Une méthode efficace de programmation dynamique pour la résolution du problème de Bin Packing en 2 dimensions.

Catherine Huyghe, Stéphane Negre, Mélanie Fontaine
Université de Picardie Jules Verne
{catherine.huyghe,stephane.negre,melanie.fontaine}@u-picardie.fr

Mots-clés *: Bin Packing, programmation dynamique, partition, bornes.*

1 Introduction

Une entreprise de composants électroniques gère quotidiennement des milliers de commandes, chacune comprenant jusqu'à mille composants électroniques. Ces composants doivent être disposés sur des panneaux avant l'expédition. Ces panneaux ont un coût élevé. L'objectif est d'optimiser le placement des composants sur les panneaux pour minimiser le nombre de panneaux utilisés, réduisant ainsi les coûts. Des contraintes existent, notamment la possibilité d'orienter librement ou non les composants, c'est-à-dire qu'ils peuvent être tournés par incréments de 90 degrés ou non avant d'être placés sur un panneau. Ce défi peut être formulé comme un problème d'emballage bidimensionnel (2D-BPP) [3][4], où des objets rectangulaires o (les composants) de largeur o.w, de hauteur o.h et avec une orientation libre ou non doivent être disposés dans des bins rectangulaires b (les panneaux) de largeur b.w et de hauteur b.h.

Pour pouvoir traiter les milliers de commandes par jour dans un délai raisonnable, nous proposons une heuristique rapide et efficace. Chaque instance (commande) doit être traitée en moins d'une minute tout en restant proche de la solution optimale.

2 Algorithme

L'idée principale de l'heuristique consiste à minimiser, à chaque décision de placement d'un objet dans un bin, la quantité d'espace qui sera forcément perdue à la fin du traitement. Inspiré du modèle de [1], nous considérons deux modèles de surfaces résiduelles dans les bins. Le premier, appelé surface maximum et noté SM, se caractérise par une surface maximale qui ne peut être entièrement contenue par aucune autre surface maximale. Elles permettent de déterminer rapidement les placements possibles (Bottom Left) de chaque objet dans les bins. Le second, appelé surface minimale, permet d'anticiper le calcul des espaces résiduels perdus à la fin de l'algorithme. Il s'agit d'une surface résultant de la projection sur un axe de tous les objets. Nous avons des surfaces minimales distinctes pour les axes des x noté Smx et pour l'axe des y noté Smy.

Pour calculer les espaces perdus, l'algorithme s'inspire de la résolution exacte du problème de Partition décrite dans [2]. Ainsi deux matrices sont construites : Mx pour l'axe des x et My pour l'axe des y. Mx a b.w + 1 colonnes et n + 1 lignes, où n est le nombre d'objets. Les objets sont ordonnés par surface croissante pour placer les plus grands en premier. Pour les objets à orientation libre, deux objets distincts sont considérés avec un ou exclusif (XOR), de sorte que le placement de l'un entraîne le placement de l'autre. Mx est ensuite calculé comme suit : Mx(0,0) = True, Mx(i,j) = True si Mx(i-1, j)

= True ou Mx(i-1,j-o_i.w) = True, ∀i ∈ [0,n], ∀j ∈ [0,b.w]. La création de My se fait de la même manière, mais en prenant la hauteur en considération.

À l'initialisation, les listes de Smx, Smy, et SM correspondent à la liste des bins disponibles (le nombre de bins initialement disponible correspond au nombre d'objets). Ensuite, pour chaque objet i dans l'ordre décroissant de leur surface et pour toutes les SM pouvant l'accueillir, nous calculons la liste des surfaces minimales sur l'axe des x tel que $L_{Smx} = \cup_{Sm \in Smx} \{Sm, Sm \cap o_i.(x,y,w,h) \neq \varnothing\}$ et la liste des surfaces minimales sur l'axe des y tel que $L_{Smy} = \cup_{Sm \in Smy} \{Sm, Sm \cap o_i.(x,y,w,h) \neq \varnothing\}$ où $o_i.(x,y,w,h)$ représente la combinaison des attributs x,y,w et h de l'objet i, c'est-à-dire sa surface à la position (x,y) dans le bin. Nous calculons également la surface perdue noté SP comme suit : $SP = \sum_{Smx \in L_{Smx}} (Smx.w - \max_{j \in [0,Smx.w]} j\ s.t.\ M(i-1,j) = \text{True}) \times Smx.h + \sum_{Smy \in L_{Smy}} (Smy.h - \max_{j \in [0,Smy.h]} j\ s.t.\ M(i-1,j) = \text{True}) \times Smy$ où Smx.w et Smy.w sont respectivement la largeur de Smx et de Smy, et Smx.h et Smy.h sont respectivement la hauteur de Smx et de Smy.

L'objet i est ensuite placé dans la surface SM qui minimise SP. Les listes des SM, Smx et Smy sont alors mises à jour avant de passer à l'objet suivant.

On peut raisonnablement considérer que le nombre de surface générée est de l'ordre de n. Sous cette hypothèse, notre heuristique a une complexité en $O(n^3 \times (b.w+b.h))$ et en mémoire en $O(n \times (b.w+b.h))$.

Notre algorithme a été testé sur des instances difficiles de la littérature décrites dans [2] qui correspondent aux pires cas des heuristiques classiques (Next Fit, First Fit et Best Fit decreasing). Nous trouvons les solutions optimales sur chacune de ces instances, en quelques secondes.

Nous avons aussi généré de nombreuses instances en partitionnant aléatoirement les bins pour définir jusqu'à mille objets. La solution optimale ne tolère aucune perte, et la moindre erreur dans l'algorithme nous éloigne de la solution optimale. Pour ajouter à la complexité, tous les objets sont librement orientés.

Sur ces instances, nous obtenons dans le pire cas un bin de plus que la solution optimale. L'ensemble des résultats est obtenu en moins d'une minute.

3 Conclusions et perspectives

Nous avons proposé un algorithme de base rapide qui fournit des solutions très proches de l'optimum, sur de grandes instances (jusqu'à mille objets), difficiles à résoudre.

Nous envisageons prochainement de tester les résultats sur les benchmarks classiques de la littérature et de comparer nos résultats avec ceux des méthodes publiées.

Pour pouvoir intégrer et prendre en compte facilement de nouvelles contraintes, un modèle objet souple est proposé.

Références

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

[2] M. R. Garey and D. S. Johnson. Computer and intractability: A Guide to the Theory of -Completeness, 1979.

[3] A. Pandey, An analysis of solutions to the 2D bin packing problem and additional complexities, *Authorea Preprints*, 2023.

[4] F. Gzara, S. Elhedhli, and B. C. Yildiz, The pallet loading problem: Three-dimensional bin packing with practical constraints, *European Journal of Operational Research*, vol. 287, no. 3,pp. 1062-1074, 2020.