Une heuristique anytime basée sur la génération de colonnes pour les problèmes de Variable-sized Bin Packing à deux dimensions avec coupes guillotine

Florian Fontan[1], Luc Libralesso[2]

[1] Artelys France, 75009 Paris
florian.fontan@artelys.fr
[2] Université Clermont Auvergne, LIMOS
luc.libralesso@uca.fr

Mots-clés : *heuristique, génération de colonnes, algorithme anytime, recherche arborescente heuristique, variable-sized bin packing, cutting stock, coupes guillotine.*

Ce travail fait suite aux travaux présentés l'année précédente à la ROADEF. Nous avions généralisé l'algorithme de recherche arborescente anytime développé dans de cadre du Challenge ROADEF/EURO 2018 [1] à l'ensemble des problèmes de Bin Packing, Knapsack et Strip Packing à deux dimensions avec coupes guilllotine de la littérature, et montré qu'il obtenait des résultats compétitifs et même état-de-l'art sur plusieurs variantes. Nous avons maintenant incorporé cet algorithme au sein d'une heuristique basée sur la génération de colonnes afin de pouvoir résoudre également les problèmes avec objectif Variable-sized Bin Packing, ou plus généralement, les problèmes où l'ordre des bins n'est pas imposé et choisir les bins à utiliser et leur ordre fait partie des décisions à prendre.

Le problème de Variable-sized Bin Packing peut se formuler ainsi. n types d'items avec demande q_j, $j = 1 \dots n$, m types de bins avec borne inférieure l_i, borne supérieure u_i, et cout c_i. Pour chaque type de bin i, $i = 1 \dots m$, soit les K_i patterns possibles avec $x^k_{i,j} = q$ si le pattern k, $k = 1 \dots K_i$ de la bin i contient q copies du type d'item j. Le problème consiste à trouver un sous-ensemble de patterns de cout minimum de sorte que les demandes de chaque type d'items soit satisfaites et que les bornes sur l'utilisation des type de bins soient respectées. Nous utilisons la formulation en programmation linéaire en nombre entier suivante :

Variables :

$$\forall i = 1 \dots m, \forall k = 1 \dots K_i, \quad y^k_i = q \qquad \text{si } q \text{ copies du pattern } k \text{ de la bin } i \text{ sont utilisés}$$

Objectif :

$$\min \sum_{i=1}^{m} \sum_{k=1}^{K_i} c_i y^k_i$$

Contraintes :

$$\sum_{i=1}^{n} \sum_{k=1}^{K_i} x^k_{i,j} y^k_i = q_j \quad \forall j = 1 \dots n \qquad \text{Satisfaction des demandes pour chaque type d'items}$$

$$l_i \leq \sum_{k=1}^{K_i} y^k_i \leq u_i \quad \forall i = 1 \dots m \qquad \text{Bornes pour l'utilisation de chaque type de bins}$$

Dans cette formulation, le nombre de variables est exponentiel. Il n'est donc généralement pas possible de la résoudre directement en pratique. La relaxation peut quand même être résolue en pratique avec une méthode de génération de colonnes. On résout la relaxation avec

un sous-ensemble de variables, c'est-à-dire ici de patterns, puis on cherche une contrainte violée dans le problème dual. S'il y en a une, on ajoute la variable correspondante dans le primal et on itère ainsi jusqu'à ce que toutes les contraintes soient respectées dans le dual ; ce qui implique que la solution du primal est optimale.

Déterminer si le dual contient une contrainte violée correspond au problème de pricing. Dans le cas du Variable-sized Bin Packing, il se réduit à un problème de Knapsack. C'est ce problème de Knapsack que nous résolvons avec notre algorithme de recherche arborescente. Cet algorithme étant heuristique, il peut ne pas trouver de contrainte violée alors qu'il en existe encore. Ainsi, dans notre résolution, nous n'avons pas de garantie que la solution que nous obtenons lors de la relaxation soit optimale. Ceci n'est pas un problème puisque notre algorithme global est lui-même heuristique, et ce sont les variables générées qui nous intéressent.

La relaxation calculée par génération de colonnes est classiquement exploitée de différentes manières : dans des algorithmes exacts de type Branch-and-Bound (Branch-and-Price) ou dans des heuristiques. Nous nous intéressons ici aux heuristiques.

L'heuristique que nous avons implémentée est inspirée des travaux de [3]. Dans cet articles, les auteurs listent et expérimentent les différentes méthodes utilisées pour passer d'une relaxation par génération de colonnes à une solution réalisable : résoudre le problème maître restreint et différentes variantes de recherches arborescentes heuristiques dans le schéma de branchement de la formulation de Dantzig-Wolfe, c'est-à-dire, en branchant directement sur les colonnes - ce qui n'est généralement pas possible dans le cadre d'un Branch-and-Price exact. Nous nous sommes concentrés sur l'heuristique basée sur la Limited Discrepancy Search. C'est l'algorithme qui donne les meilleurs résultats pour les problèmes considérés par les auteurs.

Toutefois, cet algorithme présente un inconvient majeur dans notre cas. Il nécessite de calculer complètement la relaxation au noeud racine, ce qui peut être très long pour des instances avec plus d'une dizaine d'items par bin. [3] utilisent ces heuristiques au sein d'algorithmes de Branch-and-Price (exacts) dans lesquels ils calculent cette relaxation de manière exacte de toute façon et la solution heuristique est utilisée pour accélérer la résolution en fixant certaines variables grâces au coûts réduits ou si beaucoup de noeuds ont leur bornes égale à l'optimum [2]. Ce n'est donc pas un problème dans ce contexte. Mais dans notre cas, nous souhaitons avoir un algorithme heuristique indépendant et anytime. Il n'est donc pas envisageable de commencer par la résolution complète de la relaxation au noeud racine.

Nous avons donc développé un nouvel algorithme anytime basé sur des exécutions successives de l'algorithme de Limited Discrepancy Search, où le nombre d'itérations de la génération de colonnes est limité à chaque noeud.

Nous avons comparé les résultats de notre heuristique par rapport aux algorithmes de la littérature pour une dizaine de variantes de problèmes de Variable-sized Bin Packing, Cutting Stock et Bin Packing. Notre heuristique obtient en moyenne de meilleures solutions [1].

Références

[1] Luc Libralesso and Florian Fontan. An anytime tree search algorithm for the 2018 ROADEF/EURO challenge glass cutting problem. *European Journal of Operational Research*, 291(3) :883–893, June 2021.

[2] Artur Pessoa, Ruslan Sadykov, Eduardo Uchoa, and François Vanderbeck. A generic exact solver for vehicle routing and related problems. *Mathematical Programming*, June 2020.

[3] Ruslan Sadykov, François Vanderbeck, Artur Pessoa, Issam Tahiri, and Eduardo Uchoa. Primal Heuristics for Branch and Price : The Assets of Diving Methods. *INFORMS Journal on Computing*, 31(2) :251–267, April 2019.

1. `https://github.com/fontanf/packingsolver/blob/master/results_rectangleguillotine.ods`