Méthodes heuristiques pour le problème de placement sur bande en deux dimensions

Giglia Gomez-Villouta

université angers

Année : 2010
N° d'ordre :1094

MÉTHODES HEURISTIQUES POUR LE PROBLÈME DE PLACEMENT SUR BANDE EN DEUX DIMENSIONS

THÈSE DE DOCTORAT

Spécialité : Informatique (STIM)

ÉCOLE DOCTORALE STIM, ED 503

Présentée et soutenue publiquement

Le 21 Septembre 2010

À Angers

Par **Giglia Gómez-Villouta**

Devant le jury ci-dessous :

Rapporteurs :	Bertrand NEVEU,	HDR, Ecole des Ponts Paristech
	Michel VASQUEZ,	HDR, Ecole des Mines d'Alès
Examinateur :	Chu Min LI	Professeur, Université de Picardie Jules Verne, Amiens
Directeur de thèse :	Jin-Kao HAO,	Professeur, Université d'Angers
Co-encadrant :	Jean-Philippe HAMIEZ,	Maître de Conférences, Université d'Angers

Laboratoire d'Étude et de Recherche en Informatique d'Angers
2, boulevard Lavoisier, 49045 Angers CEDEX 01

Remerciements

Cette partie de la thèse est à la fois facile et difficile à rédiger. Facile, parce qu'exprimer des sentiments et des remerciements est quelque chose de naturel et que cela fait du bien. Difficile, car tant de gens m'ont aidé que je pourrais oublier d'en citer certains. Cela fut une expérience très enrichissante de venir étudier dans un pays étranger. J'y ai rencontré des professionnels de haut niveau qui ont partagé leur expertise avec moi. D'autres personnes m'ont accompagné différemment, en me faisant découvrir la culture, la langue et l'art de vivre à la française, ou en m'apportant amitié et conseils... Tant de choses.

Je remercie donc infiniment mon directeur de thèse, Jin-Kao Hao, pour son professionnalisme, ses précieux conseils, sa sagesse et ses grandes qualités humaines.

J'adresse également mes remerciements à Jean-Philippe Hamiez, co-encadrant, pour sa disponibilité, le partage de ses connaissances et l'amélioration du manuscrit.

Merci à Bertrand Neveu et Michel Vasquez, rapporteurs de ma thèse, pour avoir accepté de donner leur avis éclairé sur mes travaux de recherche. Je remercie Chu-Min Li d'avoir présidé le jury de cette thèse.

Mes remerciements vont aussi au personnel du laboratoire LERIA pour leur disponibilité, en particulier les secrétaires et les techniciens.

Un grand merci à Marcela et Laetitia, mes amies, pour leur soutien inconditionnel, les rires et les larmes partagés, les cigarettes, les nuits de travail passées ensemble pendant la rédaction...

À Juan et Alvarita, mes parents, parce qu'ils sont le manteau qui m'abrite à chaque instant. À Yorka, ma soeur, pour être toujours présente. Je vous aime.

Je dédie finalement tout ce travail et cette jolie expérience à Sofía, ma fille, qui est toute ma vie : Hija, este esfuerzo es por ti, te amo... (Tu mamá).

Table des matières

Introduction générale

Contexte

Les problèmes d'optimisation combinatoire sont des problèmes dont les solutions peuvent se trouver dans un ensemble de configurations de grande taille. Il s'agit de trouver une meilleure solution en vérifiant un critère particulier défini par une fonction objectif. Beaucoup de problèmes d'optimisation combinatoire sont NP-difficiles [Garey and Johnson, 1979], car l'ensemble des configurations possibles peut être d'une taille telle que son énumération exhaustive n'est pas envisageable en temps raisonnable.

Voici quelques exemples de problèmes d'optimisation combinatoire classiques : l'affection quadratique, les problèmes de découpe ou de placement, le sac à dos, la couverture d'ensemble quadratique, les problèmes de conception d'emplois du temps, le problème du voyageur de commerce.

L'objet de cette thèse est l'étude et la résolution d'un problème de placement en deux dimensions connu sous le nom de "strip packing problem" SPP.

Le SPP considère un ensemble d'objets de formes rectangulaires à placer dans un espace bidimensionel "strip" (bande) en minimisant la hauteur. Dans ce problème, une solution est qualifiée de "parfaite" quand il n'y a pas d'espace vide.

Les méthodes de résolution qui existent pour ce type de problème peuvent être classées en deux grands groupes : les méthodes exactes et les méthodes approchées.

Les méthodes exactes consistent à effectuer une énumération implicite de toutes les solutions (méthode envisageable pour les instances de petites tailles). Ces méthodes garantissent de trouver une solution optimale. Comme méthode exacte pour SPP, la technique la plus fréquemment utilisée est le "Branch and Bound" [Fekete and Schepers, 1997a; Lesh *et al.*, 2004; Martello *et al.*, 2003; Belov *et al.*, 2009; Clautiaux *et al.*, 2008; Soh *et al.*, 2008].

Les méthodes approchées consistent à chercher une solution de bonne qualité dans un contexte de ressources limitées, (temps de calcul ou mémoire par exemple). Les méthodes approchées pour SPP pouvent être clasées en deux groupes : les algorithmes gloutons et les méta-heuristiques.

L'algoritme glouton le plus utilisé pour SPP est l'algorithme BLF de complexité $O(n^3)$ [Baker *et al.*, 1980] à $O(n^2)$ [Chazelle, 1983] selon l'implémentation. Cet algorithme utilise un critère de placement qui prend un par un les objets à placer jusqu'à placer le dernier. La solution finale trouvée par l'algorithme, dans la majorité des cas, n'est pas très proche de l'optimum. En voici quelques variantes : BLD [Hopper and Turton, 2001a], BF [Burke *et al.*, 2004], BFDH [Mumford-Valenzuela *et al.*, 2003], $BFDH^*$ [Bortfeldt, 2006].

Dans les méta-heuristiques appliquées au SPP, on peut citer des algorithmes de recherche locale comme la recherche tabou (RT) ou le recuit simulé (RS), des approches à base de population dont les algorithmes génétiques (AG) et des méthodes hybrides (utilisant BLF ou ses variantes avec AG, RT, RS).

Un autre point important est la représentation utilisée pour modéliser et résoudre le SPP. Nous pouvons citer la représentation à l'aide de graphes [Fekete and Schepers, 1997a], d'arbres

binaires [Imahori *et al.*, 2006], de permutations (certainement la plus utilisée) [Jakobs, 1993; Soke and Bingul, 2006; Yeung and Tang, 2004; Burke *et al.*, 2006; Gomez and de la Fuente, 2000; Lesh *et al.*, 2004] et la représentation directe [Hamiez *et al.*, 2009].

Motivations et objectifs

Les méta-heuristiques (dont la recherche locale) sont les méthodes les plus utilisées et les plus efficaces pour la résolution approchée du SPP.

L'algorithme génétique AG est probablement la méta-heuristique la plus utilisée pour SPP. Il existe même des AG hybrides avec des décodeurs, comme par exemple BLF [Hopper and Turton, 2001a; Iori *et al.*, 2003; Soke and Bingul, 2006]. Les composants les plus utilisés sont la représentation par permutations et des opérateurs de croisement génériques de connaissance spécifique du SPP. Notre première motivation a été de développer un opérateur de croisement dédié au SPP appelé "WAX" (pour "wasted area based crossover") dans un algorithme génétique hybride avec BLF comme algorithme décodeur.

Les algorithmes hybrides de type AG utilisent des fonctions d'évaluation concentrées sur l'objectif de minimisation de la hauteur. Ces fonctions sont très générales, car elles ne sont pas capables de distinguer deux solutions différentes mais avec la même hauteur. Notre deuxième motivation a été de créer une fonction d'évaluation hiérarchique pour distinguer les placements de même hauteur. Dans ce cas là, une deuxième fonction analyse la distribution des objets dans la configuration.

Les méthodes les plus efficaces pour le SPP sont l'algorithme GRASP développé par Alvarez-Valdes et al. [Alvarez-Valdes *et al.*, 2008b] et "Intensification Diversification Walk" (IDW) [Neveu *et al.*, 2008].

En général, les algorithmes de recherche locale développés pour SPP ont une fonction d'évaluation qui mesure la qualité d'une configuration **complète** (i.e. quand les objets sont placés). De même, le voisinage utilisé est rarement spécifique au SPP. Finalement, un composant très intéressant et qui n'a pas été très utilisé, est la diversification. Notre dernière motivation a été de développer un algorithme tabou avec une fonction d'évaluation capable de mesurer la qualité d'une solution **partielle**, avec un voisinage consistant et spécifique au problème et avec un mécanisme de diversification original basé sur l'historique de la recherche.

Principales contributions

Les principales contributions réalisées au cours de cette thèse concernent la conception de deux algorithmes avec des composants novateurs et une analyse de résultats comparatifs avec les algorithmes les plus fréquents et efficaces.

Le premier algorithme développé est un algorithme génétique hybride DGA ("Dedicated Genetic Algorithm") [Gómez-Villouta *et al.*, 2007]. Il utilise la représentation par permutations et BLF comme décodeur. DGA repose en particulier sur deux composants nouveaux : un opérateur de croisement spécifique au problème appelé WAX (pour "Wasted Area based crossover") et une fonction d'évaluation hiérarchique composée de deux sous-fonctions.

L'opérateur WAX considère les espaces vides et a pour objectif de tenter de reconnaître les bonnes zones de placement dans la permutation, c'est à dire, les zones où les rectangles sont bien placés. WAX repose sur des connaissances spécifiques au SPP alors que les autres opérateurs de croisements classiquement appliqués au SPP (OBX, CX, OX, PMX, UX et SJX [Soke and Bingul, 2006]) sont génériques à de nombreux autres problèmes pouvant se modéliser aisément à l'aide de permutations.

La fonction d'évaluation hiérarchique est composée de deux sous-fonctions. La première guide la recherche et mesure la qualité des individus de différentes hauteurs. Ensuite, quand deux individus différents ont la même hauteur nous appliquons une seconde fonction qui se concentre sur les grands espaces vides.

Le deuxième algorithme développé est un algorithme de recherche tabou appelé "CTS" (pour "Consistent Tabu Search") [Gómez-Villouta *et al.*, 2010]. CTS repose en particulier sur trois composants :

- Un voisinage consistant composé de solutions éventuellement partielles (avec des objets restant à placer).
- Une fonction d'évaluation de placements, éventuellement partiels, utilisant des connaissances du problème.
- Un mécanisme de diversification reposant sur l'historique de la recherche.

Organisation du manuscrit

Dans le premier chapitre nous décrivons les problèmes de placement et de découpe ainsi que leurs caractéristiques principales. Nous définissons le cas spécifique du SPP et effectuons un état de l'art orienté sur le SPP.

Dans le deuxième chapitre, nous détaillons un nouvel algorithme génétique (AG) nommé DGA pour le SPP. Dans une première partie, nous décrivons l'état de l'art par rapport aux AG appliqués au SPP. Ensuite, nous expliquons l'algorithme DGA et ses composants : représentation par permutations, hybridation avec l'algorithme de placement BLF, opérateur de croisement WAX spécifique au problème, et une fonction hiérarchique. Finalement nous présentons et analysons les résultats obtenus.

Dans le troisième chapitre, nous étudions les méthodes de recherche locale appliquées au SPP. On trouvera ici les dernières techniques développées pour le SPP et les plus efficaces. Finalement, nous nous concentrons sur les algorithmes de recherche tabou.

Le dernier chapitre est dédié à la présentation de CTS, un nouvel algorithme de recherche tabou pour le SPP. Les principales caractéristiques de CTS sont : une représentation directe, une fonction d'évaluation capable de mesurer la qualité des solutions complètes ou partielles, un voisinage consistant et un mécanisme de diversification original. Finalement, nous présentons et analysons les résultats obtenus.

Nous terminons par une conclusion générale dans laquelle nous faisons un résumé de nos contributions et apportons quelques perspectives de recherche.

Chapitre 1

Les problèmes de placement et de découpe

Sommaire

Les problèmes de placement et de découpe ont de nombreuses applications dans la vie réelle : publicité, industrie textile, entreprises forestières, industrie maritime etc. Ces problèmes combinatoires sont généralement NP-difficiles ou NP-complets [Garey and Johnson, 1979]. La grande variété de ces problèmes (formes des objets ou des containers, objectifs à atteindre, contrainte “guillotine”, possibilité de rotation ou non...) a naturellement conduit à diverses classifications [Wäscher *et al.*, 2007; Dykhoff, 1990]

Dans ce chapitre nous parlons de la classification des problèmes de placement et de découpe, en particulier du SPP, centre d’intérêt de cette thèse. Ensuite nous faisons une description globale des différentes techniques de résolution existantes.

1.1 Introduction

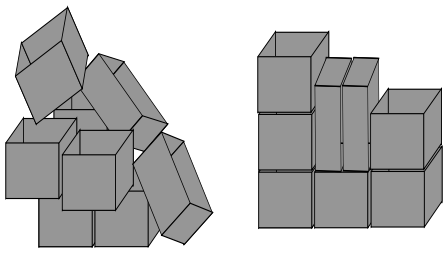


FIGURE 1.1 – Des objets (à gauche) et un placement possible (à droite)

Les problèmes de placement considèrent un ensemble d'objets à placer dans un ou plusieurs "containers" (aussi nommés bande, strip, bin (récipient), entre autres) de dimensions fixes ou libres, en respectant un ensemble de contraintes. Une application est par exemple l'optimisation du placement de publicités dans un magazine sur une ou plusieurs pages. Il faut placer une certaine quantité d'annonces publicitaires. L'idée est de calculer la capacité de chaque page, la taille des annonces, et la quantité minimale de pages à utiliser. Ils sont généralement NP-difficiles ou NP-complets suivant l'objectif à atteindre [Fowler *et al.*, 1981; Garey and Johnson, 1979]. Par exemple, si nous devons remplir une cave avec des cartons, nous pouvons le faire en les prenant dans un ordre quelconque (cf. fig. 1.1). Une fois que la cave a été remplie, s'il y a des cartons restant à stocker, nous devons les ranger dans une autre cave. Dans cet exemple on peut avoir plusieurs limitations comme la quantité de caves disponibles, les capacités des caves, l'orientation des cartons etc.

Les problèmes de placement varient donc selon les caractéristiques des objets (aussi nommés rectangles, pièces, items entre autres). Les principales sont la dimension du problème (1D, 2D, 3D ou plus), et les contraintes.

Dans ce chapitre nous présentons succinctement les différentes caractéristiques des problèmes de placement. Nous décrivons aussi les travaux récents portant sur la résolution de ces problèmes. Dans la section 1.2 nous donnons la définition générale d'un problème particulier de "Strip packing" en deux dimensions (SPP) étudié dans cette thèse. Ensuite, nous décrivons les types de méthodes appliquées au SPP (exactes et approchées) avant de conclure.

1.2 Un aperçu des problèmes de placement

Les problèmes de placement appartiennent à une classe plus générale de problèmes connus sous le nom de "Cutting and Packing" ("Découpe et Placement"). À priori un problème de placement est facile sauf s'il y a un objectif à atteindre. Choisir quelle pièce à placer dans quel container est parfois un problème combinatoire fort. Ce type de problème a généralement une complexité exponentielle (problèmes NP-difficiles ou NP-complets) [Fowler *et al.*, 1981; Garey and Johnson, 1979].

Quelques contraintes ou caractéristiques liées aux objets :

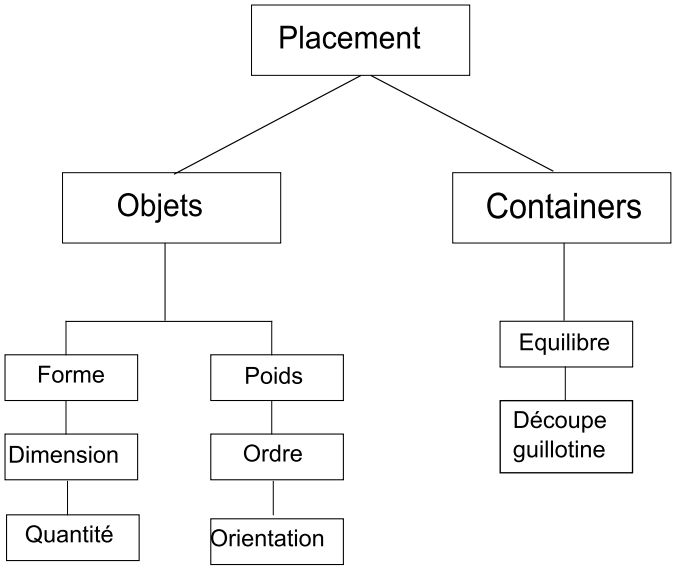


FIGURE 1.2 – Quelques contraintes ou caractéristiques d'un problème de placement

- Forme : les objets peuvent être réguliers (rectangles, cercles, etc.) ou irréguliers.
- Dimension : la largeur et hauteur (pour un rectangle), rayon (pour un cercle) etc.
- Quantité : nous pouvons trouver plusieurs objets de mêmes formes et de mêmes dimensions (répliques).
- Poids : le poids de l'objet peut être important selon la classe de problème, parce que cette caractéristique peut ajouter de la difficulté à la priorité de placement de l'objet. Par exemple, dans un container destiné à un déménagement, les objets fragiles comme la vaisselle ou le petit électroménager, doivent être placés sur les objets les plus résistants comme des tables ou des commodes.
- Ordre : l'ordre des objets peut être important dans le cas du transport. Par exemple, la classification des colis de la poste pour les transporter à différentes adresses.
- Orientations : une orientation (libre ou restreinte) peut être autorisée ou non sur les objets.

Quelques contraintes ou caractéristiques liées aux containers :

- Équilibre : les objets placés dans le(s) container(s) ayant différentes caractéristiques (dimension, poids, forme etc.), ceux-ci peuvent affecter l'équilibre à l'intérieur du (des) container(s).
- Découpe guillotine : cette contrainte impose un motif pouvant, par la suite, être "sectionné" par des découpes dites "de bout à bout" (i.e. parallèles aux côtés du container) cf. fig. 1.3.

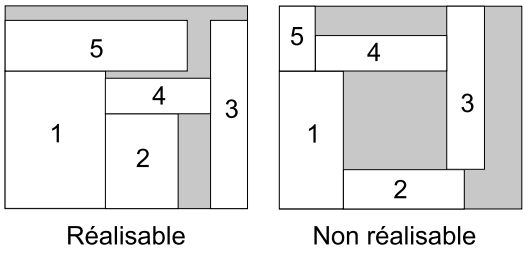


FIGURE 1.3 – La contrainte de découpe guillotine

1.3 Le problème de placement sur bande (strip packing problem) en deux dimensions SPP

Le problème de "Strip Packing" (placement sur bande SPP), considère un ensemble de rectangles à placer dans une bande (strip) de largeur donnée W (pour "width") et de hauteur H (pour "Height") infinie (cf. figure 1.4). Tous les rectangles doivent être stockés sans chevauchement, cf. figure 1.5, tels que leurs côtés soient parallèles aux bords de la bande. La rotation des rectangles est interdite et la contrainte de découpe guillotine n'est pas imposée. L'objectif est de minimiser la hauteur de la bande après le placement de tous les rectangles.

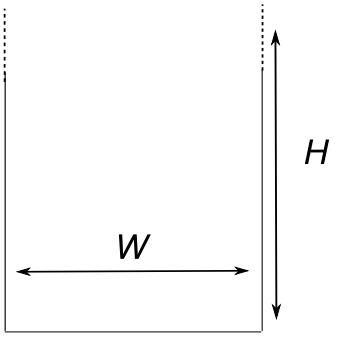


FIGURE 1.4 – Les deux principales caractéristiques d'une bande

1.3.1 Définition du SPP

Soit P un espace bidimensionnel (i.e. un plan muni d'un repère $x - y$) de largeur fixe W et de hauteur infinie H. L'origine de P est le point de coordonnées $(0, 0)$ l'ensemble des $n > 1$ objets *rectangulaires* à placer dans P est $R = \{r_1, \ldots, r_n\}$. La largeur (resp. hauteur) de chaque $r_i \in R$ est $0 < w_i^r \leq W$ (resp. $h_i^r > 0$).

Il s'agit donc de déterminer les coordonnées (x_i^r, y_i^r) du coin bas gauche de tous les $r_i \in R$ en minimisant la valeur $y_i^r + h_i^r$ de l'objet r_i placé le plus haut dans P. Plus formellement :

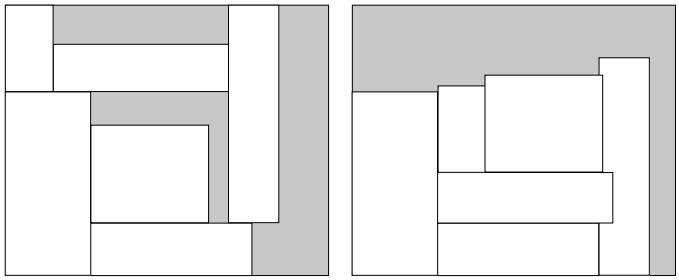


FIGURE 1.5 – Placement de rectangles sans et avec chevauchement

$$\text{Minimiser :} \quad H = \max_{1 \leq i \leq n} (y_i^r + h_i^r) \tag{1.1}$$

$$\text{Sujet à :} \quad 0 \leq x_i^r \leq W - w_i^r, y_i^r \geq 0 \tag{1.2}$$

$$\text{et } (x_i^r \geq x_j^r + w_j^r \text{ ou } x_i^r + w_i^r \leq x_j^r \tag{1.3}$$

$$\text{ou } y_i^r \geq y_j^r + h_j^r \text{ ou } y_i^r + h_i^r \leq y_j^r) \tag{1.4}$$

où 1.2 force chaque r_i à être placé dans P et 1.3 et 1.4 assurent le non chevauchement de r_i et r_j ($i \neq j$).

1.3.2 Différentes modélisations

Il existe différentes façons de modéliser le SPP ou ses proches variantes.

Nous rappelons ici les représentations par permutations ou à l'aide de graphe(s), mais il en existe d'autres, par exemple paire de permutations, arbre binaire, représentation directe etc. [Imahori *et al.*, 2006; Ibaraki *et al.*, 2007; Fekete and Schepers, 1997a; Hamiez *et al.*, 2009]. Cette dernière est détaillée dans les chapitres 3 et 4.

Représentation par permutation

On considère ici une permutation π des objets à placer, ordonnés aléatoirement ou triés selon différents critères (ordre croissant ou décroissant) comme, par exemple : longueur, hauteur, aire, périmètre, etc. (cf. figure 1.6).

Les objets sont ensuite placés dans la bande (initialement vide) en respectant l'ordre imposé par π à l'aide d'un algorithme dit "décodeur". De nombreux décodeurs existent [Neveu *et al.*, 2007; Burke *et al.*, 2009; Soke and Bingul, 2006; Hopper and Turton, 2001a; Gomez and de la Fuente, 2000; Yeung and Tang, 2004; Neveu *et al.*, 2008; Bortfeldt, 2006], expliqués dans la section 1.4.2. En particulier, l'algoritme BLF, (pour "Bottom Left Fill") [Hopper and Turton, 2001a] est utilisé dans le chapitre 2.

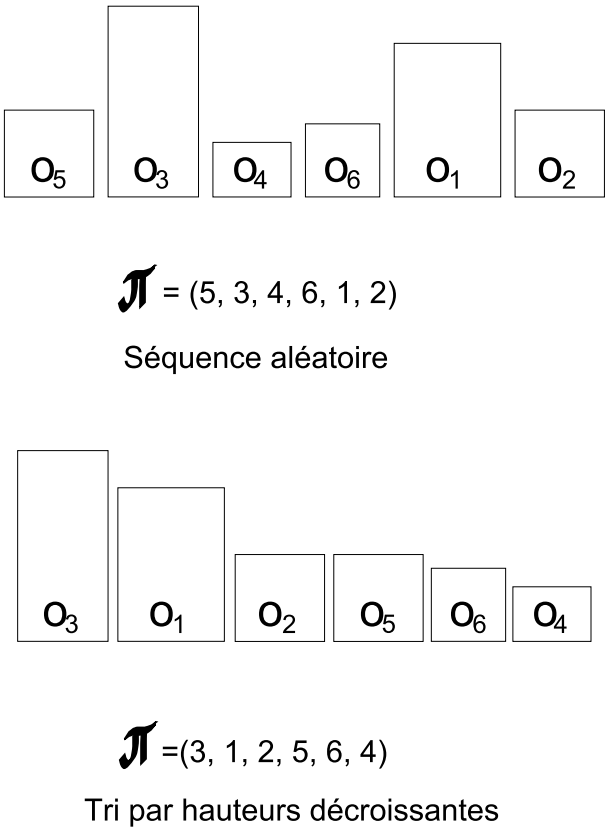


FIGURE 1.6 – Deux exemples de permutations

La représentation par permutations est certainement la plus utilisée [Jakobs, 1993; Soke and Bingul, 2006; Yeung and Tang, 2004; Burke *et al.*, 2006; Gomez and de la Fuente, 2000; Lesh *et al.*, 2004]. La figure 1.7 montre un exemple de permutation et la solution construite avec l'algorithme BLF.

Modélisation sous forme de graphes

Cette représentation [Fekete and Schepers, 1997a] repose sur la notion de graphe d'intervalle : après projections horizontale et verticale des objets, le sommet v_i (associé à l'objet r_i) est relié à v_j si les intervalles correspondants ont une intersection non vide, cf. fig. 1.8.

1.4 Un aperçu des méthodes de résolution existantes

Nous rappelons ici les principales méthodes exactes (cf. 1.4.1) ou approchées (1.4.2-1.4.6) disponibles pour le SPP.

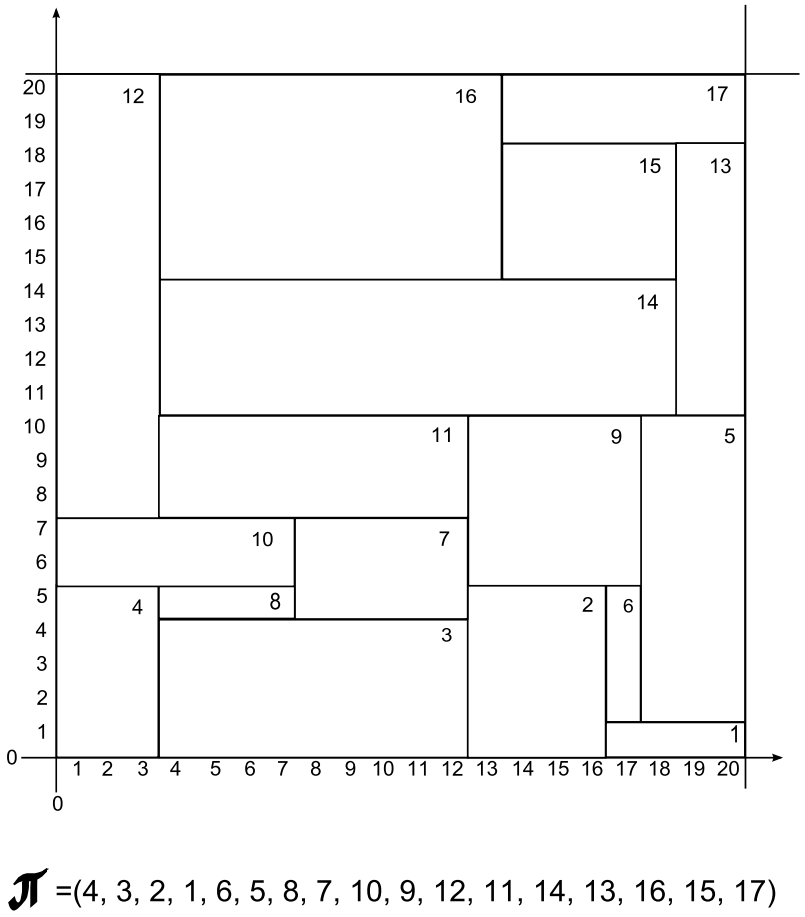


FIGURE 1.7 – Une permutation π et le motif associé après décodage par BLF

1.4.1 Méthodes exactes

Les approches exactes sont en général limitées à de petites instances [Lesh *et al.*, 2004].

Fekete and Schepers [Fekete and Schepers, 1997a] ont proposé un algorithme de type “branch and bound” (B&B) pour résoudre différents problèmes de placement en deux dimensions ou plus. Ils abordent en particulier un problème de type “sac à dos” qui est une généralisation du SPP.

Lesh et al. [Lesh *et al.*, 2004] utilisent également le B&B pour résoudre des instances “parfaites” (aucun espace vide dans les solutions optimales), avec ou sans rotation.

L’approche B&B de Martello et al. [Martello *et al.*, 2003] résout certaines instances jusqu’à 200 rectangles. Cet algorithme propose de nouvelles “bornes” obtenues par des considérations géométriques et une relaxation du problème.

Voir également [Beasley, 1985b; Alvarez-Valdes *et al.*, 2008a; Kenmochi *et al.*, 2009; Belov *et al.*, 2009; Clautiaux *et al.*, 2008; Soh *et al.*, 2008; Lesh *et al.*, 2004] par exemple pour d’autres méthodes exactes.

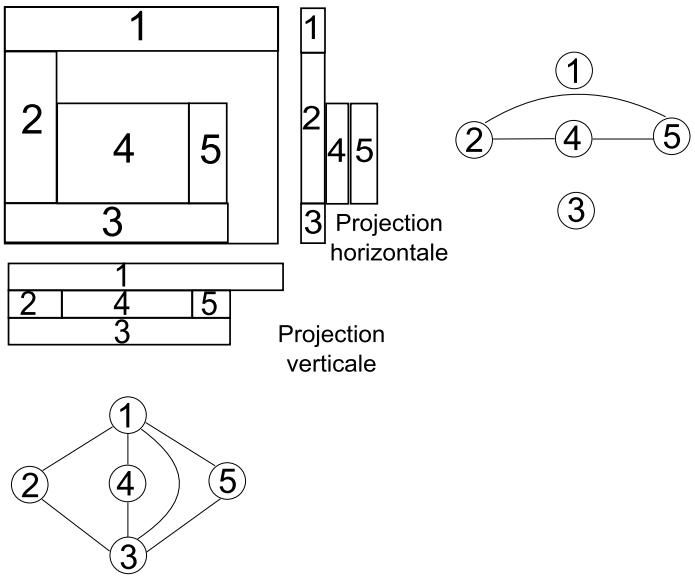


FIGURE 1.8 – Représentation par graphes d'intervalles

1.4.2 Algorithmes gloutons

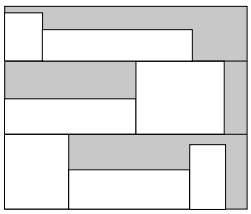


FIGURE 1.9 – Placement d'objets par niveaux

Les premières heuristiques approchées développées pour résoudre le SPP correspondent aux algorithmes de placement précédemment évoqués en 1.3.2 (décodeurs liés à la modélisation par permutations).

Il existe des algorithmes qui placent les objets par niveaux ("level algorithms"). Le premier niveau est la base de la bande et les suivants sont créés par superposition sur les précédents, cf. fig 1.9.

Ces algorithmes, habituellement de complexité O(n log n), respectent généralement la contrainte guillotine.

Concernant les algorithmes qui placent les objets sans la formation de niveaux ("non-level algorithms"), le plus utilisé est l'algorithme BL (bottom-left) de [Baker *et al.*, 1980], cf. algorithme

1.1.

Algorithme 1.1: Pseudo-code de l'algorithme BL

répéter
  Prendre le rectangle suivant
  Placer le rectangle dans le coin supérieur droit de la bande
  répéter
    Descendre le rectangle à la verticale le plus possible (sans chevauchement)
    Glisser horizontalement le rectangle le plus possible à gauche
  jusqu'à *Plus de déplacement (horizontal ou vertical) possible*;
jusqu'à *tous les rectangles sont placés*;

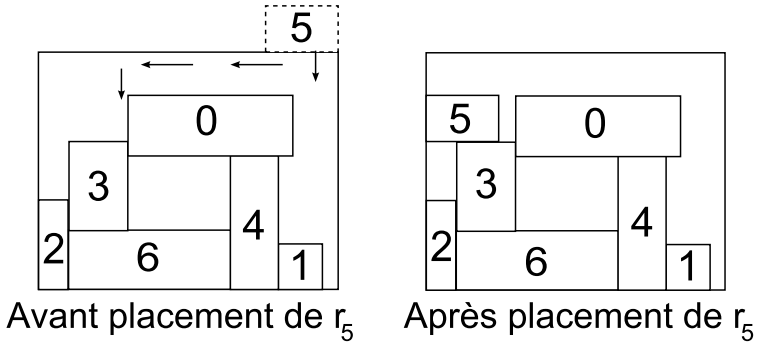


FIGURE 1.10 – Un exemple d'exécution de BL

La figure 1.10 montre le placement des objets dans la permutation [2, 6, 4, 3, 0, 1, 5]. Le principal problème de cet algorithme est la création d'espaces vides inutilisables. La complexité de cet algorithme est en $O(n^3)$. Cette méthode a été améliorée par Chazelle [Chazelle, 1983], sous le nom de BLF, de complexité $O(n^2)$. La différence par rapport au premier algorithme est la possibilité de placer une pièce dans un espace vide enclavé, cf. Fig, 1.11.

D'autres variations de BL existent : dans BLD [Hopper and Turton, 2001a] le critère de tri repose sur la largeur, la longueur, la surface etc. Un tri dynamique est utilisé dans BF (pour "Best-Fit") de Burke et al. [Burke *et al.*, 2004].
Voir aussi [Neveu and Trombettoni, 2008; Lesh *et al.*, 2004; Lesh *et al.*, 2005; Mumford-Valenzuela *et al.*, 2003; Bortfeldt, 2006], par exemple, pour d'autres variantes.

1.4.3 Recherche locale

Comme méthodes approchées la recherche locale est une des méthodes la plus eficace appliquée au SPP.

Alvarez-Valdes et al. [Alvarez-Valdes *et al.*, 2005] ont proposé un algorithme "Greedy Randomized Adaptative Search Procedure" (GRASP). Cet algorithme, qui ne permet pas la rotation, utilise différents critères de placement pour décider quelle est la meilleur pièce suivante à placer.

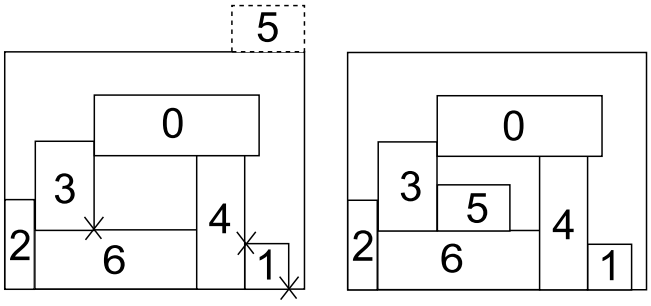


FIGURE 1.11 – Un exemple d'éxecution de BLF

Alvarez-Valdes et al. [Alvarez-Valdes *et al.*, 2008b] ont développé une approche de type "Reactive GRASP" résumée dans l'Algo. 1.2. Il utilise une première fonction d'évaluation pour décider quel mouvement faire, et une deuxième fonction d'évaluation particulière pour distinguer les cas similaires.

Les résultats obtenus par cet algorithme sont parmi les meilleurs, sur différentes catégories d'instances.

Alvarez-Valdes et al. [Alvarez-Valdes *et al.*, 2007] ont également développé un algorithme de RT qui travaille avec une solution initiale créée par l'algorithme GRASP [Alvarez-Valdes *et al.*, 2005]. Notons que les auteurs considèrent ici un problème de découpe légèrement différent du SPP. L'approche serait néanmoins applicable aux instances qualifiées de "parfaites" (i.e. sans espace vide dans les solutions optimales), comme celles de [Hopper and Turton, 2001a] considérées dans ce manuscrit.

L'algorithme tabou est capable d'améliorer les résultats obtenus par GRASP à travers deux voisinages : "Block reduction" (Algo.1.3) et "Block insertion" (Algo. 1.4). Cet algorithme travaille avec $R = (L, W)$ une bande de longueur L et largeur W. Un ensemble de pièces à placer $\mathcal{P}$, un ensemble d'espaces vides $\mathcal{L}$ et un ensemble de blocs $\mathcal{B}$[1]. Chaque pièce i est caractérisée par sa largeur w_i et sa hauteur l_i avec orientation fixe et une quantité x_i de copies de chaque pièce tel que $0 \leq P_i \leq x_i \leq Q_i$ bornes inférieures et supérieures respectivement.

Dans le voisinage "Block reduction", l'idée est d'éliminer un bloc de pièces pour générer un espace vide plus grand dans $\mathcal{L}$ (étape 1), de déplacer certains blocs et de recombiner les espaces vides restants (étape 2), de remplir les espaces vides avec de nouveaux blocs et de mettre à jour la liste $\mathcal{L}$ avec les nouveaux espaces vides générés (étape 3). Finalement, combiner les blocs avec côtés communs pour faciliter les configurations de découpe non guillotine.

Dans le voisinage "Block insertion", l'idée est de sélectionner un bloc pour être inseré dans un espace vide (étape 1 et 2). Dans le cas d'un chevauchement, les pièces en conflit sont retirées pour finalement, mettre à jour les listes des espaces vides et blocs (étape 3).

Neveu et al. [Neveu *et al.*, 2004] utilisent la stratégie "Intensification, Diversification Walk" (IDW) qui réalise des mouvements d'insertion/suppression (de rectangles) à l'aide de procédures particulièrement efficaces, (qualifiées d'*incrémentales*).

1. Des pièces du même type peuvent apparaître groupées dans des blocs rectangulaires.

Algorithme 1.2: Reactive GRASP [Alvarez-Valdes *et al.*, 2008b]

Initialisation : Soit δ un paramètre à être déterminé ($0 < \delta < 1$) Soit
$D = \{0.1, 0.2, ..., 0.9\}$, l'ensemble de valeurs possibles pour δ
Soit S_{best} et S_{worst} la meilleure et la moins bonne solution respectivement
$S_{best} \leftarrow \infty; S_{worst} \leftarrow 0$
$n_{\delta^*} \leftarrow 0$, nombre d'itérations avec $\delta^*, \forall \delta^* \in D$
$Sum_{\delta^*} \leftarrow 0$, somme des évaluations des solutions obtenues avec δ^*
$P(\delta \leftarrow \delta^*) \leftarrow p_{\delta^*} \leftarrow 1/|D|, \forall \delta^* \in D$
$numIter \leftarrow 0$
tant que $(numIter < maxIter)$ **faire**
  Choisir δ^* depuis D avec une probabilité p_{δ^*}
  $n_{\delta^*} = n_{\delta^*} + 1$
  Appliquer la Phase Constructive avec δ^* pour obtenir la solution S
  Appliquer la Phase d'Améliorations pour obtenir la solution $S^{'}$
  si $S^{'} < S_{best}$ **alors**
    $S_{best} \leftarrow S^{'}$
  fin
  si $S^{'} > S_{worst}$ **alors**
    $S_{worst} \leftarrow S^{'}$
  fin
  $Sum_{\delta^*} \leftarrow Sum_{\delta^*} + S^{'}$
  si $mod(numIter, 200) == 0$ **alors**
    $eval_\delta = \frac{S_{worst} - mean_\delta}{S_{worst} - S_{best}}^{\alpha} \forall \delta \in D \; p_\delta = \frac{eval_\delta}{\sum_{\delta^{'} \in D} eval_{\delta^*}}$
  fin
fin

Algorithme 1.3: Le voisinage Block reduction de Alvarez-Valdes

- Étape 0. Initialisation :
  $\mathcal{B} \leftarrow$ liste des blocs dans la solution actuelle.
  $\mathcal{L} \leftarrow$ liste des espaces vides.
- Étape 1. Choisir le bloc à réduire :
  prendre B, un bloc de $\mathcal{B}$ avec k colonnes et l files de pièces p_i
  sélectionner le nombre r de colonnes (files) à éliminer
  $1 \leq r \leq k (1 \leq r \leq l)$,
  laisser le nombre de pièces dans la solution $x_i \geq P_i$
  Si $P_i = 0$, le bloc pourrait complètement disparaître.
  le nouveau espace vide R est ajouté à $\mathcal{L}$.
- Étape 2. Déplacer certains blocs :
  la liste des espaces vides $\mathcal{L}$ est mise à jour
- Étape 3. Remplir les espaces vides avec les nouveaux blocs :
  appliquer l'algorithme constructif de [Alvarez-Valdes *et al.*, 2005]
  L'algorithme commence depuis les listes actualisées $\mathcal{L}$ et $\mathcal{B}$, et $\mathcal{P}$ contient les pièces qui peuvent encore être placées et ajoutées à la solution courante. Avant de passer au processus de construction, les espaces vides dans $\mathcal{L}$ sont considérés pour arranger de la meilleure manière les pièces de $\mathcal{P}$. La pièce éliminée dans le étape 1 n'est pas considérée jusqu'à qu'une autre pièce soit incluse dans le solution modifiée.
- Étape 4. Combiner les blocs avec la même structure :
  on essaye de combiner les blocs des mêmes pièces avec les mêmes longueurs ou les mêmes largeurs s'ils sont adjacents et ont un côté commun, ou si l'un d'eux peut-être bougé pour être adjacents d'un côté commun.

Algorithme 1.4: Le voisinage Block insertion de Alvarez-Valdes

- Étape 0. Initialisation :
  $\mathcal{B} \leftarrow$ liste des blocs dans la solution courante.
  $\mathcal{L} \leftarrow$ liste des espaces vides.
- Étape 1. Choisir le bloc à insérer :
  prendre p_i, une pièce dont $x_i < Q_i$ et considérer un bloc de cette pièce avec k colonnes et l files $(k * l \leq Q_i - x_i)$
- Étape 2. Sélectionner la position pour insérer le nouveau bloc.
- Étape 3. Déplacer les pièces de la solution qui présentent un chevauchement avec le bloc inséré :
  mettre à jour $\mathcal{B}$ (dans le cas de quelques blocs réduits ou éliminés)
  mettre à jour $\mathcal{L}$ (dans le cas de nouveaux espaces vides générés)
- Étape 4. Remplir l'espace vide avec le nouveau bloc (même cas du voisinage "block reduction" étape 4).
- Étape 5. Combiner les blocs avec la même structure (même cas du voisinage "block reduction" étape 5).

Cette stratégie de placement présente le concept de “maximal holes” (trous maximaux) : espaces vides maximaux où l'on peut placer les rectangles sans nécessairement respecter le critère BL [2]. Un mouvement consiste à déplacer un rectangle depuis le haut de la bande vers un trou maximal ou à l'emplacement d'un autre rectangle. En cas de chevauchement, les rectangles en conflit sont retirés puis replacés.

La procédure ADD Rectangle (Algorithme 1.5) travaille avec un ensemble de rectangles R à placer, une bande C et un ensemble d'espaces vides maximaux S. Initialement, on considère S comme un seul espace vide. Ensuite, on sélectionne un rectangle en R pour placer en C. Immédiatement s'applique un opérateur pour récupérer l'espace vide restant en S et ne garder que les espaces vides maximaux.

La procédure “RemoveRectangle” (Algorithme 1.6) est plus complexe, car elle récupère les espaces vides maximaux produits quand un rectangle est retiré de la bande. Ensuite, la liste S est mise à jour, R est remplacé par un trou H et l'opération Plus est appliquée à H ainsi qu'aux éventuels espaces vides contigus à H. Un processus ensuite est appliqué pour obtenir les espaces vides maximaux résultants de l'ajout de l'espace vide généré par le rectangle retiré R et les espaces vides autour.

Algorithme 1.5: Algorithme ADD Rectangle de l'opérateur incrémental IDW

ADD Rectangle (in R, in/out C, in/out S)

1. Ajouter R dans le container C
2. Pour chaque trou de S intersectant R, ajouter dans un ensemble *setH* les trous retournés par Minus(H, R)
3. Filtrer *setH* et ne garder que les trous *maximaux*

Finalement, Hamiez et al. [Hamiez *et al.*, 2009] proposent un algorithme tabou incluant divers composants spécifiques : diversification, voisinages complémentaires et critères d'aspiration par exemple.

1.4.4 Algorithmes à base de population

En général dans la application des algorithmes à base de population pour le SPP on commence par un placement initial construit par un algorithme décodeur (BL, BLF, etc.) et puis amélioré itérativement.

Jakobs [Jakobs, 1993] utilise la méta-heuristique AG avec le décodeur BL.

Concernant les problèmes avec ou sans rotation, [Bortfeldt, 2006] a développé l'AG certainement le plus efficace pour SPP. Cet algorithme crée une population initiale avec l'algorithme de placement BFDH* [Bortfeldt, 2006], une version améliorée de l'algorithme de Mumford-Valenzuela et al. [Mumford-Valenzuela *et al.*, 2003].

Voir e.g. [Gomez and de la Fuente, 2000] [Liu and Teng, 1999] [Yeung and Tang, 2004] pour d'autres AG, la méta-heuristique probablement la plus utilisée pour SPP.

2. Le plus en bas et le plus à gauche possible

Algorithme 1.6: La procédure Remove Rectangle de Neveu

RemoveRectangle (in R, in/out C, in/out S) Remove R from C ; Add R in S
Initialiser une liste `Holes` avec les espaces vides maximaux dans S qui sont contigus aux rectangles R
Initialiser une liste `HolePairs` avec paires (R, H) tel que H est un espace maximal dans Holes
tant que *`HolePairs` n'est pas vide* **faire**
    Sélectionner une paire d'espaces vides (`hole1`, `hole2`) dans Holepairs
    Retirer (`hole1`, `hole2`) de `HolePairs`
    `newHoles` ← Plus (`hole1`, `hole2`) /*newHoles contient au moins 2 espaces vides*/
    pour *chaque nouveau espace vide `newHole` dans `newHoles`* **faire**
        pour *chaque `hole` dans `Holes` /* garde que les espaces vides maximaux*/* **faire**
            si *`newHole` ⊂ `hole`* **alors**
                éliminer `newHole` de `newHoles`
                break
            sinon
                si *`hole` ⊂ `newHole`* **alors**
                    éliminer `hole` de `Holes`
                fin
            fin
        fin
        si *`newHole` ∈ `newHoles` /* met à jour Holes et HolePairs */* **alors**
            ajouter `newHole` à `Holes`
        sinon
            ajouter à `HolePairs` toutes les paires (`newHole`, `H`) tel que `H` est dans `Holes`
        fin
    fin
fin

1.4.5 Méthodes hybrides

Hopper et Turton [Hopper and Turton, 2001a] font une analyse des différentes méta-heuristiques appliquées au SPP. Les auteurs comparent les algorithmes de placement BL et BLF et proposent différents algorithmes hybrides (AG+BL, RS+BL, etc.).

Iori et al. [Iori *et al.*, 2003] ont utilisé les méta-heuristiques RT et AG et leur hybridation. Ils ont obtenu de bons résultats mais AG est le meilleur pour les petites instances.

Burke et al. [Burke *et al.*, 2004] utilisent les méta-heuristiques AG et RS avec un algorithme décodeur BF. Quelques années après, Burke et al. [Burke *et al.*, 2006] améliorent leurs résultats en combinant les algorithmes décodeurs BF et BLF.

D'autres auteurs ont travaillé avec les méta-heuristiques classiques (RT, RS, AG) et avec des algorithmes de placement (décodeurs). L'objectif est de trouver la meilleure séquence de pièces pour l'algorithme de placement, voir e.g. [Soke and Bingul, 2006].

1.4.6 Hyperheuristiques

Récemment, le concept d'hyperheuristiques a été testé avec succès sur différents problèmes [Burke *et al.*, 2003]. Une hyperheuristique combine, de manière contrôlée, plusieurs heuristiques pour renforcer les atouts de certaines et ainsi couvrir les faiblesses des autres.

Zhang et al. [Zhang *et al.*, 2006] ont développé une hyperheuristique *récursive* "HR" capable de diviser le SPP en sous-problèmes.

Une hyperheuristique nommée "HH" est également proposée dans [Neveu *et al.*, 2008; Araya *et al.*, 2008]. Basée sur les critères d'autres méthodes (comme BLF, HR, BFDH*, BF), HH sélectionne la méthode heuristique à employer à chaque étape à l'aide de paramètres adaptatifs.

1.5 Jeux de données

De nombreuses instances existent pour le SPP, ou ses variantes proches (avec rotation possible des pièces ou contrainte de découpe guillotine par exemple), cf. [Hopper and Turton, 2001a; Beasley, 1985a; Beasley, 1985b; Christofides and Whitlock, 1977; Christofides and Hadjiconstantinou, 1995; Fekete and Schepers, 1997b].

1.6 Conclusion

Les problèmes de placement et de découpe, dont SPP, sont très nombreux. La bibliographie concernant le SPP est donc riche et les techniques utilisées (exactes ou approchées) sont très variées.

Dans le SPP, les contraintes principalement considérées sont : l'orientation des objets (fixe ou libre) et la découpe guillotine (ou non).

Les techniques approchées sont les plus efficaces pour résoudre des instances de grandes tailles et la majorité est basée sur la combinaison de méta-heuristiques (AG, RS, RT, etc.) avec des heuristiques de placement (BL, BLF, BF, etc.). Les heuristiques de placement sont souvent utilisées comme algorithmes décodeurs pour construire et reconstruire les solutions du problème. L'heuristique la plus utilisée est BL.

Finalement, le développement des techniques récentes appelées “hyperheuristiques” est une proposition intéressante pour combiner intelligemment d’autres heuristiques déjà utilisées dans certains problèmes. La technique cherche la manière plus efficace et dynamique d’appliquer les heuristiques pour tenter de trouver une meilleure performance dans la résolution des problèmes.

Chapitre 2

Un nouvel algorithme génétique pour le SPP

Sommaire

Dans ce chapitre nous faisons un état de l'art des AG appliqués au SPP. Ensuite nous proposons un nouvel algorithme génétique pour le SPP non guillotine (DGA). DGA intègre deux caractéristiques importantes : une fonction d'évaluation **hiérarchique** et un opérateur de croisement **spécifique** (WAX pour "Wasted Area based crossover"). La fonction d'évaluation considère non seulement la hauteur finale du packing (pour la minimiser), mais aussi les espaces vides. L'objectif de WAX est de préserver les bonnes caractéristiques des parents. Pour évaluer DGA nous avons fait des expérimentations avec un ensemble d'instances bien connues et nous les avons comparées à d'autres approches.

2.1 Introduction

Dans les techniques liées au SPP, l'une des techniques la plus utilisée est l'algorithme génétique (AG). La théorie des AG est basée sur la théorie de l'évolution de Darwin [Aarts and Lenstra, 1997].

L'évolution est un processus lent de changement que présentent les espèces de génération en génération.

Charles Darwin est le père de la Théorie de l'évolution. Pendant les années 1831 à 1836 il a fait des observations sur la flore et la faune de différents endroits du monde dans un navire de l'armée anglaise. Après cette expérience, il a travaillé pendant 20 ans à la formulation d'une explication cohérente concernant la façon dont les individus varient dans le temps et conformément à l'environnement. En 1858, A. R. Wallace a travaillé sur une théorie similaire et, avec Darwin, publie un article sur la théorie de l'évolution [1]. Plus tard, en 1859, Darwin écrit le livre "The Origin of Species" [2]. Il y explique l'évolution organique et la manière dont les variations se transmettent à travers les générations. En général, Darwin a observé que les individus plus adaptés à l'environnement sont ceux qui transmettent ses variations. Ce phénomène est appelé "sélection naturelle". Dans la sélection naturelle les individus plus adaptés auraient de meilleures probabilités de subsister et de transmettre, à travers l'hérédité, les meilleures caractéristiques adaptées à l'environnement à leurs descendants.

Les principes de base de la théorie de Darwin et Wallace concernant l'évolution sont :

- Les organismes engendrent des organismes similaires comme conséquence d'une stabilité dans le processus de reproduction.
- La quantité d'individus capables de se reproduire est moindre que la quantité totale qui existe dans la population initiale.
- Il existe des changements aléatoires dans les individus transmis à travers l'hérédité mais qui ne sont pas produits par adaptation à l'environnement.
- Les variantes aléatoires héritées et les caractéristiques de l'environnement sont déterminantes pour que l'individu puisse se reproduire. Darwin a appelé ce phénomène "variations favorables". Darwin appelle la "sélection naturelle au processus" quand les variations favorables sont plus communes entre générations .
- Après un certain temps, la sélection naturelle provoque une accumulation des changements en remarquant la différentiation entre les organismes.

Dans les réflexions de Darwin, il parle de différentes espèces et de variations dans les espèces. Cependant l'union de deux espèces différentes provoque ce que Darwin a appelé un "hybride".

Hugo de Vries [1901] a créé une nouvelle théorie de l'évolution connue sous le nom de "mutationnisme". Le mutationnisme élimine la sélection naturelle comme processus principal de l'évolution.

Les Algorithmes Évolutionnistes (AE) sont connus en intelligence artificielle. Ils sont utilisés comme méthode de résolution pour des problèmes complexes là où d'autres méthodes ne sont pas capables de trouver des solutions en un temps raisonnable. Ce type d'algorithme appartient à une classe générale de méthodes pour résoudre des problèmes combinatoires (méta-heuristiques).

1. "Journal of the Proceedings of the Linnean Society", 1858, vol. 3, pp. 45-62.
2. Édition originale numérisée : `http://darwin-online.org.uk/EditorialIntroductions/Freeman_OntheOriginofSpecies.html`

La terminologie de la théorie de l'évolution est utilisée dans les AE. Les entités qui représentent les solutions candidates du problème sont appelées des individus et l'ensemble de ceux-ci est la population. Les individus sont modifiés par des opérateurs génétiques, principalement : le croisement, qui mélange l'information génétique entre deux ou plusieurs individus ; la mutation, qui fait un changement (plus ou moins aléatoire) dans l'information génétique ; la sélection, processus qui choisit les individus survivants pour la prochaine génération. L'objectif est de trouver l'individu le plus adapté à l'environnement. Les individus sont évalués à l'aide d'une fonction d'évaluation par rapport à leur capacité d'adaptation à l'environnement.

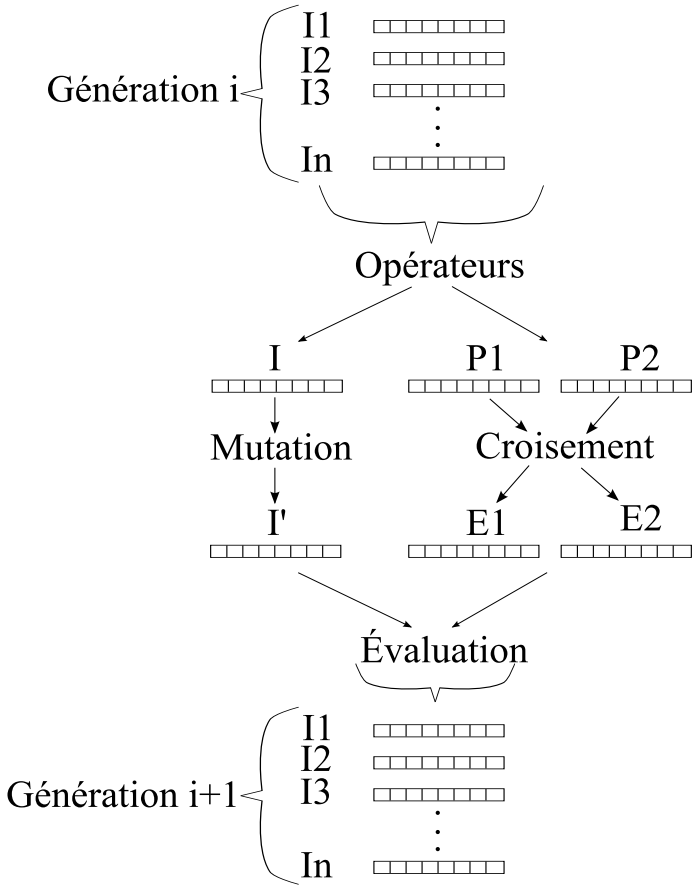


FIGURE 2.1 – Principe général d'un algorithme génétique

Un des paradigmes des AE sont les Algorithmes Génétiques (AG), proposés par John Holland [Holland, 1975]. Les AG sont des algorithmes d'optimisation qui utilisent des composants dérivés du concept de la théorie de l'évolution.

Dans la figure 2.1 nous pouvons observer le principe général d'un algorithme génétique.

Cinq éléments composent habituellement un AG :

1. Un codage de représentation des données pour les solutions candidates du problème. Holland a utilisé le codage binaire pour représenter les solutions. Avec ce codage, il a simulé

l'information génétique d'un être vivant mais de manière artificielle. Un chromosome est représenté par une chaîne de bits. Chaque gène du chromosome est représenté par chaque bit de la chaîne. La représentation binaire permet de manipuler facilement l'information.

2. Un mécanisme de génération de la population initiale. La population initiale est importante car elle peut parfois rendre plus ou moins rapide la convergence vers un optimum global. Dans le cas où nous ne connaissons rien du problème à résoudre, il peut-être intéressant que la population soit distribuée sur tout (ou une part suffisante de) l'espace de recherche.
3. Une fonction d'adaptation ou d'évaluation. La fonction permet d'évaluer la qualité d'un individu par rapport à un autre. Celle-ci retourne une valeur appelée "fitness" ou évaluation.
4. Des opérateurs capables de diversifier la population au cours des générations et d'explorer l'espace de recherche. Pour Holland il existe trois opérateurs : le croisement, la mutation et la sélection. La mutation de Holland consiste à changer la valeur d'un gène (de 0 à 1 ou vice-versa) comme le montre la figure 2.2. Le croisement de Holland combine l'information génétique de chaque individu en prenant une partie des valeurs de chaque parent pour construire un ou plusieurs fils, cf. figure 2.3 pour un exemple. Finalement, la sélection est un mécanisme qui choisit les individus les plus adaptés (à l'aide de la fonction d'évaluation).
5. Des paramètres de dimensionnement : taille de la population, nombre total de générations ou critères d'arrêt, probabilités d'application des opérateurs de croisement et de mutation.

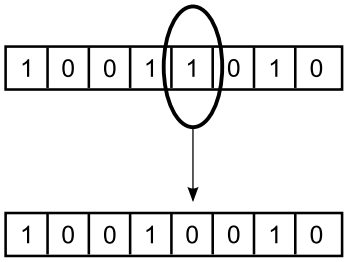


FIGURE 2.2 – Exemple de l'opérateur de mutation créé par Holland

2.2 Les algorithmes génétiques (AG) et le SPP

Les premiers AG avec une représentation différente de celle de Holland apparaissent avec Jakobs [1993]. Il utilise un AG avec une représentation sous forme de permutations. Pour construire une solution, l'auteur utilise la méthode BL, expliquée dans le chapitre 1, en prenant un vecteur unidimensionnel pour enregistrer une séquence de rectangles placés.

Plus tard, Hopper et Turton [Hopper and Turton, 1999] utilisent un AG et deux algorithmes de construction, BL et BLF, pour construire un placement.

Voir également [Iori *et al.*, 2003; Gomez and de la Fuente, 2000; Liu and Teng, 1999; Yeung and Tang, 2004] pour des autres AG et variations de l'algorithme BF [Soke and Bingul, 2006].

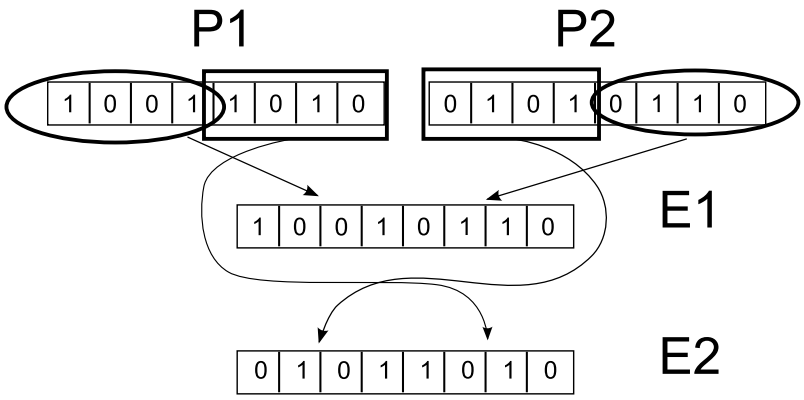


FIGURE 2.3 – Exemple de l'opérateur de croisement créé par Holland

Tous les travaux mentionnés précédemment utilisent la représentation par permutations dont la principale caractéristique est de reposer sur un algorithme de construction pour élaborer un placement.

2.2.1 Composants usuels des AG appliqués au SPP

Il existe des composants communs entre les AG appliqués au SPP. Dans cette section, nous faisons un résumé des composants les plus fréquemment rencontrés dans le cadre des AG appliqués au SPP.

Individu

Les individus sont représentés par des permutations des objets. La permutation est une représentation naturelle du problème puisqu'elle permet facilement de reconstruire un placement (à l'aide d'un décodeur). Une permutation représente une séquence de rectangles à placer où j est l'index du rectangle r_j et $\pi = (r_1, ..., r_n)$. La représentation d'un individu s'appelle "patron de placement". Pour faire un patron de placement il faut utiliser un algorithme de placement, comme par exemple BL. Dans une permutation, l'ordre des rectangles est très important car ils seront pris par l'algorithme de placement, en revanche le patron résultant d'un placement ne sera pas toujours unique. La figure 2.4 montre un exemple de deux individus (deux permutations) et le patron de placement commun.

Initialisation de la population

Dans le SPP il existe différentes façons d'initialiser la population. Nous pouvons citer des initialisations de la population au hasard [Gomez and de la Fuente, 2000; Yeung and Tang, 2004; Gómez-Villouta *et al.*, 2007] en respectant des critères spécifiques selon un ordre croissant ou décroissant de différentes dimensions (largeur, hauteur, surface, etc.) [Jakobs, 1993].

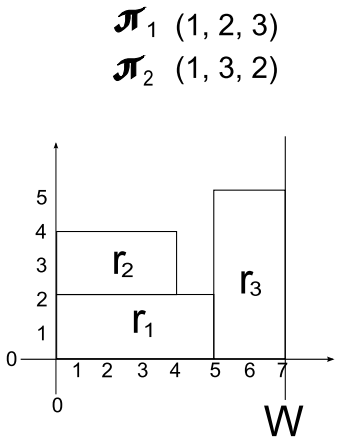


FIGURE 2.4 – Deux permutations avec le même patron de placement selon BL

Fonction d'évaluation

La fonction d'évaluation est utilisée pour calculer la qualité de l'individu par rapport aux autres. Dans le SPP, il existe différents types de fonctions d'évaluation, reposant sur différents critères plus ou moins fins :

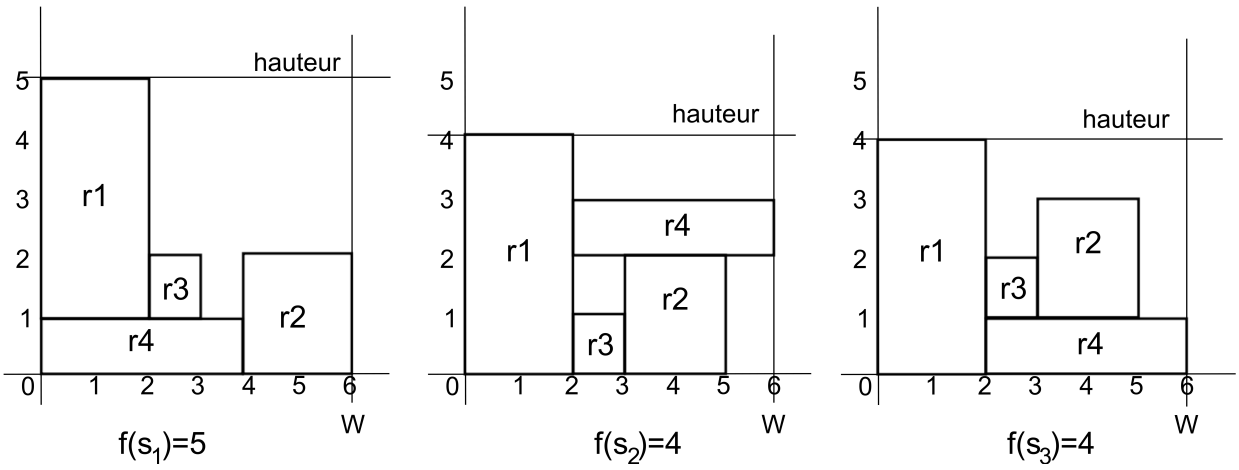


FIGURE 2.5 – Fonctions d'évaluation reposant sur le critère de hauteur

Hauteur : les premiers travaux concernant le SPP utilisent la hauteur atteinte par un rectangle placé le plus haut. Cette fonction n'est pas très efficace parce qu'elle peut donner la même évaluation à des individus différents cf. Fig. 2.5 [Yeung and Tang, 2004].

Aire gaspillée : cf. Fig. 2.6 [Soke and Bingul, 2006].

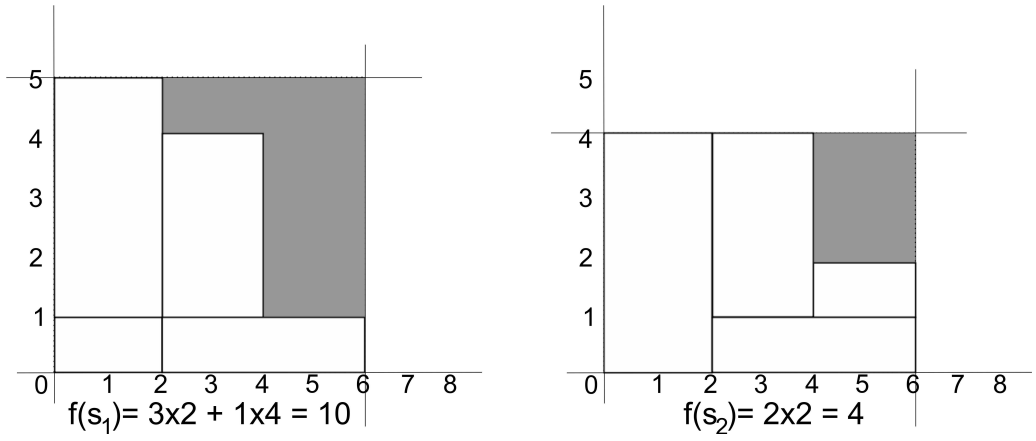


FIGURE 2.6 – Fonctions d'évaluation reposant sur le critère d'aire gaspillée

Distribution des rectangles : c'est une variation de l'évaluation reposant sur la hauteur pour différencier la qualité par rapport à la distribution des rectangles. On ajoute à la hauteur atteinte (à minimiser) la somme (à maximiser) des produits ($y_i^r + h_i^r$) x poids, où "poids" est un paramètre. Dans l'exemple de la figure 2.7 le poids est 0.01 [Gomez and de la Fuente, 2000].

Fonction hiérarchique : une fonction d'évaluation hiérarchique se compose de plusieurs sous-fonctions qui respectent un ordre de priorité décroissant [Rodríguez-Tello, 2007; Vasquez and Hao, 2001]. Chaque sous-fonction f_i correspond typiquement à une contrainte du problème ou à un objectif initial. Étant donnés deux individus x et y, ils sont comparés avec la relation suivante, où ">" signifie "meilleur que" :

$$f(x) > f(y) \equiv \begin{cases} f_1(x) > f_1(y) \\ \text{ou} \quad f_1(x) = f_1(y) \quad \text{et} \quad f_2(x) > f_2(y) \\ \text{ou} \quad f_1(x) = f_1(y), ..., f_{k-1}(x) = f_{k-1}(y) \quad \text{et} \quad f_k(x) > f_k(y) \end{cases} \tag{2.1}$$

Il semblerait qu'une seule approche pour le SPP utilise cette possibilité [Alvarez-Valdes *et al.*, 2007]. Nous présenterons une telle fonction (hiérarchique) pour le SPP dans le chapitre 3.

Sélection

La sélection des individus est un processus qui permet de choisir les individus les mieux qualifiés pour accéder à la reproduction (croisement) et créer une prochaine génération avec ses descendants. Il existe différentes formes de sélection :

Roulette : les individus sont choisis avec une probabilité proportionnelle à leur évaluation [Holland, 1975].

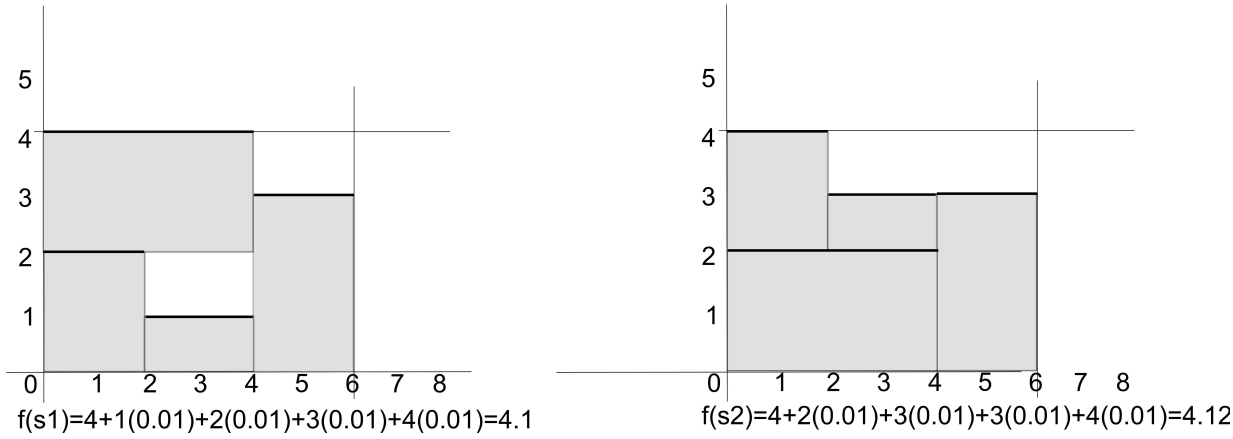


FIGURE 2.7 – Fonctions d'évaluation reposant sur la distribution des rectangles

Tournoi : un sous-ensemble d'individus est choisi au hasard, et le meilleur d'entre eux est sélectionné [Miller and Goldberg, 1995]. On a donc ici, la possibilité de choisir le meilleur d'un ensemble d'individus mais qui ne sont pas nécessairement les meilleurs de la population.

Croisement

L'opérateur de croisement génère de nouveaux individus en mélangeant l'information des parents. Habituellement, comme dans la nature, les opérateurs de croisement sont définis pour être appliqués avec deux individus (nommés "parents"). Mais il existe aussi des variantes s'appliquant à plus de deux individus. Les croisements classiques sont :

Croisement un point : un point de croisement est choisi (souvent aléatoirement) pour diviser en deux parties chaque parent et ensuite intervertir les moitiés. Par exemple, si nous avons π_1=($r_{1_1}, ..., r_{1_k}, ..., r_{1_n}$) et π_2=($r_{2_1}, ..., r_{2_k}, ..., r_{2_n}$) avec k, le point de croisement, les deux descendants sont : π_3=($r_{1_1}, ..., r_{1_k}, r_{2_{(k+1)}}, ..., r_{2_n}$) et π_4=($r_{2_1}, ..., r_{2_k}, r_{1_{(k+1)}}, ..., r_{1_n}$).

Croisement partiellement correspondant (PMX pour "partially matched crossover") : le croisement partiellement correspondant proposé par Goldberg et al. [Goldberg and Lingle, 1985], prend un sous-ensemble du premier parent et le transmet directement au descendant à la même position. Ensuite, les gènes dupliqués sont remplacés par les gènes de l'autre parent en respectant le même ordre. Par exemple, si nous avons π_1=(1,8,4,6,2,5,3,7) et π_2=(5,6,3,8,7,1,2,4), et le sous-ensemble de croisement est (4,6,2) en π_1, le premier descendant π_3 est créé avec les gènes de π_2 mais en remplaçant le sous-ensemble (3,8,7), donc, π_3 serait (5,6,**4,6,2**,1,2,4). Après, les gènes dupliqués (5,**6**,4,6,2,1,**2,4**) sont remplacés par le sous-ensemble (3,8,7) dans la même séquence de (4,6,2) : π_3 = (5,*,4,6,2,1,*,*) $\rightarrow$ (5,8,4,6,2,1,7,3).

Croisement ordonné (OX pour "ordered crossover") : le croisement ordonné proposé par Davis [Davis, 1985], commence de la même façon que PMX en choisissant deux points de

croisement. La différence par rapport à PMX, est que les sous-ensembles ne sont pas remplacés directement, mais sont temporairement éliminés dans l'autre parent. Finalement, on fait un déplacement à gauche des gènes à partir du premier point au deuxième point de croisement jusqu'à laisser les gènes à remplacer entre les deux points. La figure 2.8 montre un exemple du croisement OX.

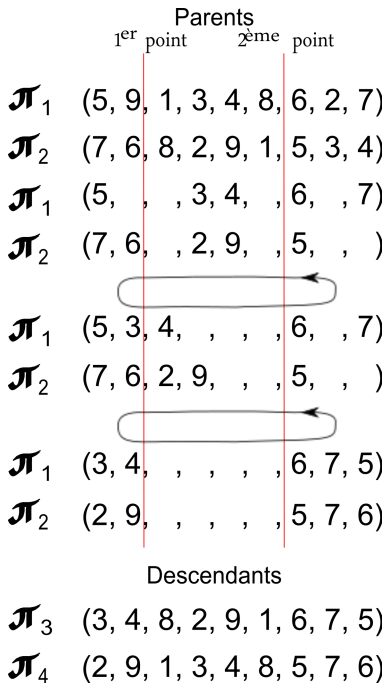


FIGURE 2.8 – Une illustration du croisement OX

Croisement basé sur l'ordre (OBX) : OBX est un opérateur qui s'intéresse à l'ordre relatif des gènes dans les parents. Deux points de croisement sont choisis pour sélectionner un sous-ensemble de gènes dans les deux parents. Le sous-ensemble du premier parent est directement transmis au premier descendant en suivant la même séquence, mais dans différentes places. Les places utilisées correspondent à celles du deuxième parent. Ensuite les gènes manquants sont sortis du deuxième parent, mais cette fois, en respectant la même place. La figure 2.9 montre un exemple du croisement OBX.

Croisement cyclique (CX) : l'opérateur CX, proposé par [Oliver *et al.*, 1987], est un opérateur cyclique qui choisit le premier gène de chaque parent pour le transmettre aux descendants. Ensuite, à partir du premier gène nous cherchons le gène dans la même position dans le deuxième parent. Le cycle est fini quand on arrive au premier gène. Les places restantes des deux descendants sont complétées par les gènes des parents opposés. Par exemple, si nous avons π_1= (9,8,2,1,7,4,5,10,6,3) et π_2= (1,2,3,4,5,6,7,8,9,10) les premiers gènes choisis pour chaque descendant sont : π_3 =(9,*,*,*,*,*,*,*,*,*) et π_4 =(1,*,*,*,*,*,*,*,*,*). Le processus suivant est montré en figure 2.10.

Croisement uniforme (UX) : UX est un opérateur qui prend deux parents p_1 et p_2 et, pour chaque

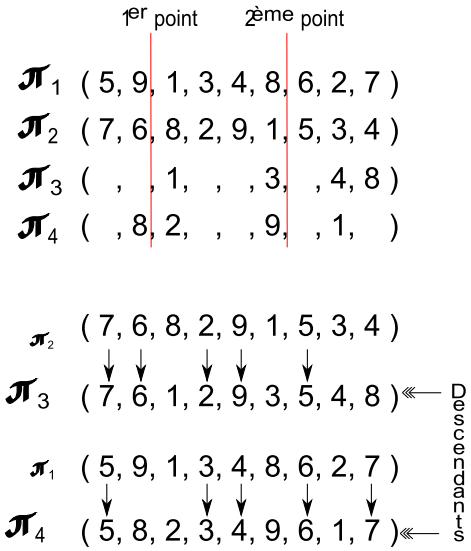


FIGURE 2.9 – Une illustration du croisement OBX

gène, décide s'il transmet celui de p_1 ou celui de p_2 selon une certaine probabilité, généralement 0.5. Donc chaque parent a théoriquement la probabilité de transmettre la moitié de ses gènes aux descendants.

Croisement de Stefan Jakobs (SJX) : l'opérateur SJX, proposé par [Jakobs, 1993], prend deux parents et crée un unique descendant. D'abord il faut choisir deux paramètres p et q avec $1 \leq p \leq n$ et $1 \leq q \leq n$. q gènes sont sélectionnés du premier parent, à partir de la position p de la permutation pour être transmis au descendant. Ensuite les gènes manquants vont être retrouvés à partir du deuxième parent, en respectant la même ordre. La figure 2.11 montre un exemple du croisement SJX.

Mutation

L'opérateur de mutation, comme dans la nature, change une partie de l'information génétique (éventuellement en garantissant la validité de l'individu résultant). Il existe plusieurs variétés de mutations pour la représentation sous forme de permutations [Hopper and Turton, 1997; Hopper and Turton, 2001b; Liang *et al.*, 2001; Soke and Bingul, 2006] mais les plus communément appliquées au SPP sont :

- Swap : échange de deux rectangles [Yeung and Tang, 2004; Gómez-Villouta *et al.*, 2007].
- Orientation : change l'orientation d'un rectangle (non applicable au SPP) [Jakobs, 1993].
- Hasard : nouvelle initialisation (généralement au hasard) de la permutation [Gomez and de la Fuente, 2000].
- Inversion : inverse complètement ou partiellement des gènes de la permutation [Yeung and Tang, 2004].

π_1 (9, 8, 2, 1, 7, 4, 5, 10, 6, 3)
π_2 (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
π_3 (9, , , 1, , 4, , , 6,)
π_3 (9, 2, 3, 1, 5, 4, 7, 8, 6, 10)

π_1 (9, 8, 2, 1, 7, 4, 5, 10, 6, 3)
π_2 (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
π_4 (1, , , 4, , 6, , , 9,)
π_4 (1, 8, 2, 4, 7, 6, 5, 10, 9, 3)

FIGURE 2.10 – Une illustration du croisement CX

π_1 (9, 8, 2, 1, 7, 4, 5, 10, 6, 3)
π_2 (10, 5, 3, 4, 6, 2, 7, 8, 1, 9)
p=2 q=4
π_3 (, 8, 2, 1, 7, , , , ,)
π_3 (10, 8, 2, 1, 7, 5, 3, 4, 6, 9)

FIGURE 2.11 – Une illustration du croisement SJX

Mise à jour de la population

La population change de génération en génération à l'aide des opérateurs génétiques. Tous les individus de meilleure qualité sont conservés à la génération suivante.

Dans les AG classiques il existe généralement deux stratégies d'évolution :

– Élitiste : cette stratégie sélectionne le meilleur individu de la population pour le passer directement à la génération suivante.
– Élimination des configurations les moins bien évaluées de la population : à chaque fois que les opérateurs génétiques s'appliquent aux individus, l'individu finalement sélectionné est celui qui présente la meilleure évaluation.

2.3 DGA : un nouvel AG pour le SPP

L'algorithme DGA (pour "Dedicated Genetic Algorithm") [Gómez-Villouta *et al.*, 2007] est un algorithme génétique pour SPP qui propose un nouvel opérateur de croisement et une nouvelle fonction d'évaluation hiérarchique.

2.3.1 Composants classiques utilisés

Nous rappelons, dans cette section, les composants classiques utilisés dans l'algorithme DGA.

Individu

Les individus sont représentés par des permutations décodées avec une heuristique de placement ϕ.

Décodage

L'algorithme décodeur utilisé correspond à BLF [Baker *et al.*, 1980].

Initialisation de la population

La population est initialisée aléatoirement.

Fonction d'évaluation

La fonction d'évaluation f mesure la *qualité* d'un individu π selon différents critères. La fonction d'évaluation peut avoir une grande influence dans le processus de recherche d'une solution.

Pour le SPP, la fonction objectif initiale f_1 (hauteur d'un patron) définie par l'Eq. 1.1 dans le chapitre 1 (section 1.3.1), peut être utilisée comme fonction d'évaluation.

f_1 ne peut pas différencier deux individus π_1 et π_2 de même hauteur. Nous proposons d'appliquer une deuxième fonction f_2 pour permettre de faire la distinction entre deux solutions de même hauteur. Pour pouvoir différencier deux individus avec la même valeur de f_1, nous considérons l'ensemble Λ des espaces vides du patron de placement. La présence de grandes aires libres implique que le nombre de zones libres est moindre. En général il est plus facile de remplir de grandes zones libres que beaucoup de petites zones libres. Formellement, f_2 est définie par :

$$f_2 = \frac{1}{\sigma_*} \tag{2.2}$$

où σ_* est la surface de l'espace vide $\lambda_* \in \Lambda$ le plus grand, cf. Fig. 2.12 pour un exemple.

DGA compare deux individus π_1 et π_2 à l'aide de la fonction suivante :

$$f = \begin{cases} f_1 & \text{si } f_1(\pi_1) \neq f_1(\pi_2) \\ f_2 & \text{autrement} \end{cases} \tag{2.3}$$

Sélection

La sélection utilisée par DGA est de type "roulette" : les individus sont choisis selon une probabilité proportionnelle à leur qualité.

$\pi_1 = (12, 17, 16, 13, 15, 14, 9, 5, 3, 2, 7, 10, 6, 4, 8, 1, 11)$
$f_1(\pi_1) = 22$
$\lambda_* = \text{C}\ (\sigma_* = 30) \Rightarrow f_2(\pi_1) \approx 0.03$

$\pi_2 = (11, 9, 5, 12, 17, 7, 10, 13, 8, 16, 4, 2, 3, 6, 1, 14, 15)$
$f_1(\pi_2) = 22$
$\lambda_* = \text{G}\ (\sigma_* = 15) \Rightarrow f_2(\pi_2) \approx 0.06$

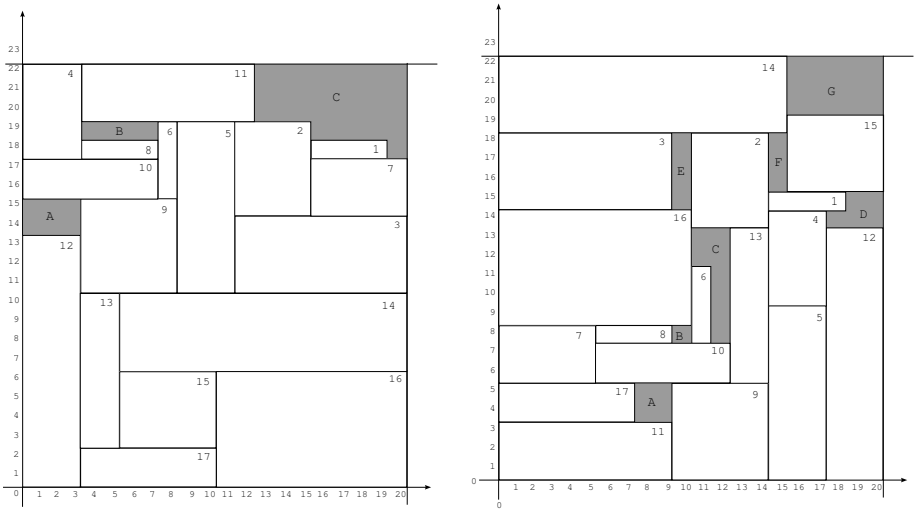


FIGURE 2.12 – Pour deux individus π_1 (à gauche) et π_2 (à droite) tels que $f_1(\pi_1) = f_1(\pi_2)$, f_1 n'est pas capable de faire la différence, donc si nous appliquons f_2, π_1 est meilleur que π_2 ($f_2(\pi_1) < f_2(\pi_2)$).

Croisement

DGA utilise un opérateur de croisement à deux parents spécifique au problème : WAX pour "Wasted Area based crossover". Cet opérateur, qui considère les espaces vides du patron de placement, a pour objectif de tenter de reconnaître les "bonnes" zones de placement.

WAX : une description intuitive

Si nous avons deux parents π_1 et π_2 de bonne qualité, il semble pertinent de préserver les zones les plus compactes, à transmettre au descendant $\pi_\times$. WAX travaille en trois étapes :

1. Identifier dans π_1 la zone libre la plus basse.
2. Copier directement (depuis π_1) le patron de placement trouvé sous la zone libre dans $\pi_\times$
3. Replacer les rectangles restants en respectant l'ordre séquentiel de π_2.

La différence de WAX, par rapport aux autres opérateurs de croisement, est qu'il travaille avec une information additionnelle liée aux espaces vides entre les rectangles. WAX sélectionne la zone $\lambda_l \in \Lambda$ (ensemble des espaces vides) la plus basse dans π_1. La partie gauche de $\pi_\times$ vient de π_1 : c'est l'ensemble des rectangles placés sous λ_l (dans le même ordre que dans π_1). La partie droite de $\pi_\times$ est complétée avec les rectangles restants dans le même ordre que dans π_2. WAX est détaillé dans l'algorithme 2.1 (complexité en $\text{O}(n)$).

La caractéristique "visuelle" de WAX peut être observée dans la Figure 2.13 ($n = 17$). La zone libre la plus basse dans π_1 est $\lambda_l = \text{A}$ avec $y_l = 13$. Les 9 objets sous A, triés par π_1

Algorithme 2.1: Schéma général de WAX.

1. **Initialisation.** $c \leftarrow 0$ (pour construire $\pi_\times$). $j \leftarrow 0$ (pour explorer π_1). Soit d un vecteur de n booléens initialisé à false.
   Soit y_l l'ordonnée du coin bas-gauche de λ_l l'aire vide la plus basse dans π_1.
2. **Héritage de** π_1 $j \leftarrow j+1$. $i \leftarrow \pi_1[j]$.
   si $y_i^r < y_l$ (r_i est sous λ_l dans π_1) alors : $c \leftarrow c+1$. $\pi_\times[c] \leftarrow i$. $d[i] \leftarrow$ true.
   Else (r_i au dessous de λ_l) : $d[i] \leftarrow$ false.
   si $j < n$ alors : allez à 2.
   $j \leftarrow 0$ (pour explorer π_2).
3. **Héritage de** π_2 $j \leftarrow j+1$. $i \leftarrow \pi_2[j]$.
   si $\overline{d[i]}$ (r_i n'est pas dans la partie gauche de $\pi_\times$, l'insérer dans la partie droite) alors :
   $c \leftarrow c+1$. $\pi_\times[c] \leftarrow i$. si $c < n$ alors : allez à 3.
4. Return $\pi_\times$.

$(12, 17, 16, 13, 15, 14, 9, 5, 3)$, sont transmis à $\pi_\times$ directement. $\pi_\times$ est finalement complété avec les 8 rectangles manquants en respectant l'ordre de π_2 : $(11, 7, 10, 8, 4, 2, 6, 1)$.

Mutation

DGA utilise la mutation *swap*. L'algorithme applique la mutation juste après avoir appliqué l'opérateur de croisement WAX. L'opérateur de mutation s'applique à l'individu $\pi_\times$ avec une probabilité p . L'opérateur choisit au hasard deux rectangles de **différentes dimensions** et les échange. Une fois l'échange effectué, la nouvelle configuration est reconstruite avec l'algorithme de placement BLF.

Mise à jour de la population

La mise à jour de la population se fait par remplacement de l'individu le moins bien évalué.

2.3.2 Schéma général de DGA

Le schéma général de DGA est donné en figure 2.2. Naturellement, DGA peut aussi s'arrêter avant d'arriver aux p_3 générations (*i.e.* $g < p_3$) à chaque fois que π_* est mis à jour (étapes 1, 3 et 4), si $f_1(\pi_*) \leq H_{OPT}$ (H_{OPT} est la hauteur optimale ou une borne supérieure).

Algorithme 2.2: Schéma général de DGA

1. **Initialisation.** $g \leftarrow 0$. Nombre de générations.
   Soit Π la population composée de p_1 permutations aléatoires des n objets.
   Exécuter BLF sur chaque $\pi \in \Pi$.
   Soit π_* un meilleur individu selon f (voir Eq. 2.3 dans la section 2.3.1).
2. **Sélection.** $g \leftarrow g + 1$.
   Soit $\pi_1 \in \Pi$ un des individus les mieux évalués
   Soit $\pi_2 \in \Pi \setminus \{\pi_1\}$ un des individus les mieux évalués.
3. **Croisement.** Appliquer à π_1 et π_2 l'opérateur de croisement WAX (voir section 2.3.1), pour créer $\pi_\times$. Appliquer BLF à $\pi_\times$.
   si $f(\pi_\times) < f(\pi_*)$ alors : $\pi_* \leftarrow \pi_\times$.
4. **Mutation.** Selon une probabilité p_2 :
   (a) Échanger deux objets choisis au hasard dans $\pi_\times$.
   Appliquer BLF à $\pi_\times$.
   si $f(\pi_\times) < f(\pi_*)$ alors : $\pi_* \leftarrow \pi_\times$.
5. **Mise à jour de la population** Soit π_1 l'un des individus les moins bien évalués
   $\Pi \leftarrow \Pi \setminus \{\pi_1\}$.
   $\Pi \leftarrow \Pi \cup \{\pi_\times\}$.
   si $g < p_3$ (nombre maximal de générations) alors : allez à 2.
6. Return $f_1(\pi_*)$ (and π_*).

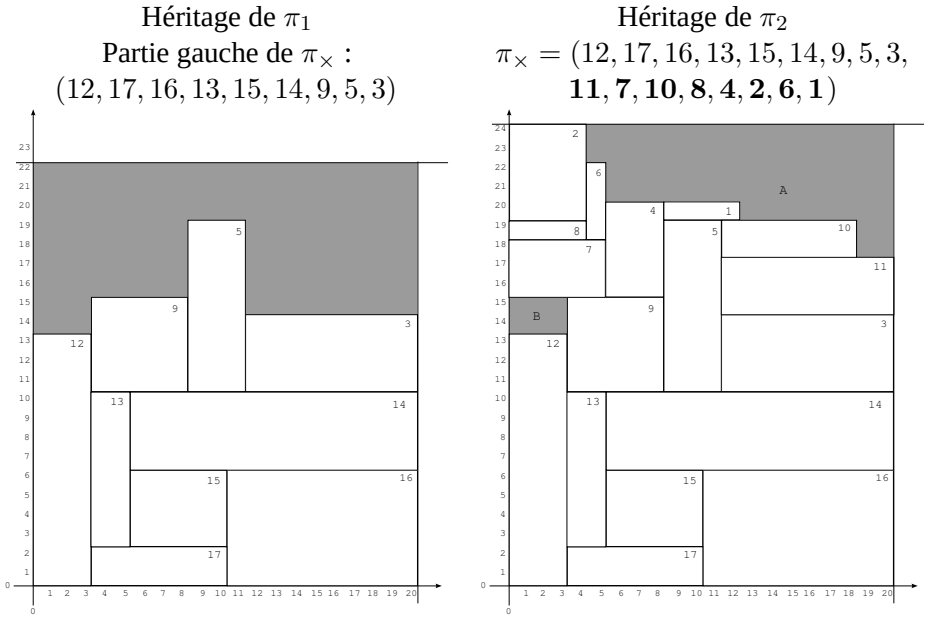


FIGURE 2.13 – WAX : un exemple (avec les parents de la figure 2.12).

2.4 Expérimentations

2.4.1 Protocole de tests

Nous avons utilisé un ensemble de 21 instances bien connues [Hopper and Turton, 2001a] dont les principales caractéristiques sont rappelées dans le tableau 2.1 [3].

Categorie	Instances	W	n	H_{OPT}
C1	C1P1, C1P2, C1P3	20	16, 17, 16	20
C2	C2P1, C2P2, C2P3	40	25	15
C3	C3P1, C3P2, C3P3	60	28, 29, 28	30
C4	C4P1, C4P2, C4P3	60	49	60
C5	C5P1, C5P2, C5P3	60	73	90
C6	C6P1, C6P2, C6P3	80	97	120
C7	C7P1, C7P2, C7P3	160	196, 197, 196	240

TABLE 2.1 – Principales caractéristiques du jeux de test de *Hopper et Turton [2001]*

Les comparaisons sont basées sur l'écart (ou "gap") d'une solution à l'optimum ou à la meilleure hauteur connue (H_{OPT}). Pour une solution π, l'écart γ est défini par γ = 1 - $H_{OPT}/f_1(\pi)$.

3. Instances disponibles depuis par exemple `http://mo.math.nat.tu-bs.de/packlib/xml/ht-eimhh-01-xml.shtml`

	GRASP		RT	AG1		DGA	
Instances	$\overline{\gamma}$	γ_*	γ_*	$\overline{\gamma}$	γ_*	$\overline{\gamma}$	γ_*
C1P1	0	0	0	-	-	**0**	**0**
C1P2	0	0	0	-	-	4.76	**0**
C1P3	0	0	0	-	-	**0**	**0**
Mean C1	0	0	0	1.59	1.59	**1.59**	**0**
C2P1	0	0	0	-	-	6.25	**0**
C2P2	0	0	0	-	-	6.25	6.25
C2P3	0	0	0	-	-	**0**	**0**
Mean C2	0	0	0	3.33	2.08	4.17	**2.08**
C3P1	0	0	0	-	-	3.23	**0**
C3P2	3.23	3.23	0	-	-	**3.23**	3.23
C3P3	0	0	0	-	-	3.23	**0**
Mean C3	1.08	1.08	0	3.16	3.13	**3.23**	**1.08**
C4P1	1.64	1.64	-	-	-	6.25	4.76
C4P2	1.64	1.64	-	-	-	6.25	4.76
C4P3	1.64	1.64	-	-	-	4.76	3.33
Mean C4	1.64	1.64	-	3.52	2.7	5.75	4.25
C5P1	1.1	1.1	-	-	-	5.26	3.23
C5P2	1.1	1.1	-	-	-	6.25	5.26
C5P3	1.1	1.1	-	-	-	5.26	4.26
Mean C5	1.1	1.1	-	2.03	1.46	5.59	4.25
C6P1	1.56	0.83	-	-	-	5.51	4.76
C6P2	1.56	0.83	-	-	-	5.51	4.76
C6P3	1.56	0.83	-	-	-	6.25	5.51
Mean C6	1.56	0.83	-	1.72	1.64	5.76	5.01
C7P1	1.64	1.64	-	-	-	6.61	5.88
C7P2	1.19	0.83	-	-	-	7.33	6.61
C7P3	1.23	1.23	-	-	-	6.25	5.88
Mean C7	1.36	1.23	-	1.52	1.23	6.73	6.12

TABLE 2.2 – Moyenne et meilleur écart ($\overline{\gamma}$ and γ_* resp.) sur les instances de *Hopper et Turton [2001]*

Les paramètres de DGA ont été déterminés de manière empirique : $p_1 = 80$ (taille de la population), $p_2 = 0.7$ (probabilité de mutation) et $p_3 = 1000$ (nombre maximum de générations). BLF a été codé selon [Hopper and Turton, 2001a]. Le gap moyen $\overline{\gamma}$ (respectivement, le meilleur écart γ_*) est la moyenne sur 25 exécutions [4] (respectivement sur les meilleurs résultats).

DGA est implémenté en C (compilateur gcc). Tous les résultats ont été obtenus en exécutant DGA sur une station Linux (2 Go RAM, Intel PIV à 2 Ghz). Pour information, le temps moyens d'éxecution de DGA, avec utilisation de la fonction d'évaluation f_2, varient de quelques secondes (pour les plus petites instances) à environ de 24 heures (pour les plus gros jeux).

2.4.2 Résultats comparatifs

Dans le tableau 2.2 nous rappelons les résultats des algorithmes les plus performants (GRASP [Alvarez-Valdes *et al.*, 2008b] et la recherche tabou RT de [Alvarez-Valdes *et al.*, 2007] et ceux d'un AG récent appelé "AG1" [Bortfeldt, 2006] [5] souvent considéré comme l'AG le plus efficace pour le problème. Dans le tableau 2.2, "-" représente une information inconnue. De plus, aucune valeur moyenne ($\overline{\gamma}$) n'est mentionnée pour RT car l'information n'est pas précisée dans [Alvarez-Valdes *et al.*, 2007].

Dans le tableau 2.2, nous pouvons observer que DGA présente une bonne performance sur les 9 premières instances (catégories C1–C3) : l'optimum est atteint pour 7 d'entre elles. Cependant, pour les 9 autres instances, DGA est moins compétitif.

Des résultats similaires ont été observés sur un autre ensemble de 35 instances proposées dans [Hopper, 2000] [6]. À notre connaissance, il semblerait que DGA soit l'unique méthode capable d'atteindre l'optimum pour les 5 premières instances (cf. tableau 2.3).

Une comparaison des opérateurs de croisement

Problème	n	H_{OPT}	WAX	OBX
Na1a	17	200	0	0
Na1b	17	200	0	0
Na1c	17	200	0	5
Na1d	17	200	0	0
Na1e	17	200	0	4
Na3	29	200	5	6

TABLE 2.3 – Comparaison meilleur écart γ_* entre WAX et OBX

Le croisement est l'un des principaux composants d'un AG. Dans cette section nous rapportons une analyse expérimentale pour comparer notre opérateur WAX avec le croisement OBX proposé

4. 10 exécutions pour les autres approches considérées dans le tableau 2.2.

5. $p_1 = 50$ et $p_2 = 0.33$. p_3 est variable : $n \leq 60 \Rightarrow p_3 = 1000$, $61 \leq n \leq 100 \Rightarrow p_3 = 500$, $n > 100 \Rightarrow p_3 = 100$

6. Instances disponibles depuis par exemple `http://www.math.tu-dresden.de/~belov/publ/downl/2D/Hopper_PhD_Thesis.pdf`

par [Leung *et al.*, 2001] (cf. la section 2.2.1 pour une description d'OBX). OBX semble être un opérateur de croisement efficace dans ce type de représentation pour le SPP, selon [Soke and Bingul, 2006]. Les jeux de test utilisés pour l'analyse correspondent à un sous-ensemble de ceux décrits dans [Hopper, 2000]. WAX a trouvé l'optimum pour les 5 premières instances et, pour Na3 WAX a présenté une meilleure performance par rapport a OBX (cf. le tableau 2.3).

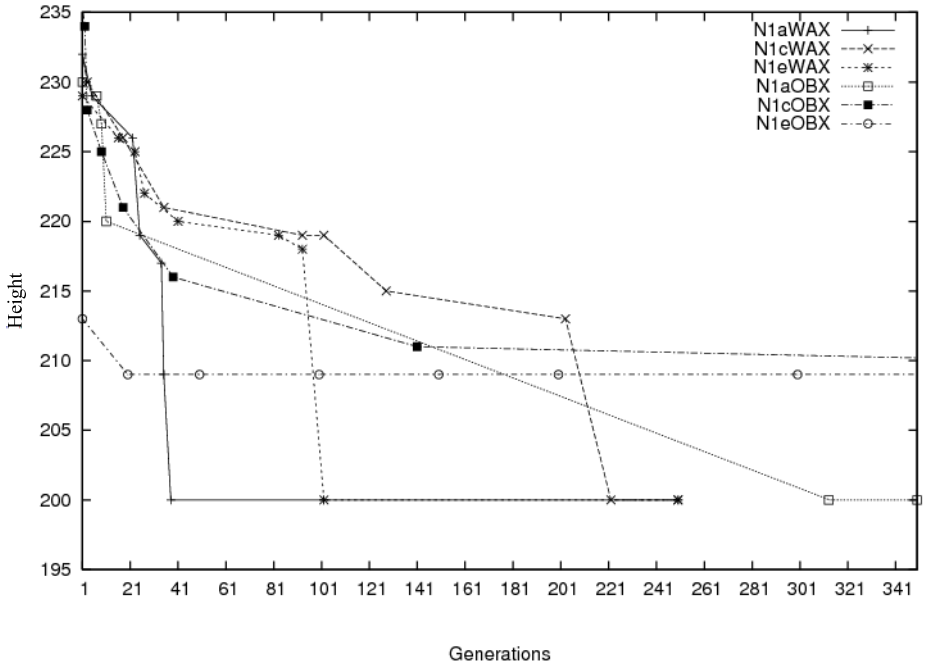


FIGURE 2.14 – Comparaison entre les croisements WAX et OBX

La figure 2.14 montre un exemple des meilleurs profils de convergence sur trois instances du même DGA en utilisant WAX et OBX (25 exécutions). Nous pouvons observer une convergence plus rapide avec WAX qu'avec OBX.

La supériorité de WAX sur OBX pourrait s'expliquer par la *spécificité* de notre croisement (dédié au SPP), alors qu'OBX est un opérateur *générique* (i.e. pouvant s'appliquer à d'autres problèmes de permutations).

2.5 Conclusion

Dans ce chapitre nous avons discuté de l'application des AG comme méthodes de résolution pour le SPP.

En général, les algorithmes génétiques existants pour le SPP sont basés sur la représentation par permutations, l'utilisation d'algorithmes décodeurs, comme BF, BL, BLF, etc. L'inconvénient majeur de ce type d'algorithmes est le temps pris pour le décodage.

Les opérateurs de croisement les plus utilisés sont génériques et peuvent s'appliquer à tout

problème représenté par des permutations. Le principal problème de ces opérateurs est qu'ils ne sont pas capables de distinguer la (ou les) bonne(s) proprieté(s) des parents à transmettre aux descendants. C'est pour cela que nous avons proposé un nouvel opérateur de croisement **"spécifique"** appelé WAX.

Il est important aussi de compter avec une fonction d'évaluation capable de donner une bonne estimation aux individus qui ont de bonnes caractéristiques. C'est pour cela que nous avons proposé l'utilisation d'une fonction **hiérarchique** capable de filtrer les individus apparemment similaires mais qui présentent une distribution différente des rectangles.

Finalement, notre algorithme DGA est compétitif par rapport à l'état de l'art sur les jeux de test de petites tailles mais il a des difficultés pour trouver des solutions sur les grandes instances.

Chapitre 3

Fonctions d'évaluation et mécanismes de diversification dans les algorithmes de recherche locale appliqués au SPP

Sommaire

La recherche locale (RL) a donné de bons résultats sur le SPP. Les composants importants d'un algorithme de RL sont principalement la représentation, la fonction d'évaluation, le voisinage et la diversification. Il existe différentes méthodes de RL : la recherche tabou, le recuit simulé ou la descente par exemple. Dans ce chapitre nous faisons d'abord une description générale de la RL, pour ensuite expliquer le cas particulier de la recherche tabou appliquée au SPP.

3.1 Introduction

Les méthodes de recherche locale (RL) sont souvent présentées comme des méthodes approchées très efficaces pour trouver des solutions de "bonne qualité" [Aarts and Lenstra, 1997].

La recherche locale se fonde sur l'idée d'améliorer une solution existante de manière itérative. Plus précisément, elle passe d'une solution à une autre en appliquant des transformations (ou "mouvements") à partir d'une solution initiale.

La recherche locale se caractérise essentiellement par trois éléments : fonction d'évaluation, voisinage et mouvements. Une description générale d'un algorithme de RL est montrée dans l'algorithme 3.1.

Concernant le SPP, il existe plusieurs techniques de résolution de type RL, comme la Recherche Tabou (RT) ou le Recuit Simulé (RS) par exemple [Aarts and Lenstra, 1997].

Ce qui les distingue est généralement le *critère d'arrêt* (ligne 3 dans l'algorithme cf. 3.1), le *voisinage* employé (ligne 4), le *critère d'acceptation* du mouvement (ligne 5) et la *mise à jour* de la meilleure configuration s_* (ligne 8).

3.2 Les méthodes de recherche locale appliquées au SPP

3.2.1 Recherche locale

Dans le SPP, nous considérons un ensemble de solutions faisables et une fonction d'évaluation pour mesurer la qualité des solutions et les comparer deux à deux. Le problème est de trouver une solution optimale. La RL est probablement la technique utilisée la plus efficace pour ce type de problèmes.

3.2.2 Représentation et configuration initiale

Pour commencer la recherche d'une solution optimale dans l'ensemble des solutions faisables, il est nécessaire d'utiliser une structure de données pour représenter une solution. La taille de la représentation, c'est à dire la quantité de bits de stockage, est liée à la taille de l'instance [Aarts and Lenstra, 1997]. Une solution est représentée par un ensemble de variables dont les valeurs possibles appartiennent à certains domaines.

3.2.3 Voisinage

Dans un problème combinatoire comme le SPP, nous ne pouvons pas examiner toutes les solutions en temps raisonnable. La recherche locale examine une partie des solutions en suivant un voisinage.

Un voisinage $\mathcal{N}(s)$ est une fonction qui définit, pour chaque solution s, un ensemble de solutions (appelées "voisins") proches selon un critère spécifique.

Un algorithme de RL commence par générer une solution initiale puis entre dans un processus itératif d'exploration du voisinage. Si un voisin a une évaluation meilleure que celle de la solution courante, celle-ci est remplacée par ce voisin pour continuer le processus de recherche. Sinon, l'algorithme accepte ce voisin sous certaines conditions.

3.2.4 Mouvement

Le processus de recherche réalise des petits changements dans la solution courante. Généralement cela consiste à changer la valeur d'une ou plusieurs variables. Un changement est appelé *"mouvement"*.

3.3 Un schéma général

Soient S l'ensemble des solutions de l'espace de recherche et f la fonction d'évaluation. Le problème consiste à trouver $s_* \in S$ tel que $f(s_*) \leq f(s), \forall s \in S$.

Algorithme 3.1: Schéma général d'une recherche locale

```
   s ← GénérerConfigurationInitiale()
   s* ← s
3  tant que Continuer() faire
4  |  s' ← ChoisirVoisin(N(s))
5  |  si Accepter(s') alors
   |  |  s ← s'
   |  |  si f(s) < f(s*) alors
8  |  |  |  s* ← s
   |  |  fin
   |  fin
   fin
   retourner s*
```

3.4 SPP et recherche locale

Dans [Lodi *et al.*, 1999], les auteurs développent un algorithme tabou pour résoudre un problème de Bin Packing en deux dimensions proche de SPP. Concernant les contraintes d'orientation des pièces et de découpe guillotine, le problème est divisé en 4 sous-problèmes. 1) Orientation fixe (i.e. sans rotation possible) avec découpe guillotine ; 2) Orientation libre avec découpe guillotine ; 3) Orientation fixe avec découpe libre ; 4) Orientation libre avec découpe libre. Le troisième cas est le SPP.

Cette procédure, résumée dans l'algorithme 3.2, comprend deux étapes. D'abord on construit une solution initiale. Après, l'algorithme tabou cherche à minimiser la quantité de récipients. Le voisinage échange des objets entre les récipients. Dans la solution initiale il y a un objet par récipient. À chaque itération, l'algorithme choisit un récipient et une séquence de mouvements avec l'objectif de le vider.

La procédure SEARCH() explore le voisinage défini pour l'actuel paramètre k, pour chaque objet j dans le récipient t. Les mouvements candidats sont évalués en exécutant l'algorithme A sur les sous-instances générées pour tous les ensembles possibles formulés par les *k-tuples* de j et les autres récipients.

Algorithme 3.2: Recherche tabou de *Lodi et. al.[1999]*

construire une solution initiale d'évaluation z^* avec l'algorithme $A(1, ..., n)$
calculer une borne inférieure L de l'évaluation d'une solution optimale
si $z^* = L$ **alors**
  stop ;
fin
initialiser la liste tabou
mettre 1 seul objet par récipient
$z \leftarrow n$ (évaluation de la solution trouvée par tabou)
tant que *le temps limite n'est pas atteint* **faire**
  choisir le récipient t à vider
  diversité $\leftarrow$ false
  $k \leftarrow 1$ (définit la taille et la structure du voisinage)
  tant que *diversité = false and* $z^* > L$ **faire**
    $k_{in} \leftarrow k$ (sauvegarde de k car il peut être modifié dans la procédure SEARCH)
    SEARCH(t,k,diversité,z)
    $z^* \leftarrow min\{z^*, z\}$
    si $k \leq k_{in}$ **alors**
      choisir le nouveau récipient t à vider
    fin
  fin
  si $z^* \neq L$ **alors**
    appliquer la diversification
  sinon
    break
  fin
fin

Les comparaisons effectuées sur les jeux de test proposés par [Berkey and Wang, 1987] ont montré que Tabou obtenait des résultats satisfaisants.

Burke et Kendall [Burke and Kendall, 1999] ont fait une comparaison de trois méta-heuristiques AG, RT et RS appliquées à un problème de "stock cutting" : AG a été le moins efficace des trois.

Hopper et Turton [Hopper and Turton, 2001a] comparent différentes heuristiques et méta-heuristiques sur des instances avec optimal connus. Dans une première partie de la recherche, les auteurs ont fait une analyse des deux algorithmes de placement les plus utilisés : BL et BLF. Les résultats ont montré que BLF, malgré sa complexité en $O(n^3)$, a été le plus efficace. Les auteurs ont ensuite comparé différentes méthodes, dont le recuit simulé (RS), la descente ou la recherche aléatoire. L'algorithme le plus efficace a été une hybridation RS + BLF.

Iori et al. [Iori *et al.*, 2003] ont proposé un algorithme hybride composé de deux méta-heuristiques et de processus locaux pour améliorer la qualité des solutions trouvées (dont la RT de [Lodi *et al.*, 1999]). L'algorithme commence par construire une solution initiale pour ensuite appliquer un algorithme de post-optimisation.

Comme deuxième algorithme, les auteurs proposent une hybridation AG + RT avec représentation par permutations et évolution élitiste. L'opérateur de croisement utilisé correspond à OX3 [Davis, 1985].

A chaque génération, une solution est sélectionnée comme solution initiale pour l'algorithme tabou. Une fois que l'algorithme tabou est fini, la solution finale est incluse dans la population.

Parmi les variantes proposées par Iori et al. 2003 [Iori *et al.*, 2003] AG + RT a présenté les meilleurs résultats sur les jeux de test de Hopper et Turton [Hopper and Turton, 2001a].

Alvarez-Valdes et al. [Alvarez-Valdes *et al.*, 2007] ont créé l'un des algorithmes tabou les plus efficaces pour un problème de découpe sans contrainte de guillotine. Les composants essentiels de l'algorithme sont une initialisation par l'algorithme GRASP d'Alvarez-Valdes et al. [2005] [Alvarez-Valdes *et al.*, 2005] et l'utilisation de deux types de mouvements pour remplir et replacer les espaces vides.

Harwig et al [Harwig *et al.*, 2006] ont développé un algorithme tabou adaptatif pour résoudre un problème de bin-packing. La solution initiale est créée par un algorithme de placement appelé pcf-BL (variante de l'algorithme BL). Les résultats sont intéressants par rapport aux meilleurs résultats publiés sur les jeux de test proposés dans [Berkey and Wang, 1987; Lodi *et al.*, 1999].

Soke et Bingul [Soke and Bingul, 2006] ont proposé un algorithme génétique hybride et un algorithme RS avec représentation par permutations. Les deux algorithmes ont montré de bons résultats sur le jeu de test de [Hopper and Turton, 2001a], mais le plus efficace a été l'AG hybride.

3.5 Le cas particulier de la recherche tabou

3.5.1 Schéma général

Les composants classiques d'un algorithme tabou sont : la représentation et la configuration initiale, la fonction d'évaluation, le mouvement et le voisinage, la liste tabou, les critères d'aspiration et la diversification (cf. Algorithme 3.3).

Algorithme 3.3: Schéma général d'un algorithme tabou.

```
Engendrer une configuration initiale s
s* ← s
T ← ∅ liste tabou
tant que condition d'arrêt non satisfaite faire
    m ← meilleur mouvement (i.e. minimisant f) parmi ceux non tabou
    ou ceux vérifiant un critère d'aspiration
    Modifier s en effectuant le mouvement m
    Mettre T à jour
    si f(s) < f(s*) alors
        s* ← s
    fin
fin
retourner s*
```

3.5.2 Représentation et configuration initiale

En général, pour obtenir une configuration initiale, on applique un algorithme de placement ou "décodeur". Les plus connus sont BF, BLF, BL (et leurs variantes). La majorité de ces algorithmes travaille avec une représentation par permutations, [Lodi *et al.*, 1999; Iori *et al.*, 2003; Harwig *et al.*, 2006; Alvarez-Valdes *et al.*, 2007; Burke *et al.*, 2009].

Représentation directe

Une représentation directe est spécifique au problème et contient l'information essentielle qui décrit une solution.

La figure 3.1 montre un exemple de représentation directe où nous pouvons voir clairement l'information de la solution.

Hamiez et al. [Hamiez *et al.*, 2009] ont proposé un algorithme avec représentation directe pour le SPP.

3.5.3 Fonction d'évaluation

La fonction d'évaluation est l'un des composants qui permet de guider la recherche dans les algorithmes de RL. Voici quelques exemples de fonctions d'évaluation utilisées par les algorithmes de type RL.

Alvarez-Valdes et al. [Alvarez-Valdes *et al.*, 2007], modélisent le SPP comme un problème de sac-à-dos avec valuation des objets, l'objectif étant de maximiser la valeur des pièces coupées :

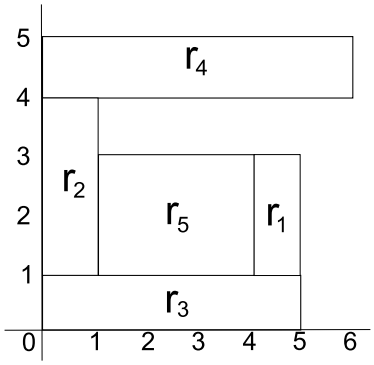
$f(x) = \Sigma_i v_i * x_i$

où x_i est la quantité de pièces de type i et v_i est la valuation associée aux pièces de type i.

Les auteurs utilisent également une deuxième fonction d'évaluation ayant d'autres objectifs :

$g(x) = k_1 S + k_2|\mathcal{L}| + k_3 C + k_4 F$

où S évalue la symétrie (on préfère des solutions non symétriques où les espaces vides sont concentrés le plus en haut et le plus à droite possible, $S = 0$ si une symétrie est détectée, sinon



Objet	Dimensions		Localisation	
	largeur	hauteur	x	y
r_1	1	2	4	1
r_2	1	3	0	1
r_3	5	2	0	0
r_4	6	1	0	4
r_5	3	2	1	1

FIGURE 3.1 – Représentation directe

$S = 1$). $|\mathcal{L}|$ est le nombre d'espaces vides (on préfère des solutions avec peu d'espaces vides). C est la concentration des espaces vides (on préfère des solutions où les espaces vides sont plus centrés et concentrés). Soit ER le plus petit rectangle contenant tous les espaces vides : $C = 1-(0.75*rd+0.25*ra)$, rd est la distance entre le centre de ER et le centre du packing divisée par la distance entre le centre du packing et le point de coordonées $(0,0)$. ra est le rapport *surface de ER* sur *aire de la bande*. $F = 1$ si la solution initiale n'est pas réalisable, $F = 0$ sinon. k_i sont des paramètres. Notons que le fait d'employer une deuxième fonction d'évaluation revient à utiliser une fonction hiérarchique.

L'algorithme tabou de Hamiez et al. [Hamiez *et al.*, 2009] a comme objectif principal de minimiser la hauteur maximale des objets placés :

$$f(s) = \begin{cases} 0 & \text{si} \lceil R \rceil = \emptyset \\ \sum_{r_i \in \lceil R \rceil} w_i^r * (y_i^r + h_i^r - H^* + p_H) & \text{autrement} \end{cases} \quad (3.1)$$

où $\lceil R \rceil \subseteq R$ est l'ensemble des objets $r_i / y_i^r + h_i^r > H^* - p_H$ avec $0 < p_H < H^*$ entier utilisé pour décrémenter la hauteur H et H^* est la meilleure hauteur atteinte.

3.5.4 Voisinage

Dans [Lodi *et al.*, 1999], un mouvement consiste à choisir un récipient et essayer de le vider.

Dans l'algorithme tabou de Iori et al. [Iori *et al.*, 2003] un mouvement consiste à couper un récipient en m récipients. Chaque objet placé est affecté à un nouveau récipient, les autres sont conservés dans le même récipient.

Voisinages relatifs au problème de bin packing

Dans [Harwig *et al.*, 2006], il existe 4 types de mouvements :

1. Ce mouvement commence par choisir deux objets à échanger. Quand les objets sont replacés ils peuvent provoquer un conflit avec un autre objet déjà placé. Dans ce cas, l'objet en conflit est retiré du récipient et ajouté à la liste des objets non placés.
2. Ce mouvement considère l'objet d'aire $h_i^r w_i^r$ maximale. Cet objet est placé dans un autre récipient et les objets qui sont en conflit sont replacés dans le même récipient sans dépasser ses bornes. Les objets qui dépassent sont éjectés et replacés dans un autre récipient.
3. Sélectionner un objet et le changer de place dans le même récipient. La nouvelle place pour l'objet correspond à un des espaces vides dans le récipient. Le remplissage de l'espace vide peut entraîner une chaîne d'éjection (cf. §1).
4. Ce mouvement consiste à échanger deux objets entre deux récipients différents. Les objets doivent être de tailles différentes et ne pas provoquer d'éjection.

Voisinages utilisés dans les problèmes de découpe

Un des algorithmes tabou les plus efficaces est présenté dans [Alvarez-Valdes *et al.*, 2007] où nous pouvons trouver deux types de mouvements reposant sur la notion de "bloc" (ensemble de rectangles) :

– Réduction d'un bloc : sélectionner un bloc déjà placé dans la solution et libérer l'espace. Après, avec un algorithme constructif, les blocs les plus proches sont replacés aux coins les plus proches, pour après remplir les espaces vides avec de nouveaux blocs non placés.
– Insertion d'un Bloc : les rectangles en conflit sont retirés.

Voisinages spécifiques au SPP

Dans [Hamiez *et al.*, 2009] il existe deux types de mouvements, chacun avec un objectif différent :

– Réduire la hauteur du packing : le mouvement replace certains objets en haut du packing dans des espaces vides situés plus bas selon les règles de placement de l'algorithme BLF.
– Éviter les solutions avec des espaces vides hauts et minces (appelés "tours") : ce mouvement essaye de supprimer les tours en y plaçant des objets (en testant les quatre coins de la tour considérée) qui n'y sont pas adjacents et dont l'aire est plus grande que l'aire de la tour considérée.

3.5.5 Liste tabou et critères d'aspiration

La liste tabou est une structure utile pour tenter d'éviter de répéter des mouvements déjà effectués. En pratique, on interdira un mouvement pendant un certain temps (cette durée est parfois appelée "teneur").

Les critères d'aspiration sont des exceptions au statut tabou.

Dans [Alvarez-Valdes *et al.*, 2007], la liste tabou est liée à la fonction d'évaluation et à certains espaces vides.

Dans [Harwig *et al.*, 2006] l'algorithme travaille avec des contraintes tabou pour prévenir l'exécution de mouvements récents. L'algorithme tabou présente deux types de contraintes tabou :

- Tabou Bin : cette contrainte empêche qu'un objet retourne au récipient d'où il est sorti dans n'importe quelle orientation.
- Tabou Order : cette contrainte empêche un objet de reprendre la place antérieurement utilisée dans un récipient.

Tabou Bin et Tabou Order utilisent une teneur tabou adaptative. Si le mouvement sélectionné n'améliore pas la valeur de la fonction d'évaluation, la teneur tabou est décrémentée de 1 (en respectant une valeur limite inférieure), par contre, si le mouvement sélectionné améliore la valeur de la fonction d'évaluation, la teneur tabou est incrémentée de 1 (en respectant une valeur limite supérieure).

Concernant les critères d'aspiration, dans [Harwig *et al.*, 2006; Alvarez-Valdes *et al.*, 2007; Hamiez *et al.*, 2009], les mouvements tabous sont acceptés s'ils permettent d'améliorer la qualité de la meilleure solution trouvée.

3.5.6 Diversification

Le mécanisme de diversification permet d'explorer de nouvelles zones dans l'espace de recherche, en particulier quand l'algorithme se stabilise dans un optimum local.

Dans [Harwig *et al.*, 2006] les auteurs appliquent une sélection de mouvements dynamiques comme stratégie de diversification. Cette stratégie consiste à modifier la fréquence et la taille de certains voisinages :

- Nombre des objets à placer.
- Quantité des objets dans un récipient.
- Nombre d'itérations depuis la meilleure solution trouvée.
- Nombre d'itérations depuis le dernier mouvement d'amélioration
- Nombre de mouvements consécutifs pour remplir un récipient depuis un autre.
- Nombre de mouvements consécutifs entre récipients ("swap" des objets).
- Nombre de mouvements consécutifs à l'intérieur du récipient.
- Stratégie de recherche spécifique pour le cas avec une quantité d'objets dans un récipient inférieure à 7 ou pas.

Dans [Alvarez-Valdes *et al.*, 2007] le processus de diversification utilise une mémoire pour garder la fréquence d'apparition de chaque objet dans la solution : on favorise les objets qui n'apparaissent pas souvent dans la solution. Cette diversification se traduit par une modification de la fonction d'évaluation.

Finalement, dans [Hamiez *et al.*, 2009], le processus de diversification repose sur un ensemble de solutions de bonnes qualités. Le processus consiste à sélectionner au hasard une solution dans l'ensemble des meilleures solutions. La solution choisie est perturbée par un mouvement aléatoire.

3.6 Conclusion

Pour le SPP, la RL est une des techniques les plus efficaces. On y trouve des algorithmes métaheuristiques comme la descente, la recherche aléatoire, le recuit simulé ou la recherche tabou

par exemple. La majorité de ces algorithmes utilise un algorithme décodeur comme BL ou ses variantes. Concernant la représentation, la plus communément utilisée est celle reposant sur des permutations mais la représentation directe commence à prendre de l'importance. Un voisinage fréquemment utilisé consiste à changer une paire d'objets.

Chapitre 4

Un nouvel algorithme de recherche tabou pour le SPP

Sommaire

Dans les méthodes approchées, la RL s'est montrée efficace dans la résolution du SPP. Entre les différentes techniques, la recherche tabou (RT) est l'une de celles les plus utilisées pour résoudre les problèmes d'optimisation [Glover and Laguna, 1997].

Notre approche dans ce chapitre est le développement d'un algorithme de RT spécifique pour le SPP. Notre algorithme a des composants novateurs incluant des connaissances du problème : fonction d'évaluation fondée sur les caractéristiques du placement, voisinage consistant et diversification basée sur l'historique de la recherche.

4.1 Introduction

Dans ce chapitre, nous présentons CTS (pour "Consistent Tabu Search") [Gómez-Villouta *et al.*, 2010], un algorithme de recherche tabou dédié au SPP. Comparé aux autres algorithmes pour le SPP, notre approche possède quelques caractéristiques particulières : CTS utilise la notion de voisinage consistant [Dupont *et al.*, 2004; Dupont *et al.*, 2005] et évalue des placements, éventuellement partiels, en utilisant plusieurs informations du problème. Finalement, notre algorithme inclut un mécanisme de diversification basé sur l'historique de la recherche. Les résultats expérimentaux montrent que CTS présente un certain intérêt pour résoudre le SPP.

La recherche tabou (RT) est l'une des méta-heuristiques les plus utilisées pour résoudre les problèmes d'optimisation [Glover and Laguna, 1997]. Elle explore un voisinage en utilisant une mémoire. Soient (S, f) l'espace de recherche et la fonction d'évaluation respectivement.

Un voisinage $\mathcal{N}$ (pour "neighborhood") sur S est une fonction qui associe à chaque individu $s \in S$ d'autres solutions $\mathcal{N}(s) \subseteq S$. Toute solution $s' \in \mathcal{N}(s)$ est un voisin de s. Pour un voisinage $\mathcal{N}$, une solution s est un "optimum local" si s est la meilleure solution entre les solutions (voisins) dans $\mathcal{N}(s)$. La transition d'une solution à un voisin est appelée "mouvement". On applique un mouvement μ à une solution s en la changeant légèrement pour visiter une solution voisine s'. Cette opération est dénotée $s' = s \oplus \mu$.

Soit $\Gamma(s)$ l'ensemble de tous les mouvements que l'on peut appliquer à s : $\mathcal{N}(s)$=$\{s \oplus \mu / \mu \in \Gamma(s)\}$.

Un algorithme de RT typique commence avec une solution initiale $s \in S$ et, dans un processus itératif, visite d'autres solutions. À chaque itération, on cherche l'un des meilleurs voisins $s' \in \mathcal{N}(s)$ pour remplacer la solution courante s, même si s' n'améliore pas la valeur de la fonction d'évaluation de la solution actuelle.

Finalement, pour tenter d'éviter les cycles et permettre une exploration plus efficace, l'algorithme utilise une mémoire à court terme appelée "liste tabou". L'utilisation d'une telle mémoire permet d'éviter de visiter des solutions déjà visitées auparavant.

4.2 Composants

Dans les sections suivantes (4.2.1-4.2.4) nous décrivons les composants spécifiques de notre recherche tabou, où toutes les variables p (avec indices) sont des paramètres dont les valeurs sont données dans la partie expérimentation (section 4.3). La procédure générale CTS est finalement résumée dans la Section 4.2.5.

4.2.1 Processus de résolution

Soient $H(s)$ la hauteur d'un placement s (objectif, à miniminiser) et $\mathsf{SPP}_{k>0}$ le problème de satisfaction suivant : existe t-il une solution s au SPP tel que $H(s) \leq k$?. Le problème d'optimisation SPP est équivalent à trouver le k minimum tel que SPP_k admette une solution.

L'algorithme commence par construire une configuration complète s_0 de hauteur $H(s_0)$, par exemple créée avec un algorithme décodeur discuté dans le chapitre 3 (section 3.5.2). CTS cherche alors une solution s pour $\mathsf{SPP}_{k<H(s_0)}$ et ensuite essaie de résoudre $\mathsf{SPP}_{H(s)-p_H}$ ($p_H > 0$).

Espace de recherche : une représentation directe

Beaucoup d'approches pour le SPP, ou ses proches variantes, considèrent un espace de recherche S composé de l'ensemble des permutations des objets [Gómez-Villouta *et al.*, 2007; Soke and Bingul, 2006; Yeung and Tang, 2004; Iori *et al.*, 2003].

Plus précisément, pour un ensemble R de n objets à placer, une permutation $s \in S$ de [1,..., n] est construite (statiquement ou dynamiquement) en utilisant une *heuristique de sélection* σ [1] suivie d'une *heuristique de placement* ϕ donnée ("décodeur"). En d'autres termes, étant donnés un opérateur de sélection σ et un décodeur ϕ, on peut placer tous les objets en utilisant ϕ conformément à l'ordre imposé par σ, voir l'algorithme 4.1 où s_ρ est l'élément de rang ρ dans la permutation s. Le problème est donc de trouver une permutation particulière $s_* \in S$ (parmi les $n!$ disponibles) tel que le placement résultant soit optimal i.e. $H(s_*) = H_{OPT}$ (H_{OPT} est la hauteur optimale).

De nombreuses heuristiques de sélection / placement (en général gloutons) ont été proposées [Asik and Özcan, 2009; Burke *et al.*, 2009; Alvarez-Valdes *et al.*, 2007; Alvarez-Valdes *et al.*, 2008b].

Algorithme 4.1: L'algorithme glouton le plus simple pour le SPP.

Entrées : un opérateur de sélection σ et une heuristique de placement ϕ
$R^{'} \leftarrow R$
pour ρ = *1 to* n **faire**
  Sélectionner un rectangle $r_i \in R^{'}$ selon σ
  $R^{'} \leftarrow R^{'} \setminus \{r_i\}$
  $s_\rho \leftarrow i$
  Placer l'objet r_i dans la bande selon ϕ
fin
retourner $f(s)$ et s

CTS ne repose pas sur des permutations [2], mais adopte une représentation directe où une solution $s \in S$ (optimale ou non, éventuellement partielle) est un ensemble $\{L, E\}$ avec :

- $L \subseteq R$ est l'ensemble des rectangles correctement placés dans la bande, i.e r_i vérifie (1.2) avec $y_i^r + h_i^r \leq k \forall_{r_i} \in L$ et (r_i, r_j) vérifie (1.3-1.4) $\forall (r_i, r_{j \neq i}) \in L \times L$. Soit $\bar{L} = R \backslash L$ l'ensemble des objets non placés dans la bande.
- E pour "Empty" est l'ensemble des espaces vides rectangulaires dans la bande. Chaque espace vide $e_i \in E$ est caractérisé par la coordonnée (x_i^e, y_i^e) de son coin inférieur gauche, une largeur $0 < w_i^e \leq W$, et une hauteur $0 < h_i^e \leq k$, avec $0 \leq x_i^e \leq W - w_i^e$ et $0 \leq y_i^e \leq k - h_i^e$.
  Chaque espace vide $e_i \in E$ est un rectangle maximal [3], i.e. e_i n'est pas inclus dans un autre espace vide e_j : $\forall (e_i, e_{j \neq i}) \in E \times E, x_i^e < x_j^e \vee x_i^e + w_i^e > x_j^e + w_j^e \vee y_i^e < y_j^e \vee y_i^e + h_i^e >$

1. σ introduit donc un ordre sur les objets.
2. D'autres approches n'utilisant pas les permutations pour modéliser le problème, ou ses variantes proches peuvent aussi être trouvées i.e. dans [Hamiez *et al.*, 2009; Soh *et al.*, 2008; Bortfeldt, 2006]
3. La notion "espace vide maximal rectangulaire" semble avoir été présentée indépendamment dans [El Hayek *et al.*, 2008] (dénommée "aire maximale") et [Neveu *et al.*, 2008] ("trou maximal"). $|E|$ est dans le pire cas en $O(n^2)$ selon [El Hayek *et al.*, 2008].

$y_j^e + h_j^e$.

Configuration initiale

La RT a besoin d'une configuration initiale s_0 qui spécifie où commence la recherche dans l'espace de recherche S. CTS utilise l'algorithme 4.1 pour construire s_0, où l'heuristique de placement ϕ est "Bottom Left Fill" (BLF) de [Baker *et al.*, 1980] et l'opérateur de sélection σ ordonne les rectangles $r_i \in R$ en premier lieu par largeur décroissante, ensuite par hauteur décroissante (quand deux objets r_i et $r_{j\neq i}$ ont la même largeur), et au hasard si nécessaire (quand r_i et $r_{j\neq i}$ ont mêmes largeur et hauteur).

BLF est capable de remplir des espaces vides complètement entourés de pièces. Dans le chapitre 1 nous avons un exemple d'une configuration construite par BLF dans la figure 1.11, où le rectangle r_5 doit être placé. BLF a une complexité, dans le pire de cas, en $O(n^3)$ [Hopper and Turton, 2001a] ou en $O(n^2)$ [Chazelle, 1983] selon les implémentations. Nous employons ce décodeur et cet ordre car quelques expériences antérieures [Imahori *et al.*, 2006; Hopper and Turton, 2001a] suggèrent que l'algorithme de placement BLF surpasse en général d'autres décodeurs [4].

Notons que s_0 est une solution pour SPP_k $\forall k \geq H(s_0)$. Ainsi, s_0 fournit une borne supérieure triviale pour le SPP : $H_{OPT} \leq H(s_0)$.

Fonction d'évaluation

Cette mesure est l'un des composants clefs de la recherche tabou parce que cela guide la recherche.

Soient $M_w = max_{r_i \in \bar{L}}\{w_i^r\}$ (resp. $M_h = max_{r_i \in \bar{L}}\{h_i^r\}$) la largeur (resp. la hauteur) maximale d'un objet non placé, $\alpha = |\{r_i \in \bar{L} / w_i^r = M_w\}|$ le nombre de rectangles non placés de largeur M_w, $\delta = \sum_{e_j \in E}(W - x_j^e)(k - y_j^e)$ la "densité" d'un placement considéré [5] et $M_a = max_{e_j \in E}\{w_j^e * h_j^e\}$ l'aire maximale d'un espace vide. CTS utilise la fonction f formellement définie par 4.1 (à minimiser) pour évaluer une solution $s \in S$ (éventuellement partielle), voir un exemple dans la figure 4.1.

$$f(s) = \begin{cases} M_w M_h \alpha & \text{si} \quad E = \emptyset \\ \frac{M_w M_h \alpha \delta}{M_a} & \text{autrement} \end{cases} \tag{4.1}$$

La valeur $f(s)$ mesure la qualité de la solution s pour le problème SPP_k considéré :

- $f(s) = 0$ signifie que s est une solution pour SPP_k. Elle est "parfaite" si elle ne contient aucun espace vide ($E = \emptyset$). Dans ce cas, s est une solution optimale pour le SPP (i. e. $\mathsf{SPP}_{k'}$ n'admet aucune solution $\forall k^{'} < k$) : $H_{OPT} = k$
- $f(s) > 0$ indique un placement partiel. Ici, $E = \emptyset$ (aucun espace vide) veut dire que $\mathsf{SPP}_{k'}$ n'a pas de solution $\forall k^{'} \leq k$ et qu'une borne inférieure triviale est trouvée pour le SPP : $H_{OPT} > k$.

4. Cependant, tandis que BLF a une garantie de performance asymptotique de $3H_{OPT}$, d'autres heuristiques sont parfois plus efficaces : 2.7 [Coffman *et al.*, 1980], 2 [Schiermeyer, 1994], 1 + *e* [Baker *et al.*, 1981].

5. Une "petite" valeur de δ indique que (presque) tous les espaces vides sont concentrés près du coin supérieur droit de la bande.

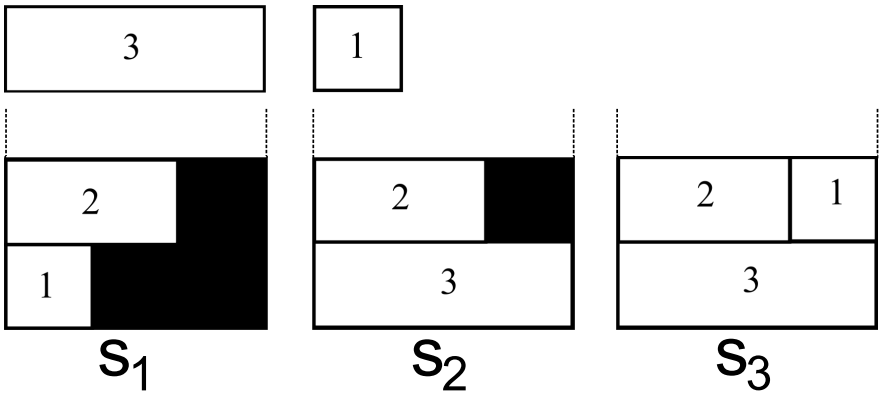


FIGURE 4.1 – Soient r_1 le carré unitaire (i.e. $w_1^r = h_1^r = 1$) et $k = 2$. $f(s_1) = 9$ car $E \neq \emptyset$, $M_w = w_3^r = 3$, $M_h = h_3^r = 1$, $\alpha = 1$, $\delta = (3.1)(2.0) + (3.2)(2.0) = 6$, et $M_a = 2$. De même, $f(s_2) = 1$ car $E \neq \emptyset$,$M_w = 1$,$M_h = 1$, $\alpha = 1$, $\delta = 1$, et $M_a = 1$. $f(s_3) = 0$ car $E = \emptyset$ et $\alpha = 0$.

La fonction d'évaluation f et la fonction objectif H, sont utilisées pour comparer deux placements s_1 et s_2 (éventuellement partiels) : s_1 est "meilleur" que s_2 si l'évaluation de s_1 est inférieure à celle de s_2, formellement $f(s_1) < f(s_2)$. Cependant, f est inappropriée quand s_1 et s_2 sont deux solutions de SPP_k, i.e quand $f(s_1) = f(s_2) = 0$. Dans ce cas, s_1 est meilleur que s_2 si $H(s_1) < H(s_2)$.

D'autres fonctions d'évaluation ont été proposées pour le SPP. Neveu et al. [Neveu *et al.*, 2007] considèrent u "le nombre d'unités sur la plus haute ligne de la bande" pour définir $f_N(s) = W * (H(s) - 1) + u$. Hamiez et al. [Hamiez *et al.*, 2009] ont proposé une fonction d'évaluation similaire formellement définie par 4.2 qui repose sur l'ensemble $\lceil R \rceil \subseteq R$ des rectangles en haut de la bande où $\lceil R \rceil = \{r_i \in R / y_i^r + h_i^r > H_* - p_H\}$, p_H est un paramètre entier, et H_* est la meilleure hauteur trouvée, initialement $H(s_0)$ la hauteur de la solution de départ. On peut observer que f_N et f_H calculent une mesure uniquement basée sur les rectangles situés en haut de la bande et ne considèrent pas ce qui se passe au-dessous de ces rectangles. Notre fonction d'évaluation est plus informée car elle comprend une mesure (le composant δ) rattachée à cette information supplémentaire.

$$f_H(s) = \begin{cases} 0 & \text{si } \lceil R \rceil = \emptyset \\ \Sigma_{r_i \in \lceil R \rceil} w_i^r * (y_i^r + h_i^r - H_* + p_H) & \text{autrement} \end{cases} \tag{4.2}$$

4.2.2 Un voisinage consistant

Le voisinage $\mathcal{N}$ est également un composant important car il définit la façon dont l'espace de recherche est exploré.

Le principal objectif du voisinage de CTS est de vider l'ensemble $\bar{L}$. Il essaie de mettre dans la bande un rectangle r_i non encore placé (r_i passe de $\bar{L}$ à L) dans le coin en bas et à gauche d'un espace vide (définissant un premier sous-voisinage $\mathcal{N}_E$) ou à la place d'un rectangle déjà dans la bande (définissant un second sous-voisinage $\mathcal{N}_L$). Notons que le voisinage de CTS semble

assez proche de celui utilisé par IDW [Neveu *et al.*, 2008] : les rectangles non placés dans CTS ($\bar{L}$) correspondent aux rectangles placés les plus hauts pour IDW et les emplacements possibles sont les mêmes dans les 2 cas.

Le déplacement d'un rectangle r_i de L vers $\bar{L}$ peut générer des conflits avec un ensemble L_i de rectangles placés : $L_i = \{r_{j\neq i} \in L / x_i^r < x_j^r + w_j^r \wedge x_i^r + w_i^r > x_j^r \wedge y_i^r < y_j^r + h_j^r \wedge y_i^r + h_i^r > y_j^r\}$. Tous les rectangles en conflit avec le rectangle r_i sont retirés de la bande ($\forall r_j \in L_i, r_j$ passe de L à $\bar{L}$). Ce principe, connu sous le nom de "chaîne d'éjection", est utilisé pour respecter les contraintes (1.3 1.4). D'autres exemples de voisinages consistants sont disponibles (pour d'autres problèmes) e.g. dans [Dupont *et al.*, 2004; Dupont *et al.*, 2005].

Finalement, quand un rectangle est éjecté, de nouveaux espaces vides sont générés et E est mis à jour à l'aide des procédures incrémentales détaillées dans [Neveu *et al.*, 2008].

Voisinage $\mathcal{N}_E$

CTS examine deux cas avec deux objectifs différents : diminuer le nombre de rectangles non placés, i. e. minimiser $|\bar{L}|$ (définissant $\mathcal{N}_E^{|\bar{L}|}$), ou faire un placement plus dense ($\mathcal{N}_E^D$).

- Le cas de $\mathcal{N}_E^{|\bar{L}|}$. Tous les rectangles non placés $r_i \in \bar{L}$, sont replacés dans la bande dans le coin en bas et à gauche de tous les espaces vides $e_j \in E$ tel que e_j puisse contenir r_i entièrement : formellement, r_i et e_j doivent vérifier $x_j^e + w_i^r \leq W \wedge y_j^e + h_i^r \leq k \wedge w_j^e \geq w_i^r \wedge h_j^e \geq h_i^r$.
- Le cas de $\mathcal{N}_E^D$. Tous les rectangles non placés $r_i \in \bar{L}$ sont replacés dans la bande dans le coin en bas à gauche de tous les espaces vides $e_j \in E$ mais ici aucune restriction de taille n'est imposée. r_i et e_j doivent juste vérifier $x_j^e + w_i^r \leq W \wedge y_j^e + h_i^r \leq k$.

Voisinage $\mathcal{N}_L$

À partir d'une solution s, tous les rectangles non placés $r_i \in \bar{L}$, sont replacés dans la bande dans le coin en bas à gauche de tous les rectangles déjà placés $r_j \in L$ tel que r_i et r_j soient de différentes tailles : r_i et r_j doivent vérifier $(w_i^r \neq w_j^r \vee h_i^r \neq h_j^r) \wedge x_j^r + w_i^r \leq W \wedge y_j^r + h_i^r \leq k$.

Stratégie de recherche

Notre voisinage $\mathcal{N}$ est composé de trois sous-voisinages : $\mathcal{N}_E^{|\bar{L}|}$, $\mathcal{N}_E^D$ et $\mathcal{N}_L$. Ils sont explorés par CTS selon un ordre hiérarchique : le premier voisinage exploré est toujours $\mathcal{N}_E^{|\bar{L}|}$, $\mathcal{N}_E^D$ est utilisé seulement si $\mathcal{N}_E^{|\bar{L}|}$ est vide, $\mathcal{N}_L$ est (peut-être) appliqué en dernier. Notre voisinage est montré formellement dans l'algorithme 4.2.

Remarquez que si la configuration s n'a pas de voisin, i.e. $\mathcal{N}(s) = \mathcal{N}_E^{|\bar{L}|}(s) = \mathcal{N}_E^D(s) = \mathcal{N}_L(s) = \emptyset$, aucun mouvement n'est réalisé et la diversification est invoquée (voir sect. 4.2.4).

4.2.3 Liste tabou

Un mouvement m de CTS d'une solution s à un voisin $s' \in \mathcal{N}(s)$ consistant en la localisation d'un rectangle non encore placé $r_i \in \bar{L}$ dans la bande, il semble assez naturel d'interdire à l'objet

Algorithme 4.2: Le voisinage est exploré selon un ordre hiérarchique

Soit $s = \{L, E\}$ une configuration
si $\mathcal{N}_E^{|\bar{L}|}(s) \neq \emptyset$ **alors**
  retourner $\mathcal{N}_E^{|\bar{L}|}(s)$
sinon
  si $\mathcal{N}_E^D(s) \neq \emptyset$ **alors**
    retourner $\mathcal{N}_E^D(s)$
  sinon
    si $\mathcal{N}_L(s) \neq \emptyset$ **alors**
      retourner $\mathcal{N}_L(s)$
    sinon
      retourner $\emptyset$
    fin
  fin
fin

r_i de quitter la bande pendant un certain temps. Ce mouvement "inverse" sera stocké dans une liste tabou τ pour une durée $0 < p_\tau \leq n$ (entier) pour indiquer que r_i ne peut pas être enlevé de la bande au moins jusqu'à l'itération $m + p_\tau$.

τ est vidée au début de la recherche ou quand CTS trouve une solution au problème de satisfaction considéré (SPP_k).

4.2.4 Diversification

Soit s_* la meilleure configuration complète trouvée par CTS à l'itération m_* (au début de la recherche s_* est la configuration initiale s_0 construite par BLF).

Quand une solution s n'a pas de voisin (i.e. $\mathcal{N}(s) = \emptyset$) ou s_* n'est pas modifiée après un certain nombre $p_* > 0$ d'itérations, CTS réinitialise la liste tabou τ et recharge s_* dans s ($s \leftarrow s_*, \tau \leftarrow \emptyset$). s est ensuite perturbée selon deux schémas différents de diversification dénommés D_I (pour "Interchange", réalisée selon une probabilité p_D) et D_T (pour "Tetris", probabilité $1 - p_D$). Après la perturbation, p_* mouvements supplémentaires sont autorisés pour modifier s_*.

D_I : échange de 2 pièces

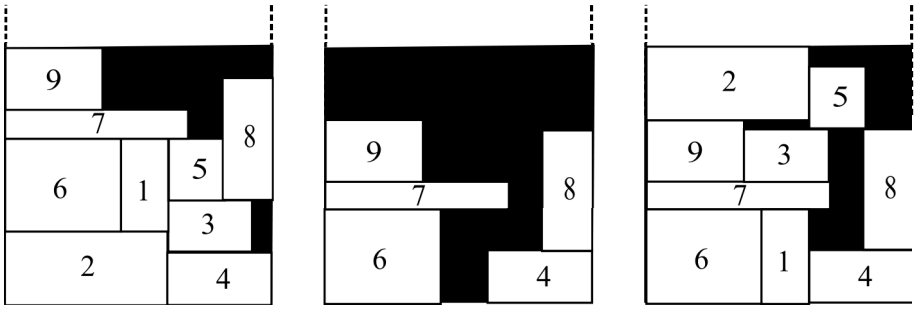
L et $\bar{L}$ sont initialisés en accord avec la définition du problème SPP_k avec $k = H(s_*) - p_H$: $L \leftarrow \{r_i \in R / y_i^r + h_i^r \leq k\}, \bar{L} \leftarrow R \backslash L$. La diversification D_I considère alors tous les couples $(r_i, r_j) \in L \times \bar{L}$ tel que r_i et r_j soient de tailles différentes, formellement r_i et r_j doivent vérifier $(w_i^r \neq w_j^r \vee h_i^r \neq h_j^r) \wedge x_i^r + w_j^r \leq W \wedge y_i^r + h_j^r \leq k$. Cela échange simplement deux éléments (sélectionnés au hasard) et met r_j à l'état tabou.

D_T : une perturbation basée sur l'historique

Durant la recherche, CTS garde pour chaque rectangle $r_i \in R$ le nombre F_i (pour "fréquence") de fois que r_i quitte la bande, i.e. le nombre de fois que r_i passe de L à $\bar{L}$ [6].

Le schéma de diversification D_T considère une permutation π_F qui ordonne tous les objets $r_i \in R$ d'abord par fréquences croissantes, ensuite par largeurs décroissantes (quand $F_i = F_{j\neq i}$), puis hauteurs décroissantes ($w_i^r = w_{j\neq i}^r$), et au hasard si cela est nécessaire ($h_i^r = h_{j\neq i}^r$). Soit $\lfloor F \rfloor$ l'ensemble composé par les premiers p_τ éléments de π_F.

Tous les rectangles $r_i \in \lfloor F \rfloor$ sont d'abord temporairement enlevés de la bande et leurs fréquences sont mises à jour [7]. Ensuite, le placement partiel est poussé en bas à la base de la bande, comme dans le fameux jeu Tetris. Finalement, tous les objets $r_i \in \lfloor F \rfloor$ sont ordonnés comme dans la Section 4.2.1 et replacés dans la bande avec BLF, voir la figure 4.2.



Soit $\lfloor F \rfloor = \{r_1, r_2, r_3, r_5\}$ — Retirer les éléments de $\lfloor F \rfloor$ et pousser en bas — Replacer les éléments de $\lfloor F \rfloor$

FIGURE 4.2 – La diversification D_T. Les expériences ont montré que presque tous les rectangles de basses fréquences (F_i) se situent dans le bas de la bande.

CTS traite maintenant SPP_k avec $k = H(s) - p_H : L \leftarrow \{r_i \in R / y_i^r + h_i^r \leq k\}$, $\bar{L} \leftarrow R \backslash L$. Cela veut dire que CTS considère éventuellement des problèmes SPP_k avec $k \geq H(s_0) \geq H(s_*)$.

4.2.5 Schéma général

La procédure CTS est résumée dans l'algorithme 4.3. CTS procède de manière itérative pour résoudre une série de problèmes de satisfaction SPP_k. S'il trouve une solution s pour SPP_k (i.e. $\bar{L} = \emptyset$), il essaie alors de résoudre le problème suivant $\mathsf{SPP}_{H(s)-p_H}$ (cf. lignes 4-11). Remarquez que $H(s) = H(s_*)$ (ligne 5) peut seulement arriver après la diversification D_T ou au début du processus de recherche.

Notons que CTS peut aussi finir avant que le nombre maximal $p_M \geq 0$ de mouvements permis ne soit atteint (ligne 3). Cela peut arriver chaque fois que s_* s'actualise, (lignes 2 et 6) ou quand la hauteur optimale H_{OPT} est connue (ou une borne supérieure) et $H(s_*) \leq H_{OPT}$.

6. $F_{1\leq i\leq n} = 0$ au début de la recherche.

7. $F_i \leftarrow 2F_i$ est utilisé ici pour éviter de considérer (presque) le même ensemble $\lfloor F \rfloor$ dans les utilisations suivantes de la diversification D_T tandis que $F_i \leftarrow F_i + 1$ est appliqué quand on réalise un mouvement.

Algorithme 4.3: CTS

Entrée : une configuration initiale s.
$m \leftarrow 0$, $s_* \leftarrow s$, $m_* \leftarrow 0$ //initialisation
tant que $m \leq p_M$ **faire**
  si $\bar{L} = \emptyset$ **alors**
    si $H(s) < H(s_*)$ *or (*$H(s) = H(s_*)$ *et probabilité* $p_{\approx}$ *verifiée)* **alors**
      $s_* \leftarrow s$
      $m_* \leftarrow m$
    fin
    $\tau \leftarrow \emptyset$//liste tabou
    $L \leftarrow \{r_i \in R / y_i^r + h_i^r \leq H(s) - p_H\}$
    $\bar{L} \leftarrow R \backslash L$
  fin
  si $\mathcal{N}(s) = \emptyset$ **alors**
    $Div \leftarrow$ TRUE //Diversification
  sinon
    $m \leftarrow m + 1$
    Soit $\lfloor \mathcal{N}(s) \rfloor$ l'ensemble des voisins s' de s les mieux évalués selon 4.1 :
    $\lfloor \mathcal{N}(s) \rfloor = \{s'_1 \in \mathcal{N}(s) / \forall s'_2 \in \mathcal{N}(s), f(s'_1) \leq f(s'_2)\}$
    si $f(s') > 0 \forall s' \in \lfloor \mathcal{N}(s) \rfloor$ **alors**
      si $(m - m_*)$ *mod* $p_* = 0$ **alors**
        $Div \leftarrow$ TRUE
      sinon
        $Div \leftarrow$ FALSE
      fin
      Sélectionner un $s' \in \lfloor \mathcal{N}(s) \rfloor$ au hasard
    sinon
      $Div \leftarrow$ FALSE
      Sélectionner un $s' \in \lfloor \mathcal{N}(s) \rfloor$ minimisant H (cf. eq 1.1) au hasard
    fin
  fin
  si Div **alors**
    //Diversification
    $s \leftarrow s_*$
    $\tau \leftarrow \emptyset$
    Modifier s en utilisant D_I ou D_T selon une probabilité p_D
  sinon
    $s \leftarrow s'$
    Mise à jour de τ
  fin
fin
retourner $H(s_*)$ et s_*

4.3 Expérimentations

Nous utilisons les 21 instances bien connues de Hopper et Turton [Hopper and Turton, 2001a] pour évaluer le fonctionnement de notre algorithme CTS. Pour mémoire, les caractéristiques principales de ces instances sont données dans le tableau 2.1 (cf. chapitre 2).

4.3.1 Protocole de tests

CTS est implémenté en langage C. Tous les résultats ont été obtenus sur un ordinateur cadencé à 2.83 Ghz (quad-core Intel® Xeon® E5440 et 8 Go de RAM) [8].

Les valeurs des paramètres de CTS sont variables selon les instances, excepté p_H qui vaut toujours 1 (p_H est utilisé pour construire la configuration de départ de SPP_k, l'actuel problème de satisfaction considéré). Déterminées de manière empirique, ces valeurs sont données dans le Tab. 4.1. Pour mémoire, $p_{\approx}$ est la probabilité que s remplace s_* quand $H(s) = H(s_*)$, p_τ est la teneur tabou, p_* est le nombre maximal de mouvements pour mettre à jour s_*, p_D est la probabilité d'appliquer la diversification D_I et p_M est le nombre maximal de mouvements par exécution.

Instances	$p_{\approx}$	p_τ	p_*	p_D	p_M
C1P1	0.5	2	200	1	7000000
C1P2	0.5	2	200	1	7000000
C1P3	0.5	2	200	1	7000000
C2P1	0.5	2	200	1	5000000
C2P2	0.5	2	200	1	5000000
C2P3	0.5	2	200	1	5000000
C3P1	0.5	3	200	1	2000000
C3P2	0.5	3	500	0.7	2000000
C3P3	0.5	3	200	1	2000000
C4P1	0.5	5	500	1	2000000
C4P2	0.5	3	200	1	2000000
C4P3	0.5	4	200	1	2000000
C5P1	0.5	6	200	1	2000000
C5P2	0.5	3	200	1	2000000
C5P3	0.5	4	500	0.7	2000000
C6P1	0.5	3	200	1	2000000
C6P2	0.5	6	500	1	2000000
C6P3	0.5	5	200	1	2000000
C7P1	0.5	3	200	1	2000000
C7P2	0.5	3	500	1	2000000
C7P3	0.5	3	500	1	2000000

TABLE 4.1 – Valeurs des paramètres variables de CTS

8. Le temps de résolution va de quelques secondes (pour les plus petites instances) à près de 33 heures (pour les instances de plus grandes tailles).

La comparaison est basée sur l'écart γ d'une solution s à l'optimum (H_{OPT}) : $\gamma(s) = 100 * (1 - H_{OPT}/H(s))$. Pour CTS, nous donnons une moyenne $\bar{\gamma}$ sur 5 exécutions et la meilleure valeur atteinte (γ_*).

4.3.2 Résultats comparatifs

CTS est comparé dans le tableau 4.2 avec cinq algorithmes : deux procédures de Recherche Tabou, dénotées TS1 [Iori *et al.*, 2003] [9] et TS2 [Hamiez *et al.*, 2009], l'un des Algorithmes Génétiques les plus efficaces (GA) de Bortfeldt [Bortfeldt, 2006], et les deux meilleures approches : "Greedy Randomized Adaptive Search Procedure" (GRASP) d'Alvarez-Valdes et al. [Alvarez-Valdes *et al.*, 2008b] et une stratégie hybride HyperHeuristique + Intensification / Diversification Walk (HH+IDW) de Neveu et al. [Neveu *et al.*, 2008].

Dans le tableau 4.2, le symbole "-" (pour HH+IDW, GA, ou TS1) signifient que $\bar{\gamma}$ ou γ_* ne peut pas être calculé ou que l'information n'est pas disponible dans [Neveu *et al.*, 2008; Bortfeldt, 2006; Iori *et al.*, 2003]. "Mean C_i" est la moyenne des valeurs sur la catégorie C_i. Les lignes agrégées "$C_1 - C_j$" donnent les valeurs moyennes pour toutes les catégories C_1 à C_j. Ces valeurs peuvent être utilisées pour identifier jusqu'à quelle catégorie l'algorithme est efficace. La dernière ligne montre le nombre de cas à l'optimum.

Selon le tableau 4.2, TS1 a la pire performance pour le jeu de test. Presque tous les autres algorithmes (exceptés TS1 et GA) ont résolu les catégories C1 et C2, voir la ligne "C1-C2" où $\gamma_* = 0.00$ ou $\bar{\gamma} = 0.00$.

À notre connaissance, les uniques approches qui résolvent les 9 cas C1P1-C3P3 à l'optimum sont la recherche tabou d'Alvarez-Valdes et al. [Alvarez-Valdes *et al.*, 2007] et les procédures exactes décrites dans [Kenmochi *et al.*, 2009; Soh *et al.*, 2008]. Dans le tableau 4.2, CTS est la seule méthode qui atteint les mêmes résultats qualitatifs, voir la ligne "C1-C3" où $\gamma_* = 0.00$ seulement pour CTS. De plus, remarquez que CTS atteint ici la meilleure valeur pour $\bar{\gamma}$ $(0.29 < 0.36 < 0.89 < 1.64 < 2.69)$.

Les résultats agrégés montrent que CTS est compétitif vis à vis des concurrents si l'on considère les cas jusqu'à C5. De plus, la ligne "C1-C4" indique la meilleure valeur de γ_* pour CTS $(0.41 < 0.68 \leq 0.68 \leq 0.68 < 2.38 < 5.40)$ et $\bar{\gamma}$ $(0.63 < 0.68 < 1.08 < 1.84 < 2.90)$. La même observation est valable en ligne "C1-C5" seulement pour γ_* $(0.62 < 0.76 \leq 0.76 \leq 0.76 < 2.20 < 5.31)$ mais la différence est faible entre le meilleur $\bar{\gamma}$ (0.76 pour GRASP) et le résultat de CTS (0.85).

CTS obtient de moins bonnes valeurs de γ_* ou $\bar{\gamma}$ que GRASP et HH+IDW seulement en ajoutant les deux plus grandes catégories. Pour C1-C6, notons, cependant, que la différence demeure raisonnable pour γ_* $(0.83 - 0.77 = 0.06)$.

4.4 Conclusion

Dans ce chapitre, nous avons présenté CTS, un algorithme à voisinage consistant de Recherche Tabou pour le problème de placement en deux dimensions (SPP). CTS repose sur quelques composants déjà utilisés ou similaires à d'autres approches, comme la représentation directe du pro-

9. TS1 est peut-être le premier algorithme de recherche tabou pour SPP.

Instances	CTS		TS1	TS2		GA		GRASP		HH+IDW	
	$\overline{\gamma}$	γ_*	γ_*	$\overline{\gamma}$	γ_*	$\overline{\gamma}$	γ_*	$\overline{\gamma}$	γ_*	$\overline{\gamma}$	γ_*
C1P1	0.0	0.0	9.09	0.0	0.0	-	-	0.0	0.0	-	-
C1P2	0.0	0.0	9.09	4.76	0.0	-	-	0.0	0.0	-	-
C1P3	0.0	0.0	4.76	0.0	0.0	-	-	0.0	0.0	-	-
Mean C1	0.0	0.0	7.65	1.59	0.0	1.59	1.59	0.0	0.0	0.48	0.0
C2P1	0.0	0.0	0.0	0.0	0.0	-	-	0.0	0.0	-	-
C2P2	0.0	0.0	6.25	0.0	0.0	-	-	0.0	0.0	-	-
C2P3	0.0	0.0	0.0	0.0	0.0	-	-	0.0	0.0	-	-
Mean C2	0.0	0.0	2.08	0.0	0.0	3.33	2.08	0.0	0.0	1.87	0.0
C3P1	0.0	0.0	3.23	0.0	0.0	-	-	0.0	0.0	-	-
C3P2	2.60	0.0	9.09	3.23	3.23	-	-	3.23	3.23	-	-
C3P3	0.0	0.0	9.09	0.0	0.0	-	-	0.0	0.0	-	-
Mean C3	0.87	0.0	7.14	1.08	1.08	3.16	3.16	1.08	1.08	2.58	1.08
C4P1	1.64	1.64	6.25	1.64	1.64	-	-	1.64	1.64	-	-
C4P2	1.64	1.64	4.76	1.64	1.64	-	-	1.64	1.64	-	-
C4P3	1.64	1.64	3.23	1.64	1.64	-	-	1.64	1.64	-	-
Mean C4	1.64	1.64	4.75	1.64	1.64	3.52	2.70	1.64	1.64	2.43	1.64
C5P1	2.17	2.17	5.26	1.10	1.10	-	-	1.10	1.10	-	-
C5P2	1.96	1.10	3.23	1.10	1.10	-	-	1.10	1.10	-	-
C5P3	1.10	1.10	6.25	1.10	1.10	-	-	1.10	1.10	-	-
Mean C5	1.74	1.46	4.91	1.10	1.10	2.03	1.46	1.10	1.10	1.78	1.10
C6P1	1.80	1.64	4.76	1.64	0.83	-	-	1.56	0.83	-	-
C6P2	2.44	2.44	3.23	0.83	0.83	-	-	1.56	0.83	-	-
C6P3	2.28	1.64	3.23	1.64	0.83	-	-	1.56	0.83	-	-
Mean C6	2.17	1.91	3.74	1.37	0.83	1.72	1.64	1.56	0.83	1.75	1.10
C7P1	2.20	2.04	-	1.23	1.23	-	-	1.64	1.64	-	-
C7P2	2.04	2.04	-	1.23	1.23	-	-	1.19	0.83	-	-
C7P3	2.20	2.04	-	1.23	1.23	-	-	1.23	1.23	-	-
Mean C7	2.15	2.04	-	1.23	1.23	1.52	1.23	1.36	1.23	1.42	1.10
C1-C2	0.00	0.00	4.86	0.79	0.00	2.46	1.84	0.00	0.00	1.18	0.00
C1-C3	0.29	0.00	5.62	0.89	0.36	2.69	2.28	0.36	0.36	1.64	0.36
C1-C4	0.63	**0.41**	**5.40**	1.08	**0.68**	2.90	**2.38**	0.68	**0.68**	1.84	**0.68**
C1-C5	0.85	**0.62**	**5.31**	1.08	**0.76**	2.73	**2.20**	0.76	**0.76**	1.83	**0.76**
C1-C6	1.07	0.83	5.04	1.13	0.77	2.57	2.11	0.90	0.77	1.82	0.82
C1-C7	1.22	1.01	-	1.14	0.84	2.41	1.98	0.96	0.84	1.76	0.86
$\#H_{OPT}/21$	9		2	8		-		8		8	

TABLE 4.2 – Moyenne et meilleur écart à l'optimum ($\overline{\gamma}$ and γ_* resp.) sur les instances de Hopper et Turton [2001]

blème ou le voisinage mentionné e.g. dans [Hamiez *et al.*, 2009]. CTS a néanmoins introduit deux nouveaux éléments qui, à notre connaissance, n'ont jamais été essayés pour le SPP :

- Une fonction d'évaluation incluant une mesure relative aux espaces vides. Cela a été motivé par le fait que la plupart des études antérieures sur le SPP emploient en général des fonctions d'évaluation uniquement basées sur les rectangles.
- Un schéma de diversification (D_T) basé sur une mesure de fréquence. Nous avons observé que certains rectangles restaient presque toujours dans la bande ou changeaient rarement de position.

Ces composants ont démontré leur utilité dans l'algorithme CTS.

Conclusion générale

Principales contributions

Les travaux réalisés tout au long de cette thèse mettent en évidence différentes contributions autour du SPP de placement sur bande en deux dimensions (sans orientation des pièces).

Tout d'abord, nous avons réalisé une étude des méthodes de résolution pour les problèmes de placement et de découpe. Nous avons classé les méthodes en deux grands groupes. Les méthodes les plus efficaces appartiennent à la classe des méthodes approchées, en particulier la recherche locale.

L'algorithme génétique (AG) est une des méta-heuristiques les plus utilisées pour SPP. En général, les AG utilisent une représentation par permutations et un algorithme décodeur comme BLF (ou ses variantes). Les opérateurs de croisement existants sont génériques à n'importe quel problème avec cette représentation. Ces opérateurs ne sont pas spécifiques au problème, donc la manipulation de l'information est "aveugle".

Concernant les fonctions d'évaluation, elles se concentrent généralement sur la hauteur. Elles ne sont pas très efficaces dès qu'il existe plusieurs individus différents avec la même hauteur.

Nous avons conçu un nouvel algorithme génétique appelé DGA (pour "Dedicated Genetic Algorithm"). Il est composé d'un opérateur de croisement "spécifique" WAX et d'une fonction d'évaluation hiérarchique.

L'opérateur WAX tente de distinguer la (ou les) bonne(s) propriété(s) des parents à transmettre aux descendants. La fonction d'évaluation hiérarchique est capable de différencier les individus de même hauteur.

Les comparaisons expérimentales entre DGA et d'autres algorithmes (les plus récents et les meilleurs) effectuées sur un ensemble d'instances bien connues et difficiles mettent en évidence la compétitivité de DGA sur les instances de petites tailles, mais DGA éprouve quelques difficultés à trouver de bonnes solutions pour les grandes instances.

Dans une deuxième étape de nos travaux, nous avons analysé d'autres méta-heuristiques classiques appliquées au SPP : Recuit Simulé, Descente, Recherche Aléatoire, Évolution Naïve et Recherche Tabou. Parmi ces approches, les algorithmes basés sur la RT sont les plus efficaces. Néanmoins, les stratégies les plus performantes actuellement sont des méthodes assez sophistiquées comme Reactive GRASP [Alvarez-Valdes *et al.*, 2008b] ou HH+IDW [Neveu *et al.*, 2008]. Elles reposent généralement sur de bonnes heuristiques gloutonnes de placement, de la recherche locale, des structures de données efficaces ou de la recherche à voisinage variable par exemple.

Nous avons analysé l'influence de composants clés sur la performance globale des algorithmes tels que : la représentation, le voisinage, la fonction d'évaluation et la diversification.

Suite à l'étude de ces composants importants, nous avons développé CTS (pour "Consistent Tabu Search") un algorithme de recherche tabou à voisinage consistant. Il est renforcé en introduisant principalement deux nouveaux éléments qui, à notre connaissance, n'ont jamais été essayés pour SPP : une fonction d'évaluation qui considère une mesure relative aux espaces vides pour diriger de manière plus efficace le processus de recherche et un mécanisme de diversification basé

sur l'historique de la recherche.

Ces composants se sont révélés utiles pour CTS. Nous pensons que les concepts ayant conduit au développement de ces composants pourraient être applicables à d'autres problèmes d'optimisation.

Perspectives de recherche

L'état actuel de nos travaux laisse entrevoir plusieurs pistes à explorer.

Dans l'algorithme génétique DGA, notre perspective de recherche est de tenter d'identifier pourquoi il atteint de moins bons résultats quand il est appliqué aux instances de grandes tailles. D'autres versions de WAX pourraient aussi être expérimentées afin de chercher à savoir si un ajout de diversification s'avèrerait utile pour l'opérateur. De même, différentes améliorations pourraient encore être apportées à la fonction d'évaluation afin de rendre l'algorithme plus performant en termes de qualité et de temps de calcul.

En ce qui concerne l'algorithme tabou CTS, les résultats ont montré une bonne performance, mais il présente de moins bons résultats sur les instances les plus grandes. Il s'avère donc nécessaire d'étudier plus en détail les composants de l'algorithme et leur influence sur la qualité des solutions, et le sur temps de calcul.

Une autre perspective intéressante est de tester nos algorithmes sur d'autres instances comme celles utilisées dans [Beasley, 1985a; Beasley, 1985b; Christofides and Whitlock, 1977; Christofides and Hadjiconstantinou, 1995; Fekete and Schepers, 1997b].

Une dernière perspective de recherche est celle de renforcer l'algorithme DGA sur des caractéristiques clés qui impactent de manière très importante sa capacité de recherche, comme par exemple : utiliser un opérateur de mutation capable d'apporter une diversification plus spécifique au problème traité ou utiliser une représentation directe. De même, comme perspective de recherche sur l'algorithme CTS, nous pensons considérer des composants plus "visuels" concernant l'information représentative du problème, par exemple une diversification échangeant des pièces proches d'un espace vide. Finalement, nous pourrions ajouter un critère d'aspiration.

Il est important de remarquer que nos algorithmes sont adaptables aux problèmes avec possibilité d'orientation.

Liste de publications personnelles

Conférences internationales avec comité de sélection

Giglia Gómez-Villouta, Jean-Philippe Hamiez and Jin-Kao Hao, "*Tabu Search with Consistent Neighbourhood for Strip Packing*", In C. Fyfe, N. Garcia-Pedrajas, F. Herrera, M. Ali (Eds.) : IEA/AIE 2010, Lecture Notes in Computer Science 6096 :1-10, 2010.

Giglia Gómez-Villouta, Jean-Philippe Hamiez and Jin-Kao Hao, "*A Dedicated Genetic Algorithm for Two-Dimensional Non-Guillotine Strip Packing*", In Proceedings of the 2007 Sixth Mexican International Conference on Artificial Intelligence (MICAI), Special Session (November 04 - 10, 2007). IEEE Computer Society, Washington, DC, 264-274. DOI= http ://dx.doi.org/10.1109/MICAI.2007.36

Revues internationales

Giglia Gómez-Villouta, Jean-Philippe Hamiez and Jin-Kao Hao,"*A Reinforced Tabu Search Approach for 2D Strip Packing*, IJAMC 2010 International Journal of Applied Metaheuristic Computing (à paraître).

Liste des figures

Listes des tables

Listes des algorithmes

Index

Références bibliographiques

[Aarts and Lenstra, 1997] cité page 22, 42, 42, 42
Emile Aarts and Jan Karel Lenstra. *Local Search in Combinatorial Optimization*. John Wiley & Sons, Inc., New York, NY, USA, 1997.

[Alvarez-Valdes *et al.*, 2005] cité page 13, 14, 14, 45
R. Alvarez-Valdes, F. Parreño, and J.M. Tamarit. A GRASP algorithm for constrained two-dimensional non-guillotine cutting problems. *Journal of the Operational Research Society ISSN 0160-5682 CODEN JORSDZ*, 56(4) :414.425, 2005.

[Alvarez-Valdes *et al.*, 2007] cité page 14, 27, 38, 38, 45, 45, 46, 48, 48, 49, 49, 53, 61
R. Alvarez-Valdes, F. Parreño, and J.M. Tamarit. A tabu search algorithm for a two-dimensional non-guillotine cutting problem. *European Journal of Operational Research*, 183-3 :1167–1182, 2007.

[Alvarez-Valdes *et al.*, 2008a] cité page 11
R. Alvarez-Valdes, F. Parreño, and J.M. Tamarit. A branch and bound algorithm for the strip packing problem. *OR Spectrum*, 31 :431–459, 2008.

[Alvarez-Valdes *et al.*, 2008b] cité page 2, 13, 38, 53, 61, 65
R. Alvarez-Valdes, F. Parreño, and J.M. Tamarit. Reactive GRASP for the strip-packing problem. *Comput. Oper. Res.*, 35(4) :1065–1083, 2008.

[Araya *et al.*, 2008] cité page 19
I. Araya, Bertrand Neveu, and María-Cristina Riff. An efficient hyperheuristic for strip-packing problems. In *Adaptive and Multilevel Metaheuristics*, pages 61–76. Springer, 2008.

[Asik and Özcan, 2009] cité page 53
Önder Baris Asik and Ender Özcan. Bidirectional best-fit heuristic for orthogonal rectangular strip packing. *Annals OR*, 172(1) :405–427, 2009.

[Baker *et al.*, 1980] cité page 1, 12, 32, 54
B.S. Baker, E.G. Coffman Jr., and R.L. Rivest. Orthogonal packing in two dimensions. *SIAM Journal on Computing*, 9(4) :846–855, 1980.

[Baker *et al.*, 1981] cité page 54
Brenda S. Baker, Donna J. Brown, and Howard P. Katseff. A 5/4 algorithm for two-dimensional packing. *J. Algorithms*, 2(4) :348–368, 1981.

[Beasley, 1985a] cité page 19, 66
J. E. Beasley. Algorithms for unconstrained two-dimensional guillotine cutting. *The Journal of the Operational Research Society*, 36(4) :297–306, 1985.

[Beasley, 1985b] cité page 11, 19, 66
J. E. Beasley. An exact two-dimensional non-guillotine cutting tree search procedure. *OPERATIONS RESEARCH*, 33(1) :49–64, January 1985.

[Belov *et al.*, 2009] cité page 1, 11
G. Belov, V. Kartak, H. Rohling, and G. Scheithauer. One-dimensional relaxations and lp bounds for orthogonal packing. *International Transactions in Operational Research*, 16(6) :745–766, 2009.

[Berkey and Wang, 1987] cité page 43, 45
J.O. Berkey and P.Y. Wang. Two-dimensional finite bin-packing algorithms. *J. Oper. Res. Soc.*, 38 :423–429, 1987.

[Bortfeldt, 2006] cité page 1, 9, 13, 17, 17, 38, 53, 61, 61
A. Bortfeldt. A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces. *European Journal of Operational Research*, 172(3) :814–837, 2006.

[Burke and Kendall, 1999] cité page 45
Edmund Burke and Graham Kendall. Comparison of meta-heuristic algorithms for clustering rectangles. *Computers and Industrial Engineering*, 37 :383–386, 1999.

[Burke *et al.*, 2003] cité page 19
Edmund Burke, Emma Hart, Graham Kendall, JimNewall, Peter Ross, and Sonia Schulenburg. Hyper-heuristics : An emerging direction in modern search technology, 2003.

[Burke *et al.*, 2004] cité page 1, 13, 19
E.K. Burke, G. Kendall, and G. Whitwell. A new placement heuristic for the orthogonal stock-cutting problem. *Operations Research*, 52(4) :655–671, 2004.

[Burke *et al.*, 2006] cité page 1, 9, 19
E.K. Burke, G. Kendall, and G. Whitwell. Metaheuristic enhancements of the best-fit heuristic for the orthogonal stock cutting problem. Technical Report NOTTCS-TR-SUB-0605091028-4370, University of Nottingham, UK, 2006.

[Burke *et al.*, 2009] cité page 9, 45, 53
Edmund K. Burke, Graham Kendall, and Glenn Whitwell. A simulated annealing enhancement of the best-fit heuristic for the orthogonal stock-cutting problem. *INFORMS J. on Computing*, 21(3) :505–516, 2009.

[Chazelle, 1983] cité page 1, 13, 54
B. Chazelle. The bottomn-left bin-packing heuristic : An efficient implementation. *IEEE Transactions on Computers*, 32(8) :697–707, 1983.

[Christofides and Hadjiconstantinou, 1995] cité page 19, 66
Nicos Christofides and Eleni Hadjiconstantinou. An exact algorithm for orthogonal 2-D cutting problems using guillotine cuts. *European Journal of Operational Research*, 83(1) :21–38, 1995.

[Christofides and Whitlock, 1977] cité page 19, 66
N Christofides and C Whitlock. An algorithm for two-dimensional cutting problems. *Operations Research*, 25 :30–44, 1977.

[Clautiaux *et al.*, 2008] cité page 1, 11
F. Clautiaux, Antoine Jouglet, Jacques Carlier, and Aziz Moukrim. A new constraint programming approach for the orthogonal packing problem. *Comput. Oper. Res.*, 35(3) :944–959, 2008.

[Coffman *et al.*, 1980] cité page 54
E. G. Jr. Coffman, M. R. Garey, D. S. Johnson, and R. E. Tarjan. Performance bounds for level-oriented two-dimensional packing algorithms. *SIAM Journal on Computing*, 9(4) :808–826, 1980.

[Davis, 1985] cité page 28, 45
L. Davis. Applying adaptive algorithms to epistatic domains. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 162–164, 1985.

[Dupont *et al.*, 2004] cité page 52, 56
A. Dupont, E. Alvernhe, and M. Vasquez. Efficient filtering and tabu search on a consistent neighbourhood for the frequency assignment problem with polarisation. *Annals OR*, 130(1-4) :179–198, 2004.

[Dupont *et al.*, 2005] cité page 52, 56
A. Dupont, M. Vasquez, and D. Habet. Consistent neighbourhood in a tabu search metaheuristics : Progress as real – problem solvers, chapter 17. In *Metaheuristics : Progress as real Problem Solvers. MIC-Kluwer*, volume 32, pages 367–386. Springer US, 2005.

[Dykhoff, 1990] cité page 5
H. Dykhoff. A typology of cutting and packing problems. *Evropean. Journal of Operational research.*, 44 :145–159, 1990.

[El Hayek *et al.*, 2008] cité page 53, 53
Joseph El Hayek, Aziz Moukrim, and Stéphane Negre. New resolution algorithm and pretreatments for the two-dimensional bin-packing problem. *Comput. Oper. Res.*, 35(10) :3184–3201, 2008.

[Fekete and Schepers, 1997a] cité page 1, 1, 9, 10, 11
S. P. Fekete and J. Schepers. A new exact algorithm for general orthogonal d-dimensional knapsack problems. In Springer Berlin / Heidelberg, editor, *Algorithms – ESA '97*, volume 1284 of *Lecture Notes Computer Science*, pages 144–156. Springer-Verlag, 1997.

[Fekete and Schepers, 1997b] cité page 19, 66
S.P. Fekete and J. Schepers. On more-dimensional packing III : Exact algorithms. Technical report zpr97-290, Mathematisches Institut, Universität zu Köln, 1997.

[Fowler *et al.*, 1981] cité page 6, 6
Robert J. Fowler, Mike Paterson, and Steven L. Tanimoto. Optimal packing and covering in the plane are NP-complete. *Inf. Process. Lett.*, 12(3) :133–137, 1981.

[Garey and Johnson, 1979] cité page 1, 5, 6, 6
M. R. Garey and D. S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[Glover and Laguna, 1997] cité page 51, 52
F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic, Dordrecht, 1997.

[Goldberg and Lingle, 1985] cité page 28
D.E. Goldberg and R. Lingle. Alleles, loci, and the TSP. In *Proceedings of First International Conference on Genetic Algorithms*, pages 154–159, 1985.

[Gomez and de la Fuente, 2000] cité page 1, 9, 9, 17, 24, 25, 26, 30
A. Gomez and D. de la Fuente. Resolution of strip-packing problems with genetic algorithms. *Journal of the Operational Research Society*, 51(11) :1289–1295, 2000.

[Gómez-Villouta *et al.*, 2007] cité page 2, 25, 30, 31, 52
Giglia Gómez-Villouta, Jean-Philippe Hamiez, and Jin-Kao Hao. A dedicated genetic algorithm for two-dimensional non-guillotine strip packing. In *MICAI '07 : Proceedings of the 2007 Sixth Mexican International Conference on Artificial Intelligence, Special Session*, pages 264–274, Washington, DC, USA, 2007. IEEE Computer Society.

[Gómez-Villouta *et al.*, 2010] cité page 3, 52
Giglia Gómez-Villouta, Jean-Philippe Hamiez, and Jin-Kao Hao. Tabu search with consistent neighbourhood for strip packing. *Lecture Notes in Artificial Intelligence, Springer-Verlag*, Vol. 6096 :1–10, 2010.

[Hamiez *et al.*, 2009] cité page 1, 9, 17, 46, 47, 48, 49, 49, 53, 55, 61, 61
Jean-Philippe Hamiez, Julien Robet, and Jin-Kao Hao. A tabu search algorithm with direct representation for strip packing. In Carlos Cotta and Peter I. Cowling, editors, *EvoCOP*, volume 5482 of *Lecture Notes in Computer Science*, pages 61–72. Springer, 2009.

[Harwig *et al.*, 2006] cité page 45, 45, 47, 48, 49, 49
J. M. Harwig, J. W. Barnes, and J. T. Moore. An adaptive tabu search approach for 2-dimensional orthogonal packing problems. *Military Operations Research*, 11-2 :5–26, 2006.

[Holland, 1975] cité page 23, 27
J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[Hopper and Turton, 1997] cité page 30
E. Hopper and B. Turton. Application of genetic algorithms to packing problems - A review. In *Proceedings of the Second On-line World Conference of Soft Computing in Engineering Design and Manufacturing*, pages 279–288. Springer Verlag, 1997.

[Hopper and Turton, 1999] cité page 24
E. Hopper and B. Turton. A genetic algorithm for a 2D industrial packing problem. *Computers & Industrial Engineering*, 37(1-2) :375–378, 1999.

[Hopper and Turton, 2001a] cité page 1, 2, 9, 9, 13, 14, 17, 19, 34, 36, 45, 45, 45, 54, 54, 58
E. Hopper and B. Turton. An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem. *European Journal of Operational Research*, 128(1) :34–57, 2001.

[Hopper and Turton, 2001b] cité page 30
E. Hopper and B.C.H. Turton. A review of the application of meta-heuristic algorithms to 2D strip packing problems. *Artif. Intell. Rev.*, 16(4) :257–300, 2001.

[Hopper, 2000] cité page 38, 38
E. Hopper. *Two-dimensional packing utilising evolutionary algorithms and other meta-heuristic methods*. PhD thesis, University of Cardiff (School of Engineering), UK, 2000.

[Ibaraki *et al.*, 2007] cité page 9
Toshihide Ibaraki, Shinji Imahori, Mutsunori Yagiura, Toshihide Ibaraki, Shinji Imahori, and Mutsunori Yagiura. Hybrid metaheuristics for packing problems. Springer Berlin / Heidelberg, 2007.

[Imahori *et al.*, 2006] cité page 1, 9, 54
S. Imahori, M. Yagiura, and H. Nagamochi. Practical algorithms for two-dimensional packing. Technical Report METR 2006-19, University of Tokyo, Graduate School of Information Science and Technology, Department of Mathematical Informatics, Japan, 2006.

[Iori *et al.*, 2003] cité page 2, 19, 24, 45, 45, 45, 47, 52, 61, 61
M. Iori, S. Martello, and M. Monaci. Metaheuristic algorithms for the strip packing problem. *Optimization and Industry : New Frontiers*, pages 159–179, 2003.

[Jakobs, 1993] cité page 1, 9, 17, 25, 30, 30
S. Jakobs. On genetic algorithms for the packing of polygons. *European Journal of Operational Research*, 88(1) :165–181, 1993.

[Kenmochi *et al.*, 2009] cité page 11, 61
M. Kenmochi, T. Imamichi, K. Nonobe, M. Yagiura, and H. Nagamochi. Exact algorithms for the two-dimensional strip packing problem with and without rotations. *European Journal of Operational Research*, 198 :73–83, 2009.

[Lesh *et al.*, 2004] cité page 1, 1, 9, 10, 11, 11, 13
N. Lesh, J. Marks, A. McMahon, and M. Mitzenmacher. Exhaustive approaches to 2D rectangular perfect packings. *Inf. Process. Lett.*, 90(1) :7–14, 2004.

[Lesh *et al.*, 2005] cité page 13
N. Lesh, J. Marks, A. McMahon, and M. Mitzenmacher. New heuristic and interactive approaches to 2D rectangular strip packing. *ACM Journal of Experimental Algorithmics*, 10 :1–18, 2005.

[Leung *et al.*, 2001] cité page 38
T.W. Leung, C.H. Yung, and M.D. Troutt. Applications of genetic search and simulated annealing to the two-dimensional non-guillotine cutting stock problem. *Computers & Industrial Engineering*, 40(3) :201–214, 2001.

[Liang *et al.*, 2001] cité page 30
Ko-Hsin Liang, Xin Yao, and Charles Newton. A new evolutionary approach to cutting stock problems with and without contiguity. *Computers and Operations Research*, 29 :1641–1659, 2001.

[Liu and Teng, 1999] cité page 17, 24
Dequan Liu and Hongfei Teng. An improved BL-algorithm for genetic algorithm of the orthogonal packing of rectangles. *European Journal of Operational Research*, 112(2) :413–420, 1999.

[Lodi *et al.*, 1999] cité page 43, 45, 45, 45, 47
Andrea Lodi, Silvano Martello, and Daniele Vigo. Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems. *INFORMS J. on Computing*, 11(4) :345–357, 1999.

[Martello *et al.*, 2003] cité page 1, 11
S. Martello, M. Monaci, and D. Vigo. An exact approach to the strip packing problem. *INFORMS Journal on Computing*, 15(3) :310–319, 2003.

[Miller and Goldberg, 1995] cité page 27
B.L. Miller and D.E. Goldberg. Genetic algorithms, tournament selection, and the effects of noise. Technical Report 95006, Illinois Genetic Algorithms Laboratory, 1995.

[Mumford-Valenzuela *et al.*, 2003] cité page 1, 13, 17
Mumford-Valenzuela, Christine L., Vick, Janis, Wang, and Pearl Y. Heuristics for large strip packing problems with guillotine patterns : An empirical study. *Metaheuristics : computer decision-making(book contents)*, pages 501–522, 2003.

[Neveu and Trombettoni, 2008] cité page 13
B. Neveu and G. Trombettoni. Strip packing based on local search and a randomized best-fit. In *Fifth International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems : FirstWorkshop on Bin Packing and Placement Constraints (CPAIOR : BPPC)*, 2008.

[Neveu *et al.*, 2004] cité page 14
Bertrand Neveu, Gilles Trombettoni, and Fred Glover. Id walk : A candidate list strategy with a simple diversification device. In Mark Wallace, editor, *CP*, volume 3258 of *Lecture Notes in Computer Science*, pages 423–437. Springer, 2004.

[Neveu *et al.*, 2007] cité page 9, 55
Bertrand Neveu, Gilles Trombettoni, and Ignacio Araya. Incremental move for 2d strip-packing. In *ICTAI (2)*, pages 489–496. IEEE Computer Society, 2007.

[Neveu *et al.*, 2008] cité page 2, 9, 19, 53, 55, 56, 61, 61, 65
Bertrand Neveu, Gilles Trombettoni, Ignacio Araya, and María-Cristina Riff. A strip packing solving method using an incremental move based on maximal holes. *International Journal on Artificial Intelligence Tools*, 17(5) :881–901, 2008.

[Oliver *et al.*, 1987] cité page 29
I. M. Oliver, D. J. Smith, and J. R. C. Holland. A study of permutation crossover operators on the TSP. In *Proceedings of the 2nd Int.Conf.on Genetic Algorithms and their applications, Lawrence Erlbaum Associates*, USA, 1987.

[Rodríguez-Tello, 2007] cité page 27
E.A. Rodríguez-Tello. *Nouvelles fonctions d'évaluation pour les problèmes d'étiquetagede graphes BMP et MinLa*. PhD thesis, Université d'Angers, 2007.

[Schiermeyer, 1994] cité page 54
Ingo Schiermeyer. Reverse-fit : A 2-optimal algorithm for packing rectangles. In Jan van Leeuwen, editor, *ESA*, volume 855 of *Lecture Notes in Computer Science*, pages 290–299. Springer, 1994.

[Soh *et al.*, 2008] cité page 1, 11, 53, 61
T. Soh, K. Inoue, N. Tamura, M. Banbara, and H. Nabeshima. A sat-based method for solving the two-dimensional strip packing problem. In & T. Mancini In M. Gavanelli, editor, *Proc. 15th Int. RCRA Workshop on Exp. Eval. of Algorithms for Solving Prob. with Comb. Explos. Germany : RWTH Aachen Univ.*, 2008.

[Soke and Bingul, 2006] cité page 1, 2, 2, 9, 9, 19, 24, 26, 30, 38, 45, 52
A. Soke and Z. Bingul. Hybrid genetic algorithm and simulated annealing for two-dimensional

non-guillotine rectangular packing problems. *Engineering Applications of Artificial Intelligence*, 19(5) :557–567, 2006.

[Vasquez and Hao, 2001] cité page 27
Michel Vasquez and Jin-Kao Hao. A heuristic approach for antenna positioning in cellular networks. *Journal of Heuristics*, 7(5) :443–472, 2001.

[Wäscher *et al.*, 2007] cité page 5
Gerhard Wäscher, Heike HauSSner, and Holger Schumann. An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183(3) :1109–1130, 2007.

[Yeung and Tang, 2004] cité page 1, 9, 9, 17, 24, 25, 26, 30, 30, 52
L.H.W. Yeung and W.K.S. Tang. Strip-packing using hybrid genetic approach. *Engineering Applications of Artificial Intelligence*, 17(2) :169–177, 2004.

[Zhang *et al.*, 2006] cité page 19
Defu Zhang, Yan Kang, and Ansheng Deng. A new heuristic recursive algorithm for the strip rectangular packing problem. *Comput. Oper. Res.*, 33(8) :2209–2217, 2006.

MÉTHODES HEURISTIQUES POUR LE PROBLÈME DE PLACEMENT SUR BANDE EN DEUX DIMENSIONS

Résumé

Les problèmes de placement sont généralement NP-difficiles, ou NP-complets suivant l'objectif à atteindre. Il s'agit ici de positionner un ensemble d'objets dans un ou plusieurs "container(s)", de dimensions données ou de hauteur infinie, en respectant des contraintes liées à certaines caractéristiques (poids, quantité, rotation, équilibre, découpe guillotine...). Ces problèmes ont de nombreuses applications pratiques. Les stratégies de résolution les plus efficaces sont généralement les méthodes approchées, en particulier la recherche locale. Dans cette thèse, nous nous intéressons à un problème de placement particulier en deux dimensions (sans rotation possible des objets (rectangulaires) ni prise en compte de la contrainte guillotine) connu sous le nom de "strip packing" (SPP). L'objectif de ce problème est de minimiser la hauteur atteinte après placement (sans chevauchement) des objets. Nous avons développé deux approches "méta-heuristiques" incluant des composants novateurs reposant sur une connaissance approfondie du problème. La première est un algorithme génétique avec un nouveau croisement (très "visuel") et une fonction d'évaluation hiérarchique. La seconde est une recherche tabou avec représentation "directe" (i.e. n'utilisant pas les habituelles permutations) dont les caractéristiques principales sont un voisinage consistant, une diversification reposant sur l'historique de la recherche et une fonction d'évaluation qui mesure la qualité de solutions éventuellement partielles. Les deux approches proposées, évaluées sur un jeux de test bien connu et très difficile, se sont révélées performantes comparées à d'autres stratégies.

Mots-clés : problème de placement sur bande, algorithme génétique, recherche tabou, heuristiques.

HEURISTIC METHODS FOR THE TWO DIMENSIONAL STRIP PACKING PROBLEM

Abstract

Packing problems are usually NP-hard, or NP-complete according to the objective. One has to locate a set of objects into one or more "container(s)", with fix dimensions or of infinite height, while respecting constraints related to some characteristics (weight, quantity, rotation, stability, guillotine cuts...). The main interest of these problems are the numerous practical applications from various domains. The most effective solution strategies for these problems are usually approximate methods, local search in particular. In this thesis, we are interested in a particular two-dimensional packing problem (without rotation nor guillotine cuts) known as "strip packing" (SPP). The objective of this problem, after locating rectangular objects without overlap, is to minimize the height of the resulting packing. We developed two "meta-heuristic" approaches for the SPP, both including innovative components based on problem knowledge. The first one is a genetic algorithm with a new (highly "visual") crossover and a hierarchical fitness function. The second one is a tabu search with "direct" representation (i.e. not using the classical permutations) whose main characteristics are a consistent neighborhood, a "well-informed" diversification (based on the search history), and a fitness function able to evaluate possibly partial solutions. The two proposed approaches, assessed on a well-known and very difficult benchmark, show good performances compared with other strategies. .

Keywords : strip packing problem, genetic algorithm, tabu search, heuristics.