La programmation mathématique en nombres entiers

Reformulation et Génération de colonnes

Stanislas Francfort

Orange Labs

5 Juin 2016

Reformulation

- Lorsque l'on change l'espace des variables d'un problème, c'est une *reformulation*
- Une reformulation peut mener à un PLNE avec un nombre exponentiel de variables
- Un algorithme de *génération de colonnes* peut être capable de gérer cela

Comment obtenir une bonne reformulation ?

- Qu'est-ce qu'une bonne reformulation ? ?
  - C'est une reformulation qui peut permettre des branchements efficaces
  - ou des bonnes valeurs de relaxation
- Il faut pouvoir résoudre une telle reformulation...
- Formulation non compacte
- $\Rightarrow$ les solveurs PLNE ne peuvent pas directement les résoudre (idem nombre exponentiel de contraintes).

Comment obtenir une bonne reformulation ?

- Plusieurs techniques de reformulations :
  - décomposition de Dantzig-Wolfe
  - découvrir soit même un espace nouveau de variables !
- Pour évaluer une formulation :
  - la valeur de relaxation : plus elle est proche de l'optimum entier, plus le branchement sera bon
  - éviter les symétries : plus il y a de symétries, plus le branchement va devoir explorer de solutions.

Un exemple de reformulation : le problème de découpes

- *Problème de découpe optimale de barres de fer*
- en anglais *cutting stock problem*
- Une entreprise possède une quantité importante $\{1, \ldots, N\}$ de barres de fer de taille identique b qui ont été produites en série
- Sur une période donnée, l'entreprise reçoit n commandes $\{1, \ldots, n\}$
- chaque commande $i \in \{1, \ldots, n\}$ correspond à la demande de n_i barres de fer de la même longueur $b_i \leq b$
- On suppose que chaque commande correspond à une taille distincte des autres commandes
- L'entreprise désire minimiser le nombre de barres initiales qui doivent étre découpées pour produire les barres demandées.

Première formulation

- Première formulation linéaire utilisant des variables x_i^j correspondant au nombre de barres de tailles $b_i, i \in \{1, \ldots, n\}$
- elles seront découpées dans la barre $j \in \{1, \ldots, N\}$
- On utilise également des variables z_j qui valent 1 s'il y a une découpe réalisée dans la barre $j \in \{1, \ldots, N\}$.

Première formulation

$$Min \sum_{j=1}^{N} z_j$$

$$\sum_{j=1}^{N} x_i^j = n_i, \ \forall i \in \{1, \ldots, n\}$$

$$\sum_{i=1}^{n} b_i x_i^j \leq b z_j, \ \forall j \in \{1, \ldots, N\}$$

$$x_i^j \in \{0, 1\}, \ \forall i \in \{1, \ldots, n\}, \ \forall j \in \{1, \ldots, N\}$$

$$z_j \in \{0, 1\}, \ \forall j \in \{1, \ldots, N\}$$

Première formulation

- Cette formulation contient un grand nombre de solutions symétriques
- car chaque barre est indifférenciée, ce qui implique qu'une même solution du problème correspond à plusieurs solution du PLNE
- $\Rightarrow$ Ajouter des inégalités
  $\sum_{i=1}^{n} x_i^1 \geq \sum_{i=1}^{n} x_i^2 \geq \cdots \geq \sum_{i=1}^{n} x_i^N$
- i.e. la barre j est utilisée seulement si la barre $j-1$ l'est
- techniques limitées : on préfère une reformulation en changeant l'espace des variables.

Seconde formulation ; nombre exponentiel de variables

- On appelle *découpe* d d'une barre de longueur b un ensemble de longueurs non-nulles $\{l_1, l_2, \ldots, l_k\}$
- telles que $\sum_{j=1}^{k} l_k = b$
- On note alors $dec(d) = \{l_1, l_2, \ldots, l_k\}$
- l'ensemble D : les découpes possibles d'une barre de longueur b
- variables t_d : nombre de barres que l'on va découper selon la découpe $d \in D$

Seconde formulation contenant un nombre exponentiel de variables

- La formulation suivante est une formulation du problème de découpe :

$$Min \sum_{d \in D} t_d$$

$$\sum_{d \in D; b_i \in dec(d)} t_d \geq n_i, \ \forall i \in \{1, \ldots, n\}$$

$$t_d \in \mathbb{N} \ \forall d \in D$$

Seconde formulation contenant un nombre exponentiel de variables

- Si on a une solution du problème :
- c'est-a-dire un ensemble de barres où les commandes peuvent être découpées à l'intérieur
- on peut facilement compter le nombre de barres initiales découpées selon la même découpe
- et on construit alors une solution du programme en fixant les variables correspondantes à ces valeurs
- Cette solution vérifie alors naturellement les contraintes.

Seconde formulation contenant un nombre exponentiel de variables

- Inversement, les variables donnent directement le nombre de barres qui doivent être découpés selon une découpe donnée
- Par les contraintes du programme, cette solution est telle que l'ensemble des barres obtenues contient bien les barres demandées.

Algorithme de génération de colonnes

- Le dual de la relaxation linéaire $\tilde{F}$ de F est le PL suivant :

$$Max \sum_{i=1}^{n} n_i \lambda_i$$

$$\sum_{i \in dec(d)} \lambda_i \leq 1, \ \forall d \in D$$

$$\lambda_i \geq 0, \ \forall i \in \{1, \ldots, n\}$$

Algorithme de génération de colonnes

- On est capable produire une solution initiale
- c'est-à-dire un ensemble de découpes D_0 dans lequel on peut découper toutes les commande
- Soit la solution optimale t_0 de $\tilde{F}$ limitée aux variables correspondant a D_0
- Et soit λ_0 la solution optimale du dual limité aux contraintes correspondant a D_0
- On pose également la solution $t^\star$ de $\tilde{F}$ obtenue en fixant a 0 toutes les colonnes n'apparaissant pas dans D_0.

Algorithme de génération de colonnes

- Avec ces définitions, on obtient alors une condition d'optimalité :

Theorem

par le principe de la dualité forte, le couple de solution $t^\star$ et λ_0 est optimal pour F et son dual
si et seulement si
λ_0 vérifie les contraintes du dual, c'est-a-dire si

$$\sum_{i \in dec(d)} \lambda_i^0 \leq 1, \ \forall d \in D \backslash \{D_0\}$$

Algorithme de génération de colonnes

- Si la condition est vérifiée
- Alors on a pu résoudre le PL primal
- Sinon, si on ajoutait des variables bien choisie dans le primal, cela permettrait de vérifier la condition !
- On appelle le fait d'ajouter des variables *la génération d'une colonne*.

Algorithme de génération de colonnes

- Le problème de génération d'une colonne pour le programme $\tilde{F}$ est-il utile pour répondre à la condition ?
- i.e. est-ce que toutes les contraintes données à la question précédente sont satisfaites par λ_0
- cette question revient a résoudre le PLNE avec des variables entières y_i associées à la quantité de barres de longueurs b_i que l'on veut découper dans la barre correspond à la nouvelle variable

$$Max \sum_{i=1}^{n} \lambda_i^0 y_i$$

$$\sum_{i=1}^{n} b_i y_i \leq b$$

$$y_i \in \mathbb{N}, \ \forall i \in \{1, \ldots, n\}$$

Algorithme de génération de colonnes

- Le PLNE précédent est le problème du sac-a-dos binaire qui est NP-difficile...
- Mais qu'on sait assez bien résoudre en pratique.
- $\Rightarrow$ Algorithme de génération de colonnes :
  - on peut déterminer si la condition est vérifiée
  - et sinon on sait ajouter une variable
  - On sait que si l'on ajoute toutes les variables possibles, alors on obtiendrait à la fin le PL avec toutes ses variables
  - donc l'algorithme converge en théorie vers l'optimum.

Algorithme de génération de colonnes

- Cette génération de colonnes : ajout d'un nombre exponentiel de variables dans le pire des cas ...
- Mais il s'agit en fait exactement de l'algorithme du simplexe
- (ajout de variables en fonction des coûts réduits)
- On sait que l'algorithme du simplexe ne nécessite que rarement un grand nombre d'ajouts de variables
- comme pour l'algorithme du simplexe, en pratique il est rarement nécessaire d'ajouter toutes les variables dans un alorithme de génération de colonnes
- Mais il est parfois difficile de faire converger l'algorithme (cas de dégénérescence des itérations de l'algorithme du simplexe).

Algorithme de génération de colonnes et branchement

- On a vu un algorithme de génération de colonnes
- pour résoudre un PL possédant un nombre exponentiel de variables
- Cette technique est facilement reproductible pour d'autres problèmes, avec succès.
- Dans le cadre d'un PLNE, il est nécessaire de coupler cette technique avec un algorithme de branchements
- ces algorithmes dit de *génération de colonnes et branchements* ou Branch-and-Price sont de plus en plus utilisés
- même si les difficultés de convergence citées plus haut restent un écueil fréquent.

le problème de multiflot continu

- Problème du multiflot (multicommodity flow)
- Soit $G = (V, A)$ un graphe orienté
- où l'on associe une capacité $c(a) \in \mathbb{N},\ \forall a \in A$
- Soit le commodités, c'est-a-dire des paires de sommets $(s_1, t_1), \ldots, (s_k, t_k)$.
- Pour chaque paire $i \in \{1, \ldots, k\}$, on désire trouver des (s_i, t_i)-flots entiers
- tels que la somme des capacités utilisés par les flots sur un même arc respecte la capacité d'un arc.
- L'objectif est de maximiser la quantité total de flot circulant sur le graphe.
- Trouver une formulation PLNE ?

le problème de multiflot continu

- Une formulation PLNE :
- utiliser des variables $x_i(a)$ indiquant la quantité de flot passant par l'arc a pour la commodité i
- lorsque $k \geq 2$, cette formulation n'est pas TU
- et sa résolution n'est pas efficace vue le nombre de variables.
- On cherche une autre formulation

le problème de multiflot continu

- Une reformulation peut être facilement obtenue en posant d'autres variables
- Soit $\mathcal{P}_{st}$ l'ensemble des chemins (élémentaires)
- allant du sommet s au sommet t
- Posons $\mathcal{P} = \cup_{i=1}^{k} \mathcal{P}_{s_i t_i}$
- Et pour tout arc a : $\mathcal{P}_a = \{\mu \in \mathcal{P} | a \in \mu\}$.
- on associe la variable f_μ associée à chaque chemin $\mathcal{P}_{s_i t_i}$, pour tout $i \in \{1, \ldots, k\}$

le problème de multiflot continu

- Cela donne la formulation PL suivante (MCF) :

$$Max \sum_{\mu \in \mathcal{P}} f_\mu$$
$$\sum_{\mu \in \mathcal{P}_a} f_\mu \leq c(a), \ \forall a \in A$$
$$f_\mu \geq 0, \ \forall \mu \in \mathcal{P}$$

- Cette formulation PL contient un nombre exponentiel de variables
- mais un nombre très petit de contraintes.
- Pour déterminer un cadre de génération de colonnes, il faut donc pouvoir déterminer un nombre réduit de variables.

le problème de multiflot continu

- Posons $\mathcal{P}'$ un sous-ensemble de chemin de $\mathcal{P}$ tel que le programme suivant ait une solution réalisable $(f'_\mu)_{\mu\in\mathcal{P}'}$ (MCF') :

$$
\begin{aligned}
Max \quad & \sum_{\mu\in\mathcal{P}'} f_\mu \\
& \sum_{\mu\in\mathcal{P}'_a} f_\mu \leq c(a),\ \forall a \in A \\
& f_\mu \geq 0,\ \forall \mu \in \mathcal{P}'
\end{aligned}
$$

- On a donc la solution $(f_\mu)_{\mu\in\mathcal{P}}$
- obtenu depuis f' en ajoutant des variables nulles pour tous les chemins absents de $\mathcal{P}'$
- elle est alors solution de (MCF)
- Donc la solution de (MCF') est inférieure a celle de (MCF)

le problème de multiflot continu

- On veut savoir si f obtenue depuis f' est optimale pour (MCF)
- Utilisons le dual $(D - MCF)$ de (MCF)
- On pose pour cela λ_a les variables duales associées aux contraintes, c'est à dire les arcs de A
- soit $(D - MCF')$, le dual de (MCF') :

$$Min \sum_{a \in A} c(a)\lambda_a$$

$$\sum_{a \in \mu} \lambda_a \geq 1, \ \forall \mu \in \mathcal{P}'$$

$$\lambda_a \geq 0, \ \forall a \in A$$

le problème de multiflot continu

- Dans le cas du dual, la solution λ' optimale de $(D - MCF')$ est solution de $(D - MCF)$
- si et seulement si λ' vérifie toutes les contraintes de $(D - MCF)$
- or il y en a une par chemin de $\mathcal{P}$, c'est-a-dire si

$$\sum_{a \in \mu} \lambda_a \geq 1, \ \forall \mu \in \mathcal{P}$$

- Dans ce cas, λ' est solution de $(D - MCF)$
- et donc solution optimal.
- Ainsi la solution f obtenue depuis f' correspond à la même valeur objective que la solution duale
- par le corollaire de la dualité forte, cela implique que f est optimale pour (MCF)

Génération de colonnes

- Cette logique induit un principe de tests polynomials pour l'optimalité
- s'il n'existe pas contrainte du dual (correspondant a un chemin) qui soit violée par λ',
- alors f est optimal,
- sinon on connait un chemin correspondant à une contrainte violée
- et on peut l'ajouter dans (MCF)
- c'est le problème de *génération d'une colonne* (ou *pricing*).

Génération de colonnes

- Réitération du principe :
- Méthode de génération de colonnes (duale de la génération de contraintes)

Theorem

Une méthode de génération de colonnes est polynomiale si et seulement si l'algorithme de génération d'une colonne est polynomial.

Génération de colonnes

- L'étape d'ajout de colonnes n'est pas simple en pratique
- La convergence de cette méthode est similaire à l'ajout de colonnes dans le simplexe
- au pire des cas on a dégénérescence et convergence très lente
- $\Rightarrow$ On doit utiliser des procédés assez complexes pour obtenir une convergence rapide