Continuous optimization

Samuel Buchet & Dorian Dumez & Brendan Guevel

Mai 2017

1 Relaxation continue

1.1 Problème de base

On peut écrire le problème de bin-packing monodimensionnel comme :
(PE) : $\min\ z = \sum_{b=1}^{n} u_b$
s.c.

$$\forall b \in [\![1, n]\!] : \sum_{p=1}^{n} s_p x_{pb} \leqslant u_b c \ (1)$$

$$\forall p \in [\![1, n]\!] : \sum_{b=1}^{n} x_{pb} = 1 \ (2)$$

$$\forall p \in [\![1, n]\!] : \forall b \in [\![1, n]\!] : u_b, x_{pb} \in \{0, 1\}$$

Avec pour paramètre :

- n, le nombre d'objet, donc un majorant du nombre de bin nécessaire
- c, la capacité d'une boite
- s, le vecteur de la taille des objets

Et pour variable :

- u_b qui est vrai si on utilise le bin numéro b
- x_{pb} qui est vrai si on met l'objet p dans le bin b

Bien que juste ce programme linéaire n'est pas très utilisable en pratique, en effet :

- il ne prend pas en compte la redondance des objets, si on a 100 fois le même objet il créera 100 variables
- bien qu'identique de part leur caractéristique tous les bin sont différencié

Donc il en résultera un nombre considérable de solution équivalente due à des symétries. Un algorithme de résolution devrai donc prendre en compte l'interprétation du problème ou y ajouter des contraintes comme :

$$\forall b \in [\![1, n-1]\!] : u_b \geqslant u_{b+1} \ (3)$$

$$\forall b \in [\![1, n-1]\!] : argmin_{p \in \{i \in [\![1,n]\!] | x_{ib}=1\}} p \leqslant argmin_{p \in \{i \in [\![1,n]\!] | x_{i,b+1}=1\}} p \ (4)$$

Où la contraintes (3) force les bin ouvert à être ceux avec les plus petit identifiant et la (4) ordonne les bin par le schéma d'objet qu'il contient. Mais même si cela ne règle pas le problème des variables redondante, et rend ce problème nécessaire par leur utilisation.

1.2 Relaxation linéaire

La relaxation linéaire de (PE) s'écrit :
(Rl) : min $z_{\text{Rl}} = \sum_{b=1}^{n} u_b$
s.c.

$$\forall b \in [\![1, n]\!] : \sum_{p=1}^{n} s_p x_{pb} \leqslant u_b c \ (1)$$

$$\forall p \in [\![1, n]\!] : \sum_{b=1}^{n} x_{pb} = 1 \ (2)$$

$$\forall p \in [\![1, n]\!] : \forall b \in [\![1, n]\!] : u_b, x_{pb} \in [0, 1]$$

Mais on peut alors toujours dire que la valeur de la relaxation linéaire est $\frac{\sum_{p=1}^{n} s_p}{c}$. Sauf que la valeur de (PE) est toujours entière donc on peut utiliser comme borne inférieure $\left\lceil \frac{\sum_{p=1}^{n} s_p}{c} \right\rceil$

Proof. Soit u_b^* et x_{bp}^* la valeur des variables dans la solution optimale de la relaxation linéaire. Alors on a bien évidement $\forall p \in [\![1, n]\!] : \sum_{b=1}^{n} x_{pb}^* = 1$ par la contrainte (2). De plus, vu que chaque u_b n'apparais que dans une contrainte, on peut dire que $u_b^* = \frac{\sum_{p=1}^{n} s_p x_{pb}^*}{c}$ par la contrainte (1). La valeur cette solution est alors $\sum_{b=1}^{n} \frac{\sum_{p=1}^{n} s_p x_{pb}^*}{c} = \frac{\sum_{p=1}^{n} s_p}{c}$ par la contrainte (2).

□

2 tâches

- réaliser un parseur ⇒ OK
- faire la relaxation continue ⇒ OK
- heuristique pour borne supérieure (best fit) ⇒ OK
- relaxation lagrangienne + réparation
- tests statistiques

Pour l'heuristique lagrangienne :

- choisir les contraintes à dualiser
- trouver un moyen de résoudre le problème relâché
- mettre en oeuvre l'algorithme de descente de gradient
- créer une heuristique de réparation

3 Relaxations lagrangiennes possibles

3.1 Dualisation de la contrainte 1

Premièrement on remarque que l'on peut séparer en deux la contrainte (1) et donc écrire (PE) comme :

(PE) : $\min z = \sum_{b=1}^{n} u_b$

s.c.

$$\forall b \in [\![1,n]\!] : \sum_{p=1}^{n} s_p x_{pb} \leqslant c \ (1')$$
$$\forall p \in [\![1,n]\!] : \forall b \in [\![1,n]\!] : x_{pb} \leqslant u_b \ (1")$$
$$\forall p \in [\![1,n]\!] : \sum_{b=1}^{n} x_{pb} = 1 \ (2)$$
$$\forall p \in [\![1,n]\!] : \forall b \in [\![1,n]\!] : u_b, x_{pb} \in \{0,1\}$$

Alors si on dualise la contrainte $(1')$ on obtient :

$RL_1(u)$: $\min z_1(u) = \sum_{b=1}^{n} u_b - \mu_b(c - \sum_{p=1}^{n} s_p x_{pb})$

s.c.

$$\forall p \in [\![1,n]\!] : \forall b \in [\![1,n]\!] : x_{pb} \leqslant u_b \ (1")$$
$$\forall p \in [\![1,n]\!] : \sum_{b=1}^{n} x_{pb} = 1 \ (2)$$
$$\forall p \in [\![1,n]\!] : \forall b \in [\![1,n]\!] : u_b, x_{pb} \in \{0,1\}$$

Où $\mu \geqslant 0$ car la contrainte $(1')$ est une contrainte en inégalité.

En écrivant la fonction objectif comme : $\min z_1(u) = \sum_{b=1}^{n}(u_b + \sum_{p=1}^{n}(\mu_b s_p x_{pb})) - \sum_{b=1}^{n} \mu_b c$. On peut alors voir $RL_1(u)$ comme un problème d'UFLP (uncapacited facility location problem) avec des coûts d'ouverture de 1 et des coûts d'association de $\mu_b s_p$. A noter que l'on ajoute un terme constant à la fonction objectif pour obtenir la vraie valeur de la relaxation lagrangienne.

3.2 Dualisation de la contrainte 2

Une autre possibilité est de dualiser la contrainte 2) : $\sum_{b=1}^{n} x_{pb} = 1$.

On obtient alors :

$RL_2(u)$: $\min z_2(u) = \sum_{b=1}^{n} u_b + \sum_{p=1}^{n} \mu_p(1 - \sum_{b=1}^{n} x_{pb})$

s.c.

$$\forall b \in [\![1,n]\!] : \sum_{p=1}^{n} s_p x_{pb} \leqslant u_b c \ (1)$$
$$\forall p \in [\![1,n]\!] : \forall b \in [\![1,n]\!] : x_{pb}, u_b \in \{0,1\}$$

Etant donné que la contrainte 2 est une contrainte d'égalité le signe des μ_i est libre.

On peut réécrire la fonction objectif comme $\min z_2(u) = \sum_{b=1}^{n} \left(u_b - \sum_{p=1}^{n} \mu_p x_{pb} \right) + \sum_{p=1}^{n} \mu_p$.

Le problème est alors décomposable par bin. En effet, le terme de droite est constant et la somme de gauche peut être décomposée par bin. Les variables d'un terme de la somme ne dépendent que d'un seul bin et il y a aussi une unique contrainte par bin. On se retrouve donc avec les n sous problèmes identiques :

$$min \quad z_{2b}(0) = u_b - \sum_{p=1}^{n} \mu_p * x_{pb}$$
$$s.c : \sum_{p=1}^{n} s_p * x_{pb} \leq u_b * c$$

Ce problème peut être décomposé en deux sous problèmes : le problème dans lequel $u_b = 0$ et celui dans lequel $u_b = 1$.

Dans le cas où $u_b = 0$, la contrainte force les variables x_{pb} à être nulles (les s_p étant tous positifs). la valeur optimale de la fonction objectif est alors 0.

Dans le cas où $u_b = 1$ la contrainte (1) devient une contrainte de capacité. Le problème peut alors être vu comme un problème de de sac à dos en inversant le signe de la fonction. Les objets de ce problème ont pour poids s_p et pour valeurs μ_p.

Enfin, la valeur optimal de ces sous problème peut être trouvée en comparant les résultat obtenus avec le problème de sac à dos et le problème avec $u_b = 0$. La valeur optimale du premier sous prpoblème étant 0, le deuxième donne une meilleure valeur si $u_b - \sum_{p=1}^{n} \mu_p * x_{pb} < 0$ ou encore $1 < \sum_{p=1}^{n} \mu_p * x_{pb}$. On peut donc résoudre le problème en résolvant le sous-problème de sac à dos et en comparant le résultat avec 1.

On peut ainsi résoudre $RL_2(u)$ en résolvant 1 problème de sac à dos, en multipliant le résultat par n et en ajoutant la somme des μ.

4 Python

Un peu de doc bien faite : `https://www.tutorialspoint.com/python/index.htm`

Pour exécuter le code : python3 bin_packing.py