

# Recherche rapide d'arbres de Steiner

Eric Bourreau  
Rodolphe Giroudeau

David Aubert  
Deguilhem Julien  
Université de Montpellier II

20 jan 2015

## 1. Résolution de programmes linéaires

---

### 1.1. Génération de colonnes

#### 1.1.1. Les outils

La génération de colonnes est une technique utilisée pour résoudre un programme linéaire en nombre réels, lorsque les variables sont trop nombreuses pour être toutes énumérées ou pour une résolution directe par un solveur. La génération de colonnes utilise en autres, la solution duale et les coûts réduits. [2]

##### 1.1.1.a. Dualité

Pour un problème linéaire appelé *Primal*, le Dual est la "transposée" du *Primal*: les variables du Primal deviennent les constantes du Dual et vice et versa.

Le problème de maximisation du primal est le problème de minimisation du dual et réciproquement.

Exemple :

Problème Primal :

$$\begin{aligned} & \text{Min} -x_1 + 4x_2 \\ & \begin{cases} -x_1 \geq -7 \\ 3x_1 - 2x_2 \geq 4 \\ x_1 + x_2 \geq 5 \\ x_1 \geq 0, x_2 \text{ libre} \end{cases} \end{aligned}$$

Problème Dual :

$$\begin{aligned} & \text{Max} -7u_1 + 4u_2 + 5u_3 \\ & \begin{cases} -u_1 + 3u_2 + u_3 \leq -1 \\ -2u_2 + u_3 = 4 \\ u_i \geq 0 \end{cases} \end{aligned}$$

##### 1.1.1.b. Coûts réduits

Pour résoudre un PL, donné par une matrice de contrainte A il faut une base B. Tel que pour un problème de minimisation il y ait:

Minimiser  $cx$  sous

$$Ax = d$$

$$x \geq 0$$

On réécrit ce problème avec la base B:

Minimiser  $c_B \cdot x_B + c_N \cdot x_N = d$  sous

$$B \cdot x_B + N \cdot x_N = d$$

$$x_B \geq 0, x_N \geq 0$$

Où  $N$  est la matrice qui complète  $B$  pour reformer  $A$ ,  $x_B$  les variables en base ( Les colonnes choisis dans  $A$  pour former  $B$ ) et  $x_N$  les variables hors-base.

Ici, il est possible de transformer les contraintes d'inégalités en égalités avec l'ajout de variables d'écarts. On les ajoute négativement pour une inégalité "superieur à" et inversement.

Pour chaque variables d'écarts, il y a un coût:  $c_j$  (Coût associé à la variable d'écart  $j$ ). Son coût réduit est l'écart qu'il y a entre le coût associé à la variable hors-base  $x_j$  et les variables duales en base pour la contrainte  $j$ .

$$c_j = c_j - c_B \cdot B^{-1} \cdot A_j$$

Calculer le coût réduit revient à donner "de combien" peut s'améliorer la fonction objective si on relâche la contrainte  $j$  d'une unité.

Exemple :

En suivant l'exemple précédent, on y ajoute les variables d'écart  $x_3$   $x_4$  et  $x_5$  :

Problème Primal	solution	solution duale	coûts réduits
$Min -x_1 + 4x_2$	$x_1 = 7$	$u_1 = 5$	$c_3 = 5$
$-x_1 - x_3 \geq -7$	$x_2 = -2$	$u_2 = 0$	$c_4 = 0$
$3x_1 - 2x_2 - x - 4 \geq 4$	$x_4 = 21$	$u_3 = 4$	$c_5 = 4$
$x_1 + x_2 - x_5 \geq 5$			

$x_1, x_2$  et  $x_4$  sont les variables en base.

### 1.1.2. La méthode

La génération de colonnes fonctionne sur un problème en nombre réels (Pas de PLNE). Elle est utilisée quand le PL possède un très grand nombre de variable. L'idée étant de résoudre le problème maître en ne prenant en compte qu'un nombre restreint et suffisant de variables.

On commence par générer un PL à partir du problème maître, qu'on va appeler problème maître restreint (PMR). Le choix de ce problème doit se faire de manière heuristique pour optimiser le développement de l'algorithme. Puis on va générer des variables améliorantes susceptibles d'améliorer le résultat.

Pour les trouver, on va s'intéresser au problème dual et on va regarder les coûts réduits. Dans un problème de maximisation, la solution optimale s'obtient s'il sont tous

négatifs (Et inversement). Si ce n'est pas le cas alors on génère de nouvelle(s) variable(s) à partir de cette solution pour obtenir un nouveau PMR et recommencer.

## 1.2. Branch and Price

### 1.2.1. Principe général

Un algorithme de branch and Price est l'équivalent d'un algorithme de Branch and Bound et de génération de colonnes.

La différence se fait par incrémentation. A chaque noeud, une résolution de PL. De ce fait on ne garantit pas à chaque noeud une solution optimal ou même valide.

La première étape d'un branch and price est de générer ce qui s'appelle le **Master Problem**.

De ce **Master Problem** devront être générés des **sous-problèmes**.

Chaque **sous-problèmes** sera résolu et à chaque itération, une résolution de PL sera effectué et chaque solution entrainera une génération de colonne et sera rajouté à la solution.

Lorsque tout les **sous-problèmes** générés admettent une solution entièrement négative c'est la fin de l'algorithme.

### 1.2.2. exemple

<b>Job(<i>i</i>)</b>			
	1	2	3
Machine <i>i</i> = 1	5	7	3
Machine <i>i</i> = 2	2	10	5

<i>i</i>	<i>C<sub>i</sub></i>
1	5
2	8

<b>Job(<i>i</i>)</b>			
	1	2	3
Machine <i>i</i> = 1	3	5	2
Machine <i>i</i> = 2	4	3	4

Soit  $k_1$  et  $k_2$  l'ensemble des solutions possibles pour les machines 1 et 2 tel que  
 $k_1 = (1,0,0), (0,1,0), (0,0,1), (1,0,1)$   
 $k_2 = (1,0,0), (0,1,0), (0,0,1), (1,1,0), (0,1,1)$

Le **Master Problem** pourra être représenté comme ci-suit:  
 $\max z = 5y_1^1 + 7y_1^2 + 3y_1^3 + 8y_1^4 + 2y_2^1 + 10y_2^2 + 5y_2^3 + 12y_2^4 + 15y_2^5$

Voici une décomposition par job:

$$\begin{aligned} y_1^1 + 0 + 0 + y_1^4 + 0 + 0 + y_2^4 + 0 &= 1 \ (u_1) \\ 0 + y_1^2 + 0 + 0 + 0 + y_2^2 + 0 + y_2^4 + y_2^5 &= 1 \ (u_2) \\ 0 + 0 + y_1^3 + y_1^4 + 0 + 0 + y_2^3 + 0 + y_2^5 &= 1 \ (u_3) \end{aligned}$$

Ici au lieu d'exprimer les **Machines** par **Job**, est exprimé les **Job** par **Machines**.  
 C'est le Dual.

$$\begin{aligned} y_1^1 + y_1^2 + y_1^3 + y_1^4 &\leq 1 \ (v_1) \\ y_2^1 + y_2^2 + y_2^3 + y_2^4 &\leq 1 \ (v_2) \end{aligned}$$

Ici ce sont les contraintes la somme doit être inférieure à 1. (  $\sum_{i=1}^5 P(i,j)$  )

## References

---

- [1] Dominique Feillet, *Résolution de problèmes de tournées par Branch and Price*. Laboratoire d'informatique, Avignon, Mars 2008.
- [2] Hélène Toussaint, *Introduction au Branch cut and price et aux solveurs SCIP (Solving constraint integer programs)*. Rapport de recherche LIMOS /RR-13-07, 19 avril 2013.
- [3] Mads Kemhet Jepsen, *Column generation and Branch and Price*.
- [4] Mohan Akella, Sharad Gupta, Avijit Sarkar, *Branch and Price column generation for solving huge integer programs*. Buffalo University (SUNY),