Un algorithme de recherche arborescente anytime pour les problèmes de Packing 2D avec coupes guillotine à 2 ou 3 niveaux

Florian Fontan[1], Luc Libralesso[2]

[1] Artelys, 75009 Paris, France
florian.fontan@artelys.com
[2] Univ. Grenoble Alpes, CNRS, Grenoble INP †, G-SCOP, 38000 Grenoble, France
luc.libralesso@grenoble-inp.fr

Mots-clés : *Packing 2D guillotine, Recherche Arborescente, Algorithme Anytime.*

Tous les deux ans, l'Association Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF) organise conjointement avec l'Association Européenne de Recherche Opérationnelle (EURO) un challenge d'optimisation ouvert à tous. Chaque édition est organisée en collaboration avec un industriel, permettant ainsi aux chercheurs d'expérimenter les derniers développements sur des problèmes à la fois difficiles et appliqués. L'édition 2018 du challenge était organisée en collaboration avec l'entreprise Saint-Gobain et était dédiée à un problème de découpe de verre. [1] Nous avions développé et soumis un algorithme de recherche arborescente pour le challenge, qui a été classé 1er lors de la phase finale.

Nous avons depuis adapté l'algorithme pour les problèmes de Packing 2D avec coupes guillotine à 2 ou 3 niveaux de la littérature.

Dans cette présentation, nous décrirons le schéma de branchement ainsi que l'algorithme de recherche arborescente utilisés, et comparerons les performances de l'algorithme ainsi obtenu par rapport aux algorithmes de la littérature.

L'espace des solutions est représenté par un arbre. La racine est la solution vide où aucune coupe n'a été effectuée. À chaque niveau de l'arbre, un nouvel objet est ajouté à la solution. Les objets sont placés dans l'ordre dans lequel ils sont extraits lors de la découpe. Quelques règles simples de dominance et une stratégie de rupture de symétries sont appliquées afin de réduire la taille de l'arbre.

Toutefois, l'arbre reste beaucoup trop grand pour être parcouru entièrement. Il est alors exploré en utilisant un algorithme nommé *Memory Bounded A** décrit dans l'Algorithme 1. C'est un algorithme proche des algorithmes classiques de recherche arborescente comme *A** ou *Beam Search*. Une file contient les nœuds à explorer. Elle contient initialement uniquement la racine. À chaque itération, le nœud le plus « intéressant » est extrait et ses fils sont ajoutés à la file. Si la file dépasse une taille donnée, les nœuds les moins intéressants qu'elle contient sont supprimés. Cet algorithme permet d'explorer en priorité les parties les plus prometteuses de l'arbre.

Sa simplicité permet à l'algorithme d'être très versatile. Ainsi, il s'adapte bien aux variantes Bin Packing, Multiple Knapsack et Strip Packing, avec coupes guillotine à 2 ou 3 niveaux, exactes ou non-exactes, avec la direction des coupes de niveau 1 imposée ou non, et avec ou sans rotation des objets.

Sur la majorité des variantes, l'algorithme obtient des résultats compétitifs par rapport aux algorithmes dédiés de la littérature. Il fourni également une implémentation pour beaucoup de variantes qui n'avaient pas encore été traitées, comme les problème de Multiple Knapsack. De

†Institute of Engineering Univ. Grenoble Alpes

1. http://www.roadef.org/challenge/2018/en/index.php

Algorithm 1 Memory Bounded A* (MBA*)

```
1: fringe ← {root}
2: while |fringe| ≠ ∅ and time < timelimit do
3:     n ← extractBest(fringe)
4:     fringe ← fringe \ {n}
5:     for all v ∈ children(n) do
6:         fringe ← fringe ∪ {v}
7:     while |fringe| > D do
8:         n ← extractWorst(fringe)
9:         fringe ← fringe \ {n}
```

plus, il s'adapte bien à l'ajout de contraintes industrielles supplémentaires, comme l'illustrent ses performances sur le challenge.