L'algorithme de la direction améliorante pour la résolution de programmes linéaires

Adrien Chan-Hon-Tong

L'algorithme de la direction améliorante pour la résolution de programmes linéaires

Adrien CHAN-HON-TONG
ONERA
adrienchanhonton@gmail.com

12 septembre 2018

1 Introduction

La résolution des programmes linéaires c'est à dire des problèmes de la forme

$$\max_{x \in \mathbb{Q}^D / Ax \geq b} (cx)$$

avec N, D deux entiers, $A \in M_{N,D}(\mathbb{Q})$, $b \in \mathbb{Q}^N$ et $c \in \mathbb{Q}^D$ est un problème très étudié dans la littérature. Il est connu que l'algorithme [4] permet de résoudre ce problème en un nombre d'opérations élémentaires dans $\mathbb{Q}$ ($+ \ - \ \times \ \backslash$ reading writting test) polynomial en N, D et L la taille nécessaire pour écrire les données d'entrées.

On propose ici un algorithme qui s'attache à la nature combinatoire du problème. Cet algorithm pourrait être intéressant dans la recherche d'un algorithme fortement polynomial c'est à dire résolvant ce problème en un nombre d'opérations élémentaires dans $\mathbb{Q}$ polynomial en N et D seulement.

2 État de l'art

La résolution des programmes linéaires est très étudiée :

- l'état de l'art reste aujourd'hui les méthodes type point intérieur (polynomiale en L, N, D [4] mais non polynomiale en N, D [1])
- il existe cependant des algorithmes polynomiaux en N, D
  - quand la matrice A ne contient que des 1 ou des -1 [6]
  - quand la matrice A ne contient pas plus de 2 valeurs non nulle par ligne [3]
  - quand la solution est connue binaire [2]
  - quand le problème peut être reformulé $\exists ? x \in \mathbb{Q}^D / Ax = 0, x \geq 1$ [5] (précisément $\exists ? x \in \mathbb{Q}^D / Ax = 0, x > 0$ dans [5] mais l'équivalence est triviale)

Cet article propose un algorithme de type point intérieur mais qui reste attacher à la nature combinatoire du problème. Il pourrait être intéressant dans la recherche d'un algorithme fortement polynomial.

3 Algorithme

3.1 Notation

Soit N, D deux entiers, $A \in M_{N,D}(\mathbb{Q})$, $b \in \mathbb{Q}^N$ et $c \in \mathbb{Q}^D$, on note :

$$K = \{x \in \mathbb{Q}^D | Ax \geq b, c \geq 0\}$$

$$\overset{\circ}{K} = \{x \in \mathbb{Q}^D | Ax > b, c \geq 0\}$$

$$K^* = \{x \in \mathbb{Q}^D | Ax \geq b, cx = 0\}$$

$$k(x) = \max_{\lambda \in \mathbb{Q}/x - \lambda c \in K} \lambda$$

$$K_n = \{x \in \mathbb{Q}^D / A_n^T x = b_n\}$$

proj désigne la projection (projection d'un vecteur sur un sous espace vectoriel).

$$e(x) = \min_{n \in \{1,...,N\}} ||x - \text{proj}(x, K_n)||$$

$$E(x) = \{n \in \{1, ..., N\} | A_n^T x - b_n = e(x)\}$$

$$O_0(x) = \{v \in \mathbb{Q}^N | \forall n \in E(x),\ A_n^T v = 0\}$$

$$O(x) = \{v \in \mathbb{Q}^N | \exists \alpha \in \mathbb{Q} | \forall n \in E(x),\ A_n^T v = \alpha\}$$

Notons $S(x)$ la matrice contenant la ligne $-c$ et les lignes A_n pour $n \in E(x)$.
Ensuite, notons $p(x, v)$ est le point résultant du déplacement maximal dans K selon v depuis x qui garde $E(x)$ constant.
Enfin, notons F l'algorithme : pour toute matrice $\Gamma \in M_{N',D'}(\mathbb{Q})$,

- $F(\Gamma) \in \mathbb{Q}^{D'}$
- $\Gamma F(\Gamma) \geq \mathbf{1}$
- ou un tel vecteur n'existe pas et le comportement peut être abérant

3.2 Precondition

L'algorithme proposé résoud le problème :

$$\min_{x \in \mathbb{Q}^D / Ax \geq b} (cx)$$

sous les préconditions :

- $K^* \neq \emptyset$
- $Ac > \mathbf{0}$
- $\forall 1 \leq i < j \leq N, A_i^T c \neq A_j^T c$

3.3 pseudo code

si $x - k(x)c \in K^*$
    retourner $x - k(x)c$
$v \leftarrow \text{proj}(-c, O(x))$
si $v \neq 0$ et $p(x, v) \in K^*$
    retourner $p(x, v)$
$w \leftarrow \text{proj}(-c, O_0(x))$
si $w \neq 0$
    $x \leftarrow p(x, w)$
    continuer
$\omega \leftarrow \text{proj}(A_{E(x)}, O(x) \cap \{v, cv = 0\}))$
si $\omega \neq 0$
    $x \leftarrow p(x, \omega)$
    continuer
si $v = 0$
    $x \leftarrow x - \frac{k(x)}{2}c$
    continuer
si $p(x, v) \in \overset{\circ}{K}$
    $x \leftarrow p(x, v)$
    continuer
$x_s \leftarrow p(x, v)$
$x = p(x_s, F(S(x_s)))$

4 Discussion

4.1 Informellement

L'idée de l'algorithme proposé est que les contraintes actives (celles qui sont les plus proches de x) poussent x sans augmenter $c^T x$ et en maintenant $E(x)$. x va donc s'éloigner des contraintes actives jusqu'à qu'une nouvelle contrainte s'active. Au bout de quelques itérations (au plus N bien moins en pratique), il n'est pas possible de s'éloigner de toute les contraintes en maintenant $E(x)$. Soit, l'orthocentre de ces contraintes est réduit à x, dans ce cas on se déplace un tout petit peu selon $-c$ et on recommance. Soit on peut continuer à se déplacer dans l'orthocentre mais cela rapproche des bords.
Dans ce dernier cas, on considère le point d'impact et on appelle l'algorithme F pour trouver un vecteur de sortie. Et, on recommence.
L'algorithme converge car on ne rencontre jamais 2 fois le même orthocentre. Et, toutes les opérations sont fortement polynomiales (projection, déplacement dans un orthocentre, F).
Ainsi, s'il s'avérait que le nombre d'orthocentre rencontré était polynomial en N, D, l'algorithme proposé serait fortement polynomiales...
Indépendamment, les préconditions de l'algorithme sont génériques.

4.2 Lemme : F est fortement polynomial

ATTENTION : les notations de chaque lemme peuvent ne pas être consistantes avec celles du reste de l'article

4.2.1 reduction

[5] assure l'existance d'un algorithme fortement polynomial G vérifiant $\forall A \in M_{N,D}(\mathbb{Q})$:

— $G(A) \in \mathbb{Q}^D$
— $G(A) \geq \mathbf{1}$
— $AG(A) = \mathbf{0}$
— ou aucun vecteur existe et dans ce cas, un certificat est produit

$\forall A \in M_{N,D}(\mathbb{Q})$, soit $H(A)$:

— $H(A) \in M_{N,2D+N}(\mathbb{Q})$
— $\forall n, d \in \{1, ..., N\} \times \{1, ..., D\}$
  — $(H(A))_{n,d} = A_{n,d}$
  — et $(H(A))_{n,d+N} = -A_{n,d}$
— $\forall n, d \in \{1, ..., N\}$
  — $(H(A))_{n,d} = -1$ si $n = d$
  — et 0 sinon

Alors, on peut construire $F(A)$ à partir de $G(H(A))$ - le comportement n'est pas garanti si un tel vecteur n'existe pas.

4.2.2 démonstration

Supposons qu'il existe $v \in \mathbb{Q}^{2D+N}$ tel que $H(A)v = 0, v \geq 1$. Notons x_+, x_- et y les sous vecteurs de taille D, D et N de v.

On a $H(A)v = Ax_+ - Ax_- - Iy = \mathbf{1}$ et $x_+ \geq \mathbf{1}$, $x_- \geq \mathbf{1}$ et $y \geq \mathbf{1}$. Soit $x = x_+ - x_-$, alors $Ax = Iy = y \geq \mathbf{1}$.

Dans on peut retourner $G(A) = x$.

Inversement, s'il existe w tel que $Aw \geq \mathbf{1}$. Soit w_+ et w_- les vecteurs définis par : $\forall d \in \{1, ..., D\}$,

— si $w_d \geq 0$, $(w_+)_d = w_d + 1$, $(w_-)_d = 1$
— sinon $(w_+)_d = 1$, $(w_-)_d = -w_d + 1$

On a : $w = w_+ - w_-$ et $w_+ \geq 1$, $w_- \geq 1$. Donc, on a $Aw_+ - Aw_- - (Aw) = 0$. Donc la concatenation de w_+, w_-, Aw est une sortie acceptable de $G(H(A))$.

CQFD

4.2.3 Remarque

Ici, on ne démontre que le bon fonctionnement de F que quand il existe un vecteur solution.

Cependant, dans l'algorithme F est appliqué sur des x_s vérifiant $x_s \notin K^*$ et $x_s \notin \overset{\circ}{K}$. Considérons la solution x^* et $x_{haha} = x^* + \lambda c$ tel que $c^T x_s > c^T x_{haha} >$

0. $x_{haha} \in \overset{\circ}{K}$ car $Ac > \mathbf{0}$ et $c^T x_s > c^T x_{haha}$. Donc $F(S(x_s))$ pourrait valoir $x_{haha} - x_s$...

4.3 Lemme : Le double primal dual

ATTENTION : les notations de chaque lemme peuvent ne pas être consistantes avec celles du reste de l'article

4.3.1 Le primal dual

Soit

$$\min_{x \in \mathbb{Q}^N} (cx)$$
$$\forall m \in \{1, ..., M\}, a_m x \geq b_m$$

un problème primal et soit

$$\max_{y \in \mathbb{Q}^M} (c'y)$$
$$\forall n \in \{1, ..., N\}, a'_m y \geq b'_m$$

le dual correspondant c'est à dire que $\forall n \in \{1, ..., N\}$, $b'_n = c_n$, $\forall m \in \{1, ..., M\}$ $c'_m = b_m$ et $\forall (n, m) \in \{1, ..., N\} \times \{1, ..., M\}$, $a'_{n,m} = a_{m,n}$.

Alors, à la fois le primal et le dual sont équivalents au système d'inéquations suivant sur x et y :

$$\forall m \in \{1, ..., M\}, a_m x \geq b_m$$
$$\forall m \in \{1, ..., M'\}, a'_m y \geq b'_m$$
$$cx = c'y$$

Ce dernier problème est équivalent à ce qu'on appelle ici la formulation primal-dual :

$$\min_{x \in \mathbb{Q}^N, y \in \mathbb{Q}^M, z \in \mathbb{Q}} (z)$$
$$\forall m \in \{1, ..., M\}, a_m x + z \geq b_m$$
$$\forall m \in \{1, ..., M'\}, a'_m y + z \geq b'_m$$
$$cx + z \geq c'y$$
$$c'y + z \geq cx$$
$$z \geq 0$$

Ce dernier problème possède un point trivial dans l'intérieur de l'espace admissible $x = y = 0, z > \max\{b_m, b'_m\}$, et possède un minimum. La valeur associée à ce minimun est 0 si et seulement si le problème d'origine est non vide et possède un minimum.

4.3.2 2 fois

Considérons la transformation qui fait passer d'un problème à son primal-dual. Si on l'applique non pas 1 mais 2 fois au problème de départ (c'est à dire si on la réapplique au primal-dual), on obtient une formulation qui possède les 3 propriétés : existence d'un point trivial, existence d'un minimum, le minimum est 0. On appelle cette formulation, la forme double primal dual.

Maintenant, dans cette transformation, on peut remarquer que toutes les contraintes ont un produit scalaire strictement positif avec le vecteur objectif. De plus, z est une variable ajoutée et il est possible de lui associé un coefficient dans chacune des contraintes :

$$\begin{gathered}\min_{x\in\mathbb{Q}^N, y\in\mathbb{Q}^M, z\in\mathbb{Q}}(z)\\ \forall m\in\{1,...,M\}, a_m x+\alpha_m z\geq b_m\\ \forall m\in\{1,...,M'\}, a'_m y+\beta_m z\geq b'_m\\ cx+\gamma z\geq c'y\\ c'y+\gamma' z\geq cx\\ z\geq 0\end{gathered}$$

et ce faisant, il est possible de faire en sorte que le produit scalaire entre le vecteur objectifs et chacune des contraintes soit différents.

4.3.3 initialisation

Comme $Ac > 0$, on peut démarer l'algorithme proposé depuis λc avec λ tel que $\lambda Ac > b$...

4.4 suppression de F

Les simulations numériques effectué jusqu'à présent invite à penser qu'il est possible de remplacer F par un algorithme adhoc regardant pour chaque $n \in M(x)$ si l'algorithme repartant à partir de $x \pm \epsilon A_n$ ne se débloque pas. Mais utiliser F a le mérite de ne pas avoir à se poser de question...

4.5 Terminaison

Actuellement, il manque un argument pour démontrer formellement que le programme termine : le lien entre on ne peut pas augmenter la distance entre x et $k_{M(x)}$ tout en restant dans $\{z, cz = cx\}$ et $\omega = 0$. L'implication est direct, mais on pourrait avoir des orthocentres bizarres (infini tout en ne permettant aucune amélioration).

Plus globalement une démonstration que chaque appel de $p(x, v)$ est bien posé serait nécessaire.

Par contre, l'argument qu'on ne rencontre jamais 2 fois le même orthocentre semble trivial par construction puisqu'on ne quitte un orthocentre que quand on l'a exploité au maximum...

Références

[1] Xavier Allamigeon, Pascal Benchimol, Stéphane Gaubert, and Michael Joswig. Log-barrier interior point methods are not strongly polynomial. *SIAM Journal on Applied Algebra and Geometry*, 2(1) :140–178, 2018.

[2] Sergei Chubanov. A strongly polynomial algorithm for linear systems having a binary solution. *Mathematical programming*, 134(2) :533–570, 2012.

[3] Dorit S Hochbaum and Joseph Naor. Simple and fast algorithms for linear and integer programs with two variables per inequality. *SIAM Journal on Computing*, 23(6) :1179–1192, 1994.

[4] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 302–311. ACM, 1984.

[5] Kees Roos. An improved version of chubanov's method for solving a homogeneous feasibility problem. *Optimization Methods and Software*, 33(1) :26–44, 2018.

[6] Eva Tardos. A strongly polynomial algorithm to solve combinatorial linear programs. *Operations Research*, 34(2) :250–256, 1986.