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

22 mai 2018

1 Introduction

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

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

avec a_m, et c des vecteurs de $\mathbb{Q}^N$ et b_m des éléments de $\mathbb{Q}$, est un problème très étudié dans la littérature. Il est connu que l'algorithme [1] permet de résoudre ce problème en un nombre d'opérations élémentaires polynomial en L la taille nécessaire pour écrire les données d'entrées (a_m, b_m et c). Mais, aujourd'hui, on ne connait aucun algorithme associé à une démonstration établissant qu'il résolve ce problème en un nombre d'opérations élémentaires polynomial en $\max(N, M)$.

2 État de l'art

De nombreuses démonstrations connues établissent des propriétés associées à cette recherche d'un algorithme fortement polynomial :

- il est connu que la méthode du simplexe [2] est exponentielle [3] en $\max(N, M)$ mais cette borne est indépendante de L - aussi une règle de pivotage efficace pour le simplexe pourrait permettre d'apporter une solution au problème [4]
- il est connu qu'il existe une solution ne dépendant que de la taille nécessaire pour écrire les a_m [7]
- il est connu qu'il existe des solutions fortement polynomiales pour certaines familles de programmes linéaires comme ceux contenant un solution binaire [5] ou un nombre de variables par contraintes inférieur à 3 [6]

— il est connu que les approches de type points intérieurs [1][8] (dont on pourra trouver une forme de synthèse dans [9]) ont des complexités inférieures à celle de [2] et certains travaux [10] conjecture que ces méthodes pourraient permettre de concevoir une solution au problème

Cet article propose un algorithme de type point intérieur mais qui s'attache à la nature combinatoire du problème. La contribution de cet article est

— de proposer un nouveau type d'algorithme de point intérieur
— d'appliquer ces déplacements à une transformation déterministe du problème de départ possédant certaines propriétés

3 Algorithme de la direction améliorante

3.1 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. Pour, $\mathbb{R}$, cela ne restreint pas la généralité car de tous vecteurs non nul v, on peut extraire le vecteur unitaire $\frac{v}{\|v\|}$. Mais ce vecteur unitaire n'est généralement pas dans $\mathbb{Q}$.

Néanmoins, je conjecture que cette algorithme peut être modifié de sorte que l'ensemble des opérations n'utilisent que des calculs dans $\mathbb{Q}$ à l'aide de structure particulaire pour représenter les nombres. Le principal argument qui m'amène à cette conjecture est que tous les tests peuvent se faire en comparant des distances au carrées plutôt que les distances et que les calculs linéaires (projection, calcul de base d'un espace vectoriel) peuvent se faire dans $\mathbb{Q}$.

Typiquement, si Υ est une matrice et $\Upsilon z = 0$ caracterise un espace vectoriel Φ alors il suffit de résoudre $\Upsilon z = 0, z_n = 1$ pour $n \in \{1, ..., N\}$ pour obtenir de façon certaine un vecteur non nul ρ de Φ. Puis, en itérant résolvant $\Upsilon z = 0, \rho z = 0, z_n = 1$ on est à nouveau certain d'obtenir un nouveau vecteur non nul jusqu'à avoir une base. L'orthonormalisation de la base à l'aide de la méthode de Gram n'est certe pas interne à Q mais des calculs de projection peuvent être effectués sans passer par l'intermédiare d'une base orthonormale.

3.2 La forme double primal-dual

Soit

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

un problème primal et soit

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

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 :

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

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.

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 :

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

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

Ainsi, dans la suite, on cherche à résoudre le problème

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

quand il vérifie les propriétés suivantes : les a_m sont unitaires, il existe un trivial point dans l'intérieur de l'espace admissible, le minimum existe, il vaut 0, toutes les contraintes a_m ont un produit scalaire strictement positif avec c et ce produit scalaire est différent deux à deux ($\forall i,j\in\{1,...,M\},a_i c\neq a_j c$).

3.3 Notations

Reintroduisons les notations nécessaires pour décrire le problème :

$$N,M\in\mathbb{N}\backslash\{0\}$$
$$U_N=\left\{z\in\mathbb{R}^N|zz=1\right\}$$
$$c\in U_N,\ a_m\in(U_N)^M,\ b\in\mathbb{R}^N$$
$$K=\left\{x\in\mathbb{R}^N|\forall m\in\{1,...,M\},\ a_m x\geq b_m\right\}$$
$$\overset{\circ}{K}=\left\{x\in K|\exists\delta\in\mathbb{R},\delta>0|\forall z\in\mathbb{R}^N,\|z-x\|_2^2\leq\delta\Rightarrow z\in K\right\}$$
$$K^*=\{x\in K|cx=0\}$$

On cherche $x\in K^*$ sachant que K^* est non vide et qu'on est capable de construire un point de $\overset{\circ}{K}$.

Pour décrire l'algorithme, on introduit, de plus, les notations suivantes (toutes les notations fonction de x sont à comprendre comme : $\forall x\in\mathbb{R}^N$, ...) :

$$e(x)=\min_{m\in\{1,...,M\}}(a_m x-b_m)$$
$$S(x)=\{m\in\{1,...,M\}|a_m z-b_m=e(x)\}$$
$$O(x)=\left\{z\in\mathbb{R}^N|\exists\alpha\in\mathbb{R}|\forall m\in S(x),\ a_m z-b_m=\alpha\right\}$$
$$O_0(x)=\left\{z\in\mathbb{R}^N|\forall m\in S(x),\ a_m z-b_m=0\right\}$$
$$G(x)=\left\{z\in\mathbb{R}^N|cz=cx\right\}$$
$$H(x)=O(x)\cap G(x)$$
$$F(x)=\{z\in K|S(x)=S(z)\}$$

$$D(x,v) = \max(\{\mu \in \mathbb{R} | \forall \tau \in [0,\mu[\ ,\ x+\tau v \in F(x)\})$$

$$p(x)\ : Proj\left(a_{S(x)}, H(x)\right)$$

$$q(x)\ : Proj\left(-c, O(x)\right)$$

Notons que chacune de ces notations correspond à des objets constructibles algorithmiquement de façon fortement polynomiale (notamment les projections comme déjà discuter). De plus $Proj\left(a_{S(x)}, H(x)\right)$ a bien un sens : quelque soit l'indice i dans $S(x)$, la projection $Proj(a_i, H(x))$ est identique (car H contient O).

Autre exemple, $D(x,v)$ peut se calculer en parcourant les M contraintes : si v n'a pas un produit scalaire identique pour tous les vecteurs de $S(x)$ c'est 0 - sinon, quelque soit λ, la distance de $x+\lambda v$ à S est $\alpha+\lambda\beta$, pour tous m hors de S on peut alors calculer l'intersection de $\alpha+\lambda\beta$ avec $a_m(x+\lambda v) - b_m$. La première intersection non nul est la valeur cherchée.

3.4 Synopsis

L'algorithme proposé part d'un point de l'intérieur de l'ensemble des points admissibles (voir sec. 3.2 pour sa construction). L'algorithme fait effectuer à ce point des déplacements élémentaires à x le point courant. Chacun des déplacements améliore le score courant cx. 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 (on a toujours $a(i)x > b(i)$) 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 $S(x)$. Ces contraintes sont appelées les contraintes actives.

Les déplacements élémentaires considérées correspondent schématiquement à l'idée suivante : chaque contrainte active m pousse x selon $a(m)$ et x est poussé selon $-c$. Cependant, les *poussées* sont symbolique de sorte à respecter la nature combinatoire du problème.

3.5 Algorithme

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

1. si $x \in K^*$ renvoyer x
2. si $x - e(x)c \in K^*$ renvoyer $x - e(x)c$
3. si $p(x) \neq 0$

   (a) $x \leftarrow x + D\left(x, p\left(x\right)\right) p\left(x\right)$
   (b) goto 1
4. si $q\left(x\right) \neq 0$
   (a) soit λ tel que $x + \lambda q\left(x\right) \in O_0\left(x\right)$
   (b) soit $\varphi = x + \lambda q\left(x\right)$
   (c) si $c\varphi \leq 0$
      i. $x \leftarrow x + D\left(x, q\left(x\right)\right) q\left(x\right)$
      ii. goto 1
   (d) sinon
      i. soit $\omega = Proj\left(q\left(x\right), G\left(x\right)\right)$
      ii. $x \leftarrow x - \frac{e(x)\omega}{2\|\omega\|}$
      iii. goto 1
5. $x \leftarrow x - e\left(x\right) c$
6. goto 1

4 Discussion

Je conjecture que l'algorithme présenté est fortement polynomial. Les arguments qui m'amènent à cette conjecture sont les suivants.

Les instructions 3.a et 4.c.i sont bien défini car e augmente strictement en se déplaçant selon p ou ω or le point $x_{-} = x - e\left(x\right) c$ vérifie $x_{-}c \geq 0$, ce qui impose qu'il n'est pas possible de se déplacer infiniment selon ces vecteurs.

Un déplacement modulé par D augmente nécessairement e donc il maintient un point de $\overset{\circ}{K}$ dans $\overset{\circ}{K}$. Les déplacements 4.d.ii et 5 ne peuvent non plus faire sortir de $\overset{\circ}{K}$: en 4.d.ii on reste strictement dans la boule contenu dans $\overset{\circ}{K}$, dans 5, grâce à l'instruction 2 on sait que l'on ne touche pas une contrainte (le seul plan ayant c comme normale est $c \geq 0$ or une contrainte ainsi tangente à la boule contenu aurait cette normale). Donc x est toujours dans $\overset{\circ}{K}$ sauf à atteindre une solution.

$q\left(x\right)$ ne peut être égale à $-c$ car les produits scalaires des contraintes à c sont différents deux à deux et que q est une projection sur O.

$S\left(x\right)$ est forcément différent entre deux passages consecutifs dans l'instruction 1 : soit on se déplace avec une amplitude utilisant D ce qui implique un changement de S - soit on se déplace selon c et les produits scalaires des contraintes à c sont différents deux à deux - soit on se déplace selon ω mais $p = 0$ donc ω n'appartient pas à H donc pas à O.

On conjecture que $S(x)$ est forcément différent entre deux passages (tout court) dans l'instruction 1. L'argument pour cela est que si $S(x)$ revient il est anormal, on se serait retrouver en ce point après 4.c.i donc ça veut dire qu'on a un $S(x)$ avec q non nul mais avec $c\varphi > 0$ **et** p nul (je ne suis même pas convaincu que ce soit possible). Le moyen le plus sur de ne plus retrouver S, serait de construire un point x tel que $cx < c\varphi$. Mais cela n'est pas trivial. Cependant, les instructions 4.d.i-ii-iii sont justement là pour permettre de sortir de ce problème bien que cette partie de la démonstration ne soit pas suffisante.

Remarquons que même si on démontrait que $S(x)$ est différent entre chaque passage dans 1, cela démontrerait seulement la terminaison de l'algorithme et non pas sa complexité car il existe un nombre exponentiel de S possibles.

Cependant, on conjecture (mais sans argument encore suffisant) que non seulement $S(x)$ est différent entre chaque passage dans 1 mais surtout, que soit S croit pour l'inclusion - soit qu'une contrainte a disparu de S et ne reapparaitra plus. Si cette conjecture était vrai, le caractère fortement polynomial serait établie puisque le nombre de passage dans 1 serait alors borné par M^2 et que toutes les opérations entre deux passages dans 1 sont fortement polynomiales.

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.

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