# Génération de colonne sur Bin Packing

Athanaël Jousselin

Octobre-Novembre 2024

Soit le problème de Bin Packing : On souhaite minimiser le nombre de bin (conteneur), de capacité $C$, nécessaire pour contenir $n$ objets de taille $w_i$ ($i \in \{1, n\}$).

On s'intéresse au cas mono-dimensionnel, c'est à dire qu'on empile les objets, il n'est pas possible d'en mettre deux côte à côte.

Les variables de décisions sont :

$$y_j = \begin{cases} 1 & \text{si le bin } j \text{ est utilisé} \\ 0 & \text{sinon} \end{cases} \quad \forall j \in \{1, n\}$$

$$x_{i,j} = \begin{cases} 1 & \text{si l'objet } i \text{ est attribué au bin } j \\ 0 & \text{sinon} \end{cases} \quad \forall i, j \in \{1, n\}$$

On pose le modèle :

$$\begin{array}{llllll} \min & z = & \sum_{j=1}^{n} y_j & & & \\ s.c. & & \sum_{j=1}^{n} x_{i,j} & = & 1 & \forall i \in \{1, n\} \\ & & \sum_{i=1}^{n} w_i x_{i,j} & \leq & C y_j & \forall j \in \{1, n\} \\ & & x_{i,j} & \in & \{0, 1\} & \forall i \in \{1, n\} \quad \forall j \in \{1, n\} \\ & & y_j & \in & \{0, 1\} & \forall j \in \{1, n\} \end{array}$$

Maintenant on considère des patterns, c'est à dire toutes les combinaisons d'objets qui rentrent dans un bin. (cette approche marche encore mieux quand les $w_i$ se répètent beaucoup, par exemple dans un problème cutting stock).
On pose $m_{i,j}$ le nombre de fois qu'un objet $j$ est dans la composition du pattern $i$, $d_j$ le nombre d'objet $j$ d'une même taille, $i \in \{1, n\}$ le nombre de pattern et $j \in \{1, k\}$ le nombre d'objet de taille différente.
Les variables de décisions sont $x_i$ le nombre de fois où le pattern $i$ est utilisé ($\forall i \in \{1, n\}$).
A chaque pattern est aussi associé une perte, ainsi on souhaite minimisé la perte. Cependant on peut prouver que minimiser le nombre de pattern total revient au même.
On pose le modèle :

$$\begin{array}{llllll} \min & z = & \sum_{i=1}^{n} x_i & & & \\ s.c. & & \sum_{i=1}^{n} m_{i,j} x_i & \geq & d_j & \forall j \in \{1, k\} \\ & & x_i & \in & \mathbb{N} & \forall i \in \{1, n\} \end{array}$$

On pourrait générer à l'avance tout les patterns possibles, cependant cela n'est pas très efficace. On va donc ce pencher sur la génération de colonne qui nous permettra de sélectionner les patterns utiles.

Pour choisir ces nouveaux patterns on s'intéresse aux variables duales $\pi$ de notre pack de contrainte. $\pi_j$ nous indique le "prix" que coûterait de rajouter une unité de demande pour un objet $j$ et inversement combien on peut "économiser" en ajoutant un pattern.
Ainsi on écrit un nouveau modèle tel qu'il maximise les économies que l'on peut faire en respectant la taille des bins :

$$\begin{array}{lllll} \max & z = & \sum_{j=1}^{k} \pi_j y_j & & \\ s.c. & & \sum_{j=1}^{k} w_j y_j & \leq & C \\ & & y_j & \in & \mathbb{N} \quad \forall j \in \{1, k\} \end{array}$$

Ainsi si ce problème (pricing problem) à une valeur objective plus grande que 1 alors on estime qu'ajouter $y$ comme nouveau pattern pourra réduire le nombre de bin utilisé.