N° d'ordre : 14/2007-M/MT

République Algérienne Démocratique et Populaire

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ DES SCIENCES ET DE LA TECHNOLOGIE HOUARI BOUMEDIENNE

FACULTÉ DES MATHÉMATIQUES

MEMOIRE

Présenté pour l'obtention du diplôme de MAGISTER

En : MATHEMATIQUES

Spécialité : Recherche Opérationnelle : Mathématiques Discrètes et Optimisation

Par : DAHMANI Isma

Thème

Assemblage Orthogonal Rectangulaire, Approche algorithmique

Soutenu publiquement, le 17/10/2007, devant le jury :

M. AIDER	Professeur	Président.
R. OUAFI	Maître de Conférences	Directeur de thèse.
M. BOUDHAR	Maître de Conférences	Examinateur.
M. MOULAI	Maître de Conférences	Examinateur.

REMERCIEMENTS

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

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

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

Mes remerciements s'adressent également à **Mr MOULAI Mustapha,** maitre de conférences à l'U.S.T.H.B et **Mr BOUDHAR Mourad**, maître de conférences à l'U.S.T.H.B pour leur gratitude et leur dévouement qui ont permis d'honorer le jury.

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

DEDICACES

À *ceux qui m'ont donné la vie, je leurs témoigne mon amour et ma reconnaissance,*

À *maman qui m'a appris à être patiente pour surmonter les difficultés,*

À *mon père qui m'a appris que vouloir c'est pouvoir**,***

À *mes sœurs et mon frère Ahmed*

À*, ma très chère grand-mère, sans oublier mes tantes et oncles*

A toutes les personnes qui m'ont connu et qui ont cru en moi.

A tous ceux qui m'ont soutenu de prés ou de loin

A mes ***AMIS***

Je dédie ce modeste travail.

***D&A* :** Problème de Découpe et d'Assemblage ;

PAR : Problème d'Assemblage Rectangulaire ;

PAC : Problème d'Assemblage Carré ;

PDSDD : Problème de Découpe de Stock à Deux Dimensions ;

PAOR : Problème d'Assemblage Orthogonal Rectangulaire ;

BFBB: Best First Branch and Bound;

BFBBLF: Best First Branch and Bound avec l'introduction de la nouvelle représentation de la Liste Fermée ;

***BFBBOL* :** Best First Branch and Bound avec l'introduction de l'Ordre Lexicographique ;

BFBBMD : Best First Branch and Bound avec l'introduction de la notion du Modèle Dominé ;

BFBBNBS : Best First Branch and Bound avec l'introduction de la Nouvelle Borne Supérieure;

BFBBA: Best First Branch and Bound Amélioré;

***RM (Opt/BS)* :** Le Rapport Moyenne entre la Solution Optimale et la Borne Supérieure (Alg1) ;

***RM (Opt/NBS)* :** Le Rapport Moyenne entre la Solution Optimale et la Nouvelle Borne Supérieure (Alg3) ;

***N.It.M* :** Nombre d'Itération Moyen ;

***T.M(s)* :** Temps Moyen d'exécution (par Second).

Résumé

Le problème d'assemblage rectangulaire consiste à regrouper des pièces de formes rectangulaires dans un rectangle final de surface minimale. Ce problème est une généralisation du problème de placement connu sous le nom de ***bin packing*** et possède de nombreuses applications intéressantes dans les processus de production de diverses industries, tels que : les placements de circuits intégrés en électronique, la conception des ouvrages en parpaing ou en briques, et autres dérivés dans l'industrie du ciment.

Le thème traité dans notre mémoire concerne la stratégie d'assemblage basée sur des constructions orthogonales dérivées des modèles de découpes guillotines. Parmi les méthodes de résolution dédiées au problème d'assemblage orthogonal rectangulaire se distingue un algorithme exact noté *BFBBORP* : *Best First Branch and Bound Orthogonal Rectangular Packing*. L'algorithme repose sur une méthode arborescente ascendante, en se basant sur trois stratégies : une borne supérieure initiale, une stratégie de branchement et une stratégie de coupe branche.

Nous avons développé des heuristiques originales largement inspirées de la méthode exacte BFBBORP, avec pour objectif principal la réduction du temps de calcul et d'espace mémoire. Nous dressons un bilan comparatif afin de mesurer l'efficacité des versions modifiées à partir des résultats des expériences numériques établies sur des instances de problèmes réels et fictifs. Ces résultats sont disséqués de manière à souligner l'effet d'amélioration porté par chaque variante.

Mots clés

Séparation et évaluation, optimisation combinatoire, heuristiques, assemblage rectangulaire.

INTRODUCTION GENERALE

L'optimisation combinatoire occupe une place très importante en recherche opérationnelle, en mathématiques discrètes et en informatique. Son importance se justifie par la grande difficulté des problèmes d'optimisation d'une part, et d'autre part, par de nombreuses applications pratiques pouvant être formulées sous la forme d'un problème d'optimisation combinatoire. Bien que les problèmes d'optimisation combinatoire soient souvent faciles à définir, ils sont généralement difficiles à résoudre. Parmi les plus rencontrés d'entre eux, on trouve : les problèmes de découpe et d'assemblage (*Cutting and Packing*) appelés *CP* par *Dyckhof* [**8**]. Ces derniers sont devenus une région de recherche de plus en plus active, car ils posséde de nombreuses applications dans la production industrielle, conditionnement, logistique, etc.

Dans notre travail, nous nous sommes intéressés à l'une des variantes du problème de découpe et d'assemblage, à savoir celui d'Assemblage Orthogonal Rectangulaire (*Orthogonal Rectangulaire Packing Problem*). Ce dernier consiste à regrouper des pièces de formes rectangulaires dans un rectangle final de surface minimale. Ce problème est une généralisation du problème de placement connu sous le nom de *Bin Packing* et possède de nombreuses applications intéressantes dans les processus de production de diverses industries, telles que : les placements de circuits intégrés en électronique, la conception des ouvrages en parpaing ou en briques et autres dérivés dans l'industrie du ciment.

La méthode de résolution traitée dans notre mémoire, appelée *« Best First Branch and Bound »*, est exacte. Elle est basée sur le principe du "meilleur d'abord" dans lequel la sélection d'un chemin à parcourir repose sur l'estimation de son coût futur

(*l'algorithme A**). Malgré son offre d'une solution optimale, elle nécessite de l'espace mémoire et un grand temps d'exécution notamment pour les instances de très grande taille.

Pour y remédier, nous proposons dans notre mémoire d'améliorer la méthode en introduisant la notion des modèles dominés, pour éliminer quelques uns qui sont d'une qualité moins bonne. Un ordre lexicographique est adapté à notre arbre de recherche pour éviter les modèles d'assemblage dupliqués. Afin de réduire le temps de calcul, nous avons amené une nouvelle présentation de la liste fermée. Enfin, nous avons introduit une nouvelle borne supérieure pour l'algorithme exact afin de réduire l'espace de recherche, ce qui a donné des résultats assez encourageants.

Notre mémoire est partagé en six chapitres organisés comme suit :

Dans le ***premier chapitre,*** nous définissons la structure générale des problèmes de découpe et d'assemblage ainsi que les différents types de variantes du problème.

Le ***second chapitre*** est consacré à la problématique d'assemblage rectangulaire avec la formulation générale de ce dernier. Nous présentons les différentes variantes du problème et nous introduisons la formulation du problème d'assemblage orthogonal rectangulaire (*PAOR*).

Aprés avoir décrit le problème d'assemblage rectangulaire, nous réservons l'essentiel du ***troisième chapitre*** à l'étude de la méthode exacte « *Best First Branch and Bound* » (*BFBB*) et à son implémentation sur un exemple illustratif.

Nous apporterons par la suite, dans le ***quatrième chapitre***, les différentes techniques et idées permettant d'améliorer l'algorithme exact *BFBB*. Elles consistent à diminuer les nœuds de l'arbre de recherche et contribuent par conséquent à accélérer le temps d'exécution. Ce qui nous amène à l'objet du ***cinquième chapitre,*** où sont exposées les nouvelles variantes de l'algorithme *BFBB* obtenues par l'introduction des ces nouvelles techniques. Cinq de celles-ci ont été développées à cet effet : technique d'élimination des Modèles Dominés **« *BFBBMD* »**, technique d'élimination des

Modèles Dupliqués **« *BFBBOL* »**, technique d'une meilleure représentation de la Liste Fermée **« *BFBBLF* »**, introduction d'une Nouvelle Borne supérieure **« *BFBBNB* »** et enfin, la variante combinée **« *BFBBA* »**.

Afin de mesurer la performance de chacune des approches développées, nous avons implémenté un logiciel pour l'algorithme *BFBB* et les cinq versions modifiées en langage de programmation Pascal sous environnement Borland Delphi 7. Dans le ***sixième chapitre***, une étude comparative a été effectuée, révélant l'amélioration de ces nouvelles variantes par rapport à celle de la méthode *BFBB*.

Chapitre 1

Les problèmes de découpe et d'assemblage

Sommaire

1.1. Introduction

Les problèmes de Découpe et d'Assemblage (D&A) (*Cutting and Packing problem C&P*) ont été posés pour la première fois dans les années 60 par *Kantorovitch* [**16**], qui s'intéressait à une modélisation cohérente des problèmes rencontrés dans l'industrie. Ils furent repris par *Gilmore* et *Gomory* en 1965 [**9**] pour la résolution de quelques variantes. Les problèmes (*D&A*) ont suscité beaucoup d'intérêt auprès des industries, en tant que moyen d'augmentation de la rentabilité tout en réduisant les coûts. Ils sont devenus à cet effet, une zone de recherche de plus en plus active. Dans le cas général, les problèmes *D&A* ont essentiellement une même structure logique qui consiste à assembler (*couper*) un groupe d'éléments représentant des formes géométriques (*rectangle, carré*,..) dans un ou plusieurs grands objets (*grand rectangle, palette, conteneur*,…). Dont le but est de réduire au minimum l'utilisation des grands objets et de satisfaire un certain nombre de contraintes [**8**].

Le processus de découpe ou d'assemblage réalise des modèles qui sont des combinaisons géométriques de petits éléments assignés aux grands objets. Les morceaux résiduels, c'est-à-dire les figures qui apparaissent dans le modèle mais qui n'appartiennent pas à l'ensemble des petits éléments, sont habituellement considérées comme pertes. Bien que tous les problèmes de (*D&A*) ont essentiellement la même structure logique, le problème de découpe consiste à découper un grand objet en de plus petites pièces alors que le problème d'assemblage (*placement, chargement, ...*) consiste à placer des petits éléments dans de grands objets.

Vu le rôle dominant des modèles et de leurs natures en tant que combinaisons géométriques, on peut dire que les (*D&A*) appartiennent à la zone des combinatoires géométriques [**8**].

1.2. Structure des problèmes de découpe et d'assemblage

Dans le cas général, les (*D&A*) sont structurés en deux groupes de données de base dont les éléments définissent des formes géométriques : un certain nombre de petits éléments et un certain nombre de grands objets. Les premiers doivent être assignés aux derniers, dont le but essentiel est de réduire l'usage des espaces tout en respectant un certain nombre de contraintes et de références géométriques.

1.3. Dualité entre les problèmes de découpe et d'assemblage

Le problème d'assemblage (*placement*) consiste à rechercher le meilleur amalgame au sens des objectifs du placement, alors que pour le problème de découpe, il s'agit de répandre à la demande. Rappelons qu'un amalgame est la manière de découper une unité d'une matière première. Le placement représente la partie la plus importante de la découpe. On ne peut résoudre efficacement le problème de découpe sans résoudre celui du placement. Ceci est dû à la dualité entre les deux problèmes qui est introduite par *Dyckhoff* [**8**]. Elle est justifiée comme suit :

- Le problème d'assemblage (*placement, chargement*) est caractérisé par le fait que les grands objets sont considérés comme un espace vide et utile (*des véhicules, voitures, palettes, conteneurs, coffres, …etc.*). L'assemblage des petits éléments dans ces grands objets peut également être vu comme étant une découpe de l'espace vide qui correspond au grand objet qui sera occupé par de petits éléments.

- Réciproquement, le problème de découpe d'un grand objet peut être vu comme étant assemblage de petits éléments correspondant à des espaces vides, qui occupent le grand objet.

Un exemple qui montre clairement cette dualité est celui de l'industrie du textile, où des patrons de papier représentant les pièces du vêtement à découper, sont placés sur le tissu pour trouver une utilisation matérielle optimale.

Ainsi, la forte relation existant entre les problèmes de (*D&A*) résulte de la dualité du matériel utilisé et de l'espace (*voir figure1.1*), c'est-à-dire entre le corps matériel et l'espace occupé par ce dernier.

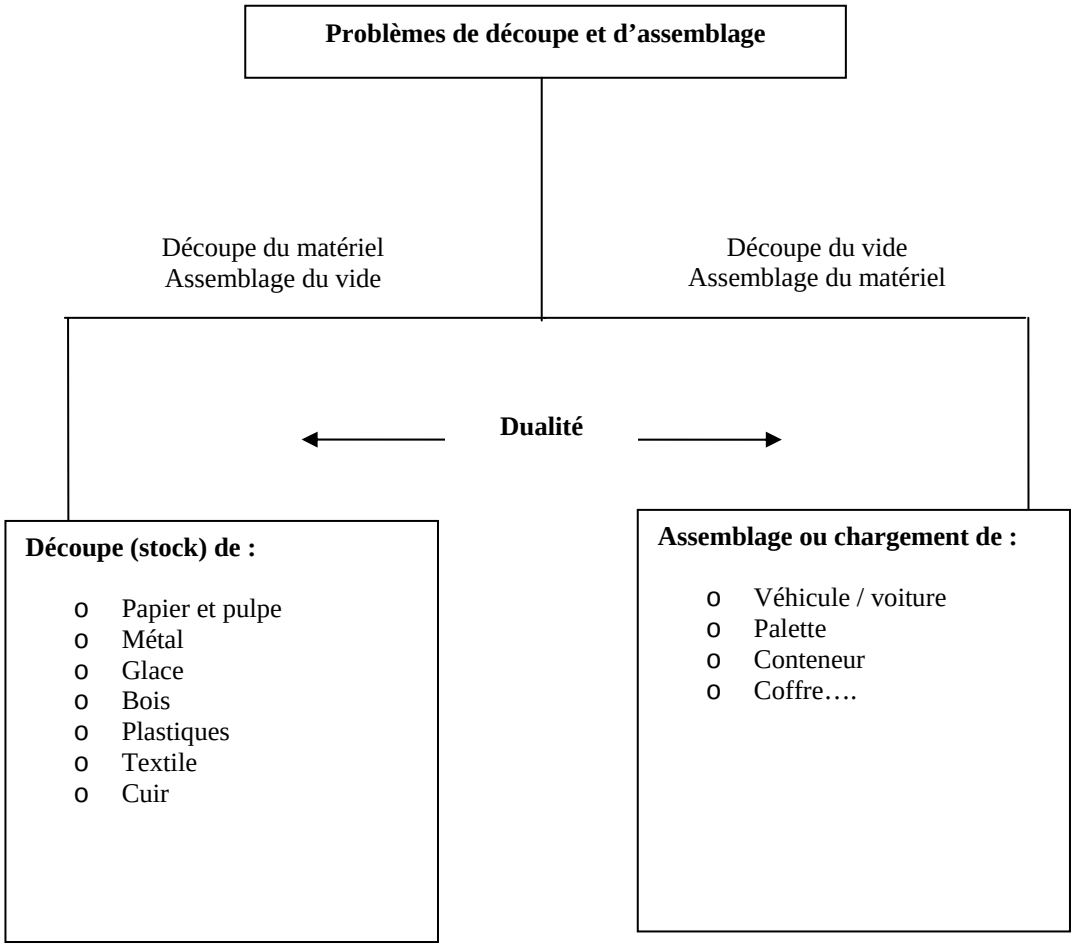


Figure 1.1 : Dualité du matériel utilisé et de l'espace vide

1.4. Critères de classification des problèmes d'assemblage

En 1990, *Dyckoff* [**8**] a proposé une typologie pour les problèmes de découpe et d'assemblage afin d'unir leurs multitudes. Cependant, cette première n'a pas été largement répandue vu les récents développements. Pour cette raison, une amélioration, proposée par *Wäscher, Haussner* et *Schumann* (2004) [**21**], s'est vue nécessaire. Pour définir les types de problèmes de *D&A*, ils ont posé les critères ci-dessous.

1.4.1. La dimension

Nous distinguons les problèmes à une, deux, trois dimensions ou plus ($n > 3$), sachant que les problèmes à dimension $n > 3$ sont rares et considérés comme variantes.

1.4.2. Genre de tâches

On présente deux situations de base :

- ***Maximiser le rendement :*** L'ensemble des grands objets n'est pas suffisant, donc les petits articles doivent être choisis pour maximiser le profit.

- ***Minimiser l'entrée :*** L'ensemble des grands objets est suffisant pour adopter tous les petits articles. Un ensemble de ces derniers doit être donc affecté à un ensemble de grands objets qui doit être minimisé.

1.4.3. Assortiment des petits articles

On distingue trois cas :

→ ***Petits articles identiques :*** ils sont de la même forme (*longueur, largeur et taille*).

→ ***Assortiment faiblement hétérogène :*** ils peuvent être groupés, relativement dans peu de classes (*par rapport au total des articles*).

→ ***Assortiment fortement hétérogène :*** ils sont caractérisés comme étant très différents (tailles et formes)

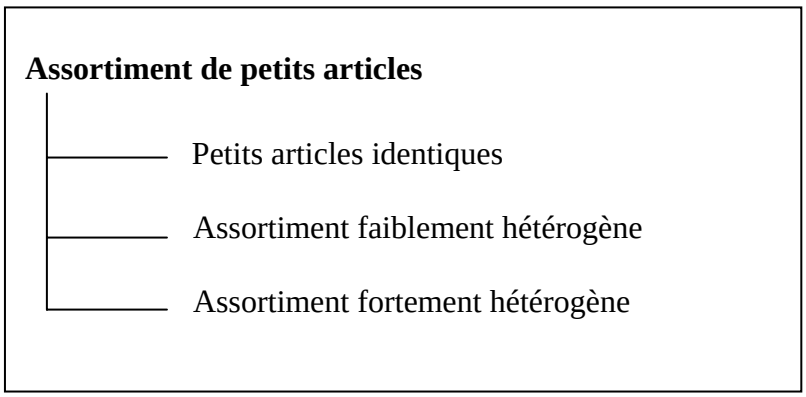


Figure 1.2** : **Les différents assortiments de petits articles

1.4.4. Assortiment de grands objets

On présente les cas suivants :

i. Un grand objet :

L'ensemble des grands objets est composé d'un élément unique. On distingue deux cas :

- Toutes les dimensions sont fixées.
- Une ou plusieurs dimensions sont variables.

ii. Plusieurs grands objets :

On distingue trois cas :

- grands objets identiques.
- grands objets faiblement hétérogènes : par analogie aux catégories d'assortiment des petits articles.
- grands objets fortement hétérogène : par analogie aux catégories d'assortiment des petits articles.

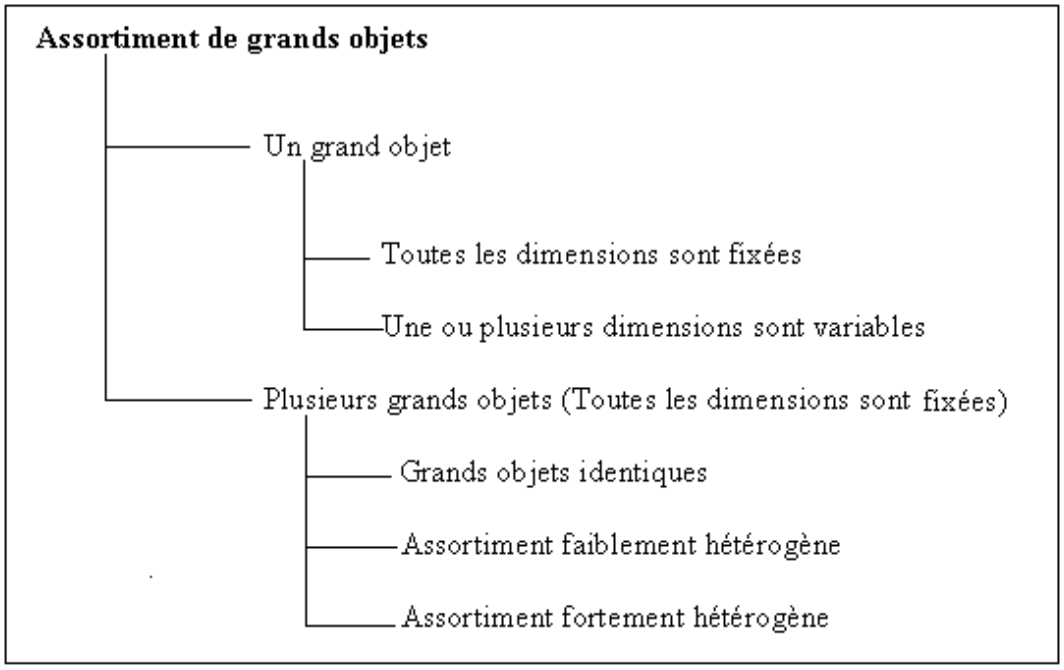


Figure 1.3 : Les différents assortiments de grands objets

1.4.4. Forme des petits articles

Dans le cas des problèmes combinés (à deux et trois dimensions) on distingue **les petits articles réguliers** (*rectangles, cercles, boites, cylindres, boules, etc*.) et **irréguliers** (*également appelés les non réguliers*).

1.5. Les différents types de problèmes de découpe et d'assemblage

En s'appuyant sur les critères cités ci-dessus, on distingue 3 types de problèmes :

1.5.1. Les problèmes de base

Les types de problèmes de base sont développés par la combinaison de deux critères « *genre de tâches assignées* » et « *assortiment de petits éléments* ». On distingue:

1.5.1.1. Problèmes de maximisation de sortie : Ce type de problème est caractérisé par le fait que les grands objets sont limités. Comme les petits articles doivent être maximisés, tous les grands objets seront utilisés.

1.5.1.2. Problèmes de minimisation d'entrée : Ce type de problèmes est caractérisé par le fait que l'approvisionnement en grands objets est assez grand pour l'adapter à tous les petits articles. La valeur des grands objets nécessaires pour adapter tous les petits articles doit être réduite au minimum.

Le schéma suivant montres les différents problèmes de base :

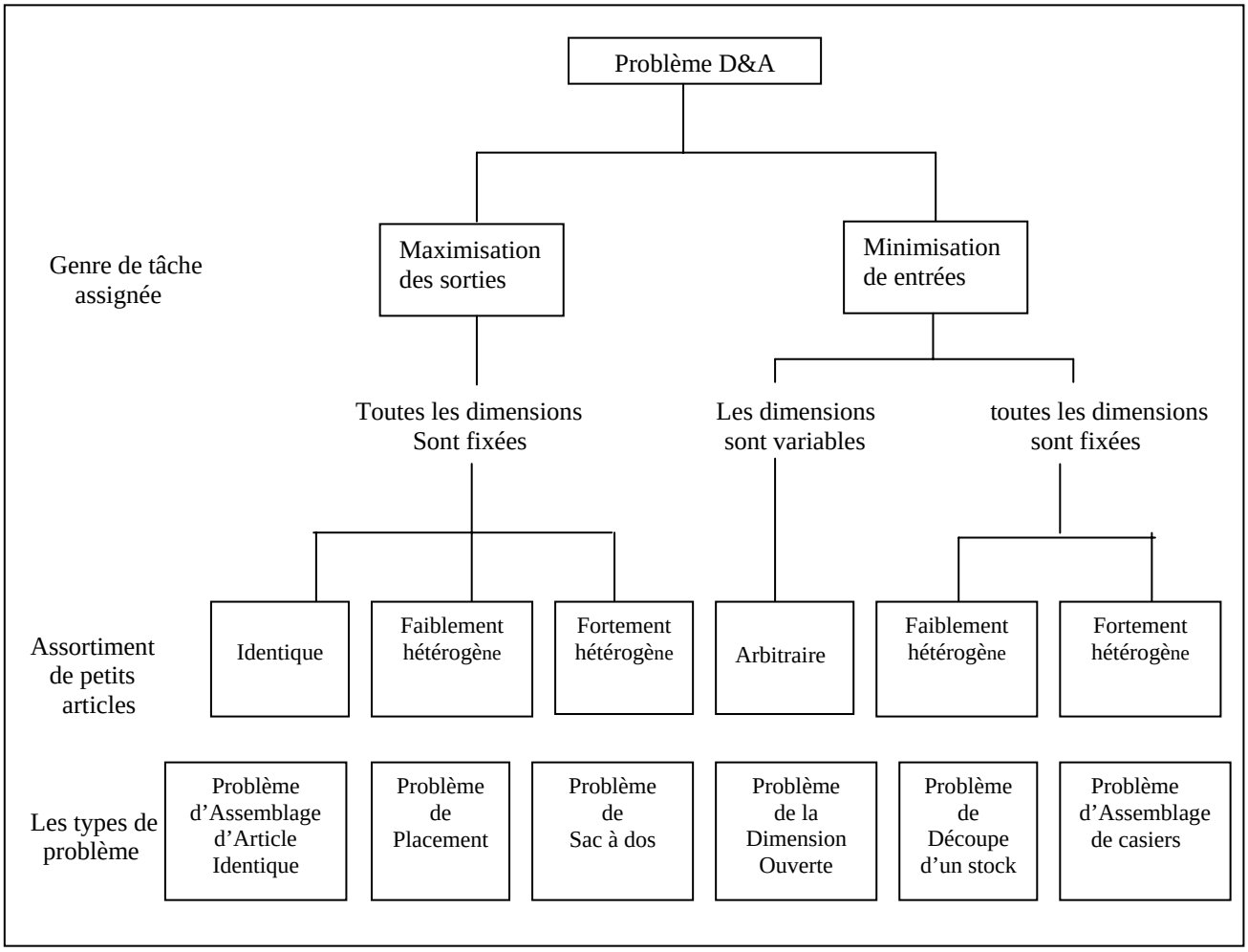


Figure 1.4: Les problèmes de base

1.5.2. Les problèmes intermédiaires

Ce type de problèmes se basant sur ce qui a été développés précédemment (*problèmes de base*), en tenant compte de l'assortiment des grands objets comme critère différent et supplémentaire. Ils sont résumés dans les deux tableaux qui suivent.

Classement des éléments / Petits caractéristiques des grands objets		Identique	Faiblement hétérogène	Fortement hétérogène
Toutes les dimensions fixées	un grand objet	Problème d'Assemblage d'Eléments Identiques IIPP	Problème de Placement d'Objet Grand Unique SLOPP	Problème de Sac à dos Unique SKP
	identique		Problème de Placement d'Objet Grand Identique Multiple MILOPP	Problème de Sac à dos Identique Multiple MIKP
	hétérogène		Problème de Placement d'Objet Grand Hétérogène Multiple MHLOPP	Problème de Sac à dos Hétérogène Multiple MHKP

Figue 1.5: Architecture des types de problème intermédiaires : maximisation de sortie

Classement Des petits éléments / Caractéristiques des objets grands		Faiblement hétérogène	Fortement hétérogène
Toutes les dimensions fixées	identique	Problème de découpe de taille unique SSSCSP	Problème d'assemblage de coffre de taille unique SBSBPP
	faiblement hétérogène	Problème de découpe de taille multiple MSSCSP	Problème d'assemblage de coffre de taille multiple MBSBPP
	fortement hétérogène	Problème de découpe résiduel RCSP	Problème d'assemblage de coffre résiduel RBPP
Dimension(s) variable(s) d'un grand objet		Problème de dimensions ouvertes	

Figure 1.6 : Architecture des types de problèmes intermédiaires : minimisation d'entrée

1.5.3. Les problèmes raffinés

Dans une étape finale, les problèmes de raffinage sont obtenus par l'application du critère « *Dimension* » (pour les problèmes à deux et trois dimensions) et du critère *« Forme »* des petits éléments. Les sous catégories résultantes sont caractérisées par des adjectifs qui sont ajoutés aux (*noms des*) types intermédiaires (*TPI*) selon le système suivant:

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

1.6. Optimisation Combinatoire

La **Combinatoire** désigne la discipline des mathématiques concernées par les structures " discrètes " ou " finies ". Citons quelques branches de cette discipline : *la théorie des graphes, la combinatoire énumérative, les problèmes de dénombrement, la combinatoire polyédrale, l'optimisation combinatoire*, etc. Les frontières entre les branches ne sont pas hermétiques, les différentes branches expriment plutôt des orientations méthodologiques différentes.

L'Optimisation, représente des termes utilisés pour recouvrir toutes les méthodes qui servent à déterminer l'optimum d'une fonction avec (ou sans) contraintes. Cette fonction modélise le choix optimal. Les ingénieurs, les économistes, la nature, font souvent des choix optimaux (ou quasi-optimaux), d'où l'importance de l'optimisation dans les sciences, qu'elles soient techniques, mathématiques, physiques, informatiques, économiques, naturelles, etc.

Un problème d'optimisation combinatoire consiste à trouver la *meilleure* solution dans un ensemble discret *dit ensemble des solutions réalisables*. En général, cet ensemble est fini mais de cardinalité très grande et il est décrit de manière implicite, c'est-à-dire par une liste, relativement courte de contraintes que les solutions réalisables doivent satisfaire.

Pour définir la notion de *meilleure solution*, une fonction, dite *fonction objectif*, est introduite. Pour chaque solution, elle renvoie un réel et la meilleure solution (ou *solution optimale*) est celle qui minimise ou maximise la fonction objectif. Clairement, un problème d'optimisation combinatoire peut avoir plusieurs solutions optimales.

1.6.1. Complexité

L'expérience montre que certains problèmes sont plus faciles que d'autres à résoudre sur un ordinateur. Une théorie de la complexité a été développée et permet mathématiquement de classer les problèmes faciles et difficiles en deux classes : la

classe P et la classe NP. Nous exposons dans ce qui suit, les grands principes de la théorie de la complexité des problèmes.

1.6.2. Les classe P, NP et NP- complet

Pour pouvoir exposer la notion de classes de problèmes, il est tout d'abord nécessaire de distinguer les problèmes de décision des problèmes d'optimisation. Un problème de décision est un problème pour lequel la réponse est « oui » ou « non ». Notons qu'il est possible d'associer à chaque problème d'optimisation, un problème de décision en introduisant un seuil *k* correspondant à la fonction objectif *f.* Le problème de décision devient : « existe-t-il une solution réalisable (S) telle que $f(S) \leq k$ (ou $f(S) \geq k$) *? ».*

Il et alors possible de définir la classe P (Polynomial) qui regroupe les problèmes de décision résolus par des algorithmes polynomiaux. Un algorithme polynomial est défini comme un algorithme dont le temps d'exécution est en O(p(x)) ou *p* est un polynôme et x est la longueur d'entrée (c.à.d. le nombre de données) d'une instance du problème.

La classe NP (*Non deterministic Polynomial*) regroupe les problèmes qui peuvent être résolus en temps polynomial par des algorithmes *non déterministes* (un algorithme est dit *non déterministe* s'il comporte des instructions de choix). Pour ces algorithmes, si à chaque instruction, le bon choix est effectué, le temps de calcul est polynomial. Si au contraire tous les choix sont énumérés, l'algorithme devient déterministe et son temps de calcul devient exponentiel. De façon informelle, un problème de décision appartient à NP si on peut vérifier en un temps polynomial si une solution « potentielle » donnée satisfait la question posée.

Les algorithmes polynomiaux sont évidemment des cas particuliers des algorithmes non déterministes. Aussi tout problème de décision qui peut être résolu par un algorithme polynomial, et qui appartient donc à la classe P, appartient également à la classe NP. D'où $P \subseteq NP$.

Parmi les problèmes de la classe NP, une large classe de problèmes, les problèmes NP-complets, sont équivalents entre eux quant à l'existence d'un algorithme polynomial pour les résoudre. C'est-à-dire, s'il existe un pour chaque problème de la classe NP. Afin de décrire cette classe d'équivalence, définissons tout d'abord la transformation (réduction) polynomiale entre deux problèmes. Soient D1 et D2, deux problèmes de décision. La transformation polynomiale de D1 vers D2 (noté D1 ∞ D2) peut être vue comme une fonction *f* de l'ensemble des instances de D1 vers l'ensemble des instances de D2 qui satisfait aux deux conditions suivantes :

1. *f* est calculable par un algorithme polynomial.
2. Pour toute instance I de D1, I a pour réponse « oui » (pour D1) si et seulement si *f* (I) a pour réponse « oui » (pour D2).

D'une manière équivalente, D1 se transforme (réduit) polynomialement à D2, s'il existe un algorithme de résolution de D1, qui fait appel à un algorithme de résolution de D2, et qui est polynomial lorsque la résolution de D2 est comptabilisée comme une opération élémentaire.

On dit alors qu'un problème de décision est *NP-complet* si tout problème de la classe NP se transforme polynomialement à lui.

Ainsi, il est nécessaire de connaître des problèmes connus pour être NP-complets. Le premier problème qui a été prouvé comme étant NP-complet, par S.A. Cook en 1970, est le problème de satisfiabilité (SAT) qui peut être défini comme suit : étant donné une expression booléenne sous forme normale conjonctive (de la forme de conjonctions de disjonctions), existe-t-il une affectation en « vrai » et « faux » de ses variables de manière à ce que l'expression booléenne prenne la valeur « vrai » ? Dans le cas où la réponse est oui, l'expression sera dite *satisfiable*.

1.6.3. Prouver la NP-complétude d'un problème

La démonstration d'appartenance d'un problème de décision à la classe des problèmes NP-complets est très importante puisque, une fois cette démonstration faite, on peut considérer comme peu vraisemblable l'existence d'un algorithme polynomial pour résoudre ce problème. Prouver qu'un problème de décision D est NP-complet consiste aux quatre étapes suivantes :

1. Montrer que D appartient à NP
2. choisir D', un problème NP-complet connu.
3. construire une transformation f de D' vers D.
4. prouver que f est une transformation polynomiale.

1.6.4. NP-complétude au sens fort

Un algorithme pseudo-polynomial est défini comme un algorithme dont le temps d'exécution est O(p(x)) ou p est un polynôme et x est la longueur de données d'une instance du problème.

Un problème de décision D est dit *NP-complet au sens fort*, si

1. D est un problème à nombres.
2. Il n'existe aucun polynôme q tel que Max[I]≤ q(Nbre[I]) pour toute instance I de D.
3. D est NP-complet.
4. Il n'existe aucun algorithme pseudo-polynomial pour le résoudre.

Ou (pour codage raisonnable A) Nbre[I] correspond au nombre de symboles utilisés pour décrire I et Max[I] correspond à la magnitude du plus grand nombre de I.

Soient D1 et D2, deux problèmes de décision. La transformation pseudo-polynomiale de D1 vers D2 (noté D1 ∞ D2) peut être vue comme une fonction f de l'ensemble

des instances de D1 vers l'ensemble des instances de D2 qui satisfait aux quatre conditions suivantes :

1. f est calculable par un algorithme polynomial en Max[I] et Nbre[I].
2. Pour toute instance I de D1, I a pour réponse « oui » (pour D1) si et seulement si f(I) a pour réponse « oui » (pour D2).
3. Il existe un polynôme q_1 tel que, pour toute instance I de D1, q_1 Nbre2[f(I)]) ≥ Nbre1[I].
4. Il existe un polynôme (à deux variables) q_2 tel que, pour toute instance I de D1, Max2[f(I)] ≤ q_2 (Max[I], Nbre1[I]).

Le problème **3-Partition** suivant est NP-complet au sens fort [59].

3-Partition :

Etant donnés un ensemble A à 3m éléments, une borne entière b et une taille a_i pour chaque élément $a \in A$, telle que $\frac{b}{4} < a_i < \frac{b}{2}$ et telle que $\sum_{a \in A} a_i = mb$. Existe-t-il m ensembles disjoints $S_1,\ldots,S_m$ tels que pour tout $1 \le j \le m$, $\sum_{a \in S_j} a_i = b$?

Pour montrer qu'un problème de décision D est NP-complet au sens fort, comme pour NP-complet (au sens faible), il suffit de considérer les quatre étapes suivantes :

1. Montrer que D appartient à NP.
2. choisir D', un problème NP- complet au sens fort, comme pour NP- complet .
3. construire une transformation f de D' vers D.
4. prouver que f est une transformation pseudo polynomiale

1.6.5. Optimisation combinatoire et problème NP- difficiles

Un problème d'optimisation combinatoire est un problème mathématique où il faut déterminer un « meilleur » élément parmi un nombre fini, mais souvent très élevé, d'éléments. La difficulté de résolution d'un tel problème réside dans le fait que la solution doit être cherchée dans un ensemble de très grande cardinalité.

Un problème d'optimisation combinatoire est dit *NP-difficile* (resp. *NP-difficile au sens fort*) si le problème de décision associé est *NP-complet* (resp. *NP-complet au sens fort*).

Chapitre 2

Le problème d'assemblage rectangulaire

Sommaire

2.1. Introduction

Dans ce qui suit, nous nous intéressons à l'assemblage rectangulaire (*Rectangular Packing Problem)* qui est une variante de découpe et d'assemblage (*D&A*), considéré comme difficile. Il appartient à la classe $NP-$difficile.

Nous allons commencer par présenter la formulation générale du problème d'assemblage rectangulaire, puis exposer ses différentes variantes.

La dernière partie de ce chapitre sera consacrée au problème d'assemblage orthogonal rectangulaire (*Orthogonal Rectangular Packing Problem)* qui constitue le thème de notre travail.

2.2. Présentation du problème

Le problème d'assemblage rectangulaire est défini par la donnée d'un ensemble de pièces $S=\{(l_1,w_1),(l_2,w_2),...,(l_n,w_n)\}$ de longueur l_i et largeur w_i, $i=1,...,n$ et un grand rectangle de dimensions fixées $L\times W$ ou non fixées. Il s'agit de placer le plus grand nombre de pièces de l'ensemble *S* dans le grand rectangle dans le cas où les dimensions du grand rectangle sont fixées et placer toutes les pièces rectangulaires à l'intérieur du plus petit rectangle fermé final dans le cas où les dimensions du grand rectangle sont variables.

Le problème modélise une grande variété de problème de la production industrielle et de conditionnement. Par exemple : étant donné une bande de tissu de longueur limitée et un ensemble de pièces à découper, le but est de trouver le meilleur placement de l'ensemble de ces découpes afin de minimiser la longueur de la bande utilisée.

2.3. Formulation du problème d'assemblage rectangulaire

Etant donnés *n* pièces et un grand rectangle, avec

l_i : La longueur de la pièce *i*

w_i : La largeur de la pièce *i*

c_i : Le profit de la pièce *i*

L, *W* : longueur et largeur du grand rectangle

b_i : Le nombre de pièce de type i

Les contraintes du problème

On affecte à chaque pièce les coordonnées x_i et y_i telles que :

1. $0 \le x_i \le W - w_i$;
2. $0 \le y_i \le L - l_i$;
3. Pour tout *i*, *j*

$$[x_i; x_i + w_i[\cup [x_j; x_j + w_j[= \phi \quad \vee$$

$$[y_i; y_i + l_i[\cup [y_j; y_j + l_j[= \phi \ ;$$

4. Le nombre de pièce de type *i* ne doit pas dépasser b_i pour chaque modèle de découpe (*réalisable*) ;

Remarque

Dans le cas où les dimensions du grand rectangle ne sont pas fixées, les contraintes 1 et 2 ne seront pas prises en considération.

Les fonctions objectifs

- Dans le cas où le grand rectangle est de dimensions fixées, l'objectif serait de choisir le sous ensemble $\Re \subset S$ de pièces, tel que la somme des profits soit maximisée.

$$z = \max \sum_{i=1}^{n} c_i r_i$$

où r_i est le nombre de rectangles de type *i* dans le modèle de découpe.

- Dans le cas où l'une ou les deux dimensions du grand rectangle ne sont pas fixées, l'objectif serait de placer toutes les pièces à l'intérieur d'un rectangle final de surface minimale.

$$z = \min\{T_1, T_2, ...T_m\}$$

tel que $T_i (1 \le i \le m)$ est un rectangle final qui contient toutes les pièces avec leurs copies.

2.4. Les différentes variantes du problème

Le problème d'assemblage rectangulaire est posé dans de nombreuses situations dans la littérature, ce qui est dû aux contraintes particulières imposées. En effet, on distingue plusieurs variantes du problème dans les deux cas où la fonction objectif est à maximiser ou à minimiser.

Comme il est impossible d'étaler toutes les variantes à cause de leur nombre important, on se contente de celles qui suivent :

→ ***Le problème d'assemblage rectangulaire contraint***

Le problème d'assemblage rectangulaire contraint impose une borne supérieure b_i à chaque pièce i $(i = 1,...,n)$ à assembler. Dans cette

variante, il s'agit de trouver un sous-ensemble de pièces de façon à maximiser le profit total des pièces à placer et sans dépasser la borne supérieure de chaque pièce placée.

→ ***Le problème d'assemblage rectangulaire non contraint***

Ce problème consiste à placer un sous-ensemble de pièces dans un grand rectangle de longueur *L* et de largeur *W*. Le nombre de pièces de chaque type n'étant pas fixé, dans ce cas une limite supérieure $b'_i = \left\lfloor \frac{L}{l_i} \right\rfloor \left\lfloor \frac{W}{w_i} \right\rfloor$ est imposée à chaque type de pièces considéré.

→ ***Le problème d'assemblage de bandes rectangulaires (Rectangular Strip Packing Problem)***

Le problème consiste à placer un ensemble d'éléments rectangulaires dans un grand rectangle de largeur fixée et de longueur variable. L'objectif est de minimiser la longueur du grand rectangle.

→ ***Le problème d'assemblage orthogonal rectangulaire (Orthogonal Rectangular Packing Problem)***

C'est sous cette variante du problème d'assemblage que notre travail s'inscrit : où les dimensions du grand rectangle sont variables (largeur et longueur). Ainsi, toutes les pièces sont placées à l'intérieur d'un rectangle final de surface minimale.

2.5. Le Problème d'Assemblage Orthogonal Rectangulaire

Le Problème d'Assemblage Rectangulaire (*PAR*) consiste à regrouper un ensemble de pièces rectangulaires à l'intérieur d'un rectangle final de surface minimale [**12**].

Dans la littérature, il s'agit d'une généralisation du Problème d'Assemblage Carré (*PAC*) qui consiste à regrouper un ensemble de petites pièces carrées à l'intérieur d'un carré final de surface minimale [**21**].

2.5.1. Problème dual du problème d'assemblage rectangulaire (*PAR*)

Le problème dual du problème *PAR* est le Problème de Découpe de Stock à Deux Dimensions (*PDSDD*) [**3**]. Ce dernier consiste à découper des petites pièces à partir d'un ou plusieurs supports initiaux de dimensions fixées.

Afin de réduire les modèles de découpe faisables (solution réalisable), la découpe guillotine est appliquée à ce problème. La découpe guillotine est effectuée d'une seule tranche en allant d'un coté à l'autre de la plaque, horizontalement ou verticalement.

De même dans le cas du problème d'assemblage rectangulaire une stratégie de ***construction orthogonale*** est appliquée au problème pour la génération des assemblages rectangulaires.

2.5.2. Présentation du problème d'assemblage orthogonal rectangulaire (*PAOR*)

Étant données un ensemble de *n* pièces $S = \{(l_1, w_1),(l_2, w_2),...,(l_n, w_n)\}$, chaque pièce i ($i = 1,...,n$), est représentée par une longueur l_i, une largeur w_i et une borne supérieure associée b_i.

Les solutions réalisables du problème *PAOR* représentées par l'ensemble $T = \{T_1, T_2,..., T_m\}$ de *m* rectangles finaux, tel que chaque solution contienne toutes les pièces avec leurs copies.

Une solution $T^* \in T$ est optimale, si elle réalise un rectangle final de surface minimale.

Nous utilisons une stratégie de ***construction orthogonale*** pour réduire les modèles d'assemblage faisables. Elle se fait en combinant les pièces avec leurs copies sous forme de constructions horizontales et verticales. A cet effet, nous adoptons les définitions suivantes.

♦ ***Définition 2.1***

Soient *p* et *p'* deux pièces de S, de dimensions (l, w) et (l', w') respectivement.

On appelle $R = (l + l', \max\{w, w'\})$ (resp. $R = (\max\{l, l'\},\ w + w')$), un assemblage rectangulaire (noté *r-assemblage*) obtenu en joignant les pièces *p* et *p'* par une construction horizontale (resp. verticale), sachant que le nombre d'occurrence d'une pièce donnée sur chaque construction ne doit pas excéder la borne supérieure b_i, $i = 1,..., n.$ (voir figure 2.1 ci-dessous)

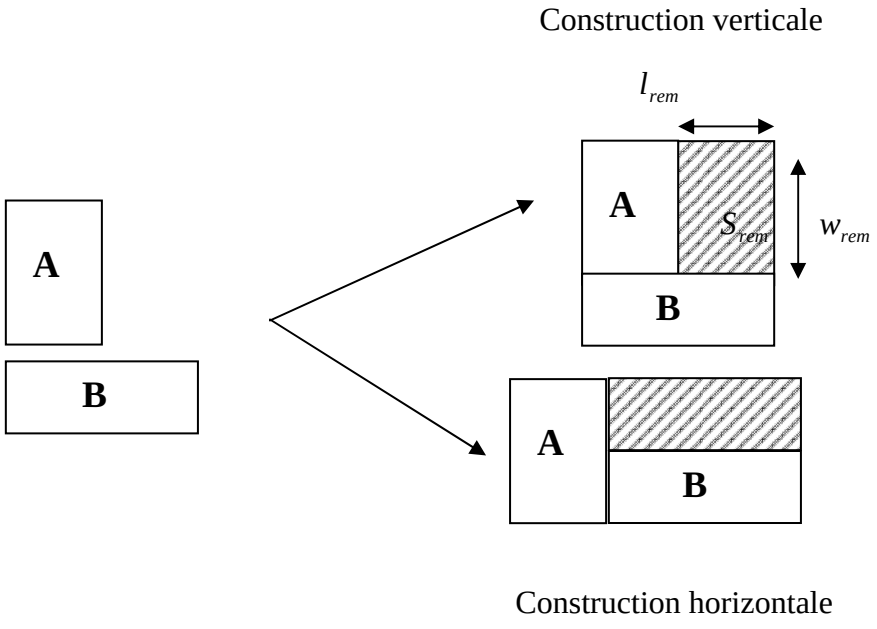


Figure 2.1* : *Constructions horizontale et verticale

S_{rem} : Sous rectangle restant (espace vide) après la combinaison des deux pièces A et B, de longueur l_{rem} et de largeur w_{rem}.

♦ ***Définition 2.2***

On appelle R un assemblage terminal (noté *t-assemblage*), s'il contient toutes les pièces avec leurs copies, voir figure 2.2.

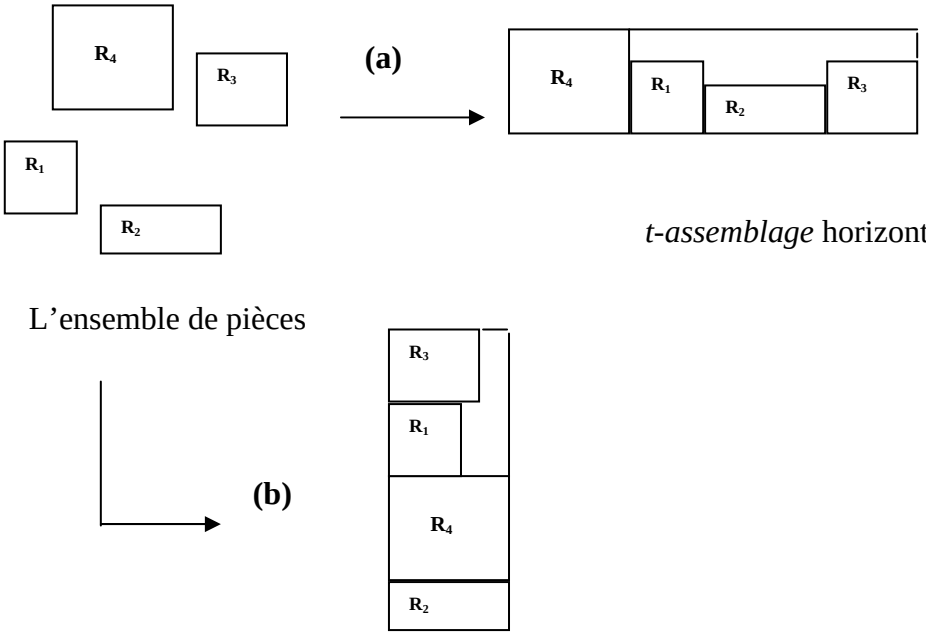


Figure 2.2* : Représentation de l'assemblage terminal (*t-assemblage)

(a) t-assemblage horizontal

(b) t-assemblage vertical

Remarque 1

Chaque assemblage peut être tourné de 90°. Dans ce cas la définition 1 est étendue à un *r-assemblage* non fixé et chaque étape de construction entre deux *r-assemblages* produit huit *r-assemblages*. Dans cette étude, seules les pièces et les *r-assemblages* d'orientations fixées sont pris en considération.

Chapitre 3

Etude de la méthode exacte Best First Branch and Bound

Sommaire

3.1. Introduction

En 1998, *Hifi. M & Ouafi. R* [**12**] ont mis en place une nouvelle méthode pour venir à résoudre le problème d'assemblage orthogonal rectangulaire connu sous le nom de *Best First Branch and Bound* (*BFBB*), que nous allons étaler dans ce chapitre.

En premier lieu, nous présentons la méthode avec ses bornes supérieure et inférieure ainsi que sa convergence.

En second lieu, trois Stratégies de branchement (*heuristique gloutonne*, *meilleur d'abord*, *coupe branch*) sont développées pour stériliser quelques nœuds inutiles de l'arborescence.

En dernier lieu, nous illustrons la méthode par le déroulement d'un exemple numérique.

3.2. Algorithme exact Best First Branch and Bound (*BFBB*)

3.2.1. Borne supérieure

Nous proposons une procédure (*gloutonne*) initiale pour produire une borne supérieure initiale pour l'algorithme exact. L'heuristique proposée commence avec la construction de deux *t-assemblages* évidents :

- Le premier étant un *t- assemblage* horizontal qui réalise la valeur suivante :

$$E_h = (\sum_{i=1}^{n} b_i l_i)(\max_{1 \le i \le n} \{w_i\}) \qquad (1)$$

- Le deuxième est un *t- assemblage* vertical avec la valeur :

$$E_v = (\sum_{i=1}^{n} b_i w_i)(\max_{1 \le i \le n} \{l_i\}) \qquad (2)$$

L'évaluation initiale de l'heuristique gloutonne consiste à choisir l'un des deux *t-assemblages* précédents qui réalise la valeur minimale :

$$E_{\sup} = \min\{E_h, E_v\} \qquad (3)$$

La valeur $E_{\sup}$ est considérée comme une première borne. Cette dernière est améliorée par une procédure gloutonne *Algo1* en utilisant une construction horizontale. Le même processus est établit pour produire une borne supérieure avec la construction verticale. Ainsi, la borne supérieure initiale $E_{\sup}$ pour l'algorithme exact est égale à la plus petite valeur des deux bornes supérieures horizontale et verticale obtenues par la procédure gloutonne.

Algo1: La procédure gloutonne pour le problème PAOR

Entrée: l'ensemble des pièces rectangulaires bornées par $b_i \quad 1 \le i \le n$

Sortie: solution sous-optimale notée $E_{\sup}$

L'étape initiale:

1) *calculer* $E_{\sup}$ *par l'équation (3)*
2) ξ *: l'ensemble des pièces avec leurs copies*
   R : une pièce de ξ *avec la plus grande longueur* $w_R = \max_{R' \in \xi}\{w_{R'}\}$
3) *Construire* $Init = (R_{Init}, l_{rem}, w_{rem})$ *avec la représentation de la liste de r- assemblage* $R_{Init} = (b_R l_R, w_R)$, l_{rem}, w_{rem} *sont respectivement la longueur et la largeur du sous-rectangle* S_{rem}.

L'étape principale:

Répéter

Choisir la pièce $R' \in \xi$ *avec la plus grande longueur* $l_{R'} = \max_{R \in \xi}\{l_R\}$ *et former la construction horizontale à* R_{Init} .

- *L'ensemble* $\xi = \xi \setminus \{R'\}$
- *Calculer* E_{Init} *la valeur de la surface de* R_{Init} *et* $S_{rem} = (l_{rem}, w_{rem})$ *avec* $l_{rem} = l_{R'}$ *et*
  $w_{rem} = w_{R_{Init}} - w_{R'}$.
- *Remplir le rectangle* (l_{rem}, w_{rem}) *avec les pièces de l'ensemble* ξ *par utilisation de la procédure gloutonne qui consiste à placer aléatoirement quelques pièces sans chevauchement et mettre à jour l'ensemble* ξ.

Jusqu'à: $(\xi = \phi)$ *ou* $(E_{\sup} \le E_{Init})$;

L'étape terminale:

Si $E_{Init} < E_{\sup}$, *alors l'ensemble* $E_{\sup} = E_{Init}$.

Fin

3.2.2. Borne inférieure

Etant donné un exemple du *PAOR* et un *r-assemblage R*, on cherche à trouver une borne inférieure pour la meilleure valeur du *t-assemblage* qui inclut R.
Soient :

- $g(R)$, la valeur interne de *R*, qui est égale à la somme des surfaces des pièces composant R.
- $c(R)$, l'aire non utilisée dans *R*, qui est égal à la différence entre la surface totale du rectangle R et la somme des surfaces des pièces qui contribuent à sa construction.
- $h(R)$, la plus petite aire représentant la surface qui couvre le reste des contraintes de la demande sans considérer l'ensemble des pièces composant R.
- $f(R) = G(R) + h(R)$ est la valeur minimale d'un *t-assemblage* faisable qui contient R, où $G(R) = g(R) + c(R)$
- $h'(R)$, l'estimation de $h(R)$ la plus petite aire représentant la surface qui couvre le reste des contraintes de la demande.

Il n'est généralement pas facile de trouver une telle estimation. Dans cette étude, l'estimation $h'(R)$ est calculée comme suit :

$$h'(R) = \sum_{i=1}^{n} c_i (b_i - x_i) \qquad (4)$$

avec :

x_i étant le nombre d'occurrences de la i^{eme} pièce dans *R* et $c_i = l_i w_i$ est l'aire de la i^{eme} pièce.

D'où $f'(R) = G(R) + h'(R)$ est la borne inférieure du t- assemblage contenant *R*.

3.2.3. L'Algorithme exact *BFBB* pour le *PAOR*

L'*Algo2* décrit l'étape principale de l'algorithme exact pour résoudre le problème *PAOR*. L'algorithme *BFBB* utilise deux listes ξ_1 et ξ_2, : la liste ξ_1 initiale contient n éléments tels que chaque élément R_i, $(i = 1,...,n.)$, de dimensions $(l_{R_i}, w_{R_i}) = (l_i, w_i)$, a une valeur interne : $g(R_i) = l_i w_i$, une valeur estimée : $h'(R_i)$, une borne inférieure : $f'(R_i)$ et un vecteur d avec $d_k \le b_k, (k = 1,..., n,)$ qui est le nombre d'occurrences de la k^{eme} pièce dans R_i. La liste ξ_2 est initialisée à l'ensemble vide.

A chaque itération, on sélectionne un élément R de l'ensemble ξ_1, ayant la plus petite borne inférieure f' qu'on place dans ξ_2. L'ensemble ξ_3 du nouvel *r-assemblage*, satisfait les conditions : $d_k \le b_k, k = 1,..., n,$ et $f'(R) < Opt$ (où Opt est la meilleure valeur de la solution produite actuellement). Il est créé en combinant l'élément R avec tous les éléments R' de ξ_2.

L'algorithme s'arrête dès que la valeur $f'(R)$ de l'élément R sélectionné de ξ_1 soit supérieure ou égale à la meilleure valeur de la solution courante. Sinon, la liste ξ_1 est réduite à l'ensemble vide.

En effet, l'arbre de recherche de l'algorithme *BFBB* est décrit comme suis :

Procédure d'initialisation :

Opt : Meilleure valeur de la solution actuelle.

La liste ouverte ξ_1 : Initialement contient n pièces (*de longueur* l_i, *largeur* w_i, *borne supérieure* b_i *et une borne inférieure* $f'(R_i)$).

La liste fermée ξ_2 : Initialement vide.

Procédure d'évaluation :

A chaque itération, on évalue la valeur de l'élément produit par le calcule de la fonction $f'(R)$ la valeur minimum d'un t-assemblage contient R (borne inférieure d'un t-assemblage contenant R).

Procédure de séparation :

Pour séparer, on choisit l'élément R' de ξ_1 de plus petite évaluation. On sélectionne après l'ensemble SE d'éléments de ξ_2 qui satisfait la contrainte $d_k \le b_k, k = 1,...,n$. Ensuite, on sépare l'élément R' en $2 \times |SE|$ (*chaque élément de SE produit deux éléments par la construction horizontale et verticale avec R'*).
On associe au sommet correspondant à R' de l'arborescence $2 \times |SE|$ sommets fils

Stérilisation :

Stériliser l'ensemble des éléments correspond à un t-assemblage ainsi que les éléments d'évaluation plus grand que Opt.

Algo2 ***: Algorithme exact BFBB pour le problème PAOR ;***

ξ_1 : L'ensemble des sous problèmes ;

ξ_2 : La liste de mémorisation des meilleurs sous problèmes ;

R et R' : *r-assemblages ;*

$f'(R)$: La borne inférieure du sous problème contenant R ;

Opt : La valeur de la meilleure solution actuelle.

Entrée: *Un exemple du problème d'assemblage orthogonal rectangulaire*

Sortie: *La valeur de la solution optimale Opt*

L'étape initiale:

$\xi_1 = \{R_1,...,R_n\}$; $\xi_2 = \emptyset$ *et* $fin =$ *faux* ;

Soit E_{sup} *la borne supérieure obtenue par l'application de la procédure gloutonne présentée dans l'Algo1 ;*

$Opt = E_{sup}$;

L'étape principale:

Répéter

Choisir le r-assemblage R avec la plus petite valeur de f' *; (note* f'_{min} *)*

Si $Opt - f'_{min} \leq 0$ *alors* $fin = vraie$

Sinon

Début

Transférer R de ξ_1 *à* ξ_2 *et construire tous les éléments de* ξ_3 *tels que*

- ξ_3 *est l'ensemble des constructions orthogonales entre R et tous les éléments de* ξ_2 ;
- *Chaque élément de* ξ_3 *satisfait les contraintes* $b_i (1 \leq i \leq n)$ *et a* $f' < Opt$;

Si $\exists$ *un assemblage terminal* $R' \in \xi_3 \setminus f'(R) < Opt$, *Alors* $Opt = f'(R)$ *; Mettre à jour l'ensemble* ξ_1 par $\xi_1 \cup \xi_3$;

Remplacer l'ensemble ξ_1 *par* ξ_1 */ {assemblage non terminal avec l'évaluation* $f' \geq Opt$ *} ;*

Si $\xi_1 = \emptyset$, *alors* $fin = vraie$;

Fin de si

Jusqu'à $fin = vraie$;

Fin

3.3. Convergence de l'algorithme

Nous allons maintenant énoncer un certain nombre de résultats qui justifient le choix des structures des assemblages utilisés par notre algorithme, de même ceux concernant sa finitude et sa convergence.

Théorème 3.1 [12]

Soient *R* et *R'* deux assemblages initiaux, une construction horizontale (resp. verticale) représente l'assemblage de surface minimale parmi toutes les constructions horizontales (resp. verticales) sur les deux assemblages *R* et *R'*.

Corollaire 3.1 **[12]**

Soient *R* et *R'* deux assemblages initiaux. Soit R_1 (resp. R_2) un assemblage engendré par construction horizontale (resp. verticale) sur les deux assemblages initiaux. La chute sur l'assemblage R_1 (resp.R_2) est supérieure ou égale à la somme des chutes sur les assemblages initiaux. Ainsi, la fonction C(*R*) est croissante par construction horizontale (resp. verticale).

Lemme 3.1 **[12]**

Chaque *r-assemblage* sélectionné de ξ_1 ne peut revenir à cet ensemble.

Théorème 3.2 **[12]**

L'algorithme BFBB s'arrête après un nombre fini d'itérations. Si $\xi_1 = \phi \; ou \; Opt - f'_{\min} \leq 0,$ alors Opt représente la valeur de la solution optimale du problème d'assemblage considéré.

3.4. Stratégies de branchement

Afin de rendre l'algorithme plus efficace, trois stratégies appropriées ont été établies pour stopper le développement de l'arborescence. Nous présentons le principe de ces différentes stratégies comme suit :

3.4.1. Heuristique gloutonne

L'heuristique gloutonne appliquée tout au début de l'algorithme *BFBB* pour l'évaluation de la borne initiale est considérée comme première stratégie de branchement dans la procédure de recherche arborescente. Cette borne consiste à rejeter les assemblages qui ont une évaluation supérieure à la meilleure évaluation en cours (d'un assemblage terminal) ; ce qui permet d'écarter quelques assemblages (non terminaux) des différentes constructions activées sur les éléments des deux listes ξ_1 et ξ_2.

3.4.2. Meilleur d'abord

Afin que l'algorithme évolue d'une manière plus efficace, nous avons adopté une deuxième stratégie qui consiste à tenir compte du nœud sur lequel le développement

de l'arborescence est effectué. Elle consiste à choisir à chaque itération de l'étape principale de l'algorithme, l'assemblage R^* qui réalise $f'(R^*) = \min_{R \in \xi_1} \{f'(R)\}$ ce qui est équivalent au choix $g(R^*) = \max_{R \in \xi_1} \{g(R)\}$, et cela permet d'atteindre plus rapidement une nouvelle borne supérieure $h'(R^*)$ (meilleure que la précédente) et donc un rejet prématuré de certains assemblages, étant donné que l'évaluation sur le meilleur assemblage en cours diminue.

3.4.3. Coupe branche

La troisième stratégie permet de stériliser certains sommets de l'arborescence créée. C'est cette stratégie (associée aux deux autres stratégies) que nous incorporons afin d'améliorer les performances de l'algorithme. Le théorème suivant illustre la stratégie développée.

Théorème 3.3 **[12]**

Soient R_1 le meilleur assemblage terminal en cours et R_2 un autre assemblage (*non nécessairement terminal*). Si :

$$(a)\ G(R_2) \geq G(R_1),$$

ou

$$(b)\ G(R_2) < G(R_1) \wedge C(R_2) \geq C(R_1)$$

Alors tout assemblage engendré par des constructions horizontales et/ou verticales sur l'assemblage R_2 conduit vers une solution moins bonne que celle fournie par R_1.

Dans l'algorithme BFBB, tout sommet de l'arborescence en cours pour lequel le théorème précédent est vérifié est stérilisé.

3.5. La phase de reconnaissance

Lors de la récupération da la solution optimale du problème d'assemblage, on est évidemment intéressé par la composition de celle-ci. Afin de retracer sa structure finale représentée par le vecteur $d = (d_1, d_2, ..., d_n)$, qui sert aussi à reconnaître le parcours de n'importe quel sommet de l'arborescence, notamment pour les besoins des retours en arrière, pour les opérations de réévaluations de la procédure de recherche arborescente. Les meilleurs sous problèmes sont stockés dans la liste ξ_2,.

Ainsi, la structure de la solution finale peut être récupérée de la liste ξ_2 qui comporte aussi tous les prédécesseurs « *parents* » de cette solution. Enfin, en introduisant certaines modifications sur chaque élément de la liste ξ_1 et ξ_2 par l'affectation d'attributs en relation avec la position, la construction verticale (respectivement horizontale), des *parents* des nouvelles constructions etc. La solution finale peut être facilement retrouvée par simple procédure récursive basée essentiellement sur les

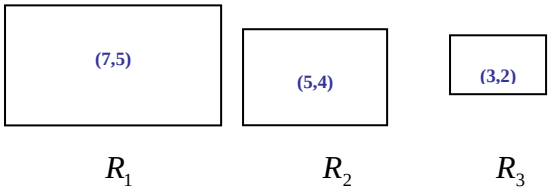
positions de chaque assemblage interne (sommet intermédiaire) du t-assemblage (sommet final).

3.6. Déroulement de la méthode *BFBB* sur un exemple numérique

Considérons l'instance *PAOR* défini par :

n=3 ; $b_1 = 1$ *;* $b_2 = 1$ *;* $b_3 = 3$ *;* $\xi =$ *{(7,5) ;(5,4) ;(3 ,2) ;(3,2) ;(3,2)}*

Type de pièces	R_1	R_2	R_3
l_i	7	5	3
w_i	5	4	2
	1	1	3

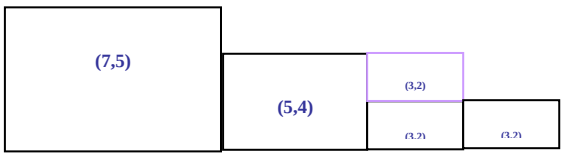


- ***L'étape initiale :***

1)- Calcul de la borne supérieure initiale E_{sup} (*Algo1*) :

La borne supérieure horizontale :

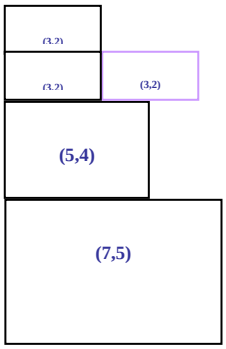
$E_{sup\,1}$ **= 90**



La borne supérieure verticale:

$E_{sup\,2}$ **= 91**

Résultat : $E_{sup} = \min\{E_{sup\,1}, E_{sup\,2}\} = 90$



2) La liste de mémorisation des meilleures sous problèmes $\xi_2 = Ø$

3) La valeur de la meilleure solution actuelle: $Opt = E_{sup} = 90$

4) calcul de la borne inférieure $f'(R)$ pour chaque sous problème contenant R :

$f'(R) = G(R) + h'(R)$

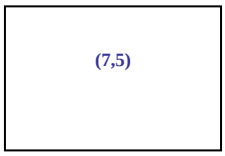
$f'(R_1) = (7 \times 5) + (5 \times 7 \times 1 + 3 \times 2 \times 3) = 73$

Type de pièces	1	2	3
$f'(R)$	73	73	73

- ***L'étape principale:***

  - *Itération 1 :* $f'min = 73$; R = (7,5), $\xi_1 = \xi_1 / \{(7,5)\}$

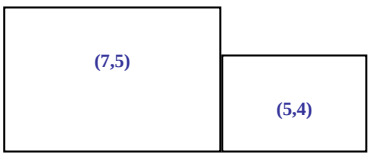
ξ_2 = {(7,5)}



  - *Itération 2 :* f'min = 73 ; R = (5,4), $\xi_1 = \xi_1 / \{(5,4)\}$

$\xi_2 = \xi_2 \cup \{(5,4)\}$

La constante de (5,4) avec :



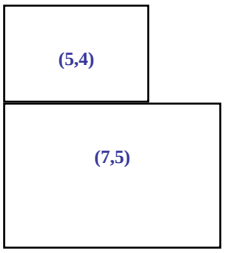
(7,5)

1) Construction horizontale : R= (12,5) ; f'$_{min}$ =78< Opt

ξ_3 = {(12,5)}, $\xi_1 = \xi_1 \cup \xi_3$

2) Construction verticale R= (7,9), f'$_{min}$ =81< Opt

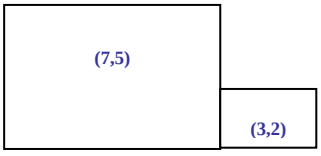
$\xi_3 = \xi_3 \cup \{(7,9)\}, \xi_1 = \xi_1 \cup \xi_3$



  - *Itération 3 :* f'$_{min}$ = 73 ; R = (3,2), $\xi_1 = \xi_1 / \{(3,2)\}$

$\xi_2 = \xi_2 \cup \{(3,2)\}$

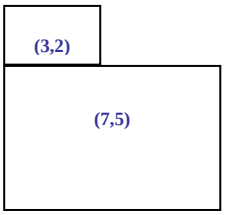
La constante de (3,2) avec :



(7,5)

1) Construction horizontale : R= (10,5) ; f'_{min} =82< Opt

$\xi_3 = \{(10,5)\}, \xi_1 = \xi_1 \cup \xi_3$

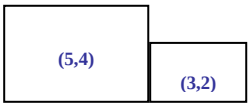


2) Construction verticale R= (7,7), f'_{min} =81< Opt

$\xi_3 = \{(7,7)\}, \xi_1 = \xi_1 \cup \xi_3$

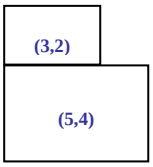
(5,4)

1) Construction horizontale : R= (8,4) ; f'_{min} =79< Opt



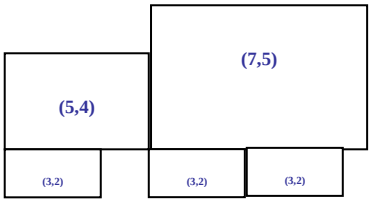
2) Construction verticale R= (5,6), f'_{min} =77< Opt

$\xi_3 = \xi_3 \cup \{(5,6)\}, \xi_1 = \xi_1 \cup \xi_3$



- …..
- ……
- ……..

On sort avec une solution optimale ***Opt =84*** où ***L=12***, ***W=7*** à l'issue de **46** itérations. Le modèle optimal est présenté dans le schéma suivant :



Chapitre 4

Différentes techniques d'amélioration

Sommaire

4.1. Introduction

L'algorithme *BFBB* est un algorithme exact qui donne une solution optimale, mais qui nécessite beaucoup d'espace mémoire. Cela est dû aux deux listes, ouverte et fermée (ξ_1 et ξ_2), qui ont tendance à devenir très larges au cours de l'exécution de l'algorithme.

Dans ce chapitre, on présente les différentes techniques utilisées pour accélérer l'algorithme et réduire la taille des deux listes. Commençons par la présentation du problème de découpe de stock, vu que les notions à suivre sont basées sur ce premier. En suite, on décrit une nouvelle représentation de la liste fermée élaborée par *V. Cung, M. Hifi et B. LeCun* 2000 [**5**]. En troisième position, nous étalons la notion de modèle dominé et de l'ordre lexicographique posé par *M. Hifi et R. M'Hallah (2004)* [**21**]. enfin, nous développons une nouvelle borne supérieure initiale améliorée qui permet de réduire l'espace de recherche ainsi que le temps d'exécution.

4.2. Définition du problème de découpe de stock contraint

Etant donnés un grand rectangle de stock S de dimensions $L \times W$, n petites pièces rectangulaires i $(i \in I = \{1,\ldots, n\})$ chacune de dimension $l_i \times w_i, i \in I$, de profit c_i et une borne supérieure b_i.

Le problème de découpe de stock consiste à découper le grand rectangle S en un ensemble de petits rectangles tels que

- L'orientation des pièces soit fixée ;
- Le type de découpe appliqué au problème soit de type guillotine ;

- Le nombre de pièce de type *i* ne doit pas dépasser b_i pour chaque modèle de découpe (*réalisable*) ;

L'objectif est de maximiser le profit total :

$$z = \max \sum_{i=1}^{n} c_i x_i \ ,$$

où x_i est le nombre de rectangles de type *i* dans le modèle de découpe.

4.3. Une nouvelle représentation de la liste fermée

L'algorithme *BFBB* est basé principalement sur les deux listes *Ouverte* ξ_1 et *Fermée* ξ_2. Le processus de travail commence par transférer le meilleur r-assemblage de ξ_1 vers ξ_2 et le combiné avec tous les éléments de ξ_2 après avoir testé si les deux conditions bi (1<=i<=n) et f<opt sont vérifiées. Selon la nature de la liste fermée qui est représentée par une liste chainée, l'algorithme teste élément par élément. Le processus ne prend pas en considération la longueur et la largeur du rectangle solution de départ (borne supérieure) c'est-à-dire on peut avoir une construction qui vérifie les deux conditions précédentes mais d'une longueur supérieure à la longueur du rectangle solution de départ ou bien de largeur supérieure à la largeur du rectangle solution de départ (cette construction est rejeté automatiquement au cour du processus).

La nouvelle représentation permet d'éviter les constructions de longueur ou largeur supérieure à la longueur ou la largeur du rectangle solution de départ. La structure matricielle utilisée permet de sélectionner l'ensemble des éléments à combiner sans passer par tous les éléments de la liste *fermée*. Le processus de la nouvelle représentation est expliqué comme suit :

A chaque itération de l'algorithme, un élément R est choisi à partir de la liste ouverte ξ_1, transféré vers la liste fermée ξ_2 et combiné avec quelques éléments de ξ_2. Désignons par ϑ, le sous ensemble de ξ_2 des éléments combinés avec R de longueur l_R et largeur w_R. Le but est de localiser un sous ensemble ϑ, sans trop d'effort de calcul.

V. Cung, M Hifi et B. Le. Cun (2000) [**5**], ont introduit une nouvelle structure de données de la liste fermée ξ_2. Cette structure contient tous les modèles ordonnés selon les deux ordres croissants de longueur et de largeur (*voir figure 6.1 ci-dessous).* Les intervalles de longueur et de largeur sont limités par la dimension (*L, W*) du rectangle de stock.

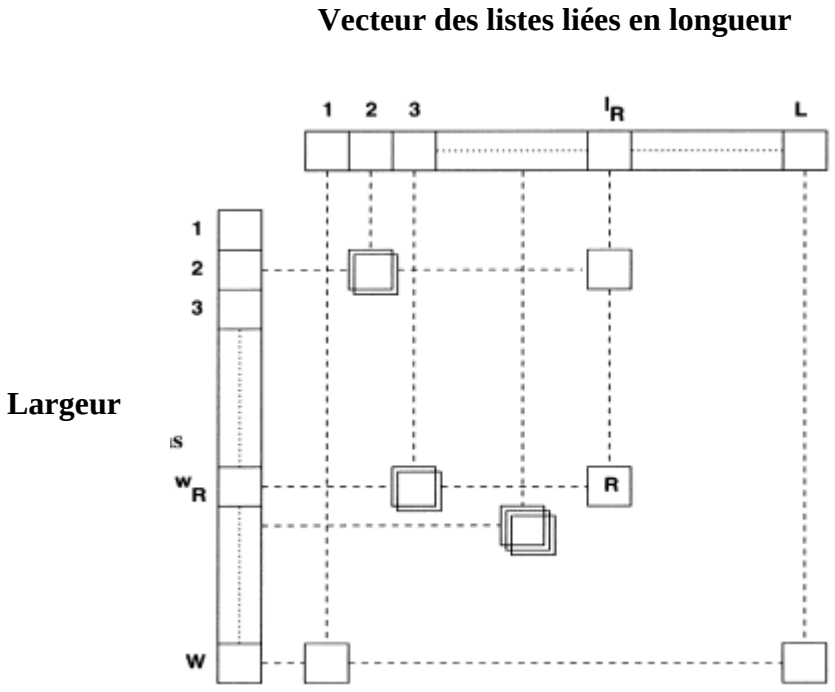


Figure 4.1 : La structure de données de la liste fermée

En suivant le schéma de la *figure 4.1*, on peut distinguer tous les éléments de ϑ comme suit :

- Soit R, un élément choisi de la liste ouverte et transféré vers la liste fermée
- Soient ϑ_{l_R} et ϑ_{w_R}, les ensembles des éléments candidats de la liste fermée pour la combinaison avec *R*, donnés comme suit :

$$\vartheta_{l_R} = \{q / l_q = l_R + l_p \le L,\ p \in \xi_2\}$$
$$\vartheta_{l_w} = \{q / w_q = w_R + w_p \le W,\ p \in \xi_2\}$$

où l_p et w_p sont la longueur et la largeur de l'élément p.

- Le sous-ensemble des candidats valides (*autorisés*) de la liste fermée est $\vartheta = \vartheta_{l_R} \cup \vartheta_{w_R}$

La nouvelle représentation de la liste fermée permet d'éviter préalablement les modèles non réalisables, sans passer par des constructions orthogonales. Donc on peut affirmer que l'algorithme donne toujours les solutions optimales et cela dans un temps minimal.

4.4. Notion des modèles dominés et de l'ordre lexicographique

Afin de réduire la taille des deux listes principales, deux stratégies sont introduites par *M. Hifi et R. M'Hallah* (2004) [**21**]. La première stratégie élimine les modèles dominés, la seconde permet d'éviter quelques modèles symétriques (*dupliqués*) en établissant un ordre lexicographique.

4.4.1. Modèles dominés

Soient *A* (l_A , w_A) et *B* (l_B , w_B) deux sous bandes (une sous bande est obtenue par combinaison horizontale de quelques pièces).
Soit *C= (A/B)* le modèle obtenu par la combinaison verticale de *A* et *B*.
Supposons que C (l_C, w_C) soit valide, c'est-à-dire que $(l_C, w_C) \leq (L, W)$ et $d_i(C) \leq b_i$ pour toute pièce *i*, où $d_i(C)$ est le nombre d'occurrences de la i^{eme} pièce dans le modèle C et b_i la borne supérieure imposée sur la pièce *i*.

Proposition 4.1 [21]

C est un modèle dominé si l'une des deux propositions suivantes sont vraies :

a) S'il existe une pièce k , $k \in I$ tel que $(l_k, w_k) \leq (L - l_A, w_A)$ et $b_k - d_k(C) > 0$, où l_k et w_k la longueur et la largeur de k
b) S'il existe une pièce k', $k' \in I$ tel que $(l_{k'}, w_{k'}) \leq (L - l_B, w_B)$ et $b_{k'} - d_{k'}(C) > 0$ où $l_{k'}$ et $w_{k'}$ soit la longueur et la largeur de k'.

Intuitivement, si les deux bandes *A* et *B* sont combinées verticalement et si l'une des bandes peut s'agrandir d'au moins une pièce, alors *C* ne peut être construite puisque une meilleure bande peut exister.

D'après ce qui précède, on peut dire que l'introduction d'une telle technique diminue le nombre de noeuds à explorer en gardant la solution optimale.

4.4.2. Ordre lexicographique

Etant donné la liste ouverte initiale, on introduit un ordre lexicographique à chaque pièce de la liste ouverte, tel que $R_i, (i \in I)$ est remplacé par l'élément $R_i^{\theta}, (i = \theta = 1...n)$ où θ est l'ordre lexicographique de certains (sous) modèles.

Notations :

- Si A est un élément composé d'une pièce R_k^{θ}, alors l'ordre lexicographique de A est noté θ_A ou $\theta_{R_k^{\theta}}$.

- Si A est composé d'au moins deux pièces, R_k^{θ} et R_s^{θ}, alors l'ordre lexicographique de A est $\theta_A = \min\{ \theta_K, \theta_S \}$

- Si $\theta_{R_k^{\theta}} \leq \theta_{R_s^{\theta}}$, alors R_s est placé dans la partie droite de A (R_k / R_s). Sinon, R_s est placé dans la partie gauche de A notée (R_s / R_k).

Théorème 4.1[21]

En utilisant les notations précédentes on peut appliquer le résultat suivant :
Soit A et B deux rectangles internes, où $A \in$ *liste ouverte* et composé d'au moins deux pièces et $B \in$ *liste fermée* et composé d'une seule pièce. Supposons que la combinaison de A et B produit un rectangle interne, alors la construction horizontale entre A et B est un rectangle interne dupliqué si $\theta_A < \theta_B$ où θ_A (resp. θ_B) est l'ordre de A (resp.B).

Le même processus est utilisé avec un nouvel ordre lexicographique pour la construction verticale. Chaque construction horizontale est considérée comme étant une nouvelle pièce, avec un nouvel ordre lexicographique *n+1*.

L'ordre lexicographique permet de détecter les modèles dupliqués (*théorème 4.2*) afin de les éliminer. L'ajout de cette technique n'a pas d'influence sur l'optimalité de la solution. Bien au contraire, il permet de l'atteindre en explorant moins de branches possibles.

4.5. La nouvelle borne supérieure

Nous développons un algorithme glouton (*Algo. 3***)** pour le calcul de la borne supérieure du problème d'assemblage orthogonal rectangulaire. L'algorithme commence par le calcul de la borne initiale (*calculer* $E_{\sup}$ *par l'équation (3)*) de départ qui sera améliorée par la suite. On considère à cet effet une liste ξ contenant l'ensemble des pièces avec leurs copies ordonnées selon l'ordre décroissant de largeur. On construit dans un premier lieu l'assemblage R_{rec} constitué uniquement de pièces de largeur maximale. Dans l'étape principale de la procédure, on effectue des constructions horizontales successives de la manière suivante : On introduit une liste intermédiaire ξ' constituée de pièces de largeur maximale, puis on sélectionne la pièce R de la liste ξ' de longueur maximale, on la place à droite du rectangle R_{rec} et on effectue par la suite un remplissage avec les pièces restantes sur le sous rectangle (*espace vide*) EV tant qu'il est possible. La procédure s'arrête si $E_{\sup} \le E_{rec}$ tel que E_{rec} représente la surface du rectangle R_{rec} résultant et on affecte la valeur de E_{rec} à $E_{\sup}$. Dans ce cas, $E_{\sup}$ est la meilleure borne supérieure de départ par construction horizontale. Sinon cette étape est répétée jusqu'à ce que $\xi = \phi$.

Algo.3 : Une procédure gloutonne pour le problème d'assemblage orthogonal rectangulaire ;

Entrée: l'ensemble des pièces rectangulaires borné par $b_i \quad 1 \le i \le n$

Sortie: $E_{\sup}$ *sous optimale solution noté*

L'étape initiale:

1) *calculer* $E_{\sup}$ *tel que :* $E_{\sup} = \min\{E_h, E_v\}$ *où*

$$E_h = (\sum_{i=1}^{n} b_i l_i)(\max_{1\le i\le n}\{w_i\}) \qquad et \qquad E_v = (\sum_{i=1}^{n} b_i w_i)(\max_{1\le i\le n}\{l_i\})$$

2) ξ *: L'ensemble des pièces avec leurs copies.*
R *: L'ensemble des pièces avect leurs copies triées selon l'ordre décroissant de leurs largeurs :* $w_1 \ge w_2 \ge ..... \ge w_{\sum_{i=1}^{n} b_i}$

3) *Soit* R_{rec} *le rectangle initial obtenu par construction horizontale de la pièce* R_1 *avec ses copies :* $R_{rec} = (b_1 l_1, w_1)$, *avec* $\xi = \xi \setminus \{R_k / l_{R_k} = l_{R_1} \ et \ w_{R_k} = w_{R_1}\}$

L'étape principale:
Répéter

1. *Soit* ξ' *l'ensemble des pièces dont la largeur est la plus grande :*

$$\xi' = \left\{ R_k : w_{R_k} = \max_{R_j \in \xi}\{w_{R_j}\};\ \forall l_{R_k} \right\}$$

2. *Choisir la pièce* $R' \in \xi' / l_{R'} = \max_{R_k \in \xi'}\{l_{R_k}\}$ *:*

   Faire la construction horizontale de R' avec R_{rec}

   Poser : $\xi = \xi \setminus \{R'\}$

3. *calculer l'espace vide EV résultat de la construction précédente, tel que :* $EV = (l_{R'}, w_{R_{rec}} - w_{R'})$.

   Tant que ($EV : (l_{EV}, w_{EV}) \ne (0,0)$ *) et (il existe une pièce qui rentre dans EV)*

   Faire :

   - *placer la pièce* R_i *dans l'espace EV*
   - *Poser* $\xi = \xi \setminus \{R_i\}$
   - *Calculer le nouvel espace vide EV*

   Fait ;

Jusqu'à: ($\xi = \phi$) *ou* ($E_{\sup} \le E_{rec}$);

L'étape terminale:

Si $E_{rec} < E_{\sup}$ *alors l'ensemble* $E_{\sup} = E_{rec}$.

Fin.

Chapitre 5

Algorithmes Best First Banch and Bound Améliorés

Sommaire

5.1. Introduction

Dans ce chapitre, on présente cinq nouvelles versions améliorées de l'algorithme *BFBB* par les techniques déjà vues dans le chapitre précédent. Ces cinq versions que nous allons développer le long de ce chapitre sont les suivantes :

1. L'algorithme *BFBB avec l'introduction de la notion du modèle dominé*.
2. L'algorithme *BFBB avec la notion de l'ordre lexicographique*.
3. L'algorithme *BFBB avec la notion de la nouvelle représentation de la liste fermée*. Puis,
4. L'algorithme *BFBB* amélioré par *l'introduction d'une nouvelle borne supérieure*.
5. L'algorithme *BFBB avec l'utilisation des quatre améliorations précédentes*.

5.2. Les versions améliorées de l'algorithme *BFBB*

5.2.1. L'algorithme Best First Branch and Bound avec l'introduction de la notion du Modèle Dominé (BFBBMD)

Vu la dualité entre les problèmes de découpe et d'assemblage, les notions introduites au chapitre 6 peuvent être facilement adaptées à notre problème.

5.2.1.1. Adaptation de la notion du modèle dominé à l'algorithme

La notion du modèle dominé énoncé dans le chapitre précèdent est adaptée à l'algorithme *Best First Branch and Bound* pour produire le nouvel algorithme ***BFBBMD*** comme suit :

Soient *R* et *R'* deux *r-assemblages*, la construction orthogonale entre *R* et *R'* est dite *modèle dominé* s'il existe $R'' \in \xi_1$ qui occupe l'espace vide S_{rem} obtenu par la construction orthogonale entre *R* et *R'*.

L'idée principale de l'algorithme ***BFBBMD*** est la même que l'algorithme ***BFBB***, sauf qu'à chaque nouveau model produit par combinaison orthogonale, on calcule l'espace vide. S'il existe une pièce de la liste ξ_1 qui peut occuper cet espace sans dépasser les contraintes exigées, alors ce modèle est un modèle dominé à éliminer.

L'arbre de recherche de notre algorithme suit le même processus que l'algorithme *BFBB* concerné par l'initialisation, la séparation et l'évaluation.

Stérilisation :

On stérilise l'ensemble des éléments correspondant à un t-assemlage, les éléments d'évaluation plus grand que *Opt* ainsi que les éléments qui représentent des modèles dominés.

5.2.1.2. Algorithme *BFBB* avec l'introduction de la notion du modèle dominé :

Algorithme BFBBMD;

ξ_1 : L'ensemble des sous problèmes ;

ξ_2 : La liste de mémorisation des meilleurs sous problèmes ;

R et R' : *r-assemblages ;*

$f'(R)$: La borne inférieure du sous problème contenant *R* ;

Opt : La valeur de la meilleure solution actuelle ;

S_{rem} : l'espace vide obtenu par la construction horizontale ou verticale entre *R* et *R'*

Entrée: *Une instance du problème d'assemblage orthogonal rectangulaire*

Sortie: *La valeur de la solution optimale Opt*

L'étape initiale:

$\xi_1 = \{R_1,...,R_n\}$; $\xi_2 = \emptyset$ *et* $fin =$ *faux* ;

Soit E_{sup} *la borne supérieure obtenue par l'application de la procédure gloutonne présentée dans l'***Algo1** ;

$Opt = E_{sup}$;

L'étape principale:

Répéter

Choisir le r-assemblage R avec la plus petite valeur de f' ; *(note* f'_{min} *)*

Si $Opt - f'_{min} \leq 0$ *alors* $fin = vraie$

Sinon

Début

Transférer R de ξ_1 *à* ξ_2 *et construire tous les éléments de* ξ_3 *tels que*

- ξ_3 *est l'ensemble des constructions orthogonales entre R et tous les éléments de* ξ_2 ;
- *Chaque élément de* ξ_3 *satisfait les contraintes* $b_i (1 \leq i \leq n)$ *et* $f' < Opt$;
- *Chaque modèle de* ξ_3 *qui satisfait le test de dominance est éliminé.*

S'il $\exists$ *un assemblage terminal* $R' \in \xi_3 \setminus f'(R') < Opt$, *alors :*

L'ensemble $Opt = f'(R')$;

Mettre à jour l'ensemble ξ_1 par $\xi_1 \cup \xi_3$;

Remplacer l'ensemble ξ_1 *par* ξ_1 */ {assemblage non terminal avec l'évaluation* $f' \geq Opt$ *} ;*

Si $\xi_1 = \emptyset$ *alors* $fin = vraie$;

Fin de si

Jusqu'à $fin = vraie$;

Fin

5.2.2. L'algorithme Best First Branch and Bound avec l'introduction de l'Ordre Lexicographique (BFBBOL)

5.2.2.1. Adaptation de l'ordre lexicographique

L'ordre lexicographique est appliqué à l'algorithme *Best First Branch and Bound*, afin d'éliminer quelques modèles de symétrie. Il est introduit dans la *liste ouverte* initiale tel qu'à chaque pièce $R_i, (i \in I)$, on associe deux ordres lexicographiques, un horizontal et un vertical, où $i = \theta_h = \theta_v = 1,2,\ldots, n$.

Pour chaque *r-assemblage A*, obtenu par construction horizontale entre *K* et *Q*, on introduit un nouvel ordre lexicographique tel que $\theta_{h(A)} = \min\{ \theta_{h(K)}, \theta_{h(Q)} \}$ et $\theta_{v(A)} = \max_{E \in \xi_1 \cup \xi_2} (\theta_{v(E)}) + 1$

On décrit l'étape principale de l'algorithme ***BFBBOL*** pour la résolution du problème PAOR. On utilise deux listes ξ_1 et ξ_2, la liste ξ_1 initiale contient *n* éléments tels que chaque élément R_i^{θ}, $(i = \theta = 1,\ldots,n.)$, est d'ordre lexicographique θ, de dimensions $(l_{R_i^{\theta}}, w_{R_i^{\theta}}) = (l_i, w_i)$, la valeur interne $g(R^{\theta}{}_i) = l_i w_i$, la valeur estimée $h'(R_i^{\theta})$, la borne inférieure $f'(R_i^{\theta})$ et le vecteur d avec $d_k \le b_k, (k = 1,\ldots,n,)$ est le nombre d'occurrences de la k^{eme} pièce dans R_i^{θ}. La liste ξ_2 est initialisée à l'ensemble vide.

Le même processus de l'algorithme *BFBB* est utilisé dans notre algorithme, avec l'introduction d'un test de détection des modèles de symétrie avant chaque combinaison.

5.2.2.2. Teste du modèle dupliqué :

Soient *R* et *R'* deux *r-assemblages*. $\theta_{h(R)}$, $\theta_{h(R')}$ les ordres lexicographiques horizontaux de *R*, *R'* respectivement. $\theta_{v(R)}$, $\theta_{v(R')}$ les ordres lexicographiques verticaux de *R* et *R'* respectivement.

Si $R \in$ *liste ouverte* composée d'au moins deux pièces et $R' \in$ *liste fermeé* composée de pièces du même type, alors :

- La construction horizontale entre *R* et *R'* est un modèle dupliqué si : $\theta_{h(R)} < \theta_{h(R')}$
- La construction verticale entre *R* et *R'* est un modèle dupliqué si : $\theta_{v(R)} < \theta_{v(R')}$

Remarque : Le même processus est utilisé pour la construction verticale.

L'arbre de recherche de notre algorithme suit le même processus que l'algorithme *BFBB* concerné par l'initialisation, la séparation et l'évaluation.

Stérilisation :

On stérilise l'ensemble des éléments correspondant à un t-assemlage, les éléments d'évaluation plus grand que *Opt* ainsi que les éléments qui représentent des modèles dupliqués.

5.2.2.3. Algorithme exact BFBB avec l'introduction de l'ordre lexicographique

Algorithme BFBBOL ;

ξ_1 : L'ensemble des sous problèmes ;

ξ_2 : La liste de mémorisation des meilleurs sous problèmes ;

R et R' : *r-assemblages ;*

$f'(R)$: La borne inférieure du sous problème contenant *R* ;

Opt : La valeur de la meilleure solution actuelle ;

$\theta_{h(R)}$: L'ordre lexicographique horizontale de *R ;*

$\theta_{v(R)}$: L'ordre lexicographique verticale de *R*

Entrée: *Une instance du problème d'assemblage orthogonal rectangulaire*

Sortie: *La valeur de la solution optimale Opt*

L'étape initiale:

$\xi_1 = \{R_1,...,R_n\}$; $\xi_2 = \emptyset$ *et* $fin = faux$;

Soit E_{sup} *la borne supérieure obtenue par l'application de la procédure gloutonne présentée dans l'***Algo1** ;

$Opt = E_{sup}$;

L'étape principale:

Répéter

Choisir le r-assemblage R avec la plus petite valeur de f' *; (notée* f'_{min} *)*

Si $Opt - f'_{min} \leq 0$ *, alors* $fin = vraie$

Sinon

Début

1. *Transférer R de* ξ_1 *à* ξ_2 *et*
2. *Construire l'ensemble* ξ_3 *tel que :*
   a) *Chaque élément R'' de* ξ_3 *obtenu par la construction horizontale ou verticale entre R et tous les éléments K de* ξ_2 ;
   b) *Tester R avec les éléments K de* ξ_2 *, Si le modèle obtenu est un modèle dupliqué, alors le nouveau modèle n'est pas construit ;*
   c) *Chaque élément de* ξ_3 *satisfait les contraintes* $b_i (1 \leq i \leq n)$ *et* $f' < Opt$;
   d) *Chaque élément* ξ_3 *est étiqueté par un ordre lexicographique ;*
3. *S'il* $\exists$ *un assemblage terminal* $R' \in \xi_3 \setminus f'(R') < Opt$ *, Alors l'ensemble* $Opt = f'(R')$;
   Mettre à jour l'ensemble ξ_1 par $\xi_1 \cup \xi_3$; *Remplacer l'ensemble* ξ_1 *par* ξ_1 */ {assemblage non terminal avec l'évaluation* $f' \geq Opt$ *} ;*
4. *Si* $\xi_1 = \emptyset$, *alors* $fin = vraie$;

Fin de si

Jusqu'à $fin = vraie$;

Fin

5.2.3. L'algorithme Best First Branch and Bound avec l'introduction de la nouvelle représentation de la liste fermée (BFBBLF)

7.2.3.1. Adaptation de la nouvelle représentation de la liste fermée

L'algorithme recherche la plus petite surface qui contient toutes les pièces avec leurs copies. D'où il s'agit d'utiliser une nouvelle représentation de la liste fermée (*voir chapitre 6*). Pour ce faire, on fixe *L* et *W* d'après la solution initiale réalisable de l'***Algo1*** telles que :

L : La longueur du rectangle solution obtenue par ***Algo1***, en utilisant la construction horizontale, et

W : La largueur du rectangle solution obtenue par ***Algo1***, en utilisant la construction verticale.

A chaque itération, on sélectionne un élément *R* de l'ensemble ξ_1, ayant la plus petite borne inférieure f' et on le place dans ξ_2. L'ensemble ξ_3 du nouvel *r-assemblage*, satisfaisant $d_k \le b_k, (k = 1,..., n,)$ et $f'(R) < OPt$ (où Opt est la meilleure valeur de la solution produite actuellement), est créé par les combinaisons horizontales de l'élément *R* avec tous les éléments *R'* de l'ensemble ϑ_{l_R} et les combinaisons verticales de l'élément R avec tous les éléments *R''* de l'ensemble ϑ_{w_R} (où ϑ_{l_R} et ϑ_{w_R} sont des sous-ensemble de la liste ξ_2).

L'algorithme s'arrête dès que la valeur $f'(R)$ de l'élément *R* sélectionné de ξ_1 est supérieure ou égale à la meilleure valeur de la solution courante. Sinon, la liste ξ_1 est réduite à l'ensemble vide.

5.2.3.2. Algorithme exact *BFBB* avec l'introduction de la nouvelle représentation de la liste fermée

Algorithme BFBBLF ;

ξ_1 : L'ensemble des sous problèmes ;

ξ_2 : La liste de mémorisation des meilleurs sous problèmes ;

R et R' : *r-assemblages ;*

$f'(R)$: La borne inférieure du sous problème contenant R ;

Opt : La valeur de la meilleure solution actuelle ;

$\theta_{h(R)}$: L'ordre lexicographique horizontal de R ;

$\theta_{v(R)}$: L'ordre lexicographique vertical de R.

Entrée: *Une instance du problème d'assemblage orthogonal rectangulaire*

Sortie: *La valeur de la solution optimale Opt*

L'étape initiale:

$\xi_1 = \{R_1,...,R_n\}$; $\xi_2 = \emptyset$ *et* $fin = faux$;

Soit E_{sup} *la borne supérieure obtenue par l'application de la procédure gloutonne présentée dans l'Algo1 ;*

$Opt = E_{sup}$;

L'étape principale:

Répéter

Choisir le r- assemblage R avec la plus petite valeur de f' *; (notée* f'_{min} *)*

Si $Opt - f'_{min} \leq 0$ *, alors* $fin = vraie$

Sinon

Début

1. *Transférer R de* ξ_1 *à* ξ_2 *et*
2. *Construire l'ensemble* ξ_3 *tel que :*
   a. *Chaque élément R'' de* ξ_3 *obtenu par la construction horizontale entre R et les éléments* ϑ_{l_R} *de* ξ_2 *tel que* $\vartheta_{l_R} = \{q / l_q = l_R + l_p \leq L,\ p \in \xi_2\}$
   b. *Chaque élément R''' de* ξ_3 *obtenu par la construction verticale entre R et les éléments* ϑ_{w_R} *de* ξ_2 *tel que* $\vartheta_{l_w} = \{q / w_q = w_R + w_p \leq W,\ p \in \xi_2\}$
   c. *Chaque élément de* ξ_3 *obtenu par construction verticale ou horizontale satisfait les contraintes* $b_i (1 \leq i \leq n)$ *et* $f' < Opt$;
3. *S'il* $\exists$ *un assemblage terminal* $R' \in \xi_3 \setminus f'(R') < Opt$ *, Alors l'ensemble* $Opt = f'(R')$ *; Mettre à jour l'ensemble* ξ_1 *par* $\xi_1 \cup \xi_3$; *Remplacer l'ensemble* ξ_1 *par* ξ_1 */ {assemblage non terminal avec l'évaluation* $f' \geq Opt$ *} ;*
4. *Si* $\xi_1 = \emptyset$, *alors* $fin = vraie$;

Fin de si

Jusqu'à $fin = vraie$;

Fin

5.2.4. L'algorithme Best First Branch and Bound avec l'introduction de la nouvelle Borne Supérieure initiale (BFBBNBS)

5.2.4.1. Algorithme exact *BFBB* avec l'introduction de la nouvelle borne supérieure initiale

L'algorithme ***BFBBNBS*** utilise une nouvelle borne supérieure basée sur l'algorithme glouton (***Algo.3***). Celle-là permet de réduire l'espace de recherche et accélère le processus de recherche.

Algorithme BFBBNBS :

ξ_1 : L'ensemble des sous problèmes ;

ξ_2 : La liste de mémorisation des meilleurs sous problèmes ;

R et R' : *r-assemblages ;*

$f'(R)$: La borne inférieur du sous problème contenant R ;

Opt : La valeur de la meilleure solution actuelle.

Entrée: *Une instance du problème d'assemblage rectangulaire orthogonal*

Sortie: *La valeur de la solution optimale Opt*

L'étape initiale:

$\xi_1 = \{R_1,...,R_n\}$; $\xi_2 = \emptyset$ *et* $fin = faux$;

Soit E_{sup} *la borne supérieure obtenue par l'application de la procédure gloutonne présentée dans l'***Algo3** ;

$Opt = E_{sup}$

L'étape principale:

Répéter

Choisir le r- assemblage R avec la plus petite valeur de f' *; (notée* f'_{min} *)*

Si $Opt - f'_{min} \leq 0$ *, alors* $fin = vraie$

Sinon

Début

Transférer R de ξ_1 *à* ξ_2 *et construire tous les éléments de* ξ_3 *tel que :*

- ξ_3 *est l'ensemble des constructions orthogonales entre R et tous les élément de* ξ_2 ;
- *Chaque élément de* ξ_3 *satisfait les contraintes* $b_i (1 \leq i \leq n)$ *et a* $f' < Opt$;

S'il $\exists$ *un assemblage terminal* $R' \in \xi_3 \setminus f'(R') < Opt$ *, alors :*

L'ensemble $Opt = f'(R')$;

Mettre à jour l'ensemble ξ_1 par $\xi_1 \cup \xi_3$;

Remplacer l'ensemble ξ_1 *par* ξ_1 */ {assemblage non terminal avec l'évaluation* $f' \geq Opt$ *} ;*

Si $\xi_1 = \emptyset$, *alors* $fin = vraie$;

Fin de si

Jusqu'à $fin = vraie$;

Fin

5.2.5. L'algorithme Best First Branch and Bound Amélioré (BFBBA)

Cette version est une combinaison de toutes les versions précédentes pour la résolution du problème *PAOR*.

5.2.5.1. Algorithme exact BFBB amélioré

Algorithme BFBBA :

ξ_1 : L'ensemble des sous problèmes ;

ξ_2 : La liste de mémorisation des meilleurs sous problèmes ;

R et R' : r- assemblages ;

$f'(R)$: La borne inférieure du sous problème contenant *R* ;

Opt : La valeur de la meilleure solution actuelle ;

S_{rem} : L'espace vide obtenu par la construction horizontale ou verticale entre *R* et *R ;*

$\theta_{h(R)}$: L'ordre lexicographique horizontal de *R ;*

$\theta_{v(R)}$: L'ordre lexicographique vertical de *R* .

Entrée: *Une instance du problème d'assemblage rectangulaire orthogonal*

Sortie: *La valeur de la solution optimale Opt*

L'étape initiale:

$\xi_1 = \{R_1,...,R_n\}$; $\xi_2 = \emptyset$ *et* $fin = faux$;

Soit E_{sup} *la borne supérieure obtenue par l'application de la procédure gloutonne présentée dans l'***Algo3** ;

$Opt = E_{sup}$;

L'étape principale:

Répéter

Choisir le r-assemblage R avec la plus petite valeur de f' *; (notée* f'_{min} *)*

Si $Opt - f'_{min} \leq 0$ *; alors* $fin = vraie$

Sinon Début

1. *Transférer R de* ξ_1 *à* ξ_2 *et*
2. *construire l'ensemble* ξ_3 *tel que :*
   a. *Chaque élément R'' de* ξ_3 *obtenu par la construction horizontale entre R et les éléments* ϑ_{l_R} *de* ξ_2 *tel que* $\vartheta_{l_R} = \{q / l_q = l_R + l_p \leq L,\ p \in \xi_2\}$
   b. *Chaque élément R'' de* ξ_3 *obtenu par la construction verticale entre R et les éléments* ϑ_{w_R} *de* ξ_2 *tel que* $\vartheta_{l_w} = \{q / w_q = w_R + w_p \leq W,\ p \in \xi_2\}$
   c. *Tester R avec les éléments K de* ξ_2. *Si le modèle obtenu est un modèle dupliqué, alors le nouveau modèle n'est pas construit.*
   d. *Chaque élément de* ξ_3 *satisfait les contraintes* $b_i (1 \leq i \leq n)$ *et* $f' < Opt$;
   e. *Chaque modèle de* ξ_3 *qui satisfait le test de dominance est éliminé.*
   f. *Chaque élément de* ξ_3 *est étiqueté par un ordre lexicographique.*
3. *S'il* $\exists$ *un assemblage terminal* $R' \in \xi_3 \setminus f'(R') < Opt$, *alors l'ensemble* $Opt = f'(R')$; *Mettre à jour l'ensemble* ξ_1 par $\xi_1 \cup \xi_3$; *Remplacer l'ensemble* ξ_1 *par* $\xi_1 / \{$*assemblage non terminal avec l'évaluation* $f' \geq Opt$ *}* ;
4. *Si* $\xi_1 = \emptyset$, *alors* $fin = vraie$;

Fin de si

Jusqu'à $fin = vraie$;

Fin

5.3. Conclusion

Le but fixé pour ce chapitre étant l'amélioration du temps d'exécution de l'algorithme *BFBB*. Les idées que nous avons introduites sont importées de la littérature toute en ayant essayé de ramener un plus. La première et la deuxième technique introduites permettent de stériliser quelques sommets par élimination des modèles dominés et des modèles dupliqués. Une nouvelle représentation de la liste fermée est adaptée pour réduire le temps de calcul gaspillé. Pour réduire l'espace de recherche nous avons proposé une nouvelle borne supérieure pour l'algorithme exact. Une adaptation de ces méthodes était nécessaire pour l'amélioration de notre problème. Selon les résultats obtenus, nous pouvons dire que nous avons contribué d'une façon efficace à bénéficier du repère temps.

Chapitre 6

Implémentation et résultats

Sommaire

6.1. Introduction

Ce chapitre est consacré à l'étude des performances des différentes versions modifiées de l'algorithme *Best First Branch and Bound* et en particulier de la nouvelle modification ***BFBBA***.

Nous avons implémenté l'algorithme de ***BFBB*** ainsi que les variantes ***BFBBMD***, ***BFBBOL***, ***BFBBLF***, ***BFBBNB*** et ***BFBBA*** en langage de programmation Pascal sous environnement Borland Delphi 7. Les différentes instances sont exécutées sur un ordinateur *Intel (R) Pentium (R) 4 CPU 3.00GHz, 512MO de RAM*.

Ces variantes ont été testées sur un nombre d'instances générées aléatoirement par la loi uniforme et sont comparées à l'algorithme ***BFBB*** afin de mettre en évidence l'évolution de la vitesse de convergence.

Dans un premier lieu, on compare le ***BFBB*** à ***BFBBNB*** afin d'analyser particulièrement l'effet de la nouvelle borne supérieure développée grâce à l'heuristique gloutonne dans la méthode exacte ***BFBB***. Ensuite, on compare le ***BFBB*** aux différentes variantes de l'algorithme. La comparaison est essentiellement basée sur le nombre d'itérations et le temps d'exécution afin de mesurer les gains en nombre d'itérations et en temps de calcul.

6.2. Performance des algorithmes

Nous considérons un ensemble d'exemples générés aléatoirement. Il comporte 37 instances, le nombre total des pièces à regrouper est donné dans l'intervalle [5,16], les dimensions des pièces sont prises dans l'intervalle [1,80]. La borne supérieure sur chaque type de pièce est générée dans [1,8]. Le coût de chaque type de pièces est égal à sa surface.

Borne superieure	*RM (Opt/BS)*	*RM (Opt/NBS)*
	0.81	*0.87*
BFBB	N. It. M	T. M (s)
	177860	4343,249
BFBBNL		T. M (s)
		2059,548
BFBBMD	N. It. M	T. M (s)
	128945	1962,728
BFBBOR	N. It. M	T. M (s)
	59875	707,952
BFBBA	N. It. M	T. M (s)
	53793	687,045

Table 1 : Performances des algorithmes BFBB améliorés

%Gain (BFBBLF/BFBB))		**T. M**
		52,58
%Gain (BFBBMD/BFBB))	**N. It. M**	**T. M**
	27,5	54,8
%Gain (BFBBOL/BFBB))	**N. It. M**	**T. M**
	66,33	83,69
%Gain (BFBBA/BFBB))	**N. It. M**	**T. M**
	69,75	84,18

Table 2 : Gain des algorithmes BFBB améliorés

RM (Opt/BS) : Le Rapport Moyen entre la Solution Optimale et la Borne Supérieure (Alg1) ;

RM (Opt/NBS) : Le Rapport Moyen entre la Solution Optimale et la Nouvelle Borne Supérieure (Alg3) ;

N.It.M : Le nombre moyen d'itérations ;

T.M(s) : Le temps moyen d'exécution (en second).

Les expériences numériques effectuées ont montré une nette amélioration de la vitesse de convergence de la méthode *Best First Branch and Bound Amélioré (**BFBBA**)*; les meilleurs résultats sont de l'ordre de 6*9,75 %* de gain en moyenne sur le nombre d'itérations effectuées et de *84,18 %* de gain en moyenne sur le temps d'exécution comparé à la méthode de ***BFBB***.

La *table 1* donne le nombre moyen d'itérations et le temps moyen d'exécution pour l'algorithme exact ***BFBB***, ainsi que pour les différentes versions proposées et donne

la qualité de la solution obtenue par la procédure gloutonne Algo1 et la qualité de la solution obtenue par la procédure gloutonne Algo3.

D'après les résultats, on constate que la nouvelle borne supérieure *Algo3* est meilleure que l'Algo1, puisqu' elle passe de 0,81 à 0.87 en moyenne.

Dans la table 2, sont représentes les gains obtenus pour chaque version en temps d'exécution et en nombre d'itérations. L'introduction de la nouvelle représentation de la liste fermée (***BFBBLF***) a réduit le temps d'exécution d'une manière très efficace, tel que le gain moyen soit de l'ordre de *52,58%*. Le gain moyen obtenu par la technique du modèle dominé (***BFBBMD***) est de l'ordre de *27,50%* par rapport au nombre d'itération et de *54,80%* par rapport au temps d'exécution. L'application de la technique de l'ordre lexicographique (***BFBBOL***) donne un gain moyen de l'ordre de 66,33% pour le nombre d'itérations et 83,69% pour le temps d'exécution.

6.3. Conclusion

Il est montré d'après les résultats obtenus, que les nouvelles versions permettent de diminuer d'une façon efficace le temps d'exécution et de réduire le nombre d'itérations de l'algorithme considéré. Chaque version agit d'une certaine manière sur le temps d'exécution. La nouvelle borne supérieure permet de réduire l'espace de recherche ainsi que le temps d'exécution. Pour la nouvelle représentation, elle évite les calculs et les tests inutiles. L'ordre lexicographique quand à lui, permet de diminuer la taille des deux listes ξ_1 et ξ_2 utilisées dans l'algorithme en éliminant les modèles de symétrie. La technique du modèle dominé permet de se débarrasser des modèles moins bons.

Enfin, l'algorithme ***BFBBA*** qui est la combinaison de toutes ces techniques, donne une réduction considérable du temps d'exécution et du nombre d'itérations de l'algorithme ***BFBB***.

CONCLUSION GENERALE

Découpe et assemblage sont une famille très célèbre. Ils appartiennent aux problèmes d'optimisation combinatoire. Ils sont essentiellement basés sur l'aspect géométrique, ce qui fait augmenter sa complexité en la classant parmi les problèmes *NP-Complet*.

Ces dits problèmes sont d'un grand intérêt car ils s'appliquent dans divers domaines tels que l'allocation de la mémoire dynamique, les problèmes de la planification multiprocesseurs et les problèmes de la disposition en général.

Le thème de notre mémoire que nous considérons être parmi les plus intéressants des problèmes d'assemblage, s'agissant *d'assemblage orthogonal rectangulaire*. Il puise son rôle dans le champ d'application en modélisant des problèmes rencontrés dans l'industrie et qui est devenu une zone d'attraction pour les chercheurs.

Dans ce mémoire, nous avons exposé les différents aspects relatifs au problème *d'assemblage orthogonal rectangulaire* après sa structura.

La partie essentielle de ce mémoire est basée sur les trois derniers chapitres dans lesquels nous avons exposé la méthode de résolution du problème, les différentes techniques d'amélioration et enfin notre contribution personnelle.

Dans le troisième chapitre, nous avons présenté la méthode exacte de résolution *Best First Branch and Bound* basée sur la séparation et l'évaluation qui utilise une solution de départ réalisable et une stratégie du meilleur d'abord. Cette méthode produit des modèles optimaux mais avec un temps d'exécution trop long. A cet effet, nous nous sommes fixé pour objectif l'amélioration du temps d'exécution dont les techniques sont étalées dans le chapitre quatre existant déjà dans la littérature. A notre tour, nous pensons avoir contribué modestement par un rapport d'une autre technique.

Nous avons adapté les quatre techniques à notre algorithme dans le chapitre cinq. Dans le chapitre six, nous avons testé notre algorithme amélioré sur des instances générées aléatoirement. Les résultats obtenus confirment bien la réduction remarquable du temps d'exécution en comparaissant avec les techniques ultérieures.

Comme perspective, nous souhaitons qu'une étude plus approfondie et dirigée vers le problème d'assemblage rectangulaire dans le cas où l'orientation des pièces rectangulaire n'est pas fixée, que nous souhaitons vivement qu'il puisse être résolu grâce à d'autres travaux.

[**1**] B. S. Baker, E. G. Coffman and R. L. Rives (1980) Orthogonal packing in two dimensions. SIAM Journal of Computing 9 (4), 846-855.

[**2**] B. S. Baker, D. J. Brown and H. P. Katserf (1981). A 5/4 algorithm for two-dimensional packing. Journal of Algorithms 2 (4), 348-368.

[**3**] N. Christofides and C. Whitiock (1977): An algorithm for two-dimensional cutting problems. Operations Research 2, 30-44.

[**4**] E. G. Coffman, and J. C. Lagarias (1989). Algorithms for packing squares: a probabilistic analysis. SIAM Journal of Computing 18 (1), 166-185.

[**5**] V.D. Cung, M. Hifi and B.L. Cun (2000): Constraind two-dimensional cutting stock problems a best-first branch-and-bound algorithm. International Transactions in Operational Research.7 pp 185-210

[**6**] R. Dechter , J.Pearl. (1998). The optimality of A*. Search in Artificial Intelligence, Spinger.

[**7**] A. J. W. Duijvestijn (1978). Simple perfect squared square of lowest order. Journal of Combinatorial Theory 25 (B), 260-263.

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

[**9**] P. Gilmore and R. Gomory (1965): Multistage cutting problems of two and more dimensions. Ops Res. 13, 94-119

[**10**] M. Hifi (1994). Study of some combinatorial optimization problems: cutting stock, packing and set covering problems, Ph. D. Thesis, Université de Paris 1 Panthéon-Sorbonne (in French).

[**11**] M. Hifi, and R. Ouafi (1997). Best-First search and dynamic programming methods for cutting problems: the cases of one more stock plates. Computers and Industrial Engineering 32(1), 187-205.

[**12**] M. Hifi and R. Ouafi (1998): A best-first branch and bound algorithm for orthogonal rectangular packing problem. International Transactions in Operational Research Vol.5, No.5, pp. 345-356

[**13**] M. Hifi and R. M'hallah (2004): An exact algorithm for constraind two dimensional two-staged cutting problems. Operations Research Vol.53, No.1, pp 140-150

[**14**] A.I. Hinxma. (1980). The tri –loss and assortment problems: a survey. European Journal of Operational Research 5 8-18.

[**15**] S. Jakobs (1996). On genetic algorithm for the packing of polygons. European Journal of Operational Research 88, 165-181.

[**16**] L. K. Kantorovich (1960): Mathematical methods of organizing and planning production. Management Science 6,363-422.

[**17**] D.J. Kleitman and M.K. Krieger (1975): An optimal bound for two dimensional bin packing. Proceedings of the 16 th Annual Symposium on foundations of computer Science, pp. 163-168.

[**18**] R. Ouafi (2004). Approche algorithmique pour une classe de problèmes de découpe. Thèse de doctorat d'état en mathématique.

[**19**] P. Sweeney, E. Paternoster (1992). Cutting and Packing problems: a categorized application-oriented research bibliography. Journal of Operation Research Society 43(7), 691-706.

[**20**] P.Y. Wang (1983). Two algorithms for constrained two-dimensional cutting stock problems. Operations Research 31(3). 573-586.

[**21**] G. Wäscher, H. Haußner, H. Schumann, (2004): An Improved Typology of Cutting andPacking Problems, 1st ESICUP Meeting, Lutherstadt Wittenberg, Germany, 18-20 March 2004