Un codage SAT pour les problèmes de placement

Stéphane Grandcolas, Cedric Pinto

Un codage SAT pour les problèmes de placement

Stéphane Grandcolas **Cédric Pinto**

Université Paul Cézanne (Aix-Marseille 3)
Avenue Escadrille Normandie-Niemen
13397 Marseille Cedex 20 (France)
{stephane.grandcolas, cedric.pinto}@lsis.org

Résumé

Le problème de placement orthogonal (OPP) consiste à déterminer si un ensemble d'objets peut etre placé dans un conteneur de taille connue. Ce problème est NP-complet [6]. Une modélisation de ce problème à base de graphes d'intervalles a été proposée par S. P. Fekete et al [3][5]. Cette modélisation permet de représenter des classes de placements équivalents, diminuant d'autant l'espace de recherche.

Dans cet article nous proposons de représenter par des formules de la logique propositionnelle la modélisation de S. P. Fekete et al. Nous avons implémenté cette approche en utilisant le solveur MiniSat, et nous l'avons comparée d'une part avec les résultats de S. P. Fekete et al. sur des problèmes classiques, et d'autre part avec l'approche de T. Soh et al. [12] basée aussi sur un codage SAT sur des problèmes de Strip Packing.

1 Introduction

Le problème de placement orthogonal en plusieurs dimensions (OPP) consiste à déterminer si un ensemble d'objets de tailles connues peut être placé dans un conteneur donné. Les objets doivent être placés parallèlement aux bords du conteneur et aucune rotation ne peut être effectuée. Bien que ce problème soit NP-complet [6], des algorithmes efficaces pour le résoudre sont essentiels du fait que OPP est souvent utilisé pour le traitement de problèmes d'optimisation comme les problèmes de *strip packing* ou de *bin-packing*.

Les graphes d'intervalles sont utilisés dans de nombreux domaines comme la génétique ou l'ordonnancement. Ils constituent une classe de graphes très particulière et possèdent des propriétés algorithmiques intéressantes [7]. S. P. Fekete et al. ont introduit une nouvelle caractérisation de OPP [3] basée sur les graphes d'intervalles. Pour chaque dimension de l'espace un graphe représente les intersections entre les objets dans cette dimension. Pour chaque graphe G_i correspondant à la dimension i, le poids d'un sommet est égal à la taille de l'objet correspondant dans la dimension i. Le problème de placement orthogonal en d dimensions est équivalent à trouver d graphes $G_1, \ldots, G_d$ tels que **(P1)** chaque graphe G_i est un graphe d'intervalles, **(P2)** dans chaque graphe G_i, tout stable est i-faisable, c'est-à-dire que la somme des poids de ses sommets est inférieure ou égale à la taille du conteneur dans la dimension i, et **(P3)** il n'y a pas d'arête qui apparaît dans chacun des d graphes. S. P. Fekete et al. proposent une procédure de recherche [5] complète qui consiste à énumerer tous les graphes d'intervalles possibles, en choisissant pour chaque arête de chaque graphe si elle appartient au graphe ou pas. Durant la recherche, la condition (P3) est toujours satisfaite en interdisant de fixer une arête apparaissant déjà dans d-1 graphes. Chaque fois qu'un graphe G_i est un graphe d'intervalles, la i-faisabilité de ses stables est vérifiée, en calculant son stable de poids maximum. Dès que les graphes $G_1, \ldots, G_d$ satisfont les trois conditions, la recherche s'arrête, les d graphes représentant alors une classe de solutions équivalentes pour le problème de placement. La figure 2 montre un exemple en deux dimensions de deux placements, parmi tant d'autres, correspondant à la même paire de graphes d'intervalles.

Il existe peu d'approches SAT pour les problèmes de placement. En 2008, T. Soh et al.[12] ont proposé un codage SAT pour le problème du *strip packing* en deux dimensions (SPP) [12]. Ce problème consiste à trouver la hauteur minimale d'une bande de largeur fixée contenant tous les objets. Pour ce calcul, ils effectuent des recherches successives avec différentes hauteurs (choisies par recherche dichotomique), en codant chaque fois le problème de décision sous la forme d'une formule CNF qui est résolue avec un solveur SAT externe (Minisat). Dans leur formula-

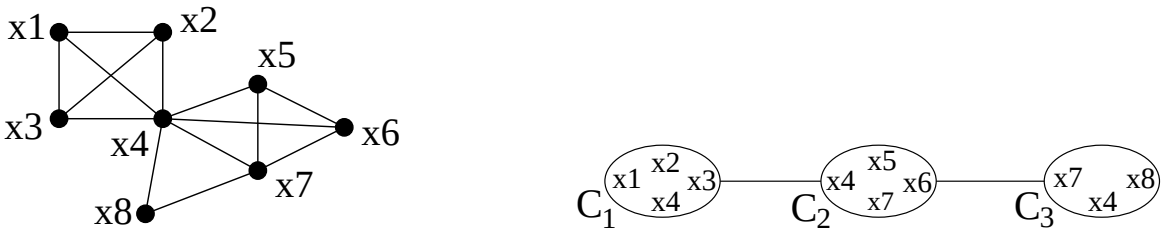

FIGURE 1 – (a) Un graphe d'intervalles, (b) une suite ordonnée de cliques de ce graphe

tion, les positions exactes des objets dans le conteneur sont représentées par des variables booléennes, ainsi que les positions relatives des objets entre eux (à gauche, à droite, au dessus, au dessous). M. D. Moffit et al. [11] se sont intéressés à une représentation de type CSP très proche de celle de T. Soh et al., étendant les travaux de R. Korf [8] avec l'ajout de variables pour décrire les positions relatives des objets.

Dans cet article nous proposons une représentation de la caractérisation de S. P. Fekete et al. avec des formules de la logique propositionnelle. L'idée consiste à représenter les arêtes des graphes d'intervalles par des variables booléennes et à coder sous forme de clauses les stables infaisables. L'article est organisé de la façon suivante. Dans la partie 2 nous présentons la caractérisation de S. P. Fekete et al. après avoir rappelé quelques propriétés sur les graphes d'intervalles. Dans la partie 3 nous revenons sur les codages SAT et CSP connus. Dans la partie 4 nous définissons le codage SAT du problème OPP que nous proposons. Enfin dans la partie 5 nous exposons les résultats expérimentaux, et nous terminons avec la conclusion.

2 Une caractérisation de OPP avec des graphes d'intervalles

Les graphes d'intervalles constituent une classe très particulière de graphes possédant de nombreuses propriétés algorithmiques intéressantes : de nombreux problèmes NP-difficiles peuvent être résolus en temps polynomial dans le cas des graphes d'intervalles [7]. Ces graphes sont utilisés dans de nombreux domaines comme la génétique ou l'ordonnancement. De plus, ils peuvent être reconnus par des algorithmes de complexité linéaire [9] en le nombre de sommets et d'arêtes du graphe.

Définition 1 *Un graphe $G = (X, E)$ est un graphe d'intervalles si et seulement si à tout sommet $x \in X$ on peut associer un intervalle I_x de l'ensemble des réels tel que :*

$$\forall\ x,\ y, \in X,\ \{x, y\} \in E \Leftrightarrow I_x \cap I_y \neq \emptyset$$

Dans un graphe G, une clique est un ensemble de sommets tel que les sommets de cette clique sont tous reliés entre eux par une arête dans G. Inversement, un stable est un ensemble de sommets dans lequel les sommets ne sont reliés à aucun autre sommet du même stable. Calculer l'ensembles des cliques maximales est un problème NP-difficile dans le cas de graphes quelconques. Dans le cas des graphes d'intervalles, ce problème peut être résolu en temps polynomial [1].

Les graphes d'intervalles ont la propriété d'être des graphes triangulés [7]. Or, l'ensemble des cliques maximales d'un graphe triangulé peut se calculer en temps linéaire en le nombre de sommets et d'arêtes du graphe [7]. De plus, le nombre de cliques maximales d'un graphe triangulé ne peut dépasser le nombre de sommets n alors qu'il peut être exponentiel en n dans le cas des graphes quelconques. Puisque les graphes d'intervalles sont triangulés, le calcul des cliques maximales peut être fait en temps polynomial et le nombre de cliques maximales est majoré par le nombre de sommets. Nous avons également la propriété suivante.

Propriété 1 *G est un graphe d'intervalles si et seulement si il existe une suite de cliques maximales de G, notées $\mathcal{C} = (C_1, \ldots, C_k)$, telle que :*

$$\forall a,\ b,\ c, \qquad 1 \leq a \leq b \leq c \leq k, \qquad C_a \cap C_c \subseteq C_b$$

Autrement dit, un graphe G est un graphe d'intervalles si et seulement si les cliques maximales de G peuvent être ordonnées de sorte que tout sommet appartient à des cliques consécutives dans cet ordre. Nous appellerons *suite ordonnée de cliques* toute suite de cliques maximales vérifiant cette propriété. Nous montrons en figure 1 un graphe d'intervalles et une suite ordonnée de cliques de ce graphe. Pour un graphe d'intervalles donné, il peut y avoir plusieurs suites ordonnées de cliques tandis qu'une suite ordonnée de cliques ne correspond qu'à un seul graphe d'intervalles.

S. P. Fekete et al. ont proposé une caractérisation des placements pour le problème de packing othogonal à l'aide de graphes d'intervalles, un pour chaque dimension de l'espace [3]. Dans ce qui suit, O désigne un ensemble d'objets

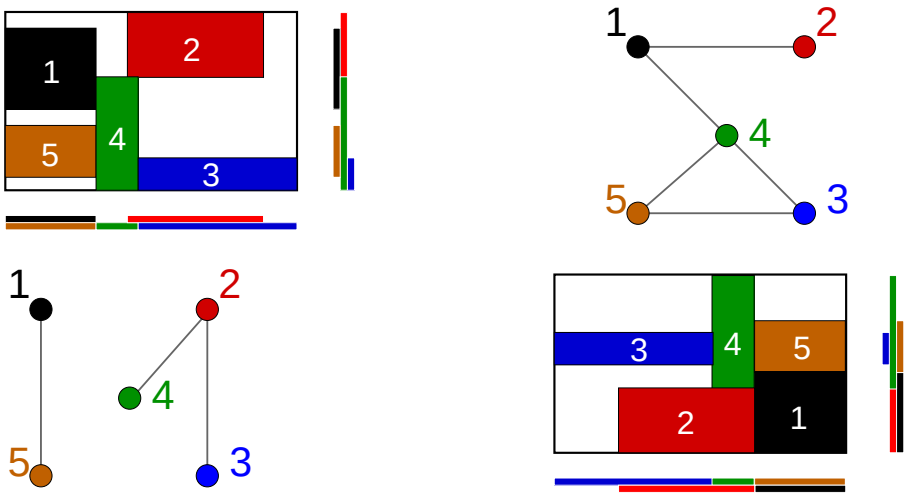

FIGURE 2 – Deux graphes d'intervalles et deux exemples de placements équivalents représentés par ces graphes.

dont les tailles sont connues. Si on note G_i le graphe associé à la dimension i, ses sommets correspondent aux objets de O et ses arêtes représentent les intersections entre les projections des objets sur la dimension i (voir figure 2). Le *poids* d'un sommet x dans un graphe G_i est égal à la taille de x dans la dimension i. Par exemple, considérons le cas d'un problème en deux dimensions. Dire que deux objets x et y s'intersectent dans la dimension horizontale signifie que x doit être au-dessus de y ou inversement. Remarquons que x et y ne peuvent s'intersecter dans toutes les dimensions à la fois auquel cas x et y se chevaucheraient. Afin de garantir la validité des placements représentés par les graphes d'intervalles, les auteurs ont ajouté deux propriétés, la première étant qu'*une arête ne peut être présente dans tous les graphes d'intervalles*. Cette propriété garantit que les objets ne se chevaucheront pas dans l'espace. La deuxième propriété devant être satisfaite par chaque graphe d'intervalles permet d'assurer que les objets seront placés à l'intérieur du conteneur. Pour cela, *dans chaque graphe d'intervalles G_i, chaque stable doit être i-faisable*. Un stable S est dit i-faisable si son poids (défini par la somme de poids des sommets qui composent S) est inférieur ou égal à la taille du conteneur dans la dimension i. Dans le cas avec deux dimensions, si un ensemble d'objets forment un stable dans le graphe associé à la dimension horizontale alors ces objets seront placés côte à côte horizontalement. Il faut donc s'assurer que la largeur nécessaire pour les placer côte à côte est inférieure ou égale à la largeur du conteneur, d'où la propriété ci-dessus.

L'intérêt de cette approche réside dans la capture d'une multitude de placements avec une seule famille de graphes d'intervalles. De plus, à partir d'une famille de graphes d'intervalles, nous pouvons facilement exhiber l'ensemble des placements représentés par cette famille. Il suffit de considérer le graphe complémentaire de chaque graphe d'intervalles. Ces graphes sont des graphes de comparabilité [7]. Dans chaque graphe de comparabilité, une orientation transitive des arêtes peut être calculée en temps linéaire en le nombre d'arêtes. Une telle orientation existe par définition des graphes de comparabilité. Chaque orientation transitive induit un ordre sur les objets. Cet ordre peut être utilisé pour en déduire un placement des objets. Si $n_1, \ldots, n_d$ désignent les nombres d'orientations transitives des graphes complémentaires $\overline{G_1}, \ldots, \overline{G_d}$ alors le nombre de placements représentés par $G_1, \ldots, G_d$ est égal à $\prod_{i=1}^{d} n_i$.

S. P. Fekete et al. ont donc démontré que résoudre OPP en d dimensions est équivalent à déterminer l'existence de d graphes tels que :

P1 : Chaque graphe est un graphe d'intervalles

P2 : Pour chaque graphe G_i, chaque stable est i-faisable

P3 : Chaque arête doit apparaître dans au plus d-1 graphes.

Dans [5], S. P. Fekete et al. définissent une méthode de résolution de OPP. Cette méthode consiste à énumérer toutes les familles de graphes d'intervalles. Pour faire cela, ils choisissent pour chaque graphe G_i et chaque arête de G_i si celle-ci est présente ou pas dans G_i. Initialement, tous les graphes sont vides, c'est-à-dire sans aucune arête. La propriété (P3) est vérifiée par construction, puisque si une arête est présente dans d-1 graphes alors celle-ci est immédiatement retirée dans le dernier graphe. Chaque fois qu'une arête est ajoutée ou retirée dans un graphe G_i, la méthode consiste à vérifier si G_i est un graphe d'intervalles. Si ce n'est pas le cas alors une autre arête est choisie (éventuellement dans un autre graphe). Si G_i est un graphe d'intervalles, un stable de poids maximum dans G_i est calculé et sa i-faisabilité est vérifiée. Si le stable calculé n'est pas i-faisable alors les arêtes reliant deux sommets

de ce stable seront choisies en priorité par la suite. Si tous les graphes sont des graphes d'intervalles et que chaque stable de poids maximum est faisable alors d graphes d'intervalles vérifiant (P1), (P2) et (P3) ont été trouvés. Dans ce cas, le problème est consistant et un placement peut facilement être exhibé en calculant une orientation transitive dans chaque graphe complémentaire. Enfin, notons que S. P. Fekete et al. ont également montré l'intérêt de leur approche en pratique [5].

3 Approches SAT et CSP pour OPP

Peu d'approches SAT existent pour les problèmes de placement. En 2008, T. Soh et al. [12] ont proposé un codage SAT pour le problème du *strip packing* en deux dimensions (SPP) [12]. Ce problème consiste à trouver la hauteur minimale d'une bande de largeur fixée contenant tous les objets. Pour ce faire, ils effectuent des recherches successives avec différentes hauteurs (choisies par recherche dichotomique). Pour chaque recherche, le problème de décision est codé sous la forme d'une formule CNF qui est ensuite résolue avec un solveur SAT externe (Minisat). Dans leur formulation, les variables représentent les positions exactes des objets dans le conteneur, c'est à dire que dans chaque dimension, pour chaque position possible une variable booléenne signale si l'objet apparaît à cette position ou non. Des variables additionnelles représentent les positions relatives des objets entre eux (à gauche, à droite, au dessus, au dessous). T. Soh et al. introduisent également des contraintes additionnelles pour éviter de reconsidérer des placements équivalents par symetrie. Enfin, durant la résolution, Minisat génère de nouvelles clauses à partir des conflits rencontrés. T. Soh et al. ont mis en place un mécanisme de mémorisation qui permet de réutiliser ces clauses dans les recherches suivantes (cette mémorisation est possible du fait que les instances SAT successives sont incrémentales). La modélisation de T. Soh et al. utilise $\mathcal{O}(W \times H \times n + n^2)$ variables booléennes pour un problème avec n objets et un conteneur de largeur W et de hauteur H.

Plusieurs approches de type CSP ont été proposées pour les problèmes de remplissage avec des rectangles. Par exemple, en 2004, R. Korf [8] définit un CSP dans lequel les variables représentent les positions des objets. Les contraintes assurant que les objets ne se chevauchent pas sont des disjonctions d'inégalités. Les tailles des domaines sont donc très proches de la taille du conteneur et les performances du solveur en sont directement affectées. En 2006, M. D. Moffit et al. [11] proposent d'améliorer cette approche en introduisant des variables additionelles représentant les positions relatives des objets entres eux. Remarquons que cette formalisation est proche de celle de T. Soh et al. Les relations d'ordre entre les objets sont représentées dans des graphes orientés, et le calcul des plus courts chemins dans ces graphes permet de détecter l'inconsistance du CSP.

4 Un nouveau codage SAT pour OPP

Nous proposons un nouveau codage SAT basé sur la caractérisation de S. P. Fekete et al. pour le problème de placement othogonal en d dimensions. L'idée consiste à représenter les graphes d'intervalles par des suites ordonnées de cliques, en utilisant des variables booléennes pour signaler la présence des objets dans les cliques. Les variables que nous utilisons pour le codage sont les suivantes :

$e^i_{x,y}$: **vrai** si l'arête $\{x, y\}$ est dans G_i,

$c^i_{x,a}$: **vrai** si le sommet x est dans la clique a,

$p^i_{x,y,a}$: **vrai** si les sommets x et y sont dans la clique a,

u^i_a : **vrai** si la clique a n'est pas vide,

Dans ce qui suit, O désigne l'ensemble des n sommets correspondants aux n objets du problème et d est le nombre de dimensions du problème. Nous allons maintenant établir l'ensemble des formules permettant de représenter les propriétés (P1), (P2) et (P3) définies par S. P. Fekete et al. Tout d'abord, chaque objet doit apparaitre dans au moins une clique dans chaque dimension.

1. **[Les objets sont placés]** $x \in O, 1 \leq i \leq d,$
$$c^i_{x,1} \vee \ldots \vee c^i_{x,n}$$

Ensuite, la suite des cliques doit être ordonnée comme indiqué dans la propriété 1.

2. **[Suite ordonnée de cliques]**
$x \in O, 1 \leq i \leq d, 1 \leq a < b - 1 < n,$
$$(c^i_{x,a} \wedge c^i_{x,b}) \Rightarrow c^i_{x,a+1}$$

Ainsi, si un objet x apparait dans deux cliques C_a et C_b alors x doit apparaitre dans les cliques $C_{a+1}, \ldots, C_{b-1}$. En fait, il suffit de forcer le placement de l'objet x dans la clique C_{a+1}, la contrainte se propage ensuite sur les cliques suivantes par récurrence.

Il faut également que les objets ne se chevauchent pas dans l'espace.

3. **[Pas de chevauchement]** $x, y \in O,$
$$\neg e^1_{x,y} \vee \ldots \vee \neg e^d_{x,y}$$

Il suffit d'imposer que chaque arête n'apparaisse pas dans tous les graphes.

Dans la formule qui suit, nous appellons stable minimal infaisable d'un graphe G_i tout stable pour lequel le retrait d'un de ses sommets le rend i-faisable. S^i désigne l'ensemble des stables minimaux infaisables. Pour satisfaire la propriété (P2), il suffit de forcer l'ajout d'au moins une arête entre deux sommets d'un stable minimal infaisable dans chaque dimension.

4. **[Stable faisabilité]** $1 \leq i \leq d, N \in S^i$,
$$\bigvee_{x,\, y \in N} e^i_{x,y}$$

Pour illustrer la formule ci-dessus, considérons l'exemple dans lequel trois objets x, y et z en deux dimensions ne peuvent être placés côte à côte dans la dimension i mais toute paire parmi ces trois objets peut être placée dans le conteneur. Nous avons donc que $\{x, y, z\}$ est un stable minimal infaisable. D'après la formule (4), nous aurons la clause $e^i_{x,y} \vee e^i_{x,z} \vee e^i_{y,z}$. Ainsi, nous obligeons l'ajout d'une de ces trois arêtes afin d'obtenir un stable i-faisable. Donc, la propriété (P2) sera bien satisfaite puisque chaque stable de chaque graphe sera faisable. Notons le cas particulier où deux objets x et y ne peuvent être placés côte à côte dans la dimension i. Dans ce cas, nous aurons la clause unitaire $e^i_{x,y}$. Le solveur SAT affectera immédiatement ce littéral et propagera cette affectation.

On considère autant de cliques qu'il y a d'objets. Lors de l'affectation des objets aux cliques il se peut que des cliques soient vides. Afin d'éviter des traitements redondants, les cliques vides sont forcées à apparaître à la fin de la suite.

5. **[Pas de cliques vides]** $1 \leq i \leq d, 1 \leq a \leq n$,
$\neg u^i_a \Rightarrow \neg u^i_{a+1}$

Ce qui signifie que si la clique a est vide alors la clique $a+1$ l'est aussi. Signalons que cette contrainte n'est pas nécessaire.

Enfin il faut établir les liens entre les différentes variables.

6. **[Liens entre les variables]**
- $x, y \in O, 1 \leq a \leq n, 1 \leq i \leq d$,
$$p^i_{x,y,a} \Leftrightarrow (c^i_{x,a} \wedge c^i_{y,a})$$
$$(p^i_{x,y,1} \vee \ldots \vee p^i_{x,y,n}) \Leftrightarrow e^i_{x,y}$$
- $1 \leq a \leq n, 1 \leq i \leq d$,
$$u^i_a \Leftrightarrow (c^i_{1,a} \vee \ldots \vee c^i_{n,a})$$

Les contraintes qui suivent ne sont pas nécessaires pour la validité des solutions, mais en contraignant le problème elles peuvent provoquer des échecs plus rapidement.

La formule suivante aide la propagation des affectations des objets aux cliques.

7. **[Suite ordonnée de cliques (bis)]**
$x \in O, 1 \leq a \leq n, 1 \leq i \leq d$,
$$(c^i_{x,a} \wedge \neg c^i_{x,a+1}) \Rightarrow (\neg c^i_{x,a+2} \wedge \ldots \wedge \neg c^i_{x,n})$$
$$(c^i_{x,a} \wedge \neg c^i_{x,a-1}) \Rightarrow (\neg c^i_{x,a-2} \wedge \ldots \wedge \neg c^i_{x,1})$$

Si un objet x est dans la clique C_a et qu'il n'est pas dans la clique C_{a+1} (resp. C_{a-1}) alors on en déduit qu'il ne sera pas dans les cliques placées après C_{a+1} (resp. avant C_{a-1}) sinon la propriété (P1) ne serait plus satisfaite.

Pendant l'énumération il se peut que des cliques successives soient incluses l'une dans l'autre. Cette situation est sans intérêt puisqu'elle peut être reproduite sans qu'aucune clique ne soit incluse dans une autre.

8. **[Cliques maximales]** $1 \leq a < n, 1 \leq i \leq d$,
$$(u_a \wedge u^i_{a+1}) \Rightarrow ((c^i_{1,a} \wedge \neg c^i_{1,a+1}) \vee \ldots \vee (c^i_{n,a} \wedge \neg c^i_{n,a+1}))$$
$$(u_a \wedge u^i_{a+1}) \Rightarrow ((\neg c^i_{1,a} \wedge c^i_{1,a+1}) \vee \ldots \vee (\neg c^i_{n,a} \wedge c^i_{n,a+1}))$$

Si deux cliques C_a et C_{a+1} ne sont pas vides alors au moins un sommet doit être présent dans C_a et pas dans C_{a+1} et inversement, au moins un sommet doit être présent dans C_{a+1} et pas dans C_a.

La présence d'objets identiques (dans toutes leurs dimensions) dans un problème génère de nombreux traitements similaires, puisque ces objets sont interchangeables. Une façon d'éviter ces traitements redondants consiste à contraindre l'ordre d'apparition de ces objets dans les cliques dans une des dimensions. Supposons que δ soit la dimension choisie pour cet ordre. Soit $\prec$ une relation d'ordre total sur l'ensemble des objets.

9. **[Objets identiques]**
$x, y \in O, x \equiv y$ and $x \prec y, 1 \leq a < n, a < b \leq n$,
$$(c^\delta_{y,a} \wedge c^\delta_{x,b}) \Rightarrow c^\delta_{x,a}$$

Si l'objet x tel que $x \prec y$ apparaît dans une clique de rang b supérieur au rang a d'une clique contenant y, alors l'objet x doit lui aussi apparaître dans la clique de rang b.

La modélisation d'un problème avec n objets en d dimensions nécessite donc $\mathcal{O}(d \times n^3)$ variables et $\mathcal{O}(d \times (n^4 + 2^n))$ clauses. Cependant, puisque seulement les stables minimaux infaisables sont générés dans l'instance, dans la pratique, nous espérons avoir moins de 2^n clauses issues de la formule (4).

5 Résultats expérimentaux

5.1 Problème de placement orthogonal (OPP)

Le problème consiste à déterminer si un ensemble d'objets peut être placé dans un conteneur donné. Nous avons comparé notre approche avec celle de S. P. Fekete et al. sur quelques problèmes en deux dimensions [2]. Les résultats obtenus par S. P. Fekete et al. sur ces problèmes sont disponibles dans [2]. Nous avons écarté les problèmes résolus trop rapidement ainsi que les deux problèmes non résolus par chacune des méthodes. Dans ces résultats, il est indiqué que S. P. Fekete et al. utilisent un Pentium IV 3 Ghz pour résoudre ces instances et le temps limite est fixé à 15 minutes pour chaque instance. Toutes nos expérimentations ont été exécutées sur un processeur Pentium IV 3.2 GHz et 1 GO de RAM, et nous avons utilisé Minisat 2.0 pour résoudre nos instances SAT. Nous avons également fixé le temps limite à 15 minutes.

Chaque instance est caractérisée par le nombre d'objets n, sa faisabilité (F si l'instance possède une solution, N sinon) et le pourcentage d'espace libre dans le conteneur

Instances				FS	M1			M2		
Nom	Espace	Fais.	n		Temps	#var.	#claus.	Temps	#var.	#claus.
E02F17	02	F	17	7	**4.95**	5474	26167	13.9	6660	37243
E02F20	02	F	20	-	5.46	8720	55707	**1.69**	10416	73419
E02F22	02	F	22	167	**7.62**	11594	105910	21.7	13570	129266
E03N16	03	N	16	**2**	39.9	4592	20955	47.3	5644	30259
E03N17	03	N	17	**0**	4.44	5474	27401	9.32	6660	38477
E04F17	04	F	17	13	**0.64**	5474	26779	1.35	6660	37855
E04F19	04	F	19	560	3.17	7562	46257	**1.43**	9040	61525
E04F20	04	F	20	22	5.72	8780	59857	**2.22**	10416	77569
E04N18	04	N	18	**10**	161	6462	32844	87.7	7790	45904
E05F20	05	F	20	491	6.28	8780	59710	**0.96**	10416	77422
Moyenne				> 217	23.9	7291	46159	**18.8**	8727	60894

TABLE 1 – Comparaison avec S. P. Fekete et al.

si tous les objets peuvent y être placés. La table 1 montre les caractéristiques des instances, les résultats de S. P. Fekete et al. **(FS)**, et les résultats de notre approche. Nous avons testé de nombreuses modélisations, résultants de différentes combinaisons avec les formules (1) à (9). Nous indiquons ici les résultats obtenus avec deux de ces modélisations qui nous semblent les plus pertinentes : la modélisation **M1** intègre les formules (1) à (6) et (9), tandis que la modélisation **M2** contient en plus les formules facultatives (7) et (8). Les temps indiqués sont en secondes.

Nous constatons que notre méthode est plus robuste sur ces instances que FS. De plus, nous surclassons FS sur les instances satisfiables, en particulier sur l'instance E02F20 qui n'a pas été résolue dans le temps limite par FS. Sur les instances non satisfiables, FS obtient de meilleurs résultats. Nous pensons que ceci est dû au fait que S. P. Fekete et al. calculent des bornes (voir DFF dans [4]) permettant de détecter de couper très haut dans l'arbre de recherche. Concernant nos différentes modélisations, nous n'avons pas encore pu déterminer dans quels cas une modélisation est meilleure qu'une autre. Nous pouvons simplement constater que l'augmentation de la taille de l'instance induit par la modélisation M2 par rapport à M1 ne garantit en rien une efficacité accrue de la résolution (et mène même quelques fois à un moins bon résultat). L'explication ne se trouve donc pas dans la taille de l'instance mais certainement dans l'influence de la présence de contraintes additionnelles sur le choix des variables effectué par MiniSat.

5.2 Le problème du Strip Packing

Nous avons également comparé notre approche avec celle de T. Soh et al. [12] sur des instances de strip packing en deux dimensions de la librairie OR disponible à l'adresse `http://www.or.deis.unibo.it/research.html`. Le problème consiste à déterminer la hauteur minimale d'un conteneur de largeur fixée pour pouvoir placer un ensemble d'objets donné. Comme T. Soh et al., nous effectuons une recherche dichotomique pour déterminer la hauteur minimale. La borne inférieure est celle proposée par Martello et Vigo [10]. Nous calculons une borne supérieure avec un algorithme glouton qui construit des placements triviaux. En appliquant cet algorithme avec différentes hauteurs on en déduit une borne maximale de la hauteur optimale. L'algorithme consiste simplement à empiler les uns sur les autres les objets, pris dans l'ordre décroissant de leurs largeurs. Chaque fois que la pile atteint la hauteur testée de la bande, la procédure continue avec une nouvelle pile. Si la somme des largeurs des piles est inférieure à la largeur de conteneur alors ce placement est valide. La recherche repart ensuite de la hauteur minimale avec laquelle l'algorithme glouton a trouvé un placement valide.

Pour la recherche dichotomique à partir des bornes, chaque fois qu'une hauteur doit être testée, nous générons une instance SAT et la traitons avec MiniSat 2.0. T. Soh et al. résolvent ces instances sur un Xeon 2.6 GHz avec 2 GO de RAM et utilisent également MiniSat 2.0. Les résultats sont présentés dans la table 2.

Pour chaque instance, la taille du codage est reportée (nombre de variables et nombre de clauses), ainsi que la hauteur minimale trouvée dans le temps limite fixé à 3600 secondes. Les hauteurs optimales sont indiquées en gras. L'optimalité peut être prouvée de deux manières : soit nous démontrons la consistance du problème avec une hauteur égale à la borne inférieure, soit nous trouvons une hauteur h telle que le problème est consistant avec h mais inconsistant avec $h - 1$. Nous avons écarté les instances dans lesquelles le nombre d'objets est trop grand, puisque le nombre de stables infaisables devient trop important et donc le nombre de clauses également.

Nous démontrons l'optimalité pour 16 instances parmi les 22. De plus, sur ces 16 instances, 14 sont résolues en moins de 30 secondes avec une de nos deux modélisa-

Instances				Soh et	M1				M2			
Nom	n	Largeur	LB	al.	Hauteur	#var.	#claus.	Temps	Hauteur	#var.	#claus.	Temps
HT01	16	20	20	**20**	**20**	4592	22963	13.3	**20**	5644	32267	19.4
HT02	17	20	20	**20**	**20**	5474	28669	744	**20**	6660	39745	444
HT03	16	20	20	**20**	**20**	4592	24222	18.5	**20**	5644	33526	25.5
HT04	25	40	15	**15**	16	16850	271500	1206	19	19396	305392	521
HT05	25	40	15	**15**	16	16850	337395	438	16	19396	372287	536
HT06	25	40	15	**15**	16	16850	494500	146	16	19396	528392	295
CGCUT01	16	10	23	**23**	**23**	4592	26745	5.89	**23**	5644	36049	9.71
CGCUT02	23	70	63	65	66	13202	115110	1043	70	15360	188222	1802
GCUT01	10	250	1016	**1016**	**1016**	1190	4785	0.11	**1016**	1606	7237	0.04
GCUT02	23	250	1133	1196	1259	8780	105810	37.3	1196	10416	123522	1241
NGCUT01	10	10	23	**23**	**23**	1190	5132	0.23	**23**	1606	7584	0.09
NGCUT02	17	10	30	**30**	**30**	5474	29662	1.6	**30**	6660	40738	2.74
NGCUT03	21	10	28	**28**	**28**	10122	108138	273	**28**	11924	128542	580
NGCUT04	7	10	20	**20**	**20**	434	1661	0.01	**20**	640	2577	0.01
NGCUT05	14	10	36	**36**	**36**	3122	15558	6.01	**36**	3930	21906	4.44
NGCUT06	15	10	31	**31**	**31**	3810	18629	1.92	**31**	4736	26361	2.91
NGCUT07	8	20	20	**20**	**20**	632	2535	0	**20**	900	3855	0
NGCUT08	13	20	33	**33**	**33**	2522	11870	2.74	**33**	3220	17010	9.73
NGCUT09	18	20	49	**50**	50	6462	33765	391	50	7790	46825	53.3
NGCUT10	13	30	80	**80**	**80**	2522	11790	0.75	**80**	3220	16930	0.39
NGCUT11	15	30	50	**52**	**52**	3810	18507	19.7	**52**	4736	26239	25.9
NGCUT12	22	30	79	**87**	**87**	11594	173575	886	**87**	13570	196931	24.5

TABLE 2 – Résultats sur les instances de la librairie OR et comparaison avec T. Soh et al.

tions. Dans leurs résultats, T. Soh et al. n'ont pas indiqué les temps de calcul. Cependant, ils résolvent optimalement plus de problèmes que nous. Nous pensons que la capacité de leur solveur à mémoriser et à réutiliser les clauses générées par Minisat durant les recherches est un réel avantage. Dans de futurs travaux, nous espérons confirmer cette hypothèse en caractérisant les clauses pouvant être réutilisées par notre approche durant les recherches successives.

6 Conclusion et discussion

Nous avons proposé un codage SAT de la modélisation de S. P. Fekete et al. qui surclasse significativement leur méthode sur des instances satisfiables. Sur les instances non satisfiables, S. P. Fekete et al. obtiennent de meilleurs résultats. Nous pensons que ceci est dû à l'apport des DFF qui permettent d'effectuer des coupures très tôt durant la recherche. Nous envisageons donc d'intégrer le calcul des DFF directement dans la résolution effectuée par MiniSat afin de bénéficier de ces mêmes coupures.

Nous avons également expérimenté notre codage sur des instances de strip packing. Bien que résolvant moins d'instances que T. Soh et al., nous obtenons des résultats encourageants. Nous allons dans des travaux futurs tenter de caractériser les situations dans lesquelles les clauses générées par le solveur SAT peuvent être réutilisées. De plus, dans la plupart des instances, certains objets sont de tailles identiques (dans toutes les dimensions). Donc de nombreux stables minimaux infaisables représentent des situations identiques dans le sens où les objets qui les composent sont de tailles identiques dans cette dimension. Afin de limiter autant que possible le nombre de clauses correspondant aux stables minimaux infaisables, nous mettrons en place dans le solveur SAT un mécanisme de détection des stables infaisables qui devrait considérablement alléger les formules produites par notre codage.

Références

[1] Andreas Brandstädt, Van Bang Le, and Jeremy P. Spinrad. *Graph classes : a survey*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1999.

[2] F. Clautiaux, J. Carlier, and A. Moukrim. A new exact method for the two-dimensional orthogonal packing problem. *European Journal of Operational Research*, 183(3) :1196–1211, 2007.

[3] S. P. Fekete and J. Schepers. A combinatorial characterization of higher-dimensional orthogonal packing. *Mathematics of Operations Research*, 29(2) :353–368, 2004.

[4] S. P. Fekete and J. Schepers. A general framework for bounds for higher-dimensional orthogonal packing problems. *Mathematical Methods of Operations Research*, 60(2) :311–329, 2004.

[5] S. P. Fekete, J. Schepers, and J. van der Veen. An exact algorithm for higher-dimensional orthogonal packing. *Operations Research*, 55(3) :569–587, 2007.

[6] M. R. Garey and D. S. Johnson. *Computers and Intractability – A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, 1979.

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

[8] Richard E. Korf. Optimal rectangle packing : New results. In Shlomo Zilberstein, Jana Koehler, and Sven Koenig, editors, *ICAPS*, pages 142–149. AAAI, 2004.

[9] Norbert Korte and Rolf H. Möhring. An incremental linear-time algorithm for recognizing interval graphs. *SIAM J. Comput.*, 18(1) :68–81, 1989.

[10] S. Martello, M. Monaci, and D. Vigo. An exact approach to the strip-packing problem. *Journal on Computing*, 15(3) :310–319, 2003.

[11] Michael D. Moffitt and Martha E. Pollack. Optimal rectangle packing : A meta-csp approach. In Derek Long, Stephen F. Smith, Daniel Borrajo, and Lee McCluskey, editors, *ICAPS*, pages 93–102. AAAI, 2006.

[12] T. Soh, K. Inoue, N. Tamura, M. Banbara, and H. Nabeshima. A sat-based method for solving the two-dimensional strip packing problem. In *Proceedings of the 15th RCRA workshop on Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion*, 2008.