الجمهوريـة الجزائـريـة الديمقراطيـة الشعبيـة
**République Algérienne Démocratique et Populaire**
وزارة التعليـم العالـي والبحـث العلمـي
**Ministère de l'Enseignement Supérieur et de la Recherche Scientifique**


**N° Ref :……**

**Centre Universitaire de Mila**

**Institut des sciences et de la technologie** **Département de Mathématiques et Informatique**

# Un algorithme modifié pour la méthode de Ye-Lustig

**Mémoire préparé En vue de l'obtention du diplôme de Master en Mathématiques**

**Préparé par : Guerba Dhiyae El hak** **Encadré par : Leulmi Assma**

**Lekcir Kamal Eddine**

**Filière : Mathématiques** **Spécialité : Mathématiques Fondamentales et appliques**

**Année universitaire :2012/2013**

# ✵✵✵Remerciement✵✵✵

*Au terme de ce travail, nous commençons par remercier DIEU pour nous avoir donné la volonté et le courage pour terminer ce modeste travail.*

*Nous tenons à exprimer notre gratitude et notre sincère remerciement à tous ceux qui ont contribués, de près ou de loin à l'élaboration de ce mémoire.*

*Nous remerciements très chaleureux sont adressé à M me .LeulmiAssmanotre encadreur.*

*Nous adressons également nos vifs remerciements aux membres de jury qui ont bien voulu et accepter d'examiner ce modeste travail.*

*Nous adressons également mes remerciements chaleureux aux membres de l'institut des sciences et de la technologie.*

*Nous remerciements sons également adressés à tous les enseignants qui ont contribué à notre formation.*

*Dhiyae el hek&kamaleddine*

# اهـــــــــداء

**هذه ثمرة حياتي اهديها الى**

**الى اعطر عطور احلامي الى احن همسات انغامي الى اروع سحر في ايامي الى ملاك احلامي الى اروع صورة فتحت عليها عيناي الى اعذب صوت سمعته اذناي الى التي طالما سقت طموحاتي وصقلت امنياتي ومسحت دمعاتي الى التي وقعت كل نجاحاتي الى التي مالات حياتي تحديا واملا وطريقي مثابرة وعملا**

**<u>الى امي ثم امي ثم امي</u>**

**الى النجم الساري في سما افقي...الى الغالي الذي سكن اعماقي...الى منبع الخير الدافق والحنان الوافر...الى المربي الفاضل الذي نسج لي طريق النجاح ...الى من تعب لنرتاح وزرع في قلوبنا بذور الكفاح**

**<u>اليك ابى الحبيب</u>**

**الى من رافقوا دربي يوما بيومالى غرسهم الخالق زهرات في ربيع حياتي اخي <u>صهيب</u> واخواتي <u>رشيدة وهاجر</u>**

**الي اختي الغالية والتي اتمنى من المولى ان يوفقها وان ييسر لها في حياتها<u>سلمى</u> وزوجها <u>فوزي</u>دون ان انسى ثمرتا زواجهما <u>لؤي سراج الدين ومحمد اياد</u>**

**الى صديقي لابل الى اخي الذي كان رفيقا وعونا لي في الحياة الجامعية الذي ادعو من الله ان يوفقه في كل خطوة يخطوها**

**<u>اليك زكريـــــــاء</u>**

**الى من قاسمني احرف هذه المذكرة ولحظات هذا العمل**

**<u>كمـــــال الـــدين</u>**

**الى ابنة العم**

**<u>الزينـــة</u>**

**الى كل من تقاسمت معهم حلاوة النجاح ومرارة الفشل اليكم اصدقائي**

**الى كل من سقط من قلمي سهوا والى كل من ترك بصمة في حياتي وغير مجراها**

**<u>ضيـــاء الـــحق</u>**

# اهـــــداء

الـــــــــــــــــــــــــــهي لا يطيب الليل إلا بشكرك

ولا يطيب النـــــــــــــــــــــــــهار إلا بطاعـــــــــتك

و لا تطيب اللحـــــــــــــــــــظات إلا بذكـــــــــــــرك

ولا تطيـــــــــــــــــــــــــــــب الآخرة إلا بـعفوك

ولا تطيـــــــــــــــــــــــب الجـنة إلا برؤية وجـــــهك

إلـــــــــــــــى من بلغ الرسالة وأدى الأمانة ونصـــــح الأمة

إلـــــــــــــــىحبـــــــــــيبي محمد- صلى الله عليه وسلم-

إلى من علمني العطاء بدون انتظار إلى من احمل اسميهما بلا افتخار أرجو من الله ان يطيل في عمرهما وستبقى كلماتهما نجوما اهتدي بها طول حياتي إلـــى من أرضعتني الحنان وعلمتني الصدق والأمان ودفعتني إلى العلم والعمل الصالح إلى من يتقطع قلبي بهواها

أمـــــــــــــــــــــــــــي الحـــــــــــــــــــــــبية

إلى سراج دربي ونور عيني إلى الصالح إلى من تعب لسنوات من اجل تحقيق أحلام حياتي

أبـــــــــــــــــــي الغـــــــــــــــــــالي

إلـــــــــــــــى النجـــــــوم المـــــــــتلألئة

إخوتي وأخواتي

إلـــــــــــــــى من شاركني في إخراج هذا العمل

ضيـــــــــــــــــــــــــــاء الـــــحق

إلـــــــــــى كل من وقف إلى جانبي

وزرع زهور الأمـــــل والتفاؤل

أصدقـــــــــــــــائي

كمـــــال الديـــــن

**ملخص:**

نقترح من خلال هذا البحث, تعديلا عمليا في الخوارزمية التساقطية ليو-ليستيغ بالنسبة للبرمجة الخطية انجر عنها انخفاض معتبر في الكلفة وعدد الخطوات هذا الطرح معزز مختلف التجارب العددية الهامة المنجزة.

**كلمات مفتاحية:**

البرمجة الخطية,طريقة كارماركر,طريقة يو-ليستيغ.

## Abstract:

We propose in this study,a partial modification for Ye-Lusting's projective algorithm for the linear programming. This modification leads to a considerable reduction of the cost and the number of iterations.

These proposed techniques are confirmed by many interesting numerical experimentation.

## Key words:

Linear Programming, Karmarkar's Method,Ye-Lustig's Method.

## Résumé:

Nous proposons dans cette étude, une modification pratique de l'algorithme projectifde Karmarkar pour la programmation linéaire entrainant une réduction considérable du cout et du nombre d'itérations.

Ces propos sont confirmés par des expérimentations numériques intéressantes.

## Mots clés:

Programmation linéaire, Méthode de Karmarkar, Méthode de Ye-Lustig.

# Table des matières

**1 Introduction générale** **3**

**2 Programmation linéaire** **6**
2.1 Introduction . . . . . 6
2.2 Éléments et notion de base . . . . . 7
2.2.1 Rappels sur l'analyse convexe . . . . . 7
2.2.2 Notations . . . . . 8
2.2.3 Classification des problèmes d'optimisation . . . . . 9
2.3 Programmation linéaire . . . . . 9
2.3.1 Forme d'un programme linéaire . . . . . 11
2.3.2 Dualité . . . . . 12
2.3.3 Méthode de résolution du programme primal . . . . . 14

**3 Méthodes de point intérieur pour la programmation linéaire** **16**
3.1 Introduction . . . . . 16
3.1.1 Préambule . . . . . 16
3.1.2 Notion fondamentale . . . . . 17
3.2 Méthode affines . . . . . 17
3.3 Méthode de trajectoire centrale . . . . . 18
3.4 Méthode de réduction du potentiel . . . . . 19
3.5 Méthode projective de Karmarkar . . . . . 20
3.5.1 Problème linéaire traité par Karmarkar . . . . . 21
3.5.2 Algorithme de Karmarkar . . . . . 22
3.5.3 Fonction potentielle & convergence . . . . . 25
3.5.4 Étude des performances de l'algorithme de Karmarkar . . . . . 26
3.5.5 Généralisation de l'algorithme de Karmarkar . . . . . 28
3.5.6 Algorithme de Ye-Lustig . . . . . 30
3.6 Calcul d'une solution initiale strictement réalisable . . . . . 31
3.7 Conclusion . . . . . 33

**4 Amélioration de l'approche de Ye-Lustig** **34**
4.1 Position du problème . . . . . 34
4.2 Amélioration de la phase 1 dans l'approche de Ye-Lustig . . . . . 35
4.2.1 Préambule . . . . . 35
4.2.2 Modification de l'algorithme de Ye-Lustig (phase 1) . . . . . 37
4.2.3 Tests numériques . . . . . 42
4.3 Amélioration de la phase 2 dans l'approche de Karmarkar . . . . . 43
4.3.1 Préambule . . . . . 43
4.3.2 Préparation du problème . . . . . 44
4.4 Amélioration de la phase 2 dans l'approche de Ye-Lustig . . . . . 44
4.4.1 Préambule . . . . . 44
4.4.2 Préparation de problème . . . . . 44
4.4.3 Modification de l'algorithme de Ye-Lustig (phase 2) . . . . . 46
4.4.4 Algorithme modifié . . . . . 48
4.4.5 Tests numériques . . . . . 49

**5 Conclusions & perspectives** **52**

**Bibliographie** **54**

# Chapitre 1

# Introduction générale

La programmation linéaire est une branche de l'optimisation permettant de résoudre de nombreux problèmes économiques et industriels (comme le transport ou l'ordonnancement). On trouve aussi plusieurs champs d'application moins connus ou plus récents, comme l'économie, la biologie, la gestion du personnel ou le secteur public. Des logiciels puissants permettent aujourd'hui, de mettre cet outil à la disposition des utilisateurs.

L'apparition, l'évolution rapide et le succès des méthodes de points intérieurs depuis leur relance par Karmarkar (1984) dans le domaine de la programmation linéaire, ont incité les chercheurs du monde entier à développer tout un arsenal de méthodes (logiciels) permettant de traiter convenablement plusieurs classes de problèmes considérées jadis difficiles à résoudre, parmi lesquelles la programmation non linéaire, la programmation semi-définie,... etc.

Notre travail concerne le problème de programmation linéaire (PL) qui est devenue un thème de recherche très convoité depuis la relance des méthodes de points intérieurs issues

des nouvelles investigations apportées par Karmarkar (1984) et autres.

En effet, ces méthodes avec leurs éléments nouveaux s'avèrent très intéressantes pour un traitement descend de la (PL) allant de l'aspect théorique pur à l'aspect numérique proprement dit. L'algorithme de Karmarkar et ses variantes principales, sont des méthodes projectives à deux phases et à convergence polynomiale.

Elles sont reconnues comme concurrents redoutables de la fameuse méthode du simplexe [8]. Leurs inconvénient majeur (à environ 80% de la valeur calculatoire) est le calcul de la projection qui domine le coût de l'itération.

A ce propos, l'étude des performances de telles méthodes a fait l'objet de travaux de recherche acharnés un peu partout dans le but d'atteindre un comportement numérique nettement meilleur que celui du simplexe.

Les travaux ainsi effectués pendant plusieurs années, sont quasiment d'ordre technique. Certes, on a enregistré certaines améliorations, mais on reste loin de l'objectif en vu.

Une idée originale, consiste à éviter de manière intelligente les calculs pénibles chaque fois que cela est possible, est proposée par D. Benterki & B. Merikhi (2001) [16], cette alternative a fait ses preuves au niveau de la phase 1 (d'initialisation).

Ce qui nous a motivés pour l'adapter à la phase 2 (d'itération). Si en évidemment, certains aménagements techniques s'avérât mi dispensables.

A ce propos, nous avons choisi une variante de l'algorithme de Karmarkar dans laquelle, nous avons introduit les modifications nécessaires.

Les expérimentations numériques effectuées sont encourageantes et montrent l'uti-

lité de cette alternative.

Le mémoire est organisé comme suit :

Dans le premier chapitre, nous présentons des notions de base sur l'analyse convexe ainsi que la programmation linéaire qui seront utiles par la suite.

Le deuxième chapitre, est réservé à une description des méthodes de points intérieurs, en particulier la méthode originelle de Karmarkar et sa variante de Ye-Lustig.

Le troisième chapitre est divisé en deux parties, dans la première, on présente une synthèse générale sur des travaux de D. Benterki & B. Merikhi (2001) qui concernant une modification dans la phase1 de l'algorithme de Ye-Lustig. En exploitant cette idée, on présente dans la deuxième partie une étude théorique et numérique d'une modification effectuée au niveau de la phase 2 de l'algorithme de Karmarkar, qui a impliqué une réduction du nombre d'itérations et du temps de calcul.

Les tests numériques que nous avons effectués, confirment l'efficacité de l'algorithme modifié vis-à-vis de celui de Karmarkar.

Enfin, une étude théorique préliminaire concernant l'extension de cette procédure à l'algorithme de Ye-Lustig est traitée à la fin de ce chapitre.

Notre étude porte des conclusions et ouvre des perspectives intéressantes pour l'élargissement de l'algorithme à d'autres classes des problèmes.

# Chapitre 2

# Programmation linéaire

## 2.1 Introduction

La programmation linéaire peut se définir comme une technique mathématique permettant de résoudre des problèmes de décisions et particulièrement ceux où le décideur cherche la meilleur utilisation des ressources pour atteindre un objectif spécifique comme la maximisation des bénéfices ou la minimisation des coûts.

La programmation linéaire est certainement, l'un des plus importants développements dans le domaine de la recherche opérationnelle. Son importance réside, d'une part, dans la puissance de modélisation offerte, malgré, sa limite inhérente imposée par la linéarité des fonctions impliquées, et d'autre part, dans la richesse de la théorie qu'elle a initiée et qui a favorisé le développement d'algorithmes extrêmement efficaces pour sa résolution. Depuis la formulation et le développement de la méthode du simplexe pour sa résolution vers la fin des années 40, la programmation linéaire demeure le modèle d'optimisation le plus utilisé par les décideurs. Ceci est, certainement, dû à la robustesse des algorithmes disponibles.

La nature combinatoire de l'algorithme du simplexe a, probablement, fait en sorte que la programmation linéaire a été longtemps considérée comme une classe de problème distincte de la programmation non linéaire.

De même, depuis les années 90, les méthodes de point intérieur sont devenues intéressantes après leurs lancements comme de vrais concurrents à la méthode du simplexe sur tout pour les problèmes de grandes dimensions [8].

## 2.2 Éléments et notion de base

### 2.2.1 Rappels sur l'analyse convexe

Dans cette partie, nous allons rappeler les plus importants d'analyse convexe nécessaire pour le développement de ce travail.

**Définition 2.2.1** *Un ensemble $C \in \mathbb{R}^n$ est dit convexe si :*

$$\forall \alpha \in [0,\ 1]\,,\ \forall x,\ y \in C,\ \alpha x + (1-\alpha)y \in C.$$

*Autrement dit : tout point du segment $[xy]$ appartient à $C$.*

**Définition 2.2.2** *Une fonction $f : C \to \mathbb{R}$ est dite convexe si :*

$$\forall t \in [0,\ 1]\,,\ \forall x,\ y \in C,\ f\left[(1-t)x + ty\right] \leq (1-t)f(x) + tf(y)$$

**Définition 2.2.3** *Un ensemble convexe de la forme*

$$P = \{x \in \mathbb{R}^n :\ Ax = b,\ x \geq 0\}\,/A \in \mathbb{R}^{m+n},\ b \in \mathbb{R}^m$$

*est appelé polyèdre convexe fermé.*

**Définition 2.2.4** *Un ensemble convexe de la forme*

$$S_n = \left\{ x \in \mathbb{R}^n_+ : \sum_{i=1}^{n} x_i = 1 \right\}$$

*est appelé n-simplexe.*

**Définition 2.2.5** *Un polyèdre convexe borné est appelé polytope convexe.*

**Définition 2.2.6** *Soit $X$ un convexe non vide de $\mathbb{R}^n$, un point $x \in X$ est dite extrémale (ou sommet de $X$) si l'on a :*

$$\forall t \in [0,\ 1]\,,\ \forall\,(y,\ z) \in X :\ x = (1-t)y + tz \Longrightarrow x = y = z.$$

**Définition 2.2.7** *L'enveloppe convexe d'un sous ensemble fini de $\mathbb{R}^n$ est appelée polytope.*

**Définition 2.2.8** *Un polytope convexe de $\mathbb{R}^n$ possédant $(n+1)$ sommets est appelé n-simplexe.*

**Exemple 2.2.9** *Lorsque $k = 0,\ 1,\ 2$ ou $3$, le n-simplexe est, respectivement, un point, un segment, une région triangulaire dans $\mathbb{R}^2$.*

### 2.2.2 Notations

Un programme mathématique $(PM)$ est un problème d'optimisation de la forme :

$$(PM) \begin{cases} \min f(x) \\ g_i(x) \leq 0,\ i = 1,\ 2,\ ...,\ k \\ h_j(x) = 0,\ j = 1,\ 2,\ ...,\ m \\ x \in C \subseteq \mathbb{R}^n \end{cases}$$

où $f$, $g_i$, $h_j$ sont des fonction définies de $\mathbb{R}^n$ dans $\mathbb{R}$.

### 2.2.3 Classification des problèmes d'optimisation

Les problèmes d'optimisation sont classifiés selon les caractéristiques des fonctions $f$, $g_i$, $h_j$. Parmi les cas particuliers les plus étudiés, on note :

- La programmation linéaire ($f$ linéaire, $g_i$, $h_j$ affines).

- La programmation convexe ($f$, $g_i$ convexes, $h_j$ affine).

- La programmation quadratique ($f$ quadratique, $g_i$, $h_j$ affines).

- La programmation en nombres entiers ($C$ discret)

**Définition 2.2.10** *L'ensemble*

$$S = \{x \in C : \ g_i(x) \leq 0, \ h_i(x) = 0\}$$

*est appelé ensemble des solutions réalisables.*

**Définition 2.2.11** *Soit $x^* \in S$. On dit que $x^*$ est un* ***minimum local*** *du problème $(PM)$, si :*

$$f(x^*) \leq f(x), \ \forall x \in S.$$

**Définition 2.2.12** *Soit $x^* \in S$. On dit que $y \in \mathbb{R}^n$ est une* ***direction admissible*** *du problème $(PM)$, s'il existe $\alpha > 0$ tel que :*

$$x^* + ty \in S, \ \forall t \in [0, \ \alpha].$$

## 2.3 Programmation linéaire

Notions que tout programme linéaire $(PL)$, c'est-à-dire tout problème d'optimisation où la fonction objective et les fonctions définissant les contraintes sont affines, peut se

mettre sous la forme standard par des manipulations algébriques simples comme suit :

$$(PL)\begin{cases} \min c^t x \\ Ax = b \\ x \geq 0. \end{cases}$$

La matrice $A$ est de dimension $m \times n$ avec $m \leq n$, avec l'hypothèse qu'elle est de plein rang. Sur des modélisations pratiques, cette l'hypothèse est rarement satisfaite. La matrice $A$ peut, toutefois, être réduite à une matrice de plein rang par une décomposition $QR$ ou une élimination de Gauss. L'ensemble des solutions réalisables :

$$S = \{x \in \mathbb{R}^n : \ Ax = b, \ x \geq 0\}.$$

Forme un polyèdre convexe, c'est-à-dire, l'intersection d'un nombre fini de demi-espaces. Un programme linéaire est réalisable si l'ensemble $S$ est non vide.

**Définition 2.3.1** *Une méthode est dite polynomiale si le nombre d'opérations élémentaires pour déterminer une solution optimale est borné par un polynôme en la taille du problème.*

**Définition 2.3.2** *Un vecteur $x$ vérifiant $\{Ax = b$ et $x \geq 0\}$ est une solution réalisable de $(PL)$.*

**Définition 2.3.3** *Un vecteur $x$ vérifiant $\{Ax = b$ et $x > 0\}$ est une solution strictement réalisable de $(PL)$.*

**Définition 2.3.4** *Un vecteur $x$ vérifiant $(PL)$ est une solution optimale de $(PL)$.*

**Définition 2.3.5** *Un programme linéaire $(PL)$ réalisable est borné sur $S$.*

**Proposition 2.3.6** *Un point $x$ est un point extrême de $S$ si et seulement si les colonnes $\{A^j : x_j > 0\}$ de $A$ sont linéairement indépendantes. On en déduit que si $S$ est non vide, alors il existe au moins un point extrême.*

**Proposition 2.3.7** *Un programme linéaire réalisable et borné possède une solution optimale en un point extrême du polyèdre de ses solutions réalisables.*

### 2.3.1 Forme d'un programme linéaire

**1 - Forme standard**

Le programme linéaire primal $(PL)$ caractérise sa forme standard.

**2 - Forme canonique**

On considère le programme linéaire primal $(PP)$ caractérisé par :

$$(pp)\begin{cases} \min c^t x \\ Ax \leq b, \ \ (\geq b) \\ x \geq 0 \end{cases}$$

représente sa forme canonique.

**Remarque 2.3.8** *a - On peut se ramener de la forme standard à la forme canonique par l'équivalence suivante :*

$$Ax = b \Leftrightarrow Ax \leq b \text{ et } Ax \geq b.$$

*b - De même en utilisant la variable d'écart, on peut se ramener de la forme canonique à la forme standard comme suit :*

$$Ax \leq b \Longrightarrow Ax + y = b \text{ avec } y \geq 0$$

*ou*

$$Ax \geq b \Longrightarrow Ax - y = b \text{ avec } y \geq 0.$$

### 2.3.2 Dualité

#### 1) Définition & Exemple

À chaque programme linéaire set associé un autre programme linéaire appelé dual $(PD)$.

Le dual du programme linéaire standard $(PP)$ est :

$$(PD) \begin{cases} \max b^t y \\ A^t y \leq c \\ y \in \mathbb{R}^m \end{cases}$$

**Définition 2.3.9** *À partir de la relation primale-duale définie pour la forme standard, il est possible de retrouver le programme dual de tous les programmes linéaires non formulés sous cette forme standard,.*

**Exemple 2.3.10** *par exemple : le dual de programme linéaire suivant :*

$$(PL) \begin{cases} \min c^t x \\ Ax \geq b \\ x \geq 0 \end{cases}$$

*Est le programme linéaire :*

$$(PD) \begin{cases} \max b^t y \\ A^t y \geq c \\ y \leq 0. \end{cases}$$

**Remarque 2.3.11** *a - Les contraintes du dual sont en bijection avec les variables du primal.*

*b - Les variables du dual sont en bijection avec les contraintes du primal.*

*c - Le dual du dual est primal.*

**Exemple 2.3.12** *Soit le programme primal :*

$$\begin{cases} \min(x_1 + x_2) \\ x_1 - x_2 = 0 \\ x_1 + x_2 + x_3 = 1 \\ x \geq 0,\ i = 1,\ 2,\ 3. \end{cases}$$

*Le dual de ce programme s'écrira :*

$$\begin{cases} \max y_2 \\ y_1 + y_2 \leq 0 \\ -y_1 + y_2 \leq 1 \\ y_2 \leq 0 \\ y_i \in \mathbb{R}^2. \end{cases}$$

*Les solutions des programmes primaux et duaux sont liées par les théorèmes de la dualité faible et forte.*

**Théorème 2.3.13** ***(Dualité faible) :*** *Si $x$ est une solution réalisable primale de $(PP)$ et $y$ est une solution réalisable dual de $(PD)$, alors :*

$$b^t y \leq c^t x.$$

**Théorème 2.3.14** ***(Dualité forte) :*** *Si $x^*$ est une solution optimale pour le programme primal $(PP)$ alors le programme dual $(PD)$ a une solution optimale $y^*$, telle que :*

$$b^t y^* = c^t x^*.$$

**2) Application de dualité**

Signalons l'importance de la dualité sur le plan pratique que théorique.

En effet, en plus de l'éclairage qu'elle apporte sur certains points d'ordre théorique permettant de mieux caractériser les propriétés du problème donné, les résultats précédents montrent qu'en résolvant un programme linéaire primal $(PP)$, on résout en même temps son dual $(PD)$.

### 2.3.3 Méthode de résolution du programme primal

**1) Méthode du Simplexe de Dantzig**

Cette méthode est une procédure qui examine les points extrémaux (réalisables) d'une façon convenable, le passage d'un sommet à un autre entraîne souvent une diminution de l'objectif si le problème est un problème de minimisation.

Lorsqu'il n'est plus possible de passer à un sommet adjacent sans diminuer la valeur de l'objectif, on stoppe l'algorithme. Dontzig a démontré d'une part que cela se produit toujours après un nombre fini d'itérations, et d'autre part que l'on a alors forcément atteint un optimum, ce qui achève de prouver la validité de la méthode.

Bien que très efficace en pratique (il est rare que le nombre d'itérations dépasse $n + m$), cette méthode ne satisfait pas les théoriciens de la complexité. En effet, il a été

prouvé que cet algorithme a une complexité de pire cas de type exponentiel (complexité théorique est de l'ordre de $2^n$ itérations).

Cela signifie qu'il existe des cas où le nombre d'itérations nécessaires pour atteindre la solution est exponentiel en fonction de la taille du problème ($n$). En fait, au moment de la découverte de la méthode du simplexe, personne ne savait si un algorithme de type polynomial (donc théoriquement meilleur que ceux de type exponentiel) existait pour la programmation linéaire.

**2) Méthode de point intérieur de Karmarkar**

A la différence de l'algorithme du simplexe, les méthodes de point intérieur partent d'un point situé à l'intérieur du polyèdre réalisable. Pour être plus précis, ces méthodes nécessitent, généralement, un point de départ strictement intérieur.

Les méthodes de point intérieur ont été relancées en 1984 par Karmarkar. Cette méthode opère l'intérieur de l'ensemble des contraintes $S$. L'atout principal de ces méthodes réside dans leur convergence polynomiale.

Actuellement, les chercheurs confirment que ces méthodes sont plus efficaces que les méthodes simpliciales pour les problèmes de grandes tailles [8]. Plusieurs alternatives sont proposées dans la littérature à savoir :

- Les méthodes projectives.
- Les méthodes de trajectoire centrale.
- Les méthodes de mise à l'échelle affine.
- Les méthodes de réduction du potentiel.

# Chapitre 3

# Méthodes de point intérieur pour la programmation linéaire

## 3.1 Introduction

### 3.1.1 Préambule

Actuellement, les méthodes de point intérieur sont devenues compétitives à la méthode du simplexe sur tout pour les problèmes de grande taille ($>$ 10000 variables ou contraintes) [8] . Il y a, principalement, trois grandes catégories de méthode de point intérieur à savoir :

- Les méthodes affines.

- Les méthodes de trajectoire centrale.

- Les méthodes de réduction du potentiel.

Ces méthodes jouent un rôle important dans l'étude et la résolution des pro-

grammes mathématiques à grandes dimensions. Elles sont développées lorsque Karmarkar (1984) a proposé un algorithme polynomial de complexité $O\left(n^{3.5}L\right)$ efficace en pratique basé une méthode de point intérieur. L'intérêt pour se méthodes, principalement. Développées depuis les années 60 par Dikin (1967) [1] et Fiacco et McCrrimick (1968) [6], a connu un renouveau pour les problèmes non linéaires et a ouvert un nouveau domaine pour les problèmes linéaire. La distinction entre la programmation linéaire et non linéaire n'était plus aussi nette.

Den Hertog (1994), a classé les méthodes de point intérieur en quatre catégories :

- Méthodes affines.

- Méthodes de réduction du potentiel.

- Méthodes de trajectoire centrale.

- Méthodes projectives avec potentiel.

### 3.1.2 Notion fondamentale

Soit le programme linéaire standard $(PL)$, son dual standard est le problème :

$$(PD)\begin{cases} \max b^t y \\ A^t y + s = c \\ s \geq 0. \end{cases}$$

## 3.2 Méthode affines

La première méthode affine a été proposée par Dikin (1967), pour la programmation linéaire et la programmation quadratique, sans tenter de prouver de la polynomialité.

Ces méthodes (affine scaling méthods), possèdent une interprétation géométrique

en tant qu'algorithme du gradient dans un espace mis à l'échelle, correspondent au cas où l'on fixe le paramètre $t$ à zéro pour chaque itération.

Nous allons à présent décrire comment s'effectue le choix du coefficient $a_k$.

**Propriétés de la méthode :**

1 - Elle est simple et ne nécessite ni transformation projective, ni fonction potentielle, ni préparation du problème.

2 - C'est une méthode primale' performante en pratique, quoique sensible au choix initial.

3 - Il n'y a pas de démonstration de complexité polynomiale, excepte pour une version primale - duale dans un cas très spécial [9].

4 - La démonstration de la convergence est, relativement, simple dans le cas non dégénérant.

## 3.3 Méthode de trajectoire centrale

De bonnes algorithmes et de bonnes preuves sont données, par les méthodes plus attractives en théorie et les plus utilisées en pratique. Elles progressent suivant une trajectoire, dite centrale.

Les méthodes de trajectoire centrale (path-following méthods) sont articulées autour du chemin central. Elles se caractérisent par un choix du paramètre $t$ différent de zéro. Leur principe revient à définir un certain voisinage autour du chemin central. et à faire évoluer les itérés à l'intérieur de ce voisinage tout en progressant vers la solution.

**Propriétés de la méthode :**

1 - Les performances numériques sont meilleures que celles des méthodes affines.

2 - La complexité polynomiale est garantie (le terme barrière joue le rôle d'un potentiel). En effet, le saut de dualité est réduit (d'une itération à l'autre) d'un montant fixe, jusqu' à la convergence, après $O(\sqrt{n})$ itérations.

## 3.4 Méthode de réduction du potentiel

Ces méthodes sont typiquement utilisées, pour résoudre le programme suivant :

$$(PRP)\begin{cases} \min f(x,\ y,\ s) = q\ln(c^t x - b^t y) - \sum_{i=1}^{n} \ln(x_i) \\ Ax = b,\ x > 0 \\ A^t y + s = c,\ s \geq 0 \end{cases}$$

Où la fonction objective $f(x,\ y,\ s)$, est appelée fonction potentielle, et $q$ un paramètre.

Les méthodes de réduction de potentiel (potentiel réduction méthodes) prennent des pas de Newton de la même forme que ceux des méthodes de trajectoire centrale. Cependant, ces méthodes ne suivent pas explicitement le chemin centrai. Elles utilisent une fonction potentielle logarithmique.

**Propriétés de la méthode**

1 - Elle n'a pas la simplicité des méthodes affines.

2 - Elle est, cependant, plus attractive pour au moins les deux raisons suivantes :

* Sa complexité polynomiale est garantie.

* Elle peut devenir très efficace, avec une bonne recherche linéaire sur la fonction potentielle.

3 - C'est une méthode primale-duale (quoiqu'il existe des versions primales et d'autres duales).

4 - Elle n'a pas reçu beaucoup d'attention au niveau des teste numériques. Ceci est peut être dû aux difficultés algorithmiques rencontrées au départ.

5 - Les transformations projectives ne sont pas nécessaires pour cette méthode ni en théorie, ni en pratique.

**Remarque 3.4.1** *Il existe plusieurs méthodes de réduction du potentielle, certaines utilisent la fonction ci-dessus, d'autres ajoutent le terme :*

$$-\sum_{i=1}^{n} \ln(s_i).$$

*Toutes ces méthodes visent, cependant, à ramener la fonction potentielle à* $-\infty$ *par des outils algorithmiques : primaux, duaux ou primaux-duaux.*

## 3.5 Méthode projective de Karmarkar

En 1984, Karmarkar ouvre un nouveau volé dans la programmation linéaire et les méthodes de point intérieur. L'algorithme de Karmarkar est, donc certainement, l'un des progrès les plus significatifs dans le domaine de la programmation mathématique.

### 3.5.1 Problème linéaire traité par Karmarkar

La méthode de Karmarkar est destinée à résoudre les problèmes de la forme :

$$(PK)\begin{cases} \min c^t x = z^* = 0 \\ Ax = 0 \\ x \in S_n = \left\{ x \in \mathbb{R}^n \ / \ e_n^t x = 1, \ x \geq 0 \right\} \end{cases}$$

où $c \in \mathbb{R}^n$ (vecteur coût), $b \in \mathbb{R}^m$, $A$ est une matrice $m \times n$ réelle de plein rang ($rgA = m \times n$) et $e_n^t = (1, 1, ..., 1)^t \in \mathbb{R}^n$.

Ayant comme solution strictement réalisable le center du simplexe $S_n$ :

$$x^{\circ} = \frac{e_n}{n}.$$

**Remarques**

1 - Si le système des contraintes n'est pas homogène ($Ax = b$, et $b \neq 0$), ou si la valeur optimale est connue et non nulle, l'égalité

$$e_n^t x = 1$$

permet de se ramener facilement à la forme $(PK)$. En effet, il suffit d'écrire :

$$Ax = b$$

sous la forme :

$$(Ax = be_n^t x) \Rightarrow \left[ (A = be_n^t)x = 0 \text{ avec } \tilde{A} = A - be_n^t \right].$$

Ainsi l'opérateur précédent devient :

$$\tilde{A}x = 0.$$

De la même manière, si :

$$c^t x = z^*.$$

Multiplions le $2^{ème}$ membre de l'equation précédente par :

$$e_n^t x = 1$$

pour obtenir :

$$(c^t x - z^* e_n^t x = 0) \Rightarrow \left[(c^t - z^* e_n^t)x = 0)\right].$$

Posons :

$$(\tilde{c}^t = c^t - z^* e_n^t) \Rightarrow (\tilde{c}^t x = 0).$$

2 - L'ignorance de la valeur optimale, ne constitue pas une difficulté, car il existe plusieurs variantes qui approximent cette valeur.

3 - Le problème $(PK)$. Peut être vu comme suit :

Trouver $x \geq 0$ tel que :

$$\begin{cases} c^t x = 0 \\ Ax = 0 \\ e_n^t x = 1 \\ x \geq 0. \end{cases}$$

### 3.5.2 Algorithme de Karmarkar

#### 1) Transformation projective de Karmarkar

D'après les opinions de plusieurs auteurs, la transformation projective joue un grand rôle pour comprendre les méthodes de point intérieur en générale. Pour cela, Kar-

markar utilise à chaque itération la transformation projective $T_k$ :

$$\begin{aligned} T_k \ : \ & S_n \longrightarrow S_n \\ & x \longmapsto y = T_k(x). \end{aligned}$$

Où

$$T_k(x) = \frac{D_k^{-1}x}{e_n^t D_k^{-1}x}$$

et

$$x = T_k^{-1}(y) \text{ avec } T_k^{-1}(y) = \frac{D_k y}{e_n^t D_k y}$$

tel que :

$$D_k = diag(x^k).$$

$X = diag(x)$ est la matrice diagonale dont les éléments diagonaux sont les composantes du vecteur $x$, d'où $Xe_n = x$. $X^{-1}$ est la matrice inverse de $X$.

On présente l'algorithme de Karmarkar pour résoudre un problème de la forme $(PK)$.

A partir de la solution initiale $x^0$, l'algorithme construit une suite de point intérieur qui converge vers une solution optimale du problème $(PK)$.

Pour ramener l'objectif $(c^t x)$ à zéro, on le projeté par la projection $p_k$ et on le minimise localement sur une sphère inscrite dans la région admissible du problème $(PK)$.

A chaque itération $k$, l'itéré $(x^k > 0)$ est renvoyé au centre du simplex $S_n$ par la transformation $T_k$ et ainsi de suite, jusqu'à ce que le test d'optimalité $(c^t x \leq \varepsilon)$ soit vérifié.

Pour plus de détails sur la description de la méthode, une présentation très détaillée de la méthode de Karmarkar est décrite dans [5] et [7].

**2) Algorithme de karmarkar**

**Début de l'algorithme**

**Initialisation :**

$$x^0 = \frac{e_n}{n},\ k = 0$$

**Tant que** $c^t x > \varepsilon$ **faire :**

**Étape 0 :** Construire la matrice des contraintes $B_k$.

$$D_k = diag(x^K)$$

$$A_K = AD_K$$

$$B_k = \begin{bmatrix} A_K \\ e_n^t \end{bmatrix}$$

**Étape 1 :** Calculer la projection $p_k$.

$$\begin{aligned} p_k &= \left[I - B_k^t (B_k B_k^t)^{-1} B_k\right] D_k c \\ &= \left[I - A_k^t (A_k A_k^t)^{-1} A_k - \frac{1}{n} e_n e_n^t\right] D_k c \end{aligned}$$

**Étape 2 :** Normaliser la projection $p_k$.

$$d^k = \frac{p_k}{\|p_k\|}$$

**Étape 3 :** Calculer l'itéré suivant :

$$y^k = \frac{e_n}{n} - \alpha r d^k;\ r = \frac{1}{\sqrt{n(n-1)}} \text{ avec } 0 < \alpha < 1.$$

$\alpha$ est le pas de déplacement.

**Étape 4 :** Revenir au problème initial par $T_k^{-1}$

$$x^{k+1} = T_k^{-1}(y^k) = \frac{D_k y^k}{e_n^t D_k y^k};\ k = k+1.$$

**Fin tant que**.

**Fin algorithme.**

**Remarque 3.5.1** *Le calcul de la projection $p_k$ constitue l'opération la plus coûteuse dans l'algorithme. L'efficacité pratique de l'algorithme dépend, en grande partie, de la manière du calcul de $p_k$. De même, la vitesse de convergence de l'algorithme dépend du pas de déplacement. En effet, plus $\alpha$ est grand ($\alpha > 1$ tout en conservant la faisabilité stricte) plus l'algorithme converge vite.*

### 3.5.3 Fonction potentielle & convergence

Pour établir la convergence de la l'algorithme, il faut montrer que :

$$\frac{c^t x^{k+1}}{c^t x^k} < q_0$$

où $0 < q_0 < 1$ est indépendant de $k$.

Ou, il est difficile de trouver directement $q_0$. Pour surmonter cette difficulté, Karmarkar associe à l'algorithme linéaire $c^t x$ la fonction potentielle logarithmique suivante :

$$f(x) = \sum_{i=1}^{n} \log\left[\frac{c^t x}{x_i}\right].$$

Définir sur l'ensemble de la stricte réalisabilité :

$$\mathring{F} = \left\{x \in \mathbb{R}^n,\ x > 0,\ Ax = 0,\ e_n^t x = 1\right\}.$$

### 3.5.4 Étude des performances de l'algorithme de Karmarkar

**1) Détermination du pas de déplace**

ment :

Le pas $\alpha$ peut être choisi de deux manières différentes :

a) $\alpha$ **constant :**

1) Karmarkar a supposé $\alpha = \frac{1}{4}$ et a montré que l'algorithme converge en un nombre polynomial ($O(nq + n \log n)$) d'itérations.

2) Padberg a amélioré le pas en modifiant légèrement la fonction potentielle comme suit :

$$h(x) = \frac{c^t x}{(\prod_{i=1}^{n} x_i)^{\frac{1}{n}}}.$$

Il a montré que l'algorithme converge après ($O(nq)$) itérations pour $\alpha = 0.7968$.

b) $\alpha$ **variable :** on distingue deux types :

i) Recherche linéaire : on minimise sur $[0, \alpha_{\max}]$, la fonction potentielle $\varphi(\alpha) = f(y(\alpha))$ où :

$$y(\alpha) = \frac{e_n}{n} - \alpha r d^k.$$

ii) Procédé technique [5] : prend $\alpha$ le plus grand possible qui vérifie :

$$\begin{cases} c^t x^{k+1} < c^t x^k \text{ monotonie} \\ y(\alpha) > 0 \text{ faisabilité} \end{cases}$$

ce qui donne :

$$\alpha = \begin{cases} \beta\alpha_{\max} \text{ si } (\mu - \rho c^t x^k) > 0 \\ -\beta\alpha_{\max} \text{ si } (\mu - \rho c^t x^k) \leq 0 \\ 0 < \beta < 1 \end{cases}$$

où :

$$\alpha_{\max} = \min\left[\left(\frac{1}{nrd_i^k}\right),\ i \in I^+\right], I^+ = \left\{i \in I :\ d_i^k > 0\right\}, I = \{1,\ ...,\ n\}$$

$$\mu = nrc^t x^k D_k d^k$$

$$\rho = nre_n^t D_k d^k.$$

**2) Calcul de la projection** $p_k$ :

On a :

$$p_k = \left[I - A_k^t (A_k A_k^t)^{-1} A_k - \frac{1}{n} e_n e_n^t\right] D_k c$$

posons :

$$u = (A_k A_k^t)^{-1} A_k D_k c$$

donc

$$A_k A_k^t u = A_k D_k c.$$

Par conséquent, la calcul de $p_k$, repose en grande partie sur la résolution du système d'équations linéaires précédent.

**Méthode de résolution du système** $(A_k A_k^t u = A_k D_k c)$ :

Les méthodes de résolution des systèmes linéaires sont de deux sortes :

1) Les méthodes directes : factorisation, par exemple : méthode de Croût, élimination de Gauss, Cholesky...etc.

2) Les méthodes itération : multiplication du type matrice-vecteur, par exemple : Jacobi, Gauss-Seidel...etc.

**Propriété du système su considéré :** La matrice $A_k A_k^t$ du système est symétrique, définie positive, les méthodes les plus réputées dans ce cas sont :

a) La factorisation de Cholesky qui est une méthode directe, cas où la dimension du système su considéré est petite.

b) Les méthodes du type gradient conjugué. Sont des méthodes itératives. Correspond au cas où la dimension du système étudié est importante.

**3) Propriétés de la méthode**

i) Elle nécessite la préparation du problème dans le cas général sur la forme $(Pk)$.

ii) Son comportement numérique est signifiant, surtout pour les grandes dimensions [8].

iii) C'est une méthode polynomiale.

iiii) L'algorithme de Karmarkar est un algorithme du gradient projeté, avec dilatation d'espace à chaque itération.

### 3.5.5 Généralisation de l'algorithme de Karmarkar

Soit le programme linéaire générale écrit sous la forme standard suivante :

$$(PL)\begin{cases} \min c^t x \\ Ax = b \\ x \geq 0 \end{cases}$$

où :

$$A \in \mathbb{R}^{m\times n},\ b \in \mathbb{R}^{m},\ c \in \mathbb{R}^{n}.$$

La transformation projective de Karmarkar est définie par :

$$T_a :\ \mathbb{R}^{n}_{+} \longrightarrow \mathbb{R}^{n+1}_{+}$$

$$x \longmapsto y = T_a\,(x)$$

tel que :

$$y = T_a(x) = \begin{cases} y_i = \dfrac{\frac{x_i}{a_i}}{1+\sum\limits_{i=1}^{n}(\frac{x_i}{a_i})} \text{ avec } i = 1,\ ...,\ n \\ \\ y_{n+1} = 1 - \sum\limits_{i=1}^{n} y_i. \end{cases}$$

Nous permet d'associer à $(PL)$ le programme linéaire suivant :

$$P(z^*) \begin{cases} \min(D_k c\ ,\ -z^*)^t y = 0 \\ \\ A_K y = 0 \\ \\ y \in S_{n+1} \end{cases}$$

où :

$$\begin{aligned} A_K &= (AD_k\ ,\ -b) \in \mathbb{R}^{m\times(n+1)} \\ D_k &= diag(x^k). \end{aligned}$$

Nous avons donc deux possibilités :

a) $z^*$ **connue :** on obtient la forme $(PK)$. Par conséquent, on utilise la transformation projective de Karmarkar.

b) $z^*$ **inconnue :** on approxime $z^*$ par l'une des trois possibilités suivantes :

i) Localisation $b^t y^k \leq z^* \leq c^t x^k$.

ii) Borne inférieure $b^t y^k = z^k \leq z^*$ condition de Todd-Burel.

iii) Borne supérieure $z^* \leq c^t x^k$ condition de Ye-Lustig.

Plusieurs technique sont proposées en vue de relaxer cette hypothèse et d'étendre ainsi l'algorithme à un programme plus général.

Parmi ces variantes, Karmarkar suggère deux méthodes :

- Borne inférieure $b^t y^k = z^k \leq z^*$ proposée par Todd et Burel, en modifiant l'algorithme de Karmarkar dans la valeur optimale $z^*$ par $z^K$.

- Borne supérieure $z^* \leq c^t x^k = z^k$ proposée par Ye-Lustig, en remplaçant $z^*$ à chaque itération par $z^k$.

Dans la suite de ce travail, on s'intéresse à la variable de Ye-Lustig. Ceci est dû aux études faites dans [5] qui affichent la supériorité de cette variante vis-à-vis de celle de Todd et Burel ou de celle de localisation. Cette caractéristique peut se résumer par les deux points suivants :

- Sa simplicité algorithmique.

- Son succès numérique vis-à-vis des autres variantes.

L'algorithme obtenu est présenté dans la section suivante.

### 3.5.6 Algorithme de Ye-Lustig

**Début algorithme**

**Initialisation :**

$$x > 0,\ k = 0$$

**Tant que** $\frac{\|d^k\|}{|c^t x^0|} > \varepsilon$ **faire :**

**Étape 1 :** Construire la matrice des contraintes $A_k$;

$$\begin{aligned} D_k &= diag(x^K) \\ A_k &= [AD_k, \ -b] \end{aligned}$$

**Étape 2 :** Calcul de projection $p_k$.

$$p_k = \left[I - A_k^t(A_k A_k^t)^{-1}A_K - \frac{1}{n}e_n e_n^t\right](D_k c, \ -c^t x^k)^t$$

**Étape 3 :** Normaliser la projection $p_k$.

$$d^k = \frac{P_k}{\|P_k\|}$$

**Étape 4 :** Calculer l'itéré suivant :

$$y^k = \frac{e_{n+1}}{n+1} - \alpha r d^k \text{ avec } \alpha \text{ le pas de déplacement.}$$

**Étape 5 :** Revenir au problème initial par $T_k^{-1}$.

$$x^{k+1} = T_k^{-1}(y^k) = \frac{D_k y^k[n]}{y_{n+1}^k}; \ k = k+1.$$

**Fin tant que.**

**Fin algorithme.**

## 3.6 Calcul d'une solution initiale strictement réalisable

Un point strictement réalisable de $(PL)$ est une solution du problème :

$$(SR)\{Ax = b, \ x > 0\}.$$

En utilisant la technique de la variable artificielle, le problème $(SR)$ est équivalent au problème suivent :

$$(P1)\begin{cases} \min \lambda \\ Ax + \lambda(b - Aa) = b \\ x > 0,\ \lambda \geq 0. \end{cases}$$

Tel que $a > 0$ est choisi arbitrairement dans l'orthant positif et  une variable artificielle. Posons $y = (x, \lambda)$, $(P1)$ est équivalent au programme linéaire :

$$(P2)\begin{cases} \min c^t x = z^* = 0 \\ \bar{A}x = b \\ y \geq 0 \end{cases}$$

où $c = (0, ..., 1)^t \in \mathbb{R}^{n+1}$, $\bar{A} = (A,\, b - Aa) \in \mathbb{R}^{n\times(n+1)}$.

$(P2)$ vérifie les hypothèses de Karmarkar :

1) $z^* = 0$.

2) $y = (a,\, 1)$ solution strictement réalisable de $(P1)$.

3) la matrice $\bar{A}$ est de plein rang, $rg(\bar{A}) = m < n + 1$.

La résolution de $(SR)$ se ramène à celle de $(P2)$. Plus précisément Karmarkar démontre le théorème suivant :

**Théorème 3.6.1** $\exists\ \varepsilon_0 > 0$ *tel que : les deux propositions suivantes sont équivalentes :*

*1) $x$ solution strictement réalisable de $(SR)$.*

*2) $(P2)$ admet une solution optimale $(x,\, \lambda)$ telle que $\lambda \leq \varepsilon_0$.*

## 3.7 Conclusion

Nous avons déjà signalé que l'opération la plus coûteuse, dans l'algorithme de Karmarkar y compris ses variantes, est le calcul de la projection à chaque itération. L'objectif du troisième chapitre est de remédier à cette difficulté. A cet égard, une modification a été mise en œuvre au niveau de la phase d'initialisation (phase 1) pour trouver une solution strictement réalisable [16]. Nous allons exploiter cette idée pour réduire le coût de l'itération au niveau de la phase 2 pour trouver une solution optimale de $(PL)$.

# Chapitre 4

# Amélioration de l'approche de Ye-Lustig

## 4.1 Position du problème

L'objectif de ce chapitre est l'extension de la modification introduite par D. Benterki & B. Merikhi [16] dans la phase 1 de l'algorithme de Ye-Lustig, et la modification introduite par A. Leulmi dans la phase 2 de l'algorithme de Karmarkar pour la phase 2 de l'algorithme de Ye-Lustig.

Ce chapitre est divisé en trois parties. Dans la première, on présente un bref résumé des travaux de D. Benterki & B. Merikhi, qui concerne la modification au niveau de la phase d'initialisation.

Soit le problème linéaire suivant :

$$(P)\begin{cases} \min c^t x \\ Ax = b \\ x \geq 0 \end{cases}$$

telles que : $c$, $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$.

## 4.2 Amélioration de la phase 1 dans l'approche de Ye-Lustig

### 4.2.1 Préambule

#### 1) Développement

Le calcul d'une solution strictement réalisable $x^0$, consiste à cherche $x$ vérifiant :

$$(PF)\begin{cases} Ax = b \\ x > 0 \end{cases}$$

le problème $(PF)$ est un problème de faisabilité (réalisabilité) stricte.

L'utilisation de la technique de la variable artificielle rend le problème $(PF)$ équivalent au problème d'optimisation suivant :

$$(PO)\begin{cases} \min \lambda \\ Ax + \lambda q = b \\ x \geq 0,\ \lambda \geq 0 \end{cases}$$

tel que : $q = b - Aa$ et $a \in \mathbb{R}^n_+$ arbitraire.

$(PO)$ est un programme linéaire sous forme standard qui peut se mettre comme

suit :

$$(\hat{P}\hat{L}) \begin{cases} \min \hat{c}^t \hat{x} \\ \hat{A}\hat{x} = b \\ \hat{x} \geq 0 \end{cases}$$

avec $c = (0,\ 0,\ ...,\ 0,\ 1)^t \in \mathbb{R}^{n+1}$, $\hat{x} = (x,\ \lambda)^t \in \mathbb{R}^{n+1}$ et $A = [Ab - Aa] \in \mathbb{R}^{m\times(n+1)}$.

Pour résoudre $(\hat{P}\hat{L})$, on utilise phase 2 de la méthode de Ye-Lustig (car $(a;\ 1)\ \forall a > 0$ est une solution strictement réalisable de $(\hat{P}\hat{L})$. L'algorithme correspondant se présente comme suit :

**2) Algorithme original**

**Début algorithme**

**Initialisation**

$$x^0 = a,\ \lambda^0 = 1,\ \hat{x}^0 = (x^0,\ \lambda^0)^t \text{ et } k = 0.$$

a)- **Si** $\left\|Ax^0 - b\right\| \leq \varepsilon$; **stop** : $x^0$est une solution approximative de $(PF)$ ;

**Sinon** aller à l'étape b.

b)- **Si** $\lambda^k \leq \varepsilon$ **stop** :$x^0$ est une solution approximative de $(PF)$ ;

**Sinon** aller à l'étape c.

c)- Poser $D_k = diag(\hat{x}^k)$, $B = \begin{bmatrix} A & b - Aa \end{bmatrix}$ ;

et

$$r = \frac{1}{\sqrt{(n+1)(n+2)}};$$

- $P_k = \left[I - B_k^t (B_k B_k^t)^{-1} B_k\right] \left[D_k \hat{c},\ -\hat{c}^t \hat{x}^k\right]^t$, tel que $B_k = [BD_k\ ,\ -b]$ ;

- prendre $y^{k+1} = \frac{e_{n+2}}{n+1} - \alpha^k r \frac{P_k}{\|P_k\|}$, tel que $\alpha^K$ est le pas de déplacement ;

- prendre

$$\hat{x}^{k+1} = T_k^{-1}(y^{k+1}) = (y_{n+2}^{k+1})D^k y^{k+1}\,[n+1]$$

d)- Faire $k = k + 1$ et retourner à l'étape b.

**Fin algorithme.**

### 4.2.2 Modification de l'algorithme de Ye-Lustig (phase 1)

**1) Développement**

Soit à l'itération $k$, on dispose d'une solution strictement réalisable $(x^k, \lambda^k)$ pour le problème $(PL)$. Pour trouver l'itération suivante $(x^{k+1}, \lambda^{k+1})$, dans l'algorithme classique, on calcul $P_k$, $\alpha_k$ puis on revient par $T_k^{-1}$. Au lieu de faire ces calculs, on essaye de trouver un vecteur $\omega^k$ qui vérifie :

$$\begin{cases} A(x^k + \omega^k) = b & (a) \\ x^k + \omega^k > 0 & (b) \end{cases}$$

d'où $x^k + \omega^k$ est une solution strictement réalisable pour $(PO)$.

♦ **Étape 1 :** Commençons par la condition $(a)$ :

cherchons $\omega^k$ tel que :

$$A(x^k + \omega^k) = Ax^k + A\omega^k = b.$$

Comme $(x^k, \lambda^k)$ est une solution strictement réalisable du problème $(\hat{P}\hat{L})$, alors on a :

$$Ax^k = b - \lambda^k q$$

donc, la condition $(a)$ devient : chercher $\omega^k$ tel que :

$$A\omega^k = \lambda^k q.$$

Comme $A \in \mathbb{R}^{m\times n}$, est une matrice de plein rang, i.e : $rgA = m$, le système ci-dessus admet aux moins une solution. Il suffit de calculer une. Parmi les quelles :

On cherche $\omega^k$ vérifiant :

$$(PQ)\left\{\min \left\|\omega^k - 0\right\|^2 :\ A\omega^k = \lambda^k q\right\}.$$

Le problème $(PQ)$ est un problème quadratique convexe. Pour calculer sa solution optimale, il suffit d'utiliser les conditions d'optimalité sur la fonction Lagrangienne :

$$\varphi(\omega^k,\ \theta) = \left\|\omega^k\right\|^2 + (A\omega^k - \lambda^k q)\theta^t,\ \theta \in \mathbb{R}^m.$$

En utilisant les conditions d'optimalités, on aura :

$$\begin{cases} A\omega^k = \lambda^k q & (c) \\ \omega^k + A^t\theta = 0 & (d) \\ A\omega^k + AA^t\theta = 0 & (e). \end{cases}$$

Où $(e)$ est une conséquence de $(d)$.

en la remplaçant dans $(c)$, on aura :

$$\theta = -(AA^t)^{-1}\lambda^k q$$

on en déduit de $(d)$ :

$$\omega^k = -A^t\theta = A^t(AA^t)^{-1}\lambda^k q.$$

**Remarque 4.2.1** *Le calcul de $\omega^k$ nécessite le calcul de l'inverse de la matrice $(AA^t)$, ce qui est indésirable en pratique. Pour remédier, on procède comme suit :*

*posons :*

$$u^k = (AA^t)^{-1}\lambda^k q$$

*qui équivalent au système linéaire symétrique, défini positif suivant :*

$$AA^t u^k = \lambda^k q.$$

**2) Avantage du système** $(AA^t u^k = \lambda^k q)$ :

Comme $q$ et $(AA^t)$ ne dépende pas de $k$ (constant), on résout une seule fois le système $AA^t u^k = \lambda^k q$ pour trouver $u^0$. Les autres vecteurs $u^k$ sont calculés, évidement, par la formule suivante :

$$u^k = \lambda^k u^0.$$

En effet :

d'après l'équation précédente, à l'itération, on peut écrire :

$$AA^t u^k = \lambda^k q.$$

De même à l'itération $(k+1)$, on a :

$$\begin{aligned} AA^t u^{k+1} &= \lambda^{k+1} q \\ &= \lambda^{k+1} \frac{1}{\lambda^k} AA^t u^k \end{aligned}$$

d'où:

$$u^{k+1} = \frac{\lambda^{k+1}}{\lambda^k} u^k.$$

Comme $\lambda^0 = 1$, on aura facilement :

$$u^k = \lambda^k u^0.$$

♦ **Étape 2** : Dans cette étape, on doit vérifier la condition $(b)$, c'est à dire que, $x^k + \omega^k > 0$.

Sous certaines conditions D. Benterki & B. Merikhi donnent une condition suffisante résumée par la proposition suivante :

**Proposition 4.2.2** *Pour tout* $(x^k,\ \lambda^k)$ *solution strictement réalisable de* $(\hat{P}\hat{L})$*, si* $\max_i \left|z_i^k\right| < 1$ *alors* $x^k + \omega^k > 0$ *et* $A(x^k + \omega^k) = b$.

*Tel que :*

$$z^k = -diag(\frac{1}{x^k})A^t u^k$$

*et* $\omega^k$ *est solution de* $(PQ)$.

**Preuve** a - On vérifie si $A(x^k + \omega^k) = b$ ?

Comme $\omega^k$ est une solution de $(PQ)$, on a :

$$\omega^k = A^t u^k$$

et

$$u^k = \lambda^k u^0$$

et comme $(x^k,\ \lambda^k)$ est une solution réalisable de $(\hat{P}\hat{L})$, alors :

$$\begin{aligned} A(x^k + A^t u^k) &= Ax^k + AA^t u^k \\ &= b - \lambda^k(b - Aa) + \lambda^k AA^t u^0 \\ &= b - \lambda^k(b - Aa) + \lambda^k\lambda^0(b - Aa) \\ &= b. \end{aligned}$$

Par conséquent :

$$A(x^k + \omega^k) = b$$

d'où le résultat.

b - On vérifie si $x^k + \omega^k > 0$ ?.

On a :

$$A(x^k + \omega^k) = b$$

donc :

$$A(diag(x^k)e_n + \omega^k) = b$$

alors :

$$A\ diag(x^k)(e_n + diag\left[(x^k)\right]^{-1}\omega^k) = b$$

comme $x^k > 0$, il suffit que :

$$e_n + diag\left[(x^k)\right]^{-1}\omega^k > 0$$

pour que $x^k + \omega^k > 0$, posons :

$$\begin{aligned} z^k &= -diag(\frac{1}{x^k})\omega^k \\ &= -diag(\frac{1}{x^k})A^t u^k. \end{aligned}$$

L'équation $e_n + diag\left[(x^k)\right]^{-1}\omega^k > 0$ devient :

$$e_n - z^k > 0.$$

Ce qui donne, alors :

$$\max_i \left|z_i^k\right| < 1$$

et par conséquent, on aura :

$$x^k + \omega^k > 0.$$

■

**3) Algorithme modifié**

**Début algorithme**

**Initialisation**

$$x^0 = a,\ \lambda^0 = 1,\ \text{et}\ k = 0.$$

*i*)- **Si** $\left\|Ax^0 - b\right\| \leq \varepsilon$; **stop** : $x^0$est une solution approximative de $(PF)$ ;

**Sinon** calculer $u^0$ solution du système linéaire :

$$AA^t u^0 = \lambda^0 (b - A).$$

*ii*)- **Si** $\lambda^k \leq \varepsilon$; **stop** : $x^0$ est une solution approximative de $(PF)$ ;

**Sinon**

- prendre $u^k = \lambda^k u^0$.

- prendre $z^k = -diag(\frac{1}{x^k})A^t u^k$,

- **Si** $\max\limits_i \left|z_i^k\right| < 1$; **stop** : $x^k + A^t u^k$ est une solution approximative de $(PF)$,

**Sinon** aller à l'étape $(iii)$.

*iii*)- Identique à étape $(c)$ de l'algorithme original.

*iiii*)- Faire $k = k + 1$ et retourner à l'étape $(ii)$.

**Fin algorithme.**

Pour calcul $u^0$ on utilise par exemple la méthode de Cholesky ou celle du gradient conjugué.

### 4.2.3 Tests numériques

Nous présentons à titre indicatif quelques exemples numériques testés par Benterki & Merikhi [16]. Dans les exemples ci-dessous, on pend $\varepsilon = 10^{-3}$ ou.$10^{-6}$.

**Exemples de taille fixes :**

| Exemple | Taille $m \times n$ | Nbr d'itér algorithme original | Nbr d'itér algorithme modifié |
|---|---|---|---|
| 1 | $3 \times 5$ | 4 | 1 |
| 2 | $3 \times 6$ | 3 | 1 |
| 3 | $5 \times 11$ | 5 | 4 |
| 4 | $6 \times 12$ | 3 | 1 |
| 5 | $6 \times 12$ | 4 | 3 |
| 6 | $16 \times 27$ | 6 | 5 |
| 7 | $11 \times 28$ | 7 | 6 |

**Exemples de taille variables :**

a) **Exemple cube** $n = 2m$, $A[i,\ j] = 0$ si $i \neq j$ ou $(i+1) \neq j$.
$A[i,\ j] = A[i,\ i+m] = 1$, $b[i] = 2$, pour $i$, $j = 1...m$

| Taille $m \times n$ | Nbr d'itér algorithme original | Nbr d'itér algorithme modifié |
|---|---|---|
| $50 \times 100$ | 3 | 1 |
| $100 \times 200$ | 3 | 1 |
| $150 \times 300$ | 3 | 1 |
| $200 \times 400$ | 3 | 1 |

b) **Exemple Hilbert :**
$n = 2m$, $A[i,\ j] = \frac{1}{i+j}$, $A[i,\ i+m] = 1$,
$b[i] = \sum_{j=1}^{m} \frac{1}{i+j}$, pour $i$, $j = 1...m$

| Taille $m \times n$ | Nbr d'itér algorithme original | Nbr d'itér algorithme modifié |
|---|---|---|
| $50 \times 100$ | 3 | 1 |
| $100 \times 200$ | 3 | 1 |
| $150 \times 300$ | 3 | 1 |
| $200 \times 400$ | 3 | 1 |

## 4.3 Amélioration de la phase 2 dans l'approche de Karmarkar

### 4.3.1 Préambule

Dans cette partie, nous allons présenter une modification dans l'algorithme de Karmarkar introduite par A. Leulmi au niveau de la phase 2 pour calculer la solution optimale $x^*$ du problème $(PL)$.

Cette étude nécessite la préparation du problème sous la forme $(P)$.

### 4.3.2 Préparation du problème

Soit le problème :

$$(PL)\begin{cases} \min\ c^t x = z^* \\ Ax = b \\ x \geq 0 \end{cases}.$$

Le problème $(PL)$ est équivalent au problème de faisabilité (réalisabilité).Ils 'agit de chercher $x$ qui vérifie :

$$(FO)\begin{cases} A_0 x = b_0 \\ x \geq 0 \end{cases}$$

où

$$A_0 = \begin{bmatrix} A \\ C^t \end{bmatrix} \in \mathbb{R}^{(m+1)\times n}$$

et

$$b_0 = \begin{bmatrix} b \\ z^* \end{bmatrix} \in \mathbb{R}^{m+1}.$$

Nous utilisons la technique de la variable artificielle, le problème $(FO)$ est équivalent au problème d'optimisation suivant :

$$(APO)\begin{cases} \min\ \lambda \\ A_0 x + \lambda q = b_0 \\ x \geq 0,\ \lambda \geq 0. \end{cases}$$

Tel que $q = b_0 - A_0 a$ et $a \in \mathbb{R}^n_+$ arbitrairement donné.

Pour résoudre $(APO)$, on doit utiliser la modification introduite dans l'algorithme modifié de la phase 1.

## 4.4 Amélioration de la phase 2 dans l'approche de Ye-Lustig

### 4.4.1 Préambule

Dans la troisième partie, nous présentons de la phase 2 pour calculer la solution optimale $x^*$de l'algorithme de Ye-Lustig, qui évité le calcul de la projection à chaque itération.

Cette étude nécessite la préparation du problème sous la forme $(P)$, et pour cela, Deux possibilités peuvent se présenter :

### 4.4.2 Préparation de problème

$1^{ère}$ **possibilité** :

Dans ce cas, on approxime $z^*$ à chaque itération par une borne supérieure, en prenant :

$$c^t x^k = z^*.$$

Le problème $(PL)$ est équivalent au problème suivant :

Chercher $x$ vérifiant :

$$\begin{cases} c^t x = c^t x^k \\ Ax = b \\ x \geq 0 \end{cases}$$

$k$ : représente l'itération à l'étape $k$ dans l'algorithme, $k = 0, 1, 2, ...$

Ce problème est équivalent au problème de faisabilité suivant : Chercher $x$ qui vérifie :

$$(F1) \begin{cases} A_1 x = b_1 \\ x \geq 0 \end{cases}$$

tels que :

$$A_1 = \begin{bmatrix} A \\ C^t \end{bmatrix} \in \mathbb{R}^{(m+1)\times n}$$

et

$$b_1 = \begin{bmatrix} b \\ c^t x^k \end{bmatrix} \in \mathbb{R}^{m+1}.$$

Le problème $(F1)$ est équivalent au problème d'optimisation suivant :

$$(AP1) \begin{cases} \min\ \lambda \\ A_1 x + \lambda q = b_1 \\ x \geq 0,\ \lambda \geq 0. \end{cases}$$

Tel que $q = b_1 - A_1 a$ et $a \in \mathbb{R}^n_+$ arbitrairement donné.

La résolution de $(AP1)$ se fait de la même manière que dans $(APO)$.

$2^{ère}$ **possibilité** :

Soit $x^k$ une solution réalisable (mais non optimale). Nous avons toujours :

$$c^t x \leq c^t x^k$$

donc le problème $(PL)$ est remplacé par une suite de problème qui vérifie :

$$(PK) \begin{cases} c^t x \leq c^t x^k \\ Ax = b \\ x \geq 0. \end{cases}$$

En introduisant une variable d'écart $x_{n+1}$, le problème $(PK)$ est équivalent au problème suivant :

$$(F2) \begin{cases} c^t x + x_{n+1} = c^t x^k \\ Ax = b \\ x \geq 0,\ x_{n+1} \geq 0 \end{cases}$$

$(F2)$ s'écrit aussi sous la forme :

$$(F2) \begin{cases} A_2 \bar{x} = b_2 \\ \bar{x} \geq 0 \end{cases}$$

où

$$A_2 = \begin{bmatrix} A & 0_{m,\,1} \\ c^t & 1 \end{bmatrix} \in \mathbb{R}^{(m+1)\times(n+1)}$$

$$b_2 = \begin{bmatrix} b \\ c^t x^k \end{bmatrix} \in \mathbb{R}^{m+1}$$

$$\bar{x} = \begin{bmatrix} x \\ x_{n+1} \end{bmatrix} \in \mathbb{R}^{n+1}.$$

Nous utilisons la technique de la variable artificielle, le problème$(F2)$ est équivalent au problème d'optimisation suivant :

$$(AP2) \begin{cases} \min\ \lambda \\ A_2\bar{x} + \lambda q = b_2 \\ \bar{x} \geq 0,\ \lambda \geq 0. \end{cases}$$

Tel que $q = b_2 - A_2 a$, $a \in \mathbb{R}^{n+1}$ et $a > 0$.

$(AP2)$ se résout dans de la même manière que $(APO)$ et $(AP1)$.

**Remarque 4.4.1** *Sur le plan pratique, la* $1^{ère}$ *possibilité apparaît plus intéressant que la* $2^{ère}$ *possibilité, car cette dernière, nécessite l'augmentation de la taille du problème* $(m,\ n)$ *à* $(m+1,\ n+1)$*, alors que dans la* $1^{ère}$ *possibilité la taille augmente de* $(m,\ n)$ *à* $(m+1,\ n)$*. Par conséquent, dans ce qui suit, on ne s'intéresse qu'à la* $1^{ère}$ *possibilité.*

### 4.4.3 Modification de l'algorithme de Ye-Lustig (phase 2)

**Développement**

Le problème donc, revient à résoudre un problème du type

$$(P\tilde{F}) \begin{cases} \tilde{A}\tilde{x} = \tilde{b} \\ \tilde{x} \geq 0 \end{cases}$$

où

$$(A1) \begin{cases} \tilde{A} = A_0 \\ \tilde{b} = b_0 \\ \tilde{x} = x \end{cases} \quad \text{dans le cas où } z^* \text{ est connue.}$$

et

$$(A2) \begin{cases} \tilde{A} = A_1 \\ \tilde{b} = b_1 \\ \tilde{x} = x \end{cases} \quad \text{dans le cas où } z^* \text{ est inconnue.}(1^{ère} \text{ possibilité}).$$

On suit les mêmes étapes faites dans la modification au niveau de la phase 1.

Le problème $(P\tilde{F})$ est équivalent (en utilisant la variable artificielle) au problème linéaire $(A\tilde{P})$ :

$$(A\tilde{P}) \begin{cases} \min\ \lambda \\ \tilde{A}\tilde{x} + \lambda q = \tilde{b},\ q = \tilde{b} - \tilde{A}a \\ (\tilde{x},\ \lambda) \geq 0. \end{cases}$$

Soit à l'itération $k$, on dispose d'un solution réalisable $(\tilde{x}^k,\ \lambda^k)$ pour le problème $(A\tilde{P})$. Pour trouver l'itération suivante $(\tilde{x}^{k+1},\ \lambda^{k+1})$, on cherche un vecteur $\omega^K$ qui vérifie :

$$\tilde{A}(\tilde{x}^k + \omega^k) = \tilde{b} \text{ et } \tilde{x}^k + \omega^k > 0.$$

♦ Étape 1 :

Cherchons $\omega^k$ tel que :

$$\tilde{A}(\tilde{x}^k + \omega^k) = \tilde{b}.$$

Comme $(\tilde{x}^k,\ \lambda^k)$ est une solution réalisable du problème $(A\tilde{P})$, alors le problème ci-dessus revient à : chercher $\omega^k$ tel que :

$$\tilde{A}\omega^k = \lambda^k q.$$

Ce système admet plusieurs solutions. Il suffit de calculer une, parmi lesquelles : on cherche $\omega^k$ vérifiant :

$$\left\{ \min \left\| \omega^k - 0 \right\|^2 :\ \tilde{A}\omega^k = \lambda^k q \right\}.$$

Considérons la fonction de lagrangienne :

$$\varphi\left(\omega^k,\ \theta\right) = \left\|\omega^k\right\|^2 + (\tilde{A}\omega^k - \lambda^k q)\theta,\ \theta \in \mathbb{R}^{m+1}.$$

En utilisant les conditions d'optimalité, on trouve la solution optimale sous la forme :

$$\omega^k = \tilde{A}^t(\tilde{A}\tilde{A}^t)^{-1}\lambda^k q.$$

**Remarque 4.4.2** *1- Le calcul de $\omega^k$ nécessite le calcul de l'inverse de la matrice $\tilde{A}\tilde{A}^t$, à chaque itération $k$. on procède comme suit :*

Posons :

$$u^k = (\tilde{A}\tilde{A}^t)^{-1}\lambda^k q$$

qui est équivalent à :

$$\tilde{A}\tilde{A}^t u^k = \lambda^k q.$$

On remarque que : le système précédent est un système linéaire symétrique défini positif.

2 - Dans le cas où $z^k$ est connue, $q$ et $(\tilde{A}\tilde{A}^t)$ sont constantes (cas 1), on résout une seul fois précédent pour trouver $u^0$, et les autres vecteurs $u^k$, on le calcule (évidement) par formule simple :

$$u^k = \lambda^k u^0.$$

♦ Étape 2 : Vérifiant si $\tilde{x}^k + \omega^k > 0$.

**Proposition 4.4.3** *Si $\max\limits_i \left|z_i^k\right| < 1$ alors $\tilde{x}^k + \omega^k > 0$ et $\tilde{A}(\tilde{x}^k + \omega^k) = \tilde{b}$.*

*Tel que :*

$$z^k = -diag(\frac{1}{\tilde{x}^k})\tilde{A}^t u^k.$$

***Preuve*** *Voir proposition* 32 *de la section* (4.2.2) *page* 45. ■

### 4.4.4 Algorithme modifié

**Présentation de l'algorithme si $z^*$connue (Karmarkar)**

**Début algorithme**
**Initialisation**

$$x^0 = a,\ \lambda^0 = 1,\ x^0 = (x^0,\ \lambda^0)^t \text{ et } k = 0.$$

**a')- Si** $\left\|A_0x^0 - b_0\right\| \leq \varepsilon$; **stop** : $x^k$ est une solution optimale de $(PL)$.
**Sinon** : Calculer $u^0$, la solution du système linéaire :

$$A_0A_0^tu^0 = \lambda^0(b_0 - A_0a).$$

**b')- Si** $\lambda^k \leq \varepsilon$ **stop** : $x^k$ est une solution optimale de$(PL)$.
**Sinon :**
- prendre $u^k = \lambda^k u^0$.
- prendre $z^k = -diag(\frac{1}{x^k})A_0^tu^k$.
- **Si** $\max\limits_i \left|z_i^k\right| \leq 1$ **stop** : $x^k + A_0^tu^k$ est une solution de $(PL)$.

**Sinon :** aller à l'étape **c'**.
**c')**- Identique à l'étape **c** de l'algorithme de original.
**d')-** Faire $k = k + 1$ et retourner à l'étape **b'**.
**Fin algorithme.**

**Présentation de l'algorithme si $z^*$ inconnue (Ye-Lustig)**

**Début algorithme**
**Initialisation**

$$x^0 = a,\ \lambda^0 = 1,\ x^0 = (x^0,\ \lambda^0)^t \text{ et } k = 0.$$

**a')- Si** $\left\|A_1x^0 - b_1\right\| \leq \varepsilon$; **stop** : $x^k$ est une solution optimale de $(PL)$.
**Sinon** : Calculer $u^0$, la solution du système linéaire :

$$A_1A_1^tu^0 = \lambda^0(b_1 - A_1a).$$

**b')- Si** $\lambda^k \leq \varepsilon$ **stop** : $x^k$ est une solution optimale de $(PL)$.
**Sinon :**
- prendre $u^k = \lambda^k u^0$.
- prendre $z^k = -diag(\frac{1}{x^k})A_1^tu^k$.
- **Si** $\max\limits_i \left|z_i^k\right| \leq 1$ **stop** : $x^k + A_1^tu^k$ est une solution de $(PL)$.

**Sinon :** aller à l'étape **c'**.
**c')**- Identique à l'étape **c** de l'algorithme de original.
**d')-** Faire $k = k + 1$ et retourner à l'étape **b'**.
**Fin algorithme**

### 4.4.5 Tests numériques

**1) $z^*$ connue (Karmarkar)**

**a - Exemples des tailles fixes**

**Exemple 4.4.4** *:*

$$A = \begin{bmatrix} -1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}, b = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \text{ et } c = \begin{bmatrix} -2 & -4 & 0 \end{bmatrix}^t.$$

*La valeur optimale est :* $z^* = -7$.

*La solution optimale exacte est :*

$$x^* = \begin{bmatrix} 0.5 & 1.5 & 0 \end{bmatrix}^t.$$

***Tableau comparatif :***

| ***Méthode*** | $\mathbf{x}^*$ | $\mathbf{z}^*$ | ***itér*** | ***temps (s)*** |
|---|---|---|---|---|
| ***Alg. original*** | $(\ 0.4999 \quad 1.4999 \quad 0)^t$ | $\mathbf{-6.9999}$ | **04** | **0.31** |
| ***Alg. modifié*** | $(\ 0.5000 \quad 1.5000 \quad 0\ )^t$ | $\mathbf{-7.0000}$ | **03** | **0.31** |

**Exemple 4.4.5** *:*

$$A = \begin{bmatrix} 2 & 3 & 1 & 2 \\ 3 & 0 & -2 & 1 \end{bmatrix}, \ b = \begin{bmatrix} 2 \\ 0 \end{bmatrix} \text{ et } c = \begin{bmatrix} 4 & 1 & 2 & 0 \end{bmatrix}^t.$$

*La valeur optimale est :* $z^* = 0.67$.

*La solution optimale exacte est :*

$$x^* = \begin{bmatrix} 0 & 0.67 & 0 & 0 \end{bmatrix}^t.$$

***Tableau comparatif :***

| *Méthode* | $x^*$ | $z^*$ | *itér* | *temps (s)* |
|---|---|---|---|---|
| *Alg. original* | $(\ 0 \quad 0.6666 \quad 0 \quad 0\ )^t$ | 0.6666 | 07 | 0.47 |
| *Alg. modifié* | $(\ 0.0004 \quad 0.6635 \quad 0.0022 \quad 0.0031\ )^t$ | 0.6700 | 03 | 0.31 |

**Exemple 4.4.6** *:*

$$A = \begin{bmatrix} 1 & -1 & 1 & 1 \\ 2 & 1 & -1 & 2 \\ 1 & 1 & 1 & 2 \end{bmatrix}, \ b = \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix} \text{ et } c = \begin{bmatrix} 3 & 2 & 1 & 3 \end{bmatrix}^t.$$

*La valeur optimale est :* $z^* = 7.67$.

*La solution optimale exacte est :*

$$x^* = \begin{bmatrix} 0.34 & 0 & 0.67 & 2 \end{bmatrix}^t.$$

***Tableau comparatif :***

| *Méthode* | $x^*$ | $z^*$ | *itér* | *temps (s)* |
|---|---|---|---|---|
| *Alg. original* | $(\ 0.3333 \quad 0 \quad 0.6666 \quad 1.9999\ )^t$ | 7.6666 | 04 | 0.32 |
| *Alg. modifié* | $(\ 0.3355 \quad 0.0011 \quad 0.6677 \quad 1.9977\ )^t$ | 7.6700 | 01 | 0.15 |

**Exemple 4.4.7** *:*

$$A = \begin{bmatrix} 2 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & -1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix},\ b = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}\ \text{et}\ c = \begin{bmatrix} 3 & -1 & 1 & 0 & 0 & 0 \end{bmatrix}^t.$$

*La valeur optimale est :* $z^* = -0.5$.

*La solution optimale exacte est :*

$$x^* = \begin{bmatrix} 0 & 0.5 & 0 & 0.5 & 0 & 0 \end{bmatrix}^t.$$

***Tableau comparatif :***

| *Méthode* | $x^*$ | $z^*$ | *itér* | *temps (s)* |
|---|---|---|---|---|
| *Alg. original* | $(\ 0 \quad 0.4999 \quad 0 \quad 0.4999 \quad 0 \quad 0\ )^t$ | $-0.4999$ | 09 | 0.78 |
| *Alg. modifié* | $(\ 0 \quad 0.4999 \quad 0 \quad 0.4999 \quad 0 \quad 0\ )^t$ | $-0.5000$ | 07 | 0.62 |

**Exemple 4.4.8** *:*

$$A = \begin{bmatrix} -1 & 1 & 1 & -1 & 1 & 0 & 0 \\ 0 & 2 & -3 & 2 & 0 & 1 & 0 \\ -3 & 2 & 1 & 0 & 0 & 0 & 1 \\ 3 & 5 & 4 & 0.5 & 0 & 0 & 0 \end{bmatrix},\ b = \begin{bmatrix} 1 \\ 2 \\ 0 \\ 2 \end{bmatrix}\ \text{et}\ c = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & -2 \end{bmatrix}^t.$$

*La valeur optimale est :* $z^* = 0$.

*La solution optimale exacte est :*

$$x^* = \begin{bmatrix} 0.5 & 0 & 0 & 1 & 2.5 & 0 & 1.5 \end{bmatrix}^t.$$

***Tableau comparatif :***

| ***Méthode*** | $x^*$ | $z^*$ | ***itér*** | ***temps (s)*** |
|---|---|---|---|---|
| ***Alg original*** | $\begin{pmatrix} 0.4999 & 0 & 0 & 0.9999 & 2.4999 & 0 & 1.4999 \end{pmatrix}$ | 0 | 10 | 0.78 |
| ***Alg modifié*** | $\begin{pmatrix} 0.5000 & 0 & 0 & 0.9999 & 2.4999 & 0 & 1.5000 \end{pmatrix}$ | 0 | 8 | 0.62 |

**Exemple 4.4.9** *:*

$$A = \begin{bmatrix} 1 & 0 & -4 & 3 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 5 & 3 & 1 & 0 & -1 & 3 & 0 & 1 & 0 & 0 & 0 & 0 \\ 4 & 5 & -3 & 3 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 2 & 1 & -5 & 0 & 0 & 0 & 1 & 0 & 0 \\ -2 & 1 & 1 & 1 & 2 & 2 & 0 & 0 & 0 & 0 & 1 & 0 \\ 2 & -3 & 2 & -1 & 4 & 5 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, b = \begin{bmatrix} 1 \\ 4 \\ 4 \\ 5 \\ 7 \\ 5 \end{bmatrix}$$

*et*

$$c = \begin{bmatrix} -4 & -5 & -1 & -3 & 5 & -8 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^t$$

*La valeur optimale est :* $z^* = 17$.

*La solution optimale exacte est :*

$$x^* = \left[\begin{array}{cccccccccccc} 0 & 0 & 2.5 & 3.5 & 0 & 0.5 & 0 & 0 & 05 & 05 & 0 & 1 \end{array}\right]^t.$$

*La solution optimale de l'algorithme original est :*

$$x^* = \left[\begin{array}{cccccccccccc} 0 & 0 & 2.5000 & 3.5999 & 0 & 0.4999 & 0 & 0 & 0.5000 & 0.4999 & 0 & 1.000 \end{array}\right]^t.$$

*La solution optimale de l'algorithme modifié est :*

$$x^* = \left[\begin{array}{cccccccccccc} 0 & 0 & 2.5000 & 3.5999 & 0 & 0.4999 & 0 & 0 & 0.4999 & 0.4999 & 0 & 1.000 \end{array}\right]^t.$$

***Tableau comparatif :***

| ***Méthode*** | $z^*$ | ***itér*** | ***temps (s)*** |
|---|---|---|---|
| ***Alg original*** | $-13.2499$ | 13 | 0.94 |
| ***Alg modifié*** | $-13.2500$ | 12 | 0.84 |

**b - Exemple de taille variable**

**Exemple 4.4.10** ***(Exemple cube)*** $n = 2m$, $A[i,\ j] = 0$ *si* $i \neq j$ *ou* $(i+1) \neq j$.

$A[i,\ j] = A[i,\ i+m] = 1$, $b[i] = 2$, pour $i,\ j = 1...m$.
La vecteurs optimale est : $z^* = -2m$.
La solution optimale exacte est :

$$x_i^* = \begin{cases} x^* = 2 \text{ pour } i = 1...m \\ x^* = 0 \text{ pour } i = 1+m...n \end{cases}$$

**Tableau comparatif :**

| Taille | Méthode | $z^*$ | itér | temps (s) |
|---|---|---|---|---|
| $50 \times 100$ | Alg original | $-100.0000$ | 4 | 0.47 |
| | Alg modifié | $-100.0000$ | 1 | 0.15 |
| $100 \times 200$ | Alg original | $-200.0000$ | 4 | 1.40 |
| | Alg modifié | $-200.0000$ | 1 | 0.31 |
| $150 \times 300$ | Alg original | $-299.9999$ | 4 | 3.60 |
| | Alg modifié | $-300.0000$ | 1 | 0.78 |
| $170 \times 340$ | Alg original | $-393.9998$ | 4 | 5.00 |
| | Alg modifié | $-340.0000$ | 1 | 1.25 |

**Commentaires**

Ces tests montrent clairement l'impact de modification sur le comportement numérique de l'algorithme, exprimé par la réduction du nombre d'itération et du temps de calcul.

De plus, on a présenté une légère modification de l'algorithme pour le problème de stricte faisabilité. Cette modification a réduit le nombre d'itération.

**2) $z^*$ inconnue(Ye-Lustig)**

Les expérimentations numériques de l'algorithme modifié cas où $z^*$ inconnue demande une étude plus profonde et beaucoup de soins. Car le choix du test d'arrêt qu'on a utilisé pour le moment ne répand pas d'avantage pour la résolution complète du problème.

# Chapitre 5

# Conclusions & perspectives

Malgré le caractère novateur des méthodes de point intérieur et de leurs résultats, Karmarkar n'imaginait, certainement pas, qu'il allait déclencher une si longue et si fructueuse série de recherches et de découvertes dans un domaine, si étudié, de l'optimisation mathématique.

Par ce modeste travail, nous comptons :

- avoir découvert le domaine est très passionnant,
- avoir dépassé le stade de la simple présentation descriptive,
- avoir apporté une compréhension plus ou moins profonde de la méthode proprement dite, de sa justification et des concepts qu'ils lui sont sous - adjacents.

Nous avons constaté que le domaine a beaucoup évolué. Les méthodes, les plus efficaces et les plus récentes, dépassent de loin celles développées au début des années quatre vingt.

L'emploi des méthodes de points intérieurs, développées en 1997, à départ non admissible, combinée d'une technique dite de Mehrota est considérée une innovation dans ce domaine par Karmakar et autres.

Cependant, certains auteurs préfèrent construire leur théorie sur d'autres formes que le couple primal - dual présenté dans notre travail.

Actuellement, la méthode de points intérieurs peut être considérée qu'elle commence à atteindre sa maturité et ce, par l'apparition des références faisant suite aux innombrables publications de recherche.

La méthode de point intérieur est plus ardue que celle de l'algorithme du simplexe.

Sur la base des travaux de D. Benterki et B Merikhi [16], on est arrivé à proposer une modification au niveau de la phase 2 de l'algorithme originel de Karmarkar.

Cette modification a permis de réduire, considérablement, le nombre ainsi que le temps de calcul des itérations. Les tests numériques effectués confirment l'efficacité de cette modification.

De plus, une étude préliminaire concernant l'extension de cette procédure à d'autres covariantes, généralisant l'algorithme de Karmarkar, a été initié dans l'algorithme de Ye-Lustig.

Cependant, il serait très intéressant de complète ce travail en s'appuyant de près sur le développement numérique de cette modification pour la variante de Ye-Lustig, et

estimer le nombre d'opérations de l'itération avec modification et le comparer avec celui de l'itération sans modification (complexité).

De même une étude théorique et numérique de cette modification peut être élargie pour d'autre problème tel que la programmation semi–définie ($SDP$), la programmation non linéaire ($PNL$).

# Bibliographie


[1] I.Dikin, "Iterative solution of prpbleme of linear and quadratic programming", Doklady Akademii Nauk SSSR, vol.174, 747-748, (1967).

[2] M.Minoux, "Programmation mathématique : théorie et algorithmes", tome 1, Dunod, Paris (1983).

[3] G. Opris, "Programmation linéaire", QPU, Algérie, (1983).

[4] N. Karmarkar , "A new polynomail-time algorithm for linear programming", Combinatorica, 373-395, 4 (1984).

[5] A.Kerghel, "Etude adaptative et comparative des principales variantes dans l'algorithme de Karmarkar", Dissertation thesis (1989), Univercité Joseph Fourier, Grenoble, France, (1989).

[6] A. V. Fiacco, G. P. McCormick, "Nonlinear Programming : Sequential unconcstrained minimization techniques", SIAM Classics in Applied Mathematics, Philadelphia, (1990).

[7] D. Benterki , "L'étude des performances de l'algorithme de Karmarkar", Thèse de magister, Université de Sétif, Algérie, (1992).

[8] I. J. Lustig, R. E. Marsten and D. F. Shanno, "Interior point method for lineair programming", Computational state of art, ORSA, Journal on Computing, 6(1), pp. 1-14. (1994).

[9] R. M. Frend, S. Mizuno, "Interior point methods :Current status and futur directions", Mathemaical Programming, N°51, Optima, (1996).

[10] S. J. Wright, "Prima - dual interior point methods", SIAM, philadelphia. PA, (1997).

[11] S. Boukaroura, "Etude du caractère newtonien dans l'algprithme de Karmarkar", Thèse de magistaire, Université de Sétif, Algérie, (1997).

[12] A. Hanachi, "Etude théorique et numérique d'une méthode de linéairisation de type Karmarkar pour la programmation quadratique convexe" Thèse de magister, Université de Sétif, Algérie, (1997).

[13] F. Gliveur, "Etudes des méthodes de point intérieur appliquée à la programmation linéaire et à la programmation semi - définie", Cours, 1997.

[14] A. Kerghel, D.Benterki, " Sur les performances de l'algorithme de Karmarkar pour la programmation linéaire", Revue Roumaine des sciences techniques - mécanique appliquée, Tome 46, n°.1. (2001).

[15] A. Kerghel, "Analyse convexe : théorie fondamentale et exercices", Editions Dar El Houda, Ain Mlila, Algérie, 2001.

[16] D. Benterki, B. Merikhi, Communicated by J. P. Crouzix, "A modified algorithm for the strict feasibility pboblem", RAIRO operation research, 395-399. (2001).

[17] G. Savard GERAD, "Introduction aux méthodes de point intérieur", Extrait de notes, Département de mathématiques et génie industriel, Ecole Polytechnique de Montréal, Fevrier 2001.

[18] P. L. Takouda, "Problèmes d'aproximation matricielle linéaire canonique : aproches par projections et via optimisation sous contraintes de sem-définie positivité", Thèse de doctdral, Université Paul Sabatier, Toulouse III. (2004).

[19] L. Menniche, "Problème d'initialisation et mise en oeuvre d'une méthode projective pour la programmation semi-définie linéaire", Thèse de magister, Université de Sétif Algérie.

[20] Leulmi. Assma, "Une procédure améliorante d'une méthode projective en programmation linéaire", Thèse de magister, Université Hadaik-Skikda Algérie.