# Résolution de grands problèmes de *Bin Packing avec contraintes* avec Hexaly Optimizer

Bienvenu Bambi

Hexaly, 251 boulevard Pereire, 75017 Paris, France
bbambi@hexaly.com

**Mots-clés** : *affectation, packing*

## 1 Introduction

Le problème de *Bin Packing*, bien connu dans la littérature de recherche opérationnelle, consiste à affecter N items à M contenants (ou bins). Parmi ses nombreuses variantes, le *Bin Packing avec exclusions* se distingue par des contraintes supplémentaires empêchant la cohabitation de certains items dans un même bin, ce qui complexifie significativement le problème[1]. Cette variante a été particulièrement pertinente pour l'un de nos clients, confronté à des défis d'optimisation dans un environnement industriel à grande échelle. Dans leur cas, avec près de 100,000 items et 10,000 bins, la modélisation traditionnelle en programmation linéaire en nombres entiers (PLNE), dont la complexité est de l'ordre de $O(M \times N)$, s'avérait prohibitivement coûteuse en termes de ressources computationnelles. Par ailleurs, en plus des contraintes d'exclusion, il fallait prendre en compte des contraintes d'exclusivité de ressources : une ressource attribuée à un bin ne peut coexister avec certaines autres ressources spécifiques dans ce même bin. C'est dans ce contexte que nous avons développé une approche alternative, en exploitant les capacités d'Hexaly Optimizer.

Hexaly Optimizer, anciennement nommé LocalSolver, est un solveur d'optimisation mathématique basé sur différentes techniques de recherche opérationnelle, combinant des méthodes exactes, telles que la programmation linéaire, non linéaire et par contraintes, et heuristiques, comme la recherche locale [2].

## 2 Modélisation

Le modèle de Bin Packing avec exclusions que l'on essaie de résoudre peut s'écrire directement avec la variable de type `set` de Hexaly Optimizer. Les contraintes du modèle se modélisent alors efficacement avec les opérateurs ensemblistes d'Hexaly Optimizer.

Pour illustrer concrètement la modélisation des contraintes clés du problème (exclusivité et exclusion), nous considérons ci-dessous des variantes du modèle de Bin Packing avec coloration.

### 2.1 Modélisation de l'exclusivité

S'il est interdit de mélanger plusieurs couleurs au sein d'un bin, alors l'opérateur `distinct` permet de modéliser directement cette exclusivité :

```
function model() {
    bins[k in 0...nbMaxBins] <- set(nbItems);
    constraint partition[k in 0...nbMaxBins](bins[k]);

    for [k in 0...nbMaxBins] {
```

```
        constraint count(distinct(bins[k], i => itemColors[i])) <= 1;
        ... // Autres contraintes et objectifs
    }
    minimize totalBinsUsed;
}
```

Dans ce modèle, l'utilisation de l'expression `distinct` avec l'opérateur `count` assure que chaque bin ne contient qu'un seul type d'item basé sur la couleur, permettant ainsi de gérer l'exclusivité des ressources dans chaque bin.

### 2.2 Modélisation de l'exclusion

Dans d'autres situations on peut vouloir interdire d'avoir plusieurs items de la même couleur dans un bin. Cette contrainte se modélise avec l'expression `intersection`. On écrira :

```
constraint count(intersection(bins[k], allItemsWithColor[c])) <= 1;
```

L'expression `intersection` permet de modéliser les incompatibilités. En contrôlant la cardinalité de l'intersection entre un ensemble et un ensemble de valeurs interdites, nous pouvons empêcher efficacement des items incompatibles d'être assignés au bin.

## 3 Résultats et conclusion

La modélisation ensembliste dans Hexaly Optimizer (avec, entre autres, ses opérateurs `distinct` et `intersection`) permet de résoudre des problèmes de bin packing de très grandes tailles. Dans nos tests, Hexaly Optimizer trouve des solutions avec un écart moyen de seulement 1.2% par rapport à l'optimum en 10 secondes pour des instances allant jusqu'à 5 000 items. Et pour le cas industriel évoqué plus haut, avec des contraintes complexes et plus de 100 000 items, Hexaly Optimizer a livré des solutions quasi-optimales en quelques minutes.

## Références


[1] F. Brandão, J. P. Pedroso. *Bin Packing and Related Problems : General Arc-flow Formulation with Graph Compression*, arXiv :1310.6887 [math.OC], 2013.

[2] F. Gardi, T. Benoist, J. Darlay, B. Estellon, et R. Megel. *Mathematical Programming Solver Based on Local Search*, Wiley, 2014