utilisée pour $2BP$. Nous notons cette borne L^{2CM}_{subs}. Le résultat suivant peut être montré de la même manière que le résultat précédent.

Théorème 5.2. *L^{2CM}_{subs} est une borne valide pour le nombre de bins en deux dimensions nécessaires lorsqu'aucun article de A_1 n'est rangé avec un article de A_2.*

5.3.0.2 Calculer la nouvelle borne inférieure

Le coût de cette borne peut être élevé. Si l'on n'utilise que les fonctions f_0, f_1 et f_2, il y a $O(n^2)$ combinaisons de fonctions f_0, f_1 et f_2 pour A_1 et le même nombre pour A_2 et donc $O(n^4)$ combinaisons au total pour les deux dimensions. La complexité totale est donc $O(n^5)$. Une manière d'améliorer le temps de calcul de cette borne est d'éviter de tester les combinaisons qui ne peuvent pas mener à de meilleures bornes. Pour ce faire, nous exploitons les deux propriétés ci-dessous.

Proposition 5.2. *Si $L_{(f)}(A_1 \cup A_3) \leq z_1$, il n'existe pas de fonction redondante g telle que $LPART(f, g)$ améliore la borne inférieure.*

Démonstration. Soit F un ensemble de fonctions redondantes.

$$\max_{g \in F}\{L_{(g)}(A_2)\} \leq z_2 \tag{5.6}$$

Si $L_{(f)}(A_1 \cup A_3) \leq z_1$, l'expression suivante est valide.

$$L_{(f)}(A_1 \cup A_3) + \max_{g \in F}\{L_{(g)}(A_2)\} \leq z_1 + z_2 = z \tag{5.7}$$

Comme $L_{(f)}(A_1 \cup A_3) + L_{(g)}(A_2)$ est une borne supérieure pour $LPART(f, g)$, la propriété est directement vérifiée. □

La proposition est valide si A_1 est remplacée par A_2.

Proposition 5.3. *Si $L_{(f)}(A_1 \cup A_3) + L_{(g)}(A_2) \leq z$, $LPART(f, g) \leq z$.*

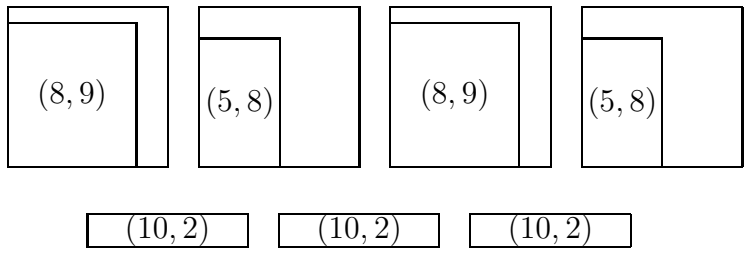


Figure 5.1 – Instance initiale

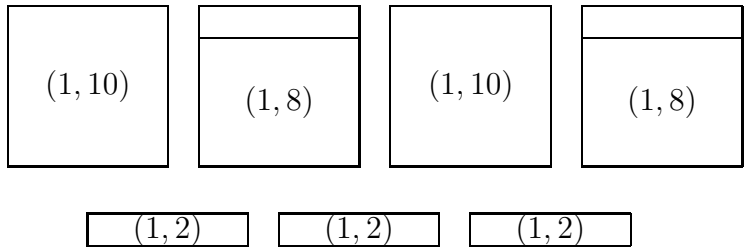


Figure 5.2 – Instance modifiée par les fonctions f_1

Démonstration. La valeur $L_{(f)}(A_1 \cup A_3) + L_{(g)}(A_2)$ est une borne supérieure pour $LPART(f, g)$. Donc, si elle ne permet pas d'atteindre une valeur plus grande que z, $LPART(f, g)$ ne peut pas le faire non plus. □

Nous donnons maintenant un exemple où les bornes sont améliorées par cette méthode. Soit $A = \{(8, 9), (8, 9), (5, 8), (5, 8), (10, 2), (10, 2), (10, 2)\}$ et $B = (10, 10)$. La valeur optimale pour A est 4 : deux bins pour les articles $(8, 9)$, un bin pour les deux articles $(5, 8)$ et un bin pour les trois articles $(10, 2)$. Supposons maintenant que la décomposition courante soit la suivante : $A_1 = \{(8, 9), (5, 8)\}$, $A_2 = \{(8, 9), (5, 8)\}$ et $A_3 = \{(10, 2), (10, 2), (10, 2)\}$. Elle est illustrée par la Figure 5.1. Cette décomposition ne peut mener à une solution en 4 bins, pourtant $OPT(A_1) + OPT(A_2) = 4$, donc aucune borne inférieure appliquée uniquement à ces deux ensembles ne peut être efficace. En appliquant la DDFF f_1^5 pour la hauteur et la largeur des deux sous-problèmes induits par $A_1 \cup A_3$ et $A_2 \cup A_3$, la borne inférieure obtenue par notre méthode est égale à 5 (voir Figure 5.2).

5.3.0.3 Améliorer les bornes

Lorsque $z = 2$, $LPART()$ peut être améliorée en exploitant l'idée suivante. Si deux articles a_j et a_k de A_3 ne peuvent pas être rangés dans le même bin, nous prenons en compte le fait que la même fonction ne peut pas être appliquée à ces deux articles. On peut généraliser cette observation pour des plus grandes valeurs de z en considérant l'ensemble des articles qui ne peuvent pas être affectés au même sous-ensemble que a_j dans une solution optimale. Soit $\overline{Q}^1_{\{a_j\}}$ (resp. $\overline{Q}^2_{\{a_j\}}$) l'ensemble des articles qui ne peuvent pas être affectés à l'ensemble A_1 (resp. A_2) dans une

solution optimale si a_j est affecté à cet ensemble. Nous discutons ultérieurement de la manière de calculer ces ensembles. Pour simplifier les notations, nous introduisons $A_4 = A_3 - (\{a_j\} \cup \overline{Q}^1_{\{a_j\}} \cup \overline{Q}^2_{\{a_j\}})$ l'ensemble des articles dont l'affectation ne dépend pas de celle de a_j. Une nouvelle borne peut être formulée de la manière suivante.

$$\begin{aligned} LPART'(f,g,a_j) = \Bigg\lceil & \sum_{a_i \in A_1} \widehat{f}(c_i) + \sum_{a_i \in A_2} \widehat{g}(c_i) + \sum_{a_i \in A_4} min\left\{\widehat{f}(c_i), \widehat{g}(c_i)\right\} \\ & + min\Bigg\{ \widehat{f}(c_j) + \sum_{a_i \in \overline{Q}^1_{a_j}} \widehat{g}(c_i) + \sum_{a_i \in \overline{Q}^2_{a_j}} min\left\{\widehat{f}(c_i), \widehat{g}(c_i)\right\}, \\ & \widehat{g}(c_j) + \sum_{a_i \in \overline{Q}^2_{a_j}} \widehat{f}(c_i) + \sum_{a_i \in \overline{Q}^1_{\{a_j\}}} min\left\{\widehat{f}(c_i), \widehat{g}(c_i)\right\} \Bigg\} \Bigg\rceil \end{aligned} \tag{5.8}$$

Proposition 5.4. *La valeur $LPART'(f,g,a_j)$ est une borne valide pour le nombre de bins nécessaires pour toute solution où aucun article de A_1 n'est rangé dans le même bin qu'un article de A_2.*

Démonstration.

$$LOPT(A_1,A_2) \geq \min_{A'_3 \cup A''_3 = A_3} \left\{ \left\lceil \sum_{a_i \in A_1 \cup A'_3} \widehat{f}(c_i) + \sum_{a_i \in A_2 \cup A''_3} \widehat{g}(c_i) \right\rceil \right\} \tag{5.9}$$

Si a_j est affecté à l'ensemble A_1, ni les articles de $\overline{Q}_{\{a_j\}}$, ni les articles de $\overline{Q}^1_{\{a_j\}}$ ne peuvent être affectés à l'ensemble A_1. La même observation peut être faite pour A_2. L'équation (5.9) peut donc être écrite de la manière suivante.

$$\begin{aligned} & LOPT(A_1,A_2) \geq \min_{A'_4 \cup A''_4 = A_4} \\ & \Bigg\{ \Bigg\lceil \sum_{a_i \in A_1 \cup \{a_j\}} \widehat{f}(c_i) + \sum_{a_i \in A_2 \cup \overline{Q}^1_{\{a_j\}}} \widehat{g}(c_i) + \sum_{a_i \in \overline{Q}^2_{\{a_j\}} \cup A'_4 \cup A''_4} min\{\widehat{f}(c_i), \widehat{g}(c_i)\} \Bigg\rceil, \\ & \Bigg\lceil \sum_{a_i \in A_1 \cup \overline{Q}^2_{\{a_j\}}} \widehat{f}(c_i) + \sum_{a_i \in A_2 \cup \{a_j\}} \widehat{g}(c_i) + \sum_{a_i \in \overline{Q}^1_{\{a_j\}} \cup A'_4 \cup A''_4} min\{\widehat{f}(c_i), \widehat{g}(c_i)\} \Bigg\rceil \Bigg\} \end{aligned} \tag{5.10}$$

Clairement, la fonction appliquée aux articles de A_1 et A_2 est la même dans les deux cas, et la valeur obtenue est indépendante de la décomposition choisie pour A_4. En simplifiant l'équation (5.10), nous obtenons la borne décrite dans l'énoncé de la proposition.

□

Lorsque $z = 2$, les ensembles $\overline{Q}^1_{a_i}$ et $\overline{Q}^2_{a_i}$ sont calculés de la manière suivante : pour chaque article a_j, nous vérifions si $w_i + w_j \leq W$ ou $h_i + h_j \leq H$. Si ce n'est pas le cas, il est ajouté aux deux ensembles. Une autre méthode est utilisée lorsque z est plus grand que trois. L'idée est de prendre en compte la surface des articles non affectés. Dans un premier temps, la surface totale R_{A_1} (resp. R_{A_2}) des articles de A_1 (resp. A_2) est calculée. Dans un deuxième temps, nous vérifions si les conditions suivantes sont respectées.

$$R_{A_1} + w_i h_i + w_j h_j > WH \times z_1 \tag{5.11}$$
$$R_{A_2} + w_i h_i + w_j h_j > WH \times z_2 \tag{5.12}$$

Si les Equations (5.11) et (5.12) sont vérifiées, a_i et a_j ne peuvent être affectés au même ensemble d'articles. Donc a_j peut être ajouté à $\overline{Q}^1_{a_i}$ et à $\overline{Q}^2_{a_i}$. Cette méthode peut être améliorée en utilisant les surfaces obtenues après application d'une DFF, mais cela augmente le temps de calcul de la borne. Si une seule des deux Equations (5.11) et (5.12) est vérifiée, l'article considéré peut être ajouté à $\overline{Q}^1_{a_i}$ ou $\overline{Q}^2_{a_i}$.

5.4 Résultats expérimentaux

Nous avons testé notre méthode avec les jeux d'essai de la littérature [5, 53] décrits dans le Chapitre 2. Initialement, nous nous servons de la borne supérieure de Boschetti et Mingozzi [8]. La borne inférieure est L^{2CM}, et les procédures de réduction sont celles du Chapitre 2. Comme dans les chapitres précédents, les programmes ont été exécutés sur un PC Linux équipé d'un processeur Pentium IV 2.6 GHz. Dans le Tableau 5.1, nous comparons nos résultats (2CM) à ceux obtenus par la méthode de Martello et Vigo [53] (MV) [2]. Pour chaque méthode, nous rapportons pour chaque ensemble de 10 instances le nombre d'instances résolues (colonnes *Opt*). Nous rapportons le nombre d'instances pour lesquelles les bornes inférieures sont égales à la borne supérieure à la racine (LB). Le temps de calcul maximum pour une instance est de 15 minutes. A l'aide de notre méthode, toutes les instances de 20 articles sont résolues, ainsi que 95% des instances ayant au plus 40 articles, 86% des instances de 60 articles ou moins, 76% pour 80 articles et 72% pour 100 items. On note que plusieurs instances non résolues sont des problèmes de faisabilité (Classe VI par exemple). Dans ce cas, notre méthode ne peut évidemment pas améliorer les résultats. Néanmoins, elle est capable de trouver la solution optimale pour certaines instances à 100 articles, même si la borne supérieure et la borne inférieure ne sont pas égales à la racine.

Le Tableau 5.2 ne permet pas une comparaison équitable entre les deux principes de séparation. Notre méthode fonctionne mieux pour de nombreuses instances, mais la différence est souvent due à la borne inférieure utilisée à la racine. Nous rapportons donc dans le Tableau 5.2 les résultats obtenus pour les instances où les bornes inférieures sont différentes des bornes supérieures à la racine. La colonne *Num* est la

2. Pour cette comparaison, nous utilisons le code de David Pisinger disponible en ligne sur sa page personnelle et décrit dans [51] pour le problème en trois dimensions.

Problème			Opt		
Classe	$H \times W$	n	LB	MV	2CM
I	10×10	20	9	10	10
		40	7	10	10
		60	7	6	9
		80	10	4	10
		100	9	2	10
II	30×30	20	10	10	10
		40	10	9	10
		60	10	5	10
		80	10	1	10
		100	10	3	10
III	40×40	20	8	10	10
		40	7	9	9
		60	7	9	7
		80	7	5	7
		100	5	6	6
IV	100×100	20	10	10	10
		40	10	7	10
		60	8	2	8
		80	7	0	7
		100	9	2	9
V	100×100	20	10	10	10
		40	7	10	10
		60	8	9	10
		80	4	6	7
		100	3	4	4
VI	300×300	20	10	10	10
		40	7	5	7
		60	10	1	10
		80	10	2	10
		100	8	0	8
VII	100×100	20	10	10	10
		40	9	8	10
		60	7	2	8
		80	5	0	5
		100	7	0	7
$VIII$	100×100	20	10	10	10
		40	9	6	9
		60	8	2	8
		80	7	3	7
		100	4	1	7
IX	100×100	20	10	10	10
		40	10	10	10
		60	10	10	10
		80	10	10	10
		100	10	9	10
X	100×100	20	8	10	10
		40	8	10	10
		60	6	5	6
		80	3	1	3
		100	1	0	1

Tableau 5.1 – Nombre d'instances résolues par les deux méthodes exactes

valeur comprise entre 1 et 50 égale à l'indice de l'instance dans le fichier de donnée. Ainsi, chaque donnée est identifiée, ce qui permettra de comparer plus facilement des résultats futurs. Pour chaque instance, nous rapportons le nombre de nœuds visités et le temps de calcul nécessaire pour chaque méthode. Nous ne rapportons pas les résultats lorsqu'aucune des deux méthodes n'a trouvé de solution. Il existe 11 instances qui sont résolues par notre méthode quand la méthode MV échoue. En revanche, on peut trouver 4 instances qui sont résolues par MV mais pas par $2CM$. Le temps de calcul en chaque nœud de notre recherche est plus grand que pour MV, mais notre méthode explore un nombre beaucoup moins important d'états, ce qui permet d'obtenir un temps de calcul inférieur pour plusieurs jeux d'essai.

Notre méthode se comporte bien grâce à la qualité de nos évaluations par défaut, qui nous permettent de réaliser un grand nombre de coupes dans l'arbre de recherche. Un élement intéressant est le faible nombre de problèmes de faisabilité qui interviennent. Cela signifie que l'efficacité de la méthode n'est pas toujours due à notre algorithme de résolution de ce problème.

Nous comparons aussi les résultats obtenus par notre méthode avec et sans la borne $LPART()$ proposée dans ce chapitre. Les colonnes $2\tilde{C}M$ contiennent les résultats avec les bornes inférieures. Le temps de calcul de la borne est important, et le temps de calcul est souvent augmenté. En revanche, le nombre de nœuds visités est réduit, ce qui signifie que des bornes moins coûteuses pourraient améliorer les résultats grâce à notre nouveau schéma d'utilisation.

5.5 Conclusion

Dans ce chapitre, nous proposons une nouvelles méthode exacte pour $2BP$. Elle repose sur la décomposition de l'ensemble des articles en deux sous-ensembles et réduit le nombre de nœuds considérés. Elle améliore strictement des résultats sur les jeux d'essai de la littérature. Le principe de séparation proposé est utile pour plusieurs instances : il permet notamment d'éviter un grand nombre de problèmes de faisabilité. Il est à noter que la méthode repose avant tout sur la qualité des évaluations. Nous proposons aussi une nouvelle borne inférieure pour des solutions partielles qui permet de réduire le nombre d'états explorés, mais augmente le temps de calcul lorsque les bornes actuelles sont utilisées. L'exploitation des affectations d'articles au cours de la méthode pourrait nous permettre d'améliorer sensiblement les résultats. La méthode proposée peut tout à fait s'appliquer au problème où l'orientation des articles peut être modifiée, mais la qualité moindre des évaluations pour ce problème risquent de rendre la méthode moins efficace.

Problem			Nodes			CPU Time		
Class	Num.	n	MV	2CM	$2\tilde{C}M$	MV	2CM	$2\tilde{C}M$
I	3	20	71	15	15	0	0	0
	11	40	42337	13	13	0	1	1
	12	40	240332	3071	2981	6	7	12
	13	40	37525	3	3	0	0	0
	21	60	-	43	43	-	2	2
	26	60	31738	2993	1742	7	13	14
	49	100	-	3485	-	-	22	-
III	3	20	96	20	20	0	1	1
	6	20	138	17	17	0	0	0
	12	40	59	5630	4552	0	23	38
	20	40	102549	6243	6048	124	13	26
	22	60	707419	-	-	12	-	-
	26	60	145765	-	-	557	-	-
V	14	40	18386	271	206	1	2	2
	15	40	82	17	17	0	1	1
	19	40	96354	10068	9793	1	32	50
	26	60	-	21044	-	-	52	-
	30	60	5680	3	3	0	3	3
	33	80	-	74	73	-	2	3
	35	80	-	2103	7784	-	10	82
	36	80	634709	18	18	5	5	5
	40	80	49406	-	-	0	-	-
	47	100	7144737	-	-	71	-	-
	49	100	-	6182	6154	-	120	199
VII	13	20	-	395	395	-	3	4
	29	40	-	91836	90244	-	171	367
$VIII$	46	100	-	6227	3846	-	63	84
	47	100	-	12	12	-	16	17
	50	100	-	456	330	-	5	7
X	9	20	18	2358	2358	0	0	0
	10	20	108	97	97	5	1	1
	12	40	172	1320	1320	0	5	6
	16	40	214	177	168	1	2	2

Tableau 5.2 – Instances résolues par une des deux méthodes exactes

Conclusion

Dans cette thèse, nous avons étudié **le problème de bin-packing en deux dimensions** $2BP$ où les articles sont rectangulaires et la rotation n'est pas autorisée. Nous avons décrit une méthode exacte qui repose sur plusieurs types d'algorithmes : des prétraitements, des bornes inférieures et des méthodes exactes pour le problème de faisabilité, ainsi que pour le problème d'optimisation $2BP$.

Les **prétraitements** que nous avons proposés permettent de réduire la taille de nombreuses instances de référence. Ils s'expriment tous deux sous forme de *fonctions identiquement réalisables*. Le premier prétraitement repose sur l'analyse de l'instance formée par les articles *hauts* ou *longs* et ceux qui peuvent être rangés avec eux. Le deuxième étudie les *grands articles* et l'ensemble des *petits articles*. Dans les deux cas, nous avons montré qu'il était parfois possible de trouver une manière optimale de ranger ces articles indépendamment du reste de l'instance. Il est alors possible de les traiter séparément.

Toutes les **bornes inférieures** que nous avons rapportées reposent sur le concept de **fonctions dual-réalisables** (dépendantes ou non de la donnée). Nous avons introduit tout d'abord trois fonctions f_0^k, f_1^k et f_2^k, qui permettent d'obtenir des bornes qui dominent celles de la littérature, en théorie et en pratique. Pour améliorer ces résultats, nous avons comparé deux méthodes de génération de fonctions.

- Nous avons étudié la composition des trois fonctions f_0^k, f_1^k et f_2^k et montré que toutes les applications itératives de ces fonctions étaient équivalentes à une fonction de la forme $f_2^k \circ f_0^l$ ou $f_1^1 \circ f_2^k \circ f_0^l$. Il existe donc un ensemble dominant de fonctions dont la cardinalité est de l'ordre de n^2.
- Nous avons aussi utilisé le générateur de fonctions de Carlier et Néron [13]. Il calcule de manière énumérative des fonctions dual-réalisables *maximales* (MDFF). Nous avons montré que f_0^k et f_2^k étaient des MDFF, puis nous avons expliqué comment les MDFF pouvaient être incorporées dans nos évaluations par défaut.

Nous avons validé les deux méthodes sur les instances de référence. Il apparaît qu'elles améliorent toutes les deux les résultats précédents [7, 8]. La composition des fonctions générées par Carlier et Néron et de la fonction f_1^1 permet d'améliorer considérablement les résultats pour $2BP$.

Nous avons aussi traité le **problème de faisabilité**. Nous avons adapté nos prétraitements à ce problème, et décrit une méthode exacte qui permet de réduire considérablement le nombre d'états visités en comparaison à la méthode classique. Elle se décompose en deux étapes.

- La première phase consiste à relâcher des contraintes d'intégrité sur la hauteur des articles et à résoudre le problème obtenu en fixant les abscisses des articles.
- Pour chaque solution trouvée dans la première étape, nous cherchons une solution correspondante pour le problème de faisabilité en utilisant une amélioration de la méthode classique, guidée par les abscisses fixées lors de la première étape.

Nous avons aussi introduit des bornes inférieures et des règles de dominance qui permettent de réduire la taille de l'arbre de recherche.

- Nous avons adapté nos bornes inférieures aux solutions partielles en agrégeant les articles placés pour former des articles plus grands.
- Nous avons introduit une première règle de dominance pour la première phase de notre méthode. Elle repose sur la détection de *pseudo-symétries*.
- Nous avons introduit une deuxième règle de dominance qui repose sur la détection de *blocs* : des sous-ensembles d'articles qui forment un rectangle.

L'incorporation des bornes inférieures réduit le nombre d'états visités, mais leur coût algorithmique est trop important pour qu'elles soient utiles dans la plupart des cas. En revanche, les règles de dominance permettent de réduire considérablement le temps de calcul en pratique. La méthode obtenue permet de résoudre des instances de bonne taille.

Enfin, la **méthode exacte**, que nous avons proposée pour le problème d'optimisation $2BP$, est originale et permet de fermer de nouvelles instances de référence. Elle repose sur la décomposition récursive de l'ensemble des articles en deux sous-ensembles. En pratique, les deux problèmes obtenus sont souvent résolus à l'aide des méthodes approchées. On trouve ainsi la solution optimale pour des instances contenant un grand nombre d'articles. Nous avons adapté nos bornes inférieures à des solutions partielles. Elles permettent de réduire le nombre d'états explorés, mais leur temps de calcul est important.

L'ensemble de nos méthodes permet de fermer un grand nombre d'instances de référence. La qualité de nos évaluations par défaut est pour beaucoup dans les résultats obtenus. Pour résoudre les jeux d'essai restants, il nous faudrait améliorer l'efficacité des évaluations au sein de la méthode exacte.

Cette thèse ouvre plusieurs perspectives. Dans un premier temps, nous comptons approfondir l'étude des fonctions dual-réalisables pour améliorer les résultats, mais aussi le temps de calcul de nos évaluations. Il serait aussi intéressant de trouver de nouvelles procédures permettant d'intégrer davantage les évaluations dans la méthode exacte. Dans un deuxième temps, nous comptons adapter notre travail à des problèmes proches de celui que nous avons traité. On peut citer, parmi d'autres, $2BP$ où l'orientation des articles peut être modifiée, mais aussi le problème de coupe guillotine, celui à trois dimensions $3BP$, et de strip-packing $2SP$. Nous y travaillons actuellement.

Bibliographie

[1] B. S. Baker, E. G. Coffman., et R. L. Rivest. Orthogonal packing in two dimensions. *SIAM Journal on Computing*, 9 :846–855, 1980.

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

[3] S. Ben Messaoud. *Caractérisation, modélisation et algorithmes pour des problèmes de découpe guillotine.* PhD thesis, Université de Technologie de Troyes, 2004.

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

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

[6] M. Boschetti. New lower bounds for the three-dimensional finite bin packing problem. *Discrete Applied Mathematics*, 140 :241–258, 2004.

[7] M. 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.

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

[9] J.M. Bourjolly et V. Rebetz. An analysis of lower bounds procedures for the bin packing problem. *European Journal of Operational Research*, to appear.

[10] D. J. Brown. An improved *BL* lower bound. *Inf. Process. lett.*, 11 :37–39, 1980.

[11] J. Carlier, F. Clautiaux, et A. Moukrim. New reduction procedures and lower bounds for the two-dimensional bin-packing problem with fixed orientation. *Submitted to Computers and Operations Research*, 2005.

[12] J. Carlier et E. Néron. A new LP-based lower bound for the cumulative scheduling problem. *European Journal of Operational Research*, 127 :363–382, 2000.

[13] J. Carlier et E. Néron. Computing redundant resources for the RCPSP. In *Proceedings of the 8th International Workshop on Project Management and Scheduling, Valencia*, pages 92–95, 2002.

[14] H. Y. Chao, M. P. Harper, et R. W. Quong. A tight lower bound for optimal bin packing. *Operational Research Letters*, 18 :133–138, 1995.

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

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

[17] F. Clautiaux, J. Carlier, et A. Moukrim. Exact method for the two-dimensional orthogonal packing problem. *Submitted to European Journal of Operational Research*, 2004.

[18] F. Clautiaux, J. Carlier, et A. Moukrim. Le problème de bin-packing en deux dimensions. Méthode exacte pour le problème de réalisabilité. In *MOSIM 2004, Nantes, France*, 2004.

[19] F. Clautiaux, J. Carlier, et A. Moukrim. Pretreatments and lower bounds for the two-dimensional oriented bin-packing problem. In *Ninth International Conference on Project Management and Scheduling, PMS 2004, April 26-28, 2004, Nancy, France*, pages 311–314, 2004.

[20] F. Clautiaux, J. Carlier, et A. Moukrim. Méthode exacte pour un problème de bin-packing en deux dimensions. In *ROADEF 2005, Tours, France*, 2005.

[21] F. Clautiaux, J. Carlier, et A. Moukrim. A new exact method for the two-dimensional bin-packing problem with fixed orientation. *Submitted to Operations Research Letters*, 2005.

[22] E. G. Coffman, 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.

[23] E. G. Coffman, M. R. Garey, et D. S. Johnson. *Approximation Algorithms NP-Hard Problems*, chapter 2, pages 46–93. D. Hochbaum (ed.), PWS Publishing, Boston, 1996.

[24] E. G. Coffman, M. R. Garey, D. S. Johnson, et R. E. Tarjan. Performance bounds for level oriented two-dimensional packing algorithms. *SIAM Journal on Computing*, 9 :808–826, 1980.

[25] E. G. Coffman et P. W. Shor. Average-case analysis of cutting and packing in two dimensions. *European Journal of Operational Research*, 44 :134–144, 1990.

[26] K. A. Dowsland et W. B. Dowsland. Packing problems. *European Journal of Operational Research*, 56 :2–14, 1992.

[27] H. Dyckhoff. A typology of cutting and packing problems. *European Journal of Operational Research*, 44 :145–159, 1990.

[28] H. Dyckoff, G. Scheithauer, et J. Terno. *Cutting and Packing*, pages 393–413. Wiley, Chichester, 1997.

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

[30] S. Fekete et J. Schepers. An exact algorithm for higher-dimensional orthogonal packing. *Revised for Operations Research*, 2003.

[31] 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.

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

[33] M. R. Garey, R. L. Graham, D. S. Johnson, et A. C. Yao. Resource constrained scheduling as generalized bin packing. *J. Combinatorial Theory Ser. A*, 21 :257–298, 1976.

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

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

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

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

[38] I. Golan. Performance bounds for orthogonal oriented two-dimensional packing algorithms. *SIAM J. Comput.*, 10 :571–582, 1981.

[39] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New York, 1980.

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

[41] M. Haouari et A. Gharbi. Fast lifting procedures for the bin packing problem. *submitted*, 2004.

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

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

[44] P. Lemaire. *Rangement d'objets multiboîtes.* PhD thesis, Université Joseph Fourier (Grenoble I), 2004.

[45] N. Lesh, J. Marks, A. McMahon, et M. Mitzenmacher. Exhaustive approaches to 2D rectangular perfect packings. *Information Processing Letters*, 90 :7–14, 2004.

[46] A. Lodi. *Algorithms for Two-Dimensional Bin Packing and Assignment Problems.* PhD thesis, University of Bologna, Italy, 2000.

[47] A. Lodi, S. Martello, et M. Monaci. Two-dimensional packing problems : a survey. *European Journal of Operational Research*, 141 :241–252, 2002.

[48] 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.

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

[50] G. S. Lueker. Bin packing with items uniformly distributed over intervals [a,b]. In *Proc. of the 24th Annual Symposium on Foundations of Computer Science (FOCS 83)*, pages 289–297. IEEE Computer Society, 1983.

[51] S. Martello, D. Pisinger, et D. Vigo. The three-dimensional bin packing problem. *Operations Research*, 48 :256–267, 2000.

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

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

[54] M. Monaci. *Algorithms for Packing and Scheduling Problems.* PhD thesis, University of Bologna, Italy, 2001.

[55] G. Scheithauer. Equivalence and dominance for problems of optimal packing of rectangles. *Ricerca Operativa*, 83 :3–34, 1998.

[56] D. Sleator. A 2.5 times optimal algorithm for packing in two dimensional. *Inf. Process. Lett.*, 10 :37–40, 1980.

[57] G. Wäscher, H. Haussner, et H. Schumann. An improved typology of cutting and packing problems. In *First ESICUP Meeting, Lutherstadt Wittenberg, Germany, March 18-20 2004*, 2004.

[58] G. Zhang. A 3-approximation algorithm for two-dimensional bin packing. *Operations Research Letters*, 2004.

Titre :

Bornes inférieures et méthodes exactes pour le problème de bin packing en deux dimensions avec orientation fixe

Résumé :

Notre problème consiste à déterminer le nombre de grands rectangles identiques nécessaires pour ranger une liste de rectangles sans modifier leur orientation. Nous proposons des méthodes pour calculer des bornes inférieures pour ce problème, essentiellement basée sur le concept de fonctions dual-réalisables. Nous proposons aussi deux méthodes exactes de type énumératives. L'une permet de déterminer si un ensemble de rectangles peut être contenu dans un rectangle unique. Elle repose sur une nouvelle relaxation du problème. La deuxième méthode permet de résoudre le problème général de bin packing en deux dimensions. Elle calcule pour cela une décomposition itérative de l'ensemble des rectangles à placer.

Mots-clés :

problème de conditionnement (bin-packing) en deux dimensions, bornes inférieures, méthodes exactes

Title :

Lower bounds and exact methods for the two-dimensional bin packing problem with fixed orientation

Abstract:

Our problem consists in determining the number of large identical rectangles used to pack a list of rectangles with a fixed orientation. We propose methods to compute lower bounds, based on the concept of dual-feasible functions. We also propose two enumerative exact methods. One is devoted to the problem with one bin. It uses a new relaxation of the problem. The other solves the optimization problem by iteratively decomposing the set of rectangles.

Keyword :

two-dimensional bin packing, lower bounds, exact methods, branch&bound