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) : $z = \min \sum_{b=1}^{m} u_b$

s.c.

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

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

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

Avec pour paramètre :

- n, le nombre d'objet
- m, le nombre de bin maximal, peut être obtenu à l'aide d'une heuristique ou par défaut n (mais sera coûteux)
- 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 :

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

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

Où la contraintes (3) force les bin ouvert à être ceux avec les plus petits identifiants 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 redondantes, et rend ce défaut nécessaire par leur utilisation.

Si on veux les écrire avec des contraintes linéaires on peut écrire la contrainte (4) avec :

$$\forall p \in [\![2,n]\!] : \forall b \in [\![2,m]\!] : x_{pb} \leqslant \sum_{pp=1}^{p-} x_{pp,b-1} \text{ (4')}$$

$$x(1,1) = 1 \text{ (4")}$$

De plus on peut remarquer que sous ses contraintes on a toujours :

$$\forall p \in [\![1,n]\!] : \forall b \in [\![p+1,m]\!] : x_{pb} = 0 \text{ (5)}$$

1.2 Relaxation linéaire

La relaxation linéaire de (PE) s'écrit :

(Rl) : $z_{\text{Rl}} = \min \sum_{b=1}^{n} u_b$

s.c.

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

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

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

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 $\forall p \in [\![1,n]\!] : \sum_{b=1}^{m} x_{pb}^* = 1$ 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). En effet on est dans un problème de min donc cette contrainte est limitante, par propriété du simplexe elle sera vérifié a l'égalité dans la solution optimale (car elles sont toutes indépendantes). La valeur cette solution est alors $\sum_{b=1}^{m} \frac{\sum_{p=1}^{n} s_p x_{pb}^*}{c} = \frac{\sum_{p=1}^{n} s_p}{c}$ par la contrainte (2).

□

2 Relaxations lagrangiennes possibles

2.1 Dualisation de la contrainte 1

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

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

s.c.

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

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

$RL_{1'}(u) : \ z_{1'}(u) = \ \min \ \sum_{b=1}^{m} u_b - \mu_b(c - \sum_{p=1}^{n} s_p x_{pb})$

s.c.

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

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

En écrivant la fonction objectif comme $z_{1'}(u) = \min \ \sum_{b=1}^{m}(u_b + \sum_{p=1}^{n}(\mu_b s_p x_{pb})) - \sum_{b=1}^{m} \mu_b c$. 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'associations 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.

Alors comme on l'a fait en cours on peut dualiser la contrainte (2) pour obtenir :

$RL_{1',2}(u) : \ z_{1',2}(u) = \ \min \ \sum_{b=1}^{m}(u_b + \sum_{p=1}^{n}(\mu_b s_p x_{pb})) - \sum_{b=1}^{m} \mu_b c + \sum_{p=1}^{n} \gamma_p(1 - \sum_{b=1}^{m} x_{pb}$

s.c.

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

Où γ est de signe libre car la contrainte associé est une contrainte d'égalité.
Comme précédemment on peut reformuler la fonction objectif comme :

$z_{1',2}(u) = \min \ \sum_{b=1}^{m}(u_b + \sum_{p=1}^{n}((\mu_b s_p - \gamma_p)x_{pb})) - \sum_{b=1}^{m} \mu_b c + \sum_{p=1}^{n} \gamma_p$

En pratique on remarque qu'il suffit de résoudre m problème indépendants :

$PI_b(u) : \ z'_b(u) = \ \min \ \sum_{b=1}^{m}(u_b + \sum_{p=1}^{n}(\mu_b s_p x_{pb}))$

s.c.

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

Alors on sait comment calculer la solution optimale de chacun d'entre eux :

1 - $u_b = 1 \Leftrightarrow \sum_{p \in [\![1,n]\!] : \mu_b s_p - \gamma_p < 0} (\mu_b s_p - \gamma_p) < -1$

2 - $\forall p \in [\![1,n]\!] : (\mu_b s_p - \gamma_p < 0) \Rightarrow x_{pb} = u_b$

3 - $\forall p \in [\![1, n]\!] : (\mu_b s_p - \gamma_p \geqslant 0) \Rightarrow x_{pb} = 0$

De plus on peut remarquer que la contrainte anti-symétrie (5) ne nuit pas au lagrangien : les sous-problèmes reste indépendants et la relaxation conserve sa liberté. Cela permet d'améliorer la vitesse de convergence de l'algorithme de sous-gradient ainsi que la vitesse de résolution d'une partie des sous-problèmes.

2.2 Dualisation de la contrainte 2

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

On obtient alors :

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

s.c.

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

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

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 $z_2(u) = \min \sum_{b=1}^{m} \left(u_b - \sum_{p=1}^{n} \mu_p x_{pb} \right) + \sum_{p=1}^{n} \mu_p$.

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 m 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 m et en ajoutant la somme des μ.

3 Algorithme de sous-gradient

Pour les 2 relaxations nous avons implémenté un algorithme de sous-gradient avec une longueur de pas dépendant de la satisfaction des contraintes.

Algorithm 1 algorithme de sous-gradient

```
1: function RelaxLagrange(instance)
2:     initialisation
3:     première résolution des sous-problèmes
4:     calcul du score et de la première valeur du pas
5:     while ¬ Critères d'arrêts do
6:         calcul des nouveaux multiplicateur
7:         résolution des sous-problèmes
8:         mise à jours de la meilleure solution
9:         calcul de la taille du pas
10:    end while
11:    return best
12: end function
```

- L'initialisation comprend la mise à 0 des variables de calcul ainsi que la fixation des paramètres, le calcul de borne et l'affectation des valeurs initiale des coefficients lagrangiens
- Toutes les résolutions se font selon des procédures spécifiques à la relaxation choisie, détaillé dans les paragraphes correspondant, de même pour le calcul de la valeur de la fonction objectif
- Le calcul du pas se fait en 2 partie :

  1 - Le calcul du pois de ce pas $\nu = \epsilon.\frac{\omega - val}{||d - Dx(\mu)||^2}$ où ω est un majorant de la valeur de la relaxation lagrangienne (la valeur de best-fit est utilisée en pratique), val est la valeur courante de la fonction objectif dualisé, et $Dx\{\leqslant, =, \geqslant\}d$ sont les contraintes qui ont été dualisé ($x(\mu)$ est la valeur optimale de x dans les sous-problème avec les coef μ).

  2 - Ensuite le pas est $\mu = max(\mu + \nu(d - Dx(\mu)), 0)$

  ϵ est un facteur qui va déterminer la taille du pas, au départ il est fixé à une valeur intermédiaire puis vas être diminué d'un facteur ρ si aucune amélioration n'a été effectué en t_{max} itérations, avec une ré-augmentation si sa valeur devient vraiment trop faible. De plus, dans le cadre de la première relaxation, on utilise deux ν et ϵ, un pour chaque type de contraintes. En effet leur nature étant profondément différente les même valeurs ne leurs conviennent pas. Enfin il faut noter que le signe de γ est libre donc sa mise a jour ne comprend pas de max.

- On dispose de plusieurs critère d'arrêt, l'activation de l'un d'entre eux suffit :
  - $\nu < \nu_{min} \wedge val > om\bar{e}ga - 1$ où ν_{min} est un seuil sur ν et $\bar{\omega}$ la valeur de la relaxation linéaire. Cette formule signifie que l'algorithme a convergé vers une valeur au moins aussi bonne que la relaxation linéaire (la valeur de la relaxation lagrangienne est sensé être au moins aussi bonne).
  - $\omega - val < 1$ qui signifie que la solution construite est optimale
  - $\nu = 0$ qui signifie que les contraintes dualisé sont toutes vérifié
  - no_improve $\geqslant$ max_no_improve qui limite le temps de calcul maximal

4 Python

Pour exécuter le code : python3 bin_packing.py