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

2012-08-06

1 Introduction

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

$$\min_x (cx)$$
$$\forall m \in \{1, ..., M\}, a_m x \geq b_m$$

avec a_m, et c des vecteurs de dimension N, b_m des scalaires est un problème très étudié dans la littérature. Ce problème est un problème polynomial [3]. Mais, les meilleurs bornes polynomiales connues dépendent du nombre de la taille L nécessaire pour écrire les données et non pas seulement de $\max(N, M)$: il n'existe aujourd'hui aucun algorithme déterministe connu ayant seulement besoin pour produire la solution d'un nombre d'opération dans $\mathbb{Q}$ polynomial en M et N. Cette dépendance en L est la conséquence d'une recherche de la solution optimale via une descente de gradient propre aux algorithmes de type points intérieurs. À l'opposé, les méthodes qui cherchent la solution optimale en explorant la combinatoire fini des ensembles de contraintes ne dépendent que du nombre de contraintes M et du nombre de variables N. Par exemple, la méthode du simplexe [1], 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 [2] car 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. La suite de cet article est structuré ainsi. Dans la section 2, un état de l'art des recherches d'un algorithme fortement polynomial pour résoudre les programmes linéaires est présentée. Dans la section 3, l'algorithme proposé est présenté. Dans la section 4, une discussion sur l'efficacité de l'algorithme est présentée. Enfin la section 5 contient la conclusion de cet article.

2 État de l'art

Depuis qu'il est démontré [3] que la résolution des programmes linéaires est un problème polynomial, deux principales voies d'améliorations sont explorés pour obtenir des algorithmes plus efficace : la recherche de règle de pivotage efficace pour le simplex [4] ou l'amélioration des méthodes de type point intérieurs.

Il existe aussi des recherches qui explorent de nouvelles voies comme [5,6].

Dans [5], un algorithme purement combinatoire est donnée pour résoudre de façon polynomiale les programmes linéaires dans lesquelles le nombre de variable par contrainte est inférieur ou égal à 2.

Dans [6], un algorithme basé sur des directions améliorantes est présenté. Il a l'avantage d'être invariant aux transformations affines du système.

Cet article se place dans la recherche de nouvelles voies et réutilise l'idée présentée dans [6] des directions améliorantes. Néanmoins au lieu d'utiliser des directions améliorantes heuristiques comme dans [6], nous proposons de chercher à construire une direction efficace compte tenu de l'objectif à chaque étape de l'algorithme.

3 Algorithme de la direction améliorante

3.1 Construction de l'algorithme

3.1.1 Meilleur direction en chaque point :

Une première idée pour construire un algorithme fortement polynomial pour résoudre un programme linéaire est de calculer pour le point courant, la meilleure direction à suivre (une direction permettant un déplacement non nul tout en restant dans l'ensemble des points admissibles et de plus fort produit scalaire avec le vecteur objectif.

Mais cette approche a 2 problèmes : le problème du calcul de la meilleure direction est un problème plus difficile que la résolution d'un programme linéaire.

De plus, même en étant capable de calculer la meilleure direction en chaque point, il n'est pas trivial de montrer que le nombre d'itérations est borné.

3.1.2 Centre de l'ensemble admissible :

Une deuxième idée est, étant donné un point dans l'intérieur de l'ensemble des points admissibles, se déplacer pour atteindre le point centrale (un point qui maximise la distance à l'ensemble des contraintes) [7].

En supposant qu'on sache calculer un tel point, il suffit à partir du centre courant z à distance aux moins d de chaque contrainte de rajouter une contrainte $cx \geq cz - \frac{d}{2}$.

Néanmoins, si la convergence est trivial, il est facilement concevable que chaque itération divise par 2 la différence entre la valeur objectif courante et

la valeur optimale mais la différence initiale entre ces deux valeurs peut croître exponentiellement en fonction de L.

3.1.3 Centre de l'ensemble admissible et orthocentre :

Mais l'idée précédente est améliorable : plutôt que de se déplacer uniquement en faisant des sauts (un ajout de contrainte forçant à améliorer la valeur objectif), on peut suivre l'orthocentre des contraintes actives.

Introduisons 2 définitions : on appellera **contraintes actives** les contraintes à distance minimale du point courant (cela est différent des **contraintes saturées** qui sont les contraintes touchées par le point courant : dans notre algorithme on ne sature des contraintes que pour rejoindre la solution optimale). **L'orthocentre** de 2 plans est le plan tel qu'un point sur ce plan est équidistant des 2 plans donnés. L'orthocentre de R plans est l'intersection des orthocentres 2 à 2 (sachant qu'il suffit de R intersections à calculer pour calculer l'orthocentre car si on est a même distance de (a_1, b_1) et de (a_2, b_2) et si on est à même distance de (a_1, b_1) et de (a_3, b_3) alors on est aussi à même distance de (a_2, b_2) et de (a_3, b_3)).

Ainsi, on peut préciser l'idée : Considérons le centre initial de l'ensemble des contraintes (un point qui maximise la distance à l'ensemble des contraintes construit à partir d'un point dans l'intérieur de l'ensemble des points admissibles). $\forall \epsilon > 0$, on considère le point centre du nouvel ensemble de contrainte où on impose à $cx \leq cz + d - \epsilon$ où d et la distance courante du centre aux contraintes.

Cet ensemble de point est continu et on conjecture qu'il est constitué d'une droite pour ϵ suffisamment petit.

Dans le cas où l'orthocentre des contraintes actives courante n'est pas réduit au point courant, cette droite est dans l'orthocentre. Sinon c'est l'orthocentre seulement d'un sous-ensemble des contraintes actives.

Dans les deux cas, il suffit de suivre cet orthocentre, jusqu'à atteindre un point solution, où jusqu'à ce que l'ensemble des contraintes actives soit modifié.

Cette opération est itérée et on conjecture qu'au moins une des contraintes abandonnées dans le cas d'un orthocentre réduit à un point n'apparaîtra plus jamais au cours des itérations suivantes.

Ce dernier algorithme est ainsi fortement polynomial si

- on sait construire et suivre la trajectoire (supposée affine par partie) des centres contraints à avoir une valeur objectif de plus en plus petite
- cette trajectoire est constitué que d'un nombre fortement polynomial de point critique (un point courant associé à un ensemble de contraintes actives, dont il est l'orthocentre unique)

3.1.4 Centre de l'ensemble admissible, orthocentre et saut :

Malheureusement, il est difficile de construire et suivre la trajectoire (supposée affine par partie) des centres contraints à avoir une valeur objectif de plus en plus petite.

La principale difficulté vient des centres où une partie des contraintes actives doit être abandonné pour augmenter la valeur objectif.

Pour remédier à cela il suffit de faire un saut selon la direction objectif (forcer la valeur objectif du point courant à être augmenter tout en restant dans l'ensemble des valeurs admissible) puis de recalculer un centre de même valeur objectif. Ainsi au lieu de suivre la trajectoire (supposée affine par partie) des centres contraints à avoir une valeur objectif de plus en plus petite, on fait un saut à chaque point critique puis on se projette sur la trajectoire.

Malheureusement, cet algorithme suppose quand même d'être capable de construire le centre d'un ensemble de contrainte c'est à dire

$$\max_x (d)$$
$$\forall m \in \{1, ..., M\}, a_m x \geq b_m + d$$

Or ce problème semble aussi difficile que de construire

$$\min_x (cx)$$
$$\forall m \in \{1, ..., M\}, a_m x \geq b_m$$

3.1.5 Approximations :

Pour ne pas avoir à calculer des centres, on peut approximer ce problème en se déplaçant dans les orthocentres des contraintes actives et en effectuant un saut chaque fois qu'un tel déplacement est impossible.

Cela conduit à l'algorithme suivant :

À chaque itération effectuer une des actions suivantes (jusqu'à atteindre la solution) :

- se déplacer selon c tant qu'on est dans l'ensemble des points admissibles si cela amène à une solution
- se déplacer selon la projection de c dans l'orthocentre tant qu'on est dans l'ensemble des points admissibles si cela amène à une solution
- se déplacer selon c tant que cela ne change pas l'ensemble des contraintes actives et augmente la distance aux contraintes actives
- se déplacer selon la projection de c dans l'orthocentre des contraintes actives tant que cela ne change pas l'ensemble des contraintes actives et augmente la distance aux contraintes actives
- se déplacer selon la projection de c dans l'iso-distance aux contraintes actives tant que cela ne change pas l'ensemble des contraintes actives
- se déplacer selon la projection de c dans l'orthocentre des contraintes actives du moment que cela conduit à un changement des contraintes actives avant de saturer une contrainte
- effectuer un saut : se déplacer selon c de sorte à rester dans l'intérieur de l'ensemble des points admissibles.

Il est possible que cet algorithme soit fortement polynomial mais il ne semble pas vrai que chaque saut élimine au moins une contrainte pour la suite des itérations.

De plus, cet algorithme a les 2 limites suivantes :

Calcul dans $\mathbb{Q}$: Le premier problème de l'algorithme proposé est que certaine opération ont naïvement lieu dans $\mathbb{R}$ et non dans $\mathbb{Q}$. Typiquement le calcul de la distance d'un point à un plan implique naïvement le calcul de la norme euclidienne du vecteur normal et donc le calcul d'une racine carré (opération non interne à $\mathbb{Q}$). Néanmoins, cet algorithme semble pouvoir être modifier de sorte que l'ensemble des tests n'utilisent que des calculs dans $\mathbb{Q}$ notamment en comparant les distances au carrées. De plus, 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 $\mathbb{Q}$.

Cas d'un orthocentre contenant la direction objectif : L'algorithme proposé boucle infiniment dans le cas où l'orthocentre des contraintes actives contient c mais que se déplacer selon c conduit à toucher les contraintes actives sans pour autant aboutir à une solution et qu'il n'est pas possible de se déplacer tout en restant à égale distance de toutes les contraintes actives. Néanmoins, ce cas semble être possible que si l'ensemble des projections du point courant sur les contraintes actives est contenu dans un demi-espace passant par le point courant. La solution est alors de se déplacer dans le demi espace opposé (typiquement, un des vecteurs normaux des contraintes actives, projeté dans le plan de normale le vecteur objectif c passant par le point courant fournit un vecteur de fuite).

Même dans le cas où ce type de situation pouvait arriver en d'autres situations, il semble qu'on puisse trouver une amélioration permettant de sortir de ce piège en utilisant les projections du point courant sur les contraintes actives.

Pour tenter de résoudre ces problèmes, l'idée est d'essayer de ne quitter la contrainte active de plus petit indice que dans la mesure où elle ne réapparaîtra plus.

3.2 Condition

L'algorithme proposé suppose que le programme linéaire soit d'intérieur non vide, de valeur minimale 0 et qu'on connaisse un point dans l'intérieur.

Il est toujours possible de former un tel programme linéaire à partir d'un programme linéaire quelconque en formant le programme primal-dual 2 fois :

$$\begin{gathered}\min_x (cx) \\ \forall m \in \{1, ..., M\}, a_m x \geq b_m\end{gathered}$$

est équivalent à

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

où a' et b' sont les contraintes correspondantes au dual.

Ce dernier problème est équivalent à

$$\begin{gathered}\min_{x}(z)\\ \forall m \in \{1,...,M\}, a_m x + z \geq b_m\\ \forall m \in \{1,...,M'\}, a'_m y - z \leq b'_m\\ cx \geq by - z\\ z \geq 0\end{gathered}$$

qu'on appelle ici le primal-dual.

Le fait de prendre à nouveau le primal-dual de ce dernier problème fournit un programme linéaire d'intérieur non vide, borné de valeur minimale 0.

3.3 Description de l'algorithme

À chaque itération effectuer une des actions suivantes (jusqu'à atteindre la solution) :

- se déplacer selon c tant qu'on est dans l'ensemble des points admissibles si cela amène à une solution
- se déplacer selon la projection de c dans l'orthocentre tant qu'on est dans l'ensemble des points admissibles si cela amène à une solution
- se déplacer selon c tant que cela ne change pas l'ensemble des contraintes actives et augmente la distance aux contraintes actives
- se déplacer selon la projection de c dans l'orthocentre des contraintes actives tant que cela ne change pas l'ensemble des contraintes actives et augmente la distance aux contraintes actives
- se déplacer selon la projection de c dans l'iso-distance aux contraintes actives tant que cela ne change pas l'ensemble des contraintes actives
- se déplacer selon la projection de c dans l'orthocentre des contraintes actives du moment que cela conduit à un changement des contraintes actives avant de saturer une contrainte
- effectuer un saut : si effectuer un saut selon c permet d'obtenir une valeur objectif inférieur à la plus petit valeur dans le plan de plus petit indice, se déplacer selon c [1], sinon se déplacer vers le plan de plus petit indice (vers la projection du point courant sur le plan) [2]. Dans les 2 cas, on peut par exemple calibrer l'amplitude du saut en réduisant moitié la distance aux contraintes actives.

4 Discussion

Dans cet article, ni la convergence de l'algorithme ni la convergence fortement polynomiale n'est démontré. Néanmoins, on conjecture que l'algorithme converge de façon fortement polynomiale.

1. en utilisant les projections dans le cas où se déplacer selon c ne conduit pas à un changement de l'ensemble des contraintes actives
2. si se déplacer vers le plan du plus petit indice aboutit à se retrouver dans la même direction, c'est qu'il faut se déplacer selon c à la place

5 Conclusion

Dans cet article, un algorithme est présenté dans le contexte de la programmation linéaire. On conjecture que cet algorithme retourne la solution du 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 en bits de l'entrée.

Références

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

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

[3] N. KARMARKAR, A NEW POLYNOMIAL-TIME ALGORITHM FOR LINEAR PROGRAMMING, 1984

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

[5] N. Megiddo, TOWARD A GENUINELY POLYNOMIAL ALGORITHM FOR LINEAR PROGRAMMING, 1981

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

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