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
CEA,LIST,DIASI,LVIC
adrienchanhonton@gmail.com

4 septembre 2013

1 Introduction

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

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

avec a_m, et c des vecteurs $\mathbf{Q}^N$ et b_m des scalaires, est un problème très étudié dans la littérature. Ce problème est un problème polynomial [1]. Mais, les meilleurs bornes polynomiales connues dépendent de la taille L nécessaire pour écrire les données et non pas seulement de $\max(N, M)$. Dit autrement, il n'existe aujourd'hui aucun algorithme déterministe connu ayant seulement besoin pour produire la solution d'un nombre d'opérations dans $\mathbb{Q}$ polynomial en M et N.

Il existe cependant des méthodes dont les complexités temporelles ne dépendent que du nombre de contraintes M et du nombre de variables N. Par exemple, la méthode du simplexe [2], très efficace en pratique, qui explore l'ensemble fini des sommets de l'espace des solutions, n'est pas sensible à L. Mais cette méthode n'est pas polynomiale [3] notamment parce que le nombre de sommets augmente exponentiellement avec la dimension de l'espace.

La question de l'existence d'un algorithme possédant ces deux avantages (caractère polynomial et indépendance vis à vis de L) est une question ouverte à ce jour, et, c'est dans ce contexte que s'inscrit cet article. Nous proposons un algorithme dont on conjecture qu'il vérifie les propriétés souhaitées.

2 État de l'art

De nombreuses approches ont été explorées pour tenter de résoudre de façon fortement polynomiale la programmation linéaire. Parmi ces approches,

on trouve :

- la recherche d'une règle de pivotage efficace pour le simplex [4]
- la recherche de solution pour des programmes linéaires ayant certaines spécificités comme l'existence d'une solution binaire [5] ou un nombre de variables par contraintes inférieur à 3 [6]
- la recherche de solution dans le cas où le nombre de caractères nécessaire pour écrire les a est borné [7]
- la recherche de critères d'améliorations pour les approches de type points intérieurs [8] dont on pourra trouver une forme de synthèse dans [9]. Parmi ces approches, certaine propose de chercher la solution pour des formes spécifiques mais néanmoins *complètes* des programmes linéaires [10]. C'est dans ce contexte que cet article se positionne.

Les approches de type points intérieurs se schématise ainsi : à partir d'un point admissible, on cherche à se déplacer étape par étape dans l'ensemble des points admissibles vers un point optimal. Ce type d'approche suppose d'être capable de trouver à chaque étape un déplacement élémentaire admissible et de pouvoir montrer que le nombre d'étapes est borné soit par une borne polynomiale [1] voire par une borne fortement polynomiale. La contribution de cet article est

- de proposer de se déplacer selon deux types de mouvements : des mouvements dans l'orthocentre des contraintes les plus proches du point courant et des sauts selon c
- appliquer au *double primal-dual*

3 Algorithme de la direction améliorante

3.1 Le double primal-dual

Pour tous programmes linéaires, on peut construire son *double primal-dual* équivalent. Ces deux problèmes sont de tailles équivalentes : si T et T' sont respectivement les maximums entre le nombres de contraintes et le nombre de variable pour respectivement le problème initial et le *double primal-dual*, on a $T' \leq 4T + 8$.

Les intérêt du *double primal-dual* sont multiples. On peut notamment remarquer

- que l'intérieur de l'ensemble des points admissible est non nul
- qu'on peut construire un de ces points trivialement
- que l'on sait que le problème a bien un minimum
- que l'on sait que ce minimum est 0
- que de tous points x, $x + \mu c$ est admissible pour $\mu \geq 0$

Il suffit pour le construire d'appliquer le primal-dual 2 fois. Soit

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

un problème primal et soit

$$\max_{y \in \mathbb{R}^M} (c'y)$$
$$\forall m \in \{1, ..., M\}, a'_m y \leq b'_m$$

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

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

Ce dernier problème est équivalent au primal-dual

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

Le primal-dual a un intérieur non vide dont on sait trouver un élément simplement : pour $z = 1 + \max\{(b_m, m \in \{1, ..., M\}\}$ et $x = y = 0$ on est dans l'intérieur des points admissibles. Mais, ce problème possède un point admissible x, y, z avec $z = 0$ seulement si le primal est faisable et borné.

Cependant, le fait de prendre à nouveau le primal-dual du premier primal-dual fournit un programme linéaire possédant les propriétés voulues.

3.2 Calcul dans $\mathbb{Q}$

Par soucis de simplification, l'algorithme présenté utilise naïvement des opérations dans $\mathbb{R}$ et non dans $\mathbb{Q}$. Typiquement, l'entrée de l'algorithme est constitué de vecteurs unitaires. Cela ne restreint pas la généralité dans $\mathbb{R}$ car de tous vecteurs non nul, on peut former le vecteur unitaire correspondant par la division par la racine de la norme. Mais ce vecteur unitaire peut ne pas être dans $\mathbb{Q}$.

Néanmoins, cet algorithme semble pouvoir être modifié de sorte que l'ensemble des tests n'utilisent que des calculs dans $\mathbb{Q}$ notamment en comparant les distances au carrées plutôt que les distances. Typiquement, bien que naïvement le calcul de la distance entre un point et un plan fasse intervenir une racine carré, le calcul de la projection du point sur le plan ne fait par contre intervenir que des opérations internes à $\mathbb{Q}$.

3.3 Notations

Afin de décrire simplement l'algorithme proposé, introduisons les notations suivantes :

$$\text{Soit } D \in \mathbb{N} \text{ avec } D \geq 1.$$
$$\text{Soit } U_D = \left\{x \in \mathbb{R}^D | xx = 1\right\}$$
$$\text{Soit } c \in U_D$$
$$\text{Soit } I \text{ un ensemble fini avec } |I| \geq 1 \text{ où } |.| \text{ désigne le cardinal.}$$
$$\text{Soit } a \text{ une fonction de } I \text{ vers } U_D \text{ et } b \text{ une fonction de } I \text{ vers } \mathbb{R}.$$
$$\text{On suppose que } \forall i \in I,\ ca\left(i\right) \geq 0 \text{ et } a\left(0\right) = c \text{ et } b\left(0\right) = 0.$$
$$\text{Soit } K = \left\{x \in \mathbb{R}^D | \forall i \in I,\ a\left(i\right) x \geq b\left(i\right)\right\}$$
$$\text{On suppose que } \overset{\circ}{K} \neq \emptyset \text{ et } \forall x \in K\ ,\ cx \geq 0 \text{ et } \exists x \in K | cx = 0$$
$$\text{Soit } K^* = \left\{x \in \mathbb{R}^D | \forall i \in I,\ a\left(i\right) x \geq b\left(i\right), cx = 0\right\}$$
$$\text{L'objectif est de déterminer un point de } K^*$$
$$\forall x \in K,\ e\left(x\right) = \min_{i \in I}\left(a\left(i\right) x - b\left(i\right)\right)$$
$$\forall x \in K,\ S\left(x\right) = \left\{i \in I | a\left(i\right) x - b\left(i\right) = e\left(x\right)\right\}$$
$$\forall H \subset I,\ O\left(H\right) = \left\{x \in \mathbb{R}^D | \exists \alpha \in \mathbb{R},\ \forall i \in H,\ a\left(i\right) x - b\left(i\right) = \alpha\right\}$$
$$\forall H \subset I,\ Ker\left(H\right) = \left\{v \in \mathbb{R}^D | \forall i \in H,\ a\left(i\right) v = 0\right\}$$
$$\forall x \in K, \text{ dans le cas où il existe}$$

$$\lambda\left(x, v\right) = \max\left(\left\{\mu \in \mathbb{R} | x + \mu v \in K \wedge \forall \nu \in \left[0, \mu\right[\ ,\ S\left(x + \nu v\right) = S\left(x\right)\right\}\right)$$

Ces notations sont très similaires aux notations standards mais sont utiles pour définir S, O, λ. On prendra garde que désormais, on utilisera des parenthèses et non plus des indices puisque a et b sont maintenant non plus des familles de vecteurs mais des fonctions de domaine fini dans l'ensemble des vecteurs.

3.4 Synopsis

L'algorithme proposé part d'un point de l'intérieur de l'ensemble des points admissibles (voir 3.1 pour sa construction). L'algorithme fait effectuer à ce point des déplacements élémentaires. Chacun des déplacements améliore le score courant qui est le score non pas du point courant cx mais qui est le maximum des scores des points contenus dans la boule maximale centrée sur x c'est à dire $c\left(x - e\left(x\right)c\right)$. Chacun des déplacements laisse le point courant dans l'intérieur de l'ensemble des points admissibles sauf pour atteindre une solution (ce qui correspond à $cx = 0$ voir 3.1).

Étant donné que l'algorithme travaille dans l'intérieur de l'ensemble des points admissibles aucune contraintes n'est saturée (i.e. $a\left(i\right) x > b\left(i\right)$) avant d'atteindre une solution. Ainsi, la notion de contraintes saturées n'est pas pertinente ici mais est remplacée par la notion de contraintes *qui sont les plus proches du point courant*. C'est à dire les contraintes de $S\left(x\right)$. Ces contraintes sont appelées les contraintes actives.

Deux types de déplacements sont considérés :

- des déplacements *surs* : des déplacements qui font croître strictement l'ensemble des contraintes actives (au sens de l'inclusion)
- des **sauts** : des déplacements selon c

L'algorithme proposé est donc constitué de deux fonctions. La fonction amelioration_locale correspond aux mouvements dans l'orthocentre. La fonction amelioration_globale correspond aux autres.

3.5 Améliorations locales

amelioration_locale : $x \in \overset{\circ}{K}$

1. si $0 \in S(x)$, $x \leftarrow x - e(x)\, c$
2. si $x \in K^*$ renvoyer x
3. $\varpi \leftarrow \underset{v \in O(S(x))}{\arg\max} ((c-v)(c-v))$
4. si $x + \lambda(x, \varpi)\, \varpi \in K^*$
   (a) retourner $x + \lambda(x, \varpi)\, \varpi$
5. $\varpi \leftarrow \underset{v \in Ker(S(x))}{\arg\max} ((c-v)(c-v))$
6. si $\varpi \neq 0$
   (a) retourner $x \leftarrow x + \lambda(x, \varpi)\, \varpi$
   (b) goto 1
7. s'il existe $\omega \in O(S(x)) \cap U_D$ et $\delta \in \mathbb{R}_+^*$ tel que $\omega \neq c$ et tel que $\forall \epsilon \in [0, \delta]$, $e(x + \epsilon\omega) = e(x) + \epsilon$
   (a) $x \leftarrow x + \lambda(x, \omega)\, \omega$
   (b) goto 1
8. $\varpi \leftarrow \underset{v \in O(S(x))}{\arg\max} ((c-v)(c-v))$
9. si $\varpi \neq 0$ et $x + \lambda(x, \varpi)\, \varpi \in \overset{\circ}{K}$
   (a) retourner $x \leftarrow x + \lambda(x, \varpi)\, \varpi$
   (b) goto 1

Remarquons que l'utilisation de λ ici justifié en 7.a. Supposons que $\forall \epsilon \in \mathbb{R}_+^*$, $S(x + \delta\omega) = S(x)$ alors on établit que $\forall \epsilon \in \mathbb{R}_+^*$, $e(x + \epsilon\omega) = e(x) + \epsilon$ et alors pour $\epsilon > \frac{cx - e(x)}{1 - c\omega}$, on obtient $c(x + \epsilon\omega - e(x + \epsilon\omega)\, c) < 0$ ce qui est une contradiction. Il en est de même en 4.a et 6.a car $\varpi c > 0$.

Remarquons aussi que le test 7 se ramène à de algèbre linéaire de base : soit $\nu_1, ..., \nu_R$ une base de $O(S(x))$, s'il existe r tel que $\nu_r \neq c$ et $\nu_r S(x) > 0$ alors la condition est vrai sinon elle est fausse. Or déterminer une base d'un espace vectoriel donnée comme une intersection d'hyperplan revient à savoir résoudre des systèmes d'équation linéaire. Soit $\phi_1, ...\phi_R$ des équations d'hyperplans, l'intersection des $\{\phi_r x = 0\}$ est de dimension 0 si et seulement si pour tous d le système $\phi_1 = 0, ...\phi_R = 0, x_d = 1$ n'a pas de solution. Sinon on obtient un vecteur ψ_1 et il suffit de rajouter ψ_1 aux ϕ pour travailler sur le sous espace résultant.

Il en est de même pour 3 et 5.

3.6 Amélioration globale

amelioration_globale : $x \in \overset{\circ}{K}$

1. $x \leftarrow$ amelioration_locale$(x, \text{drapeau})$
2. si $x \in K^*$ retourner x
3. $x \leftarrow x - e(x)\, c$
4. goto 1

4 Discussion

La fonction amelioration_locale vérifie par sa construction les propriétés souhaitées : chaque déplacement améliore strictement le score et laisse le point courant dans l'intérieur de l'ensemble des points admissibles sauf pour retourner une solution. De plus, de part sa construction, cette fonction termine toujours et n'effectue qu'un nombre polynomial en T' d'opérations arithmétiques. Cette dernière propriété est du au fait que le cardinal de $S(x)$ croit strictement à chaque itération.

Par contre, il n'est pas trivial que la fonction amelioration_globale vérifie ces deux propriétés de convergence. Il n'est même pas évident que la fonction termine.

4.1 Terminaison

Nous conjecturons qu'en sortie de amelioration_locale, on a nécessairement $O(S(x)) = \{x\}$.

Cette intuition vient du fait que si on venait à trouver au cours des itérations x puis x' tel que $S(x) = S(x')$ alors c'est que soit $e(x) < e(x')$ ce qui signifie qu'on aurait du appliquer 7 dans amelioration_locale soit c'est que $e(x) = e(x')$ et que $cx < cx'$ (car le score, lui, croit) et on aurait du appliquer 5 soit c'est que $e(x) > e(x')$ mais qu'on n'a pas intérêt à appliquer 7 de x' vers x donc cela signifie que $x' = x + \tau c$ et que 7 ne s'applique pas. Mais si x' vérifie que $O(S(x'))$ est le vectoriel de c c'est que 4 s'applique. Dans tous les cas, il semble y avoir une contradiction.

Si cette conjecture était vrai, si on regarde la suite des sauts on ne peut jamais avoir le même $S(x)$. Comme il n'y en a qu'un nombre fini, cela impliquerait la terminaison.

Cela ne montrerait pas la terminaison fortement polynomial car il faudrait pouvoir montrer que le nombre de saut est polynomial en T'.

4.2 Nombre d'itérations

Nous conjecturons qu'un des indices ayant disparu entre 2 sauts ne réapparaîtra plus. Si cela était vrai alors il n'y aurait au maximum que T' saut.

Cette conjecture semble se ramener à la conjecture suivante : dans le cas où $O(S(x)) = \{x\}$ alors il existe $i \in S(x)$ tel que $K \cup \{a(i)\, p = b(i)\}$ ne contient aucun point p avec $cp \leq cx - e(x)$.

5 Conclusion

Cet article s'intéresse à la programmation linéaire. On conjecture que l'algorithme présenté résolve la programme linéaire en effectuant un nombre d'opérations dans $\mathbb{Q}$ polynomial en le nombre de contraintes et le nombre de variables. Cet algorithme représente potentiellement une avancée majeure par rapport aux algorithmes polynomiaux actuelles ayant besoin d'un nombre d'opérations dans $\mathbb{Q}$ polynomial en le nombre de contraintes et le nombre de variables et linéaire en la taille nécessaire pour écrire le programme linéaire.

Références

[1] N. Karmarkar, A new polynomial-time algorithm for linear programming, 1984

[2] G. B. Dantzig, Maximization of linear function of variables subject to linear inequalities, 1951

[3] V. Klee et G. J. Minty, How good is the simplex algorithm ?, 1972

[4] J.A. Kelner et D.A. Spielman, A randomized polynomial-time simplex algorithm for linear programming, 2006

[5] S. Chubanov, A strongly polynomial algorithm for linear systems having a binary solution,2012

[6] N. Megiddo, Toward a genuinely polynomial algorithm for linear programming, 1981

[7] E. Tardos, A strongly polynomial algorithm to solve combinatorial linear programs, 1986

[8] M. Bàràsz et S. Vempala, A new approach to strongly polynomial linear programming,2010

[9] I. Lavallée et B.M. Ndiaye et D. Seck,Une approche spécifiquement informatique pour la programmation linéaire,2012

[10] J. Plesník, Deepest point of a polyhedron and linear programming, 2013