REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE DE CONSTANTINE

FACULTE DES SCIENCES

DEPARTEMENT DES MATHEMATIQUES

N° D'ORDRE :
SERIE :

Mémoire
PRESENTE POUR L'OBTENTION DE DIPLOME DE MAGISTERE EN MATHEMATIQUES

OPTION
MATHEMATIQUES APPLIQUEES

THEME

UNE NOUVELLE APPROCHE SIMPLICIALE EN PROGRAMMATION LINEAIRE

PAR: HAMEURLAIN ABDELAZIZ

DEVANT LE JURY :

Président :	***Mr. M.DENCHE***	***Prof.***	***U .de Constantine***
Rapporteur :	***Mr. A.AYADI***	***M.C.***	***U. de Constantine***
Examinateurs :	***Mr. M.DEGHDAK***	***M.C.***	***U. de Constantine***
	Mr. M. BOUSHABA	***M.C.***	***U. de Constantine.***
	Mr. B.TENIOU	***M.C.***	***U. de Constantine***

Soutenu, le: 17/04/2006

Introduction générale

Le terme de «programmation linéaire » a été introduit en même temps que la méthode du simplexe par G.B. DANTZIG au lendemain de la seconde guerre mondiale.

Cette méthode est un outil mathématique très riche qui constitue la technique la plus célèbre de la recherche opérationnelle, elle a motivé toute seule (40) années de développements.

En (1979), le mathématicien soviétique L.G KHACHIAN a proposé un algorithme polynomial ; ce fut un grand succès théorique, malheureusement cet algorithme est beaucoup moins efficace que celui du simplexe malgré certaines améliorations de détail.

En (1984) KARMAKAR propose un algorithme polynomial lui aussi dont il proclamait (sans toutefois en fournir une preuve convaincante) la grande efficacité pratique, ce qui a provoqué quelques controverses dans la communauté de la programmation mathématique, bien qu'il a incité plusieurs recherches dans le domaine de la programmation linéaire.

Ceci étant, avec la méthode de KARMAKAR sont nées de nombreuses directions et idées de recherches particulièrement en optimisation. La méthode du simplexe et donc loin d'être périmée.

C'est une procédure itérative permettant d'effecteur une exploration dirigée de l'ensemble des points extrémaux, c'est à dire de l'ensemble des solutions réalisables. L'application de la méthode nécessite la connaissance d'une base réalisable de départ et à calculer à chaque itération une base adjacente (c'est à dire un point extrémal voisin) au moins aussi bonne que la précédente.

Connaître une base réalisable de départ est la difficulté principale de cette méthode.

L'algorithme de KARMAKAR minimise une fonction linéaire (qui vaut zéro à l'optimum) sur un simplexe régulier S qui est un ensemble de n-vecteurs non-négatifs (≥ 0) *dont la somme des composantes est égale à une constante (par exemple 1 ou n).*

A chaque itération, KARMAKAR utilise une transformation projective de S dans lui-même qui envoie la solution réalisable courante au centre de S. L'algorithme

trouve alors une direction de descente suivant laquelle on effectue un déplacement convenable à partir du centre. Finalement, on revient aux variables d'origine en utilisant l'inverse de la transformation projective. On répète le processus jusqu'à la vérification du test d'optimalité.

La difficulté principale de cette méthode est de supposer, au départ, connue la valeur optimale. Condition très restrictive en pratique. Plusieurs techniques sont proposées dans la littérature en vue de relaxer cette hypothèse et d'entendre ainsi l'algorithme à un programme générale.

L'objet de notre travail est la présentation d'une nouvelle méthode de résolution des problèmes linéaires (ne nécessitant pas la connaissance d'une base de départ) et la comparaison théorique et en terme d'efficacité numérique de celle-ci à celle du simplexe.

L'approche que nous présentons est inspirée de la méthode du simplexe qui coïncide même avec celle-ci si la base courante est réalisable.

Les trois chapitres constituent une étude allant du développement théorique à l'implantation numérique.

Le premier chapitre est consacré aux fondements théoriques et à la présentation de la méthode du simplexe.

Dans le deuxième chapitre on trouve la présentation générale de la méthode de KARMAKAR et son algorithme de base avec sa description.

Le troisième chapitre est consacré à la présentation de la nouvelle approche (que nous proposons) qui élimine l'hypothèse de connaître au départ une base réalisable.

Enfin, les expérimentations numériques et les commentaires sont exposés à la fin de ce dernier.

Notons que les résultats obtenus sont nettement en faveur de la nouvelle approche simpliciale proposée, du moins pour les exemples étudiés.

Notations générales:

Soient x et y deux vecteurs du IR^n*:*

La notation $<x,y>=x^t y$ *désigne le produit scalaire de x par y.*

e_n *désigne le vecteur (1,…,1) de* IR^n.

On note par $\sum_n xi$ *la somme des n composantes de x.*

Une matrice A à m lignes et n colonnes sera considérée comme un élément de $IR^{m\times n}$ *On écrit* $A\in IR^{m\times n}$.

Soient A de $IR^{m\times n}$ *et* $A\in IR^n$ *alors:*

A=diag (b) signifie que A est une matrice diagonale dont les éléments diagonaux sont les composantes du vecteur b.

A^t *,* b^t *désignent respectivement la matrice transposée de A et le vecteur transposé de b. L'inverse d'une matrice A sera noté* A^{-1}.

CHAPITRE -I-

Fondements théoriques et présentation générale de la méthode du simplexe

Introduction

Dans ce chapitre on parlera essentiellement de la méthode du simplexe.

La premiere partie de ce chapitre constitue un bref rappel de certaines propriétés fondamentales dans la programmation linéaire.

La deuxième partie est consacrée a la description de méthode.

I. 1- Notions fondamentales

I-1. 1- définition d'un problem d' optimisation mathématique.

Etant donné un domaine D de IR" et une fonction f définie de D dans IR" un problème mathématique consiste a optimiser (maximiser ou minimiser) la fonction f sur le domaine D.

*Autrement dit, il s'agit de trouver un point x * de D (si un tel point existe) pour le quel on a :*

1°) $f(x) \geq f(x^*)$ *pour tout* $x \in D$

Dans le cas de minimisation

2°) $f(x) \leq f(x^*)$ *pour tout* $x \in D$

Dans le cas de maximisation

Symboliquement, ce problème d'optimisation mathématique est présentée par la notation suivante :

$$(P) \qquad \begin{cases} \min f(x) \\ x \in D \end{cases}$$

I' ensemble D est appelé ensemble admissible ou réalisable.

Un point de D est appelé solution réalisable.

Une solution réalisable x réalisant l'optimum est appelé solution optimale.*

La valeur de f en x est appelé «valeur optimale » et est notée f*=f(x*)*

I.1.2- Définition d'un programme linéaire

Un programme linéaire est un problème d'optimisation mathématique dans lequel:

1°) Le domaine D des solutions réalisables est défini par un ensemble d'équations ou d'inéquations linéaires ou les deux à la fois appelées «contraintes».

2°) La fonction f, dite «fonction objectif» est linéaire.

Notons que les inéquations linéaires strictes sont interdites car elles sont dépourvues de signification physique.

I.1.3 Forme CANONIQUE et forme STANDARD d'un PROGRAMME LINEAIRE

On dit qu' un programme linéaire est écrit sous forme canonique si le domaine des solutions réalisables est défini par un système d'inéquations linéaires,

Si par contre, en dehors des contraintes de non-négativité toutes les contraintes sont des égalités, on dit que le programme linéaire, est mis sous «forme standard».

On peut toujours mettre un programme linéaire quelconque sous forme standard en ajoutant des variables supplémentaires appelées «variables d'écart»

Exemple

$$(P)\begin{cases} MaxZ = x_1 + x_2 \\ x_1 + x_2 \le 2 \\ 2x_1 + x_2 \ge 1 \end{cases}$$

$$x_1, x_2 \ge 0$$

En introduisant les variables d'écart $x_3 \ge 0$ *et* $x_4 \ge 0$

dans la première et la deuxième contrainte on met le problème sous la forme équivalente :

$$(P')\quad \begin{cases} MaxZ = x_1 + x_2 + 0x_3 + 0x_4 \\ x_1 + 2x_2 + x_3 = 2 \\ 2x_2 + x_2 - x_4 = 1 \end{cases}$$

$$x_1, x_2, x_3, x_4 \ge 0$$

Le problème (P') est la forme standard du problème (P).

Notons que les problèmes de minimisation et de maximisation sont en fait équivalents puisque :

$$\underset{D}{Min}\ f(x) = -\underset{D}{Max}\left(-f(x)\right)$$

Pour cette raison, on ne traitera dans ce qui suit que des programmes linéaires sous forme standard du type :

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

Où $A \in IR^{m \times n}$ *est la matrice des contraintes, n désigne le nombre des variables et m désigne le nombre des contraintes,*

$C \in IR^n$ *est le vecteur coût*

$b \in IR^n$ *est le second membre*

$Z = C.x = \sum_{j=0}^{n} C\,j\,X\,j$ *est la «fonction objectif» avec m< n.*

Remarquons qu 'on peut toujours supposer que rang(A)=m
En effet, si rang (A)<m, une ou plusieurs lignes de la matrice A peuvent être exprimées comme combinaisons linéaires des autres.

Suivant la valeur des coefficients b_i *, les contraintes correspondantes sont soit redondantes, auquel cas on peut les éliminer, soit incompatibles avec les autres, auquel cas le domaine des solutions réalisables est vide.*

I.1.4- Bases, bases réalisables, solution de base

Défmition1

Etant donné un programme linéaire sous forme standard :

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

On appelle base du programme linéaire (P), toute sous matrice carrée régulière ($m \times m$) extraite de A notée B.

On pose $N = \{1,2,...,n\}$, *on note* $J = \{1,2,...,m\}$ *l'ensemble des indices des colonnes de* B *et* $\bar{J} = N - J$ *l'ensemble des indices des colonnes hors base.*

Soit B une base, alors en permutant les colonnes de A, on peut toujours mettre la matrice A sous la forme A= (B,R) où R est la sous matrice formée par les colonnes de A qui ne sont pas dans la base B. De même on peut partitionner x en $(x_J, x_{\bar{J}})$ *et C en* $(c_J, c_{\bar{J}})$

Toute solution de (P) vérifie Ax =b et par suite on a $Bx_J + Rx_{\bar{J}} = b$ (1)

Définition 02

On appelle solution de base (associée à la base B) la solution particulière de (1)

$$\begin{cases} x_J = B^{-1}.b \\ x_j = 0,\ j \in \bar{J} \end{cases}$$

Définition 03

Une base B d'un programme (P) est dite réalisable si la solution de base correspondante est réalisable, c'est-à-dire si

$x_J = B^{-1}.b \geq 0$

Exemple

Soit le programme linéaire:

$$(P_1) \qquad \begin{cases} MaxZ = 4x_1 + 5x_2 \\ x_1 + 2x_3 - x_4 = 3 \\ x_2 - x_3 + x_4 = 2 \\ x_3 - 2x_4 + x_5 = 1 \\ x_1, x_2, x_3, x_4 x_5 \geq 0 \end{cases}$$

L'ensemble J= {1,2,5} est l'ensemble des indices des colonnes d'une base réalisables, où :

$$B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

La solution de base correspondante est:

$x_1 = 3, x_2 = 2, x_5 = 1$

Les variables de base: $x_3 = x_4 = 0$

Les variables hors-base : $x_3 = x_4 = 0$

Par contre la base dont J= {1,3 ,5} est une base mais non réalisable, la solution associée est:

Les variables de base: $x_1 = 7, x_3 = -2 \leq 0, x_5 = 3$

Les variables hors-base: $x_2 = x_4 = 0$

On écrit $\begin{cases} x_J = (7,-2,3) \\ x_{\bar{J}} = (0,0) \end{cases}$

Définition 04

On dit qu'un programme linéaire est mis sous forme canonique relativement à une base B dont l'ensemble des indices des colonnes est J si :

1 °) $c_J = 0$ *, c'est à dire* $c_j = 0$ *pour tout* $j \in J$

2°) B est, à une permutation prés, la matrice unité.

Remarque 01

Connaissant une base du problème (P), on peut toujours le mettre sous forme canonique relativement à une base. En effet, soit le problème suivant:

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

B une base de (P) et J l'ensemble des indices de ses colonnes, on a d'une part

$Ax = Bx_J + Rx_{\bar{J}} = b$

D'où

$x_J + B^{-1} R x_{\bar{J}} = B^{-1} b$ *et* $x_J = B^{-1} b - B^{-1} R x_{\bar{J}}$

Où B^{-1} *est l'inverse de la matrice B, posons* $M = (U, B^{-1}R)$ *(U la matrice unité d'ordre m) et* $b' = B^{-1} b$ *et d'autre part* $cx = c_J x_J + c_{\bar{J}} x_{\bar{J}}$

Ecrivons x_j *en fonction de* $x_{\bar{j}}$ *c'est à dire les variables de base, on trouve*

$$\begin{aligned} c^t x &= c_J (B^{-1} b - B^{-1} R x_{\bar{J}}) + c_{\bar{J}} x_{\bar{J}} \\ &= c_J B^{-1} b + (c_{\bar{J}} - c_J B^{-1} R) x_{\bar{J}} \\ &= Z_0 + c'_{\bar{J}} x_{\bar{J}} \end{aligned}$$

Où on a posé $z_0 = c_J . B^{-1} . b$ *qui est une constante et* $C_{\bar{J}} = C_J B^{-1} R$

Le problème (P) est alors équivalent à (P'):

$$(P)\begin{cases} Max\ z = z_0 + c_{\bar{J}} \\ Mx = b' \\ x \geq 0 \end{cases}$$

qui est écrit sous forme canonique.

I.1.5- *Caractérisation des bases et des solutions de bases optimales*

Théorème 1 : (théorème d'optimalisé)

Soit (P) mis sous forme canonique relativement à une base réalisable.

Si le vecteur coût c est négatif ou nul alors la solution de base est optimale.

Une telle base est dite base optimale.

Théorème 02 :

Soit (P) écrit sous forme canonique relativement à une base réalisable.

S'il existe un élément s de J tel que $C^s > 0$ *alors:*

(a) Ou bien la colonne $A^s \leq 0$ *, au quel cas la fonction objectif est non bornée.*

(b) Ou bien le contraire, au quel cas on peut mettre en évidence une nouvelle base B' *donnant à la «fonction objectif» une nouvelle valeur* Z' $\geq$ Z .

L'intérêt des deux théorèmes ci dessus vient du fait que l'algorithme du simplexe en découle directement.

I.2.1- L'algorithme du simplexe

Pour pouvoir appliquer l'algorithme du simplexe il est nécessaire de connaître une base réalisable. On suppose donc que l'on dispose d'une telle base de départ.

A itération k, soit B la base courante, J l'ensemble des indices de B, ceux-ci sont placés dans un vecteur ligne noté (col (1), col(2), …,col (m)).

Les différentes étapes sont les suivantes:

Etape1:- *calculer:* B^{-1} *(la matrice inverse de B)*

$p = c_J B^{-1}$ *(les multiplicateurs du simplexe)*

Etape2: *Calculer*

$c_{\bar{J}} = c_{\bar{J}} - p\ R$ *(les coûts réduits)*

Etape3: *Déterminer s de* $\bar{J}$ *tel que* $c_s = \underset{j \in \bar{J}}{Max}\{c_j\}$

S 'il existe plusieurs indices, choisir le plus petit.

Si $C'_s \leq 0$ *; Arrêt : l'optimum est atteint.*

Sinon

Etape4: *calculer la colonne* $A^{'s}$ *par la formule*

$A^{'s} = B^{-1}A^s$

Etape5: *Déterminer l'ensemble I:* $I = \left\{ i / a'_{is} > 0 \right\}$

Si $I = f$ *; Arrêt : optimum non borné*

Sinon :

Etape6: *calculer b' par la formule b '=*$B^{-1}b$.

Etape7: *déterminer l'indice r tel que :*

$$\frac{b'_r}{a'_{rs}} = Min_{i \in I}\left\{\frac{b'_i}{a'_{is}}\right\}$$

S'ils existent plusieurs indices qui vérifient la condition choisir le plus petit.

Etape8: *Mise à jour de la base :*

$$J' = J \cup \{s\} - col(r), col(r) = s$$

la colonne dont l 'indice est col(r)=s est remplacée par la colonne A^s.

etape9: *retourner à* ***etape1***.

Géométriquement, la procédure s'interprète comme un cheminement de point extrémal en point extrémal voisin le long de la frontière de l'ensemble réalisable du problème.

Algébriquement, elle s'interprète comme la détermination d'une suite de bases adjacentes.

I.2.2 Résolution d'un exemple

Considérons le programme linéaire

$$(P)\begin{cases} Max z = 4x_1 + 5x_2 \\ 2x_1 + x_2 + x_3 = 8 \\ x_1 + 2x_2 + x_4 = 7 \\ x_2 + x_5 = 3 \end{cases}$$

$$x_1, x_2, x_3, x_4, x_5 \geq 0$$

écrit sous forme canonique par rapport à la base réalisable dont les colonnes forment l'ensemble J= {3,4,5} et on a col(l) =3 , col(2)=4, col(3)=5

La solution de base est : $x_1=x_2=0$, $x_3=8$, $x_4=7$, $x_5=3$.

Les conditions d'application du théorème d'optimalité ne sont pas réunies puisque

$C_1=4>0$, $c_2=5>0$

On voit qu'on a intérêt à augmenter x_2 *(puisque* $c_2=5>c_1=4$*) pour faire croître z, donc s=2.*

Le domaine de variation de x_2 *est:*

$$D = \left\{ x_2 / 0 \le x_2 \le Min\left[\frac{8}{1}, \frac{7}{2}, \frac{3}{1}\right] \right\}$$
$$D = \{ x_2 / 0 \le x_2 \le 3 \}$$

C'est la troisième contrainte

$$\frac{b_3}{a_{321}} = \underset{i \in I}{Min}\left\{ b_i \diagup a_{i2} \right\}$$

qui limite l'augmentation x_2 , la 2ème colonne entre dans la base et la 5ème en sort.

La nouvelle base est alors formée par les colonnes 3,4 et 2 c 'est à dire que

$J' = J \cup \{2\} - \{5\} = \{3,4,2\}$ *et col(3) =2*

Ecrivons (P) sous forme canonique par rapport à J'

$$(P_1)\begin{cases} Max\ z = 4x_1 - 5x_5 + 15 \\ 2x_1 + x_3 - x_5 = 5 \\ x_1 + x_4 - 2x_5 = 1 \\ x_2 + x_5 = 3 \end{cases} \quad x_i \ge 0$$

La base B' dont les colonnes sont 3,4 et 2 n'est pas optimale puisque c_1 =4 > 0 et s=1.

Cherchons l'indice d'entrée r :

$$\underset{i \in I}{Min}\left\{ \frac{b_i}{a_{i\,s}} \right\} = \underset{i \in I}{Min}\left\{ \frac{b_i}{a_{i1}} \right\}$$
$$= \underset{i \in I}{Min}\left\{ \frac{5}{2}, \frac{1}{1} \right\}$$

D'où r=2

Par conséquent l'ensemble des indices des colonnes devient

*J" = J' **U** {1} -{col (2} ={3,1,2} et col (2)=1*

Ecrivons le problème (P) sous forme canonique par rapport à la nouvelle base dont les colonnes sont 3,1 et 2.

$$(P_2)\begin{cases} Max\ z = -4x_4 + 3x_5 + 19 \\ x_3 - 2x_1 + 3x_5 = 3 \\ x_1 + x_4 + -2x_5 = 1 \\ x_2 + x_5 = 3 \end{cases} \quad x_i \ge 0$$

La base n 'est pas optimale puisque $c_5 = 3 > 0$. On applique le même processus l'indice r:

$$\underset{i\in I}{Min}\left\{\frac{b_i}{a_{is}}\right\} = \underset{i\in I}{Min}\left\{\frac{b_i}{a_{i5}}\right\} = \underset{i\in I}{Min}\left\{\frac{3}{3},\frac{3}{1}\right\} = \frac{b_1}{a_{15}}$$

donc r=1 et par conséquent la l^{ere} colonne de la base sort (puisque r=1), et la colonne $A^s = A^5$ entre dans la base.

L'ensemble des indices devient alors:

J''' ={5, 1, 2} et col(1) =5.

La base en question est alors constituée des colonnes 5,1 et2

Ecrivons encore une fois (P) sous forme canonique par rapport à cette nouvelle base.

$$(P_3)\begin{cases} Maxz = -x_3 - 2x_4 + 22 \\ \frac{1}{3}x_3 - \frac{2}{3}x_4 + x_5 = 1 \\ x_1 + \frac{2}{3}x_3 - \frac{1}{3}x_4 = 3 \\ x_2 - \frac{1}{3}x_3 + \frac{2}{3}x_4 = 2 \end{cases}$$

$$x_1, x_2, x_3, x_4, x_5 \geq 0$$

On a

$$Z = 22 + 0x_1 + 0x_2 - x_3 - 2x_4 + 0x_5 = -x_3 - 2x_4 + 22$$

$$c' est\ à\ dire\ que\ c_1 = c_2 = c_5 = 0, c_3 = -1, c_4 = -2$$

Le vecteur coût est négatif ou nul par conséquent la base courante est optimale.

La solution optimale est:

X =(3,2,0,0,1) et z*=22*

I.2.3- Initialisation de l'algorithme du simplexe

Comme nous l'avons indiqué auparavant, l'application de l'algorithme du simplexe nécessite la connaissance d'une base réalisable de départ.

Or, en pratique on ne dispose pas toujours d'une telle base (par exemple lorsqu 'il y 'a des contraintes d'égalité).

Nous cherchons ici à trouver une base réalisable et écrire (P) sous forme canonique par rapport à celle ci, si toutefois une telle base réalisable existe ou bien à démontrer que (P) ne peut admettre de solution réalisable.

Sans perte de généralité, on peut toujours supposer que $b \geq 0$ *sinon il suffit de multiplier par (-1) les équations pour les quelles* $b_i < 0$.

On associe à (P) le «programme linéaire auxiliaire»

$$(PA)\begin{cases} Max\, y = \sum_{i=1}^{m} y_i \\ A.x + U.y = b \\ x, y \geq 0 \end{cases}$$

Dans lequel U est la $m \times m$ *matrice unité.*

Les variables y_i *sont dites «artificielles»*

Soit e le n-vecteur ligne tel que $e_i = 1$ *pour tout i.*

(PA) se met sous forme canonique par rapport à la base réalisable dont l'ensemble des indices des colonnes est $J_0 = \{n+1, n+2, \ldots, n+m\}$ *et on écrit :*

$$(PA)\begin{cases} Max\, y = -eb + eAx \\ A.x + U.y = b \end{cases}$$

On montre que si $\hat{x}, \hat{y}$ *est une solution optimale de (PA), alors (P) admet une solution réalisable si et seulement si* $\hat{y} = 0$

Dans le problème initial (P) il se peut qu 'une variable disons x_s *soit contenue dans une seule équation, disons la* r^{ieme} *alors si* a_{rs} *et* b_r *sont de même signe il n'est pas nécessaire d'introduire une variable artificielle correspondant à cette équation. On peut en effet considérer que* x_s *est la variable de base correspondant à la* r^{eme} *équation.*

Définition 05

L'application de l'algorithme du simplexe à (PA) s'appelle phase 1 et à (P) s'appelle phase 2.

II. 2.4 Description de la phase 1

On applique l'algorithme du simplexe à (PA) sans conserver trace de variables artificielles qui quittent la base.

Si le maximum de la valeur de la «fonction objectif» de (PA) est négatif terminer le programme linéaire initiale (P) n 'a pas de solution réalisable.

Si le maximum de la «fonction objectif» de (PA) est nul, le problème linéaire initial (P) se trouve alors écrit sous forme canonique par rapport à une base réalisable après élimination des variables artificielles de la base si elle en contient.

Exemple:

$$(P)\begin{cases} Max\, z = x_1 + x_2 \\ 2x_1 + x_2 \le 4 \\ x_1 - 2x_2 \le -3 \\ x_1 + x_2 = 5 \end{cases} \qquad x_1, x_2 \ge 0$$

Après introduction des variables d'écart et en multipliant la 2ème équation par (-1) on a:

$$(P)\quad \begin{cases} Max\, z = x_1 + x_2 \\ 2x_1 + x_2 + x_3 = 4 \\ -x_1 + 2x_2 - x_4 = 3 \\ x_1 + 3x_2 = 5 \end{cases} \qquad x_1, x_2, x_3, x_4 \ge 0$$

En introduisant les variables artificielles y_1 et y_2 dans la 2ème et 3ème équation, on obtient le programme auxiliaire suivant:

$$(PA)\quad \begin{cases} Max\, y = 5x_2 - x_4 - 8 \\ 2x_1 + x_2 + x_3 = 4 \\ -x_1 + 2x_2 - x_4 + y_1 = 3 \\ x_1 + 3x_2 + y_2 = 5 \end{cases} \qquad x, y \ge 0$$

qui constitue un programme linéaire présenté sous forme canonique dont les variables de base sont la variable x_3 et les variables artificielles y_1 et y_2

On peut appliquer, alors, l'algorithme du simplexe au problème auxiliaire (PA).

I. 3 Dualité

Ce paragraphe traite en bref du concept de dualité en insistant sur son importance pratique.

A tout problème de programmation linéaire on peut associer un autre problème de même nature appelé dual. Cette association est involutive, c 'est à dire que le dual du dual est le problème de départ appelé primal.

En fait, il faut considérer que deux problèmes linéaires duaux ne constituent pas deux problèmes distincts, mais deux aspects du même problème.

Dans ce paragraphe, les problèmes en dualité seront toujours écrits sous forme canonique.

I.3.1- Définition

Soit le programme linéaire écrit sous forme canonique

$$(P) \quad \begin{cases} MaxZ = c^t x \\ Ax \le b \\ x \ge 0 \end{cases}$$

On appelle «dual» de (P) le programme linéaire suivant :

$$(D) \quad \begin{cases} MinW = y^t b \\ yA \ge c \\ y \ge 0 \end{cases}$$

Où le vecteur inconnu y est un vecteur ligne de IR^m.

I.3.2- L'algorithme dual du simplexe

Considérons le programme linéaire

$$(P) \quad \begin{cases} MaxZ = cx \\ Ax \le b \\ x \ge 0 \end{cases}$$

écrit sous forme canonique. Si $b \ge 0$ *alors (P) est dit «primal réalisable» puisque la solution de base x=0 est une solution réalisable du primal.*

Si $c \leq 0$ alors (P) est dit «dual réalisable » .En effet, le dual de (P) s'écrit:

$$(D) \qquad \begin{cases} MinW = yb \\ yA \geq c \\ x \geq 0 \end{cases}$$

et la solution y=0 est réalisable.

Notons qu'il est démontré qu'une base est optimale si et seulement si lorsque le programme linéaire est écrit sous forme canonique par rapport à cette base il est à la fois primal et dual réalisable.

Dans certains cas le programme linéaire et donné sous forme duale réalisable mais non primal réalisable, il serait alors maladroit d'utiliser la phase1 pour le rendre primal réalisable puis d'appliquer la phase2. On applique plutôt l'algorithme dual qui consiste à maintenir la duale réalisabilité en tentant de le rendre primal réalisable.

C'est le principe de l'algorithme dual du simplexe qui est présenté dans ce paragraphe.

On considère alors le programme linéaire.

$$(P) \qquad \begin{cases} MaxZ = cx \\ Ax \geq b \\ x \geq 0 \end{cases}$$

écrit sous forme canonique par rapport à une base. J désigne l'ensemble des indices des colonnes de A associées à la base et tel que (P) soit dual réalisable, c'est à dire qu'on a:

(1) A^J est, à une permutation près des colonnes, la matrice unité.

(2 $c \leq 0$

(3 $c^J = 0$.

L'algorithme dual s'énonce comme suit:

Etape1: *Trouver l'indice r tel que : $b_r = \min_i \{b_i\}$*

Si plusieurs indices réalisent ce minimum, choisir le plus petit d'entre eux

Etape2: *Si $b_r<0$ alors déterminer l'ensemble L tel que $L = \{j / a_r^j < 0\}$.*

Si $L \neq \emptyset$ *arrêt : pas de solution réalisable.*

Si $b_r \geq 0$ *arrêt : la base courante est optimale.*

Etape3: *Trouver le plus petit indice s tel que* $\frac{c^s}{A_i^s} = \underset{j \in L}{Min}\left\{\frac{c^j}{A_s^j}\right\}$

Mettre à jours le programme linéaire et itérer.

Notons que l'algorithme dual n'est autre que l'algorithme primal appliqué au problème dual, mais tous les calculs sont effectués sur les données du problème primal.

CHAPITRE -II-

Présentation générale de la méthode de KARMAKAR et description de son algorithme

II.0 Introduction:

Dans ce chapitre on parlera essentiellement de la méthode de KARMAKAR en signalant, chaque fois qu'il est nécessaire de le faire, les difficultés pratiques de cet algorithme.

Cette méthode comprend deux grandes idées innovatrices:

1-Plonger le problème initial dans l'espace projectif (Contenant le simplexe unitaire).

2- Utiliser une projection pour trouver la direction de déplacement.

Contrairement à la méthode du simplexe qui –géométriquement- s'interprète comme un cheminement de sommet en sommet adjacent le long de la frontière de l'ensemble des solutions réalisables, la méthode de KARMAKAR est "une méthode de point intérieur" qui produit une suite de solutions réalisables (dans l'intérieur relatif de la région admissible) convergeant vers une solution optimale du problème traité.

II.1-Préléminaires:

II.1.1- Minimisation d'une fonction linéaire sur un ellipsoïde:

Considérons le problème suivant:

$$(0)\begin{cases}\min c'x\\ x\in E\end{cases}$$

Où $x,c\in IR^n$ *et E un ellipsoïde défini par son centre* a^0 *, par r et la matrice symétrique définie positive A.*

Algébriquement: $E=\left\{x\in IR^n : (x-a^0)\,A(x-a^0)\le r^2\right\}$

Lemme 1: *La solution du problème (0) est donnée par:*

$$x^*=a^0-\frac{A^{-1}cr}{\sqrt{<c,A^{-1}c>}}$$

Preuve:

En posant y=x-a^0*, le problème revient à résoudre:*

$$(0')\begin{cases}\min c^t y\\ y^t Ay\le r^2\end{cases}$$

En écrivant les conditions d'optimalité en programmation convexe, y^* est solution de (0') si et seulement si, il existe un réel $l^* \geq 0$ tel que

$$\begin{cases} c + l^* A y^* = 0 \\ l^* (y^{*t} A y^* - r^2) = 0 \end{cases}$$

On en déduit: $l^* > 0$, $y^* = (-1/l^*) A^{-1} c$, $y^{*t} A y^* = r^2$ *et donc*

$$\frac{1}{l^{x2}} c^t A^{-1} c = r^2$$

$$y^* = -\frac{A^{-1} c r}{\sqrt{c^t A^{-1} c}} \Rightarrow x^* = a^0 - \frac{A^{-1} c r}{\sqrt{c^t A^{-1} c}} \quad c.q.f.d$$

Cas particulier du lemme 1:

Si l'ellipsoïde E est réduit à une sphère (A=I) alors la solution de (0) est

$$x^* = a^0 - \frac{c}{\|c\|} r$$

Autrement dit, il suffit de se déplacer (figure1) à partir du centre a^0 dans la direction opposée du vecteur coût (i.e. -c)d'une distance égale au rayon (r) de la sphère.

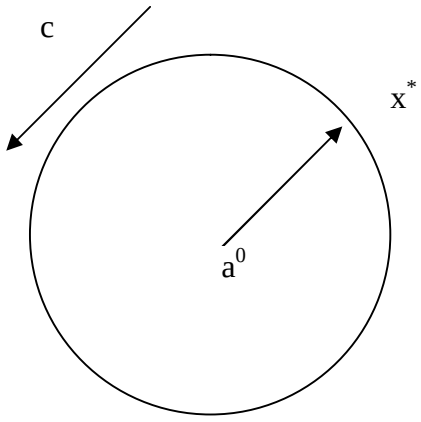


Figure 1.1

II.1.2- L'idée générale de l'algorithme:

Soit A un élément de $iR^{m\times n}$, $b\in IR^m$ *et* $c\in IR^n$ *et posons par définition*

$\Omega=\{x\in IR^n : Ax=b\}$

$P=\{x\in IR^n : Ax=b, x\geq 0\}=\Omega \cap IR^n{}_+$, *où* $IR^n{}_+$ *désigne l'orthant positif*

Considérons alors le programme linéaire générale:

$$(p.l.g)\begin{cases}\min c^t x\\ x\in P\end{cases}$$

Si l'on remplace $IR^n{}_+$ *par une sphère (ou ellipsoïde)* $E_1\subset IR^n{}_+$, *on obtient le problème:*

$$(1)\begin{cases}\min c^t x\\ x\in \Omega \cap E_1\end{cases}$$

Or, l'intersection d'une sphère et d'un espace affine est une sphère de dimension plus petite dans cet espace affine, donc $(\Omega \cap E_1 = E_2)$. *Le problème devient:*

$$(2)\begin{cases}\min c'^t x\\ x\in E_2\end{cases}$$, *c' désigne la projection orthogonale de c sur* Ω

D'après ce qui précède, le problème (2) est trivial: Il suffit de se déplacer à partir du centre de E_2 *dans la direction (-c') d'une distance égale au rayon de* E_2.

Finalement, tout revient à remplacer P par un ellipsoïde (ou une sphère) E de centre a^0 $(a^0\in P : a^o{}_i>0, i=1,...,n)$ *et minimiser* $c^t x$ *sur E au lieu de P.*

II.1.3- La transformation projective de KARMAKAR

Soit $a=(a_1, a_2, \ldots, a_n)^t$ *un point intérieur à P (i.e. tel que* $a\in\Omega et\, a_1>0)$ *et posons* $D=diag\{a\}$.

Définissons alors le simplexe S_{n+1} *de dimension n contenue dans* IR^{n+1}:

$S_{n+1}=\{x\in IR^{n+1}, x\geq 0 \text{ et } e^t{}_{n+1}=1\}$

La transformation projective (notée T) est une fonction:

$IR^n{}_+ \dashrightarrow IR^{n+1}{}_+$ *définie par:*

$$y = T(x) \text{ avec } \begin{cases} y_i = \dfrac{x_i / a_i}{1 + \sum_{i=1}^{n} x_i / a_i} \quad , \quad i = 1,...,n \\ \text{et} \\ y_{n+1} = 1 - \sum_{i=1}^{n} y_i \end{cases}$$

Il est facile de voir que: $y_i = \dfrac{x_i}{a_i} y_{n+1} \quad , i = 1,....,n$

ou encore: $y_{[n]} = (D^{-1}x)y_{n+1}$ *, où* $y_{[n]}$ *désigne les n premières composantes de y. Ceci montre que T est bijective: En effet,*

$T^{-1}(y) = x = \dfrac{Dy[n]}{y_{n+1}}$

Evidement:

$T(IR^n{}_+) = S_{n+1} \subset IR^{n+1}$

$T(a) = a^0 = (1/n+1)e_{n+1} \qquad (centre\, de\, S_{n+1})$

Quelques Propriétés de T:

1-L'image d'un espace affine Ω *par T est un espace affine* Ω'**,** *donc puisque a* $\in \Omega$ *, son image* $a^0 \in \Omega'$.

2- L'image d'une face $(x_i = 0)$ *de* $IR^n{}_+$ *est la face correspondante* $y_i = 0, i = 1,...n)$ *du simplexe* S_{n+1}*. Quant à la face* $y_{n+1} = 0$ *de* S_{n+1} *, elle est l'image des points à l'infini.*

Il est facile de voir que la plus grande sphère inscrite dans S_{n+1} *et la plus petite sphère contenant* S_{n+1} *centrées en* a^0 *sont respectivement:*

$B(a^0, r) = \left\{ x \in IR^{n+1} \quad e^t{}_{n+1} x = 1, \left\| x - a^0 \right\| \leq r \right\}$

$B(a^0, R) = \left\{ x \in IR^{n+1} : e^t{}_{n+1} x = 1, \left\| x - a^0 \right\| \leq R \right\}$

Où $r = 1/\sqrt{n(n+1)}$ *et* $R = \sqrt{n/(n+1)}$ *d'où* $(R/r) = n$

La situation est donc la suivante: $B(a^0, r) \subset S_{n+1} \subset B(a^0, R), ce\ qui\ entraîne$ $B(a_0, r)$ **I** $\Omega' \subset S_{n+1}$ **I** $\Omega' \subset B(a_0, R)$ **I** Ω'.

Or, S_{n+1} **I** $\Omega' = P' = T(P = \Omega$ **I** $IR^n{}_+)$*. De plus l'intersection d'une sphère B avec un espace affine* Ω *est une sphère B' de dimension plus petite et de même rayon que B si* Ω *contient le centre de B, c'est le cas ici puisque*

$a^0 \in \Omega'$. *Ainsi:*

$B(a^0, r) \subset P' \subset B'(a^0, R)$, *ou* $B'(a^0, r) = B(a^0, r)$ **I** Ω'. *et* $B'(a^0, R) = B(a^0, R)$ **I** Ω

Ceci prouve que la minimisation sur la sphère inscrite dans la région admissible réduit la valeur de l'objective d'au moins (1-1/n).

II.2-Description de la méthode:

II.2.1-Le problème traité par KARMAKAR et les hypothèses de travail:

On se place désormais dans IR^n*: Le implexe* $S_n = \{x \in IR^n : e_n{}^t x = 1, x \geq 0\} \subset IR^n$ *est donc de dimension (n-1).*

La méthode projective de KARMAKAR résoud directement le programme linéaire suivant:

$$(p.l.r)\begin{cases} \min c'x \\ Ax = 0 \\ x \in S_n \end{cases}$$

En supposant que:

1-La valeur optimale est nulle, i.e: si x^* *est une solution optimale de (p.l.r) alors* $C^tX^*=Z^*=0$

2- Le point $a^0 = (1/n)e_n$ *(centre du simplexe* S_n*) est une solution réalisable de (p.l.r).*

3-La matrice A est de plein rang: rg(A)=m.

Ces trois hypothèses seront appelées les conditions de KARMAKAR.

On suppose également que $c^ta^0>0$*, puisque si* $c^ta^0=0$*on s'arrête immédiatement avec* a^0 *optimal. Ceci implique que* c^tx *n'est pas constant sur la région admissible et par conséquent il est strictement positif pour tout x réalisable.*

Remarques:

a- *Si la valeur optimale est connue à priori mais non nulle l'égalité* $e_{n^1} x = 1$ *permet de se ramener à un objectif nul. En effet, soit x une solution optimale du problème et* c^* *la valeur optimale de l'objectif.*

Alors $c^t x = z^* = z^* e^1{}_n x \Rightarrow (c - z^* e_n)^t x = (c')^t x = 0$. *On minimise alors l'objectif* $(c')^tx$ *au lieu de* c^tx*, où* $c'_i = c_i - z^*, i = 1...n.$

b- *Si le système de contraintes est de la forme* $Ax = b, b \neq 0$, *on se raméne facilement à un système homogène, il suffit d'écrire:* $Ax = be_n^{\ t}x \Rightarrow (A - be_n^{\ t})x = 0$

Autrement dit on obtient un système de la forme: $A'x = 0$ *ou les éléments de A' sont :* $a'_{ij} = (a_{ij} - b_i), \{i = 1.....m,\ j = 1,.....,n\}$.

c-*Le problème* (p.l.r) *ainsi défini peut être vu comme un problème de faisabilité*
En effet, puisqu'on suppose la valeur optimale (Z) nulle ou connue à priori, il s'agit alors:*

de trouver $x \geq 0$ *tel que:* $\begin{cases} c^t x = z^* \\ Ax = 0 \\ e_n^1 x = 1 \end{cases}$

II.2.2-L'Algorithme de base:

Nous présentons dans ce paragraphe l'algorithme de base de KARMAKAR pour résoudre un programme linéaire du type (p.l.r) pour cela on se donne une précision e *(par exemple* $e = 2^{-q}$ *où q est un entier,* $q \geq 1$).

Partant de la solution initiale $x^0 = a^0$, *l'algorithme produit une suite de points intérieurs qui converge vers une solution optimale du problème en un temps polynomial.*

Dans le but de ramener l'objectif $c^t x$ *à zéro on le minimise localement sur une sphère inscrite dans la région admissible. A chaque itération (k) l'itéré (*x^k *>0) est ramené au centre de* S_n *par la transformation projective* T_k *définie par:*

$T_k :\ x \in S_n$ ----------▶ $T_k(x) = y \in S_n$ *avec*

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

et ainsi de suite jusqu'à ce que le test d'optimalité $(C^t x^k \leq e)$ *soit vérifié.*

Début algorithme

Initialisation $x^0 = a^0 = (1/n)e_n$,$k=0$

Tant que $C^t x^k > e$ *Faire*

Pas 0 $\begin{cases} D_k = diag\{x^k\} \\ B_k = \begin{bmatrix} A_k \\ e_n^t \end{bmatrix} \end{cases}$ $\qquad A_k = AD_K$

Pas 1 $p_k = \{I - B_k^t (B_k B_k^t)^{-1} B_k \} D_k c = \{I - A_k^t (A_k A_k^t)^{-1} A_k - \frac{1}{n} e_n e_n^t \} D_k c$

Pas 2 $d_k = \frac{P_k}{\|P_k\|}$

Pas3 $y^k = a^0 - ard_k \quad , \; r = \frac{1}{\sqrt{n(n-1)}} \; , 0 < a < 1$

Pas 4 $x^{k+1} = \frac{D_k y^k}{e_n^t D_k y^k} = T_k^{-1}(y^k) \quad , \; k = k+1$

fin tant que

fin algorithme

Dans le pas 0, on ne fait que construire la matrice des contraintes (i.e B_k)

Le pas 1 consiste à projeter $c_k = D_k c$ sur le noyau de B_k (c'est l'opération la plus coûteuse de l'algorithme). La formule donnant p_k est obtenue par un calcul élémentaire en utilisant le fait que $A_k e_n = 0$.

Au pas 2 on calcule le vecteur normé d_k correspondant à p_k.

Au pas 3 on choisit y^k à une distance a de a^0 dans la direction $-d_k$ KARMAKAR choisit $a = 1/4$

Et enfin au pas 4 on effectue la transformation inverse T_k^{-1} pour calculer le nouvel itéré x^{k+1}.

II.2.3-Dérivation et analyse de l'algorithme:

Le but de ce paragraphe est de montrer en quelque sorte comment est obtenu l'algorithme précédent.

On a vu que la transformation T_k applique le simplexe S_n dans lui-même, en même temps l'itéré $x^k > 0$ (dont les composantes forment la matrice diagonale D_k) est envoyé au centre de S_n. Cependant le transformé du programme linéaire (p.l.r) est le programme suivant:

$$(p.n.l)\begin{cases}\min \dfrac{c^t D_k y}{e_n^t D_k y} \\ \dfrac{AD_k y}{e_n^t D_k y} = 0 \\ e_n^t y = 1\ ,\ y \geq 0\end{cases}$$

Mais on a pour tout $y \in S_n$: $e_n^t D_k y = \sum_{i-1}^{n} x_i^k y_i \geq \{x_i^k : i = 1,....,n\}$

Les égalités $\dfrac{c^t D_k y}{e_n^t D_k y} = 0$ *et* $\dfrac{AD_k y}{e_n^t D_k y} = 0$ *sont donc satisfaites si et seulement si:*

$c^t D_k y = 0$ *et* $AD_k y = 0$

(p.n.l) est alors équivalent au programme linéaire:

$$(p.l)\begin{cases}\min c^t D_k y \\ AD_k y = 0 \\ e_n^t y = 1\ ,\ y \geq 0\end{cases}$$

qui est de la forme (p.l.r) et vérifie les conditions de KARMAKAR.

II.3- Extension de l'algorithme de KARMAKAR:

II.3.0-Introduction:

La mise en œuvre de l'algorithme de KARMAKAR, son implémentation effective nécessite l'étude de ces deux problèmes:

(1)- La transformation du problème (p.l.g) (ci-dessous) à la forme réduite:

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

(2)- La modification de l'algorithme décrit dans le paragraphe2 en vue de son extension au cas où la valeur optimale z n'est pas connue.*

II.3.1- Résolution du problème de faisabilité (phase 1)

Un point réalisable de (p.l.g) est une solution du problème:

$$(P1)\begin{cases} Ax = b \\ x \geq 0 \end{cases}$$

Or, d'après KARMAKAR (P1) est équivalent au problème de minimisation:

$$(P2)\begin{cases} \min\ l \\ Ax + l\,(b - Ax^0) = b \\ x \geq 0\ ,\ l \geq 0 \end{cases}$$

Où $x^0>0$ est choisi arbitrairement dans l'orthant positif et l une variable artificielle.

Notons que (P2) est toujours réalisable, il suffit de prendre: $x = x^0$ et $l = 1$.

Soit l_m la valeur minimale de l'objectif. Alors, $l_m = 0$ correspond à une solution réalisable de (P1), cependant on peut se contenter d'une solution presque nulle. Plus précisément KARMAKAR démontre le théorème suivant:

Théorème:

$\exists e_0 > 0$ tel que: les deux propositions suivantes sont équivalentes:

1- (P1) est réalisable.

2- (P2) admet une solution (x, l) telle que $l \leq e_0$

La résolution de (P1) se ramène donc à celle de (P2), problème auquel on appliquera L'algorithme de base; puisque connaissant la valeur optimale et une solution réalisable de ce dernier il est facile de le mettre sous la forme (p.l.r).

Remarque

Le problème de faisabilité (P1) peut être résolu par une quelconque méthode à condition que la solution trouvée soit dans l'intérieur relatif du domaine de faisabilité $\{x \geq 0,\ Ax = b\}$. Par exemple la méthode du simplexe ne convient pas puisqu'elle donne une solution sur la frontière.

II.3.1.2- Problème de minimisation (Phase 2)

Soit $a(a_i > 0,\ i = 1,.....,n)$ *une solution réalisable de (p.l.g) obtenue (par exemple par la méthode précédente). En appliquant la transformation T le système Ax=b devient A'y=0 où A' est une* $m \times (n+1)$ *matrice définie par:*

$$A' = \left[AD_a - b \right] \quad , \quad D_a = diag\{a\}.$$

Quant à l'objectif il est transformé en une fonction non linéaire:

$$\frac{c^1 D_a y[n]}{y_{n+1}}$$

Où y[n] est un vecteur de IR^n *formé des n premières composantes de* $y = T(x) \in IR^{n+1}$.

On se retrouve alors avec le problème:

$$(p.l.f) \begin{cases} \min \dfrac{c^t D_a y[n]}{y_{n+1}} \\ A' y = 0 \\ e_{n+1}^t y = 1 \\ y \geq 0 \end{cases}$$

Or si la valeur optimale z est connue, on peut écrire*

$$\frac{c^t D_a y[n]}{y_{n+1}} = z^* \Rightarrow c^t D_a y[n] - z^* y_{n+1} = c'^t\ y = 0 \quad , \ où \begin{cases} c'_i = c_i a_i \ , \ i = 1,...,n \\ c'_{n+1} = - z^* \end{cases}$$

On obtient le programme linéaire réduit:

$$(p.l.r) \begin{cases} \min c'^t\ y \\ A' y = 0 \\ e_{n+1}^t = 1 \\ y \geq 0 \end{cases}$$

Il est clair que toute solution réalisable de (p.l.g) est transformée par T en une solution réalisable de (p.l.r) et réciproquement, toute solution réalisable y de (p.l.r) avec $(y_{n+1} > 0)$ *est transformée par* T^{-1} *en une solution réalisable x de (p.l.g).*

Notons en fin que si la valeur optimale est inconnue, le problème (p.l.f) ne peut pas être mis sous la forme (p.l.r). Il faudra avoir recours à d'autres techniques que nous verrons dans le paragraphe suivant.

II.3.2- Modifications de l'algorithme de base:

On s'intéresse toujours à la résolution du problème linéaire générale

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

en supposant cette fois la valeur optimale z^* inconnue KARMAKAR propose deux variantes:

a- La méthode "primale- duale" qui n'est autre que l'application de la phase 2 de l'algorithme de base à un problème de faisabilité obtenu par la combinaison de (p.l.g) avec son dual.

b- La "sliding objective function method" qui consiste à localiser la valeur optimale dans un intervalle suffisamment petit.

II.3.2.1- Méthode primale-duale

Considérons le dual de (p.l.g):

$$(D)\begin{cases}\max b^t y \\ A^t y \leq c \\ y \text{ quelconque dans } IR^m\end{cases}$$

En combinant les deux problèmes et en utilisant le théorème de dualité de la programmation linéaire, le problème revient tout simplement à chercher une solution réalisable du programme "primal-dual":

$$\begin{cases}Ax = b\ ,\ x \geq 0 \\ A^t y \leq c\ ,\ y \text{ quelconque} \\ c^t x - b^t y = 0\end{cases}$$

En écrivant y sous forme d'une différence de deux vecteurs non négatifs et en ajoutant quelques variables d'écart, le système précédent prend la forme:

(1) $SX = b_s \quad , X \geq 0$ *avec*

$$S = \begin{bmatrix} A & 0 & 0 & 0 \\ 0 & A^t & -A^t & I \\ c^t & -b^t & b^t & 0 \end{bmatrix} \quad , \quad b_s = \begin{bmatrix} b \\ c \\ 0 \end{bmatrix}$$

$$S \in IR^{(m+n+1)\times(2m+2n+1)} \quad , \ b_s \in IR^{(m+n+1)} \quad , \ X \in IR^{(2m+2n+1)}$$

Evidement (1) est réalisable "si et seulement" (p.l.g) admet une solution optimale finie. De plus, d'après les résultats précédents le problème (1) est équivalent au problème d'optimisation:

$$(2)\begin{cases}\min\ l \\ SX+(b_s-SX_0)\,l=b_s \\ \begin{pmatrix}X\\ l\end{pmatrix}\geq 0\end{cases}$$

$X_0>0$ *est choisi arbitrairement, par exemple* $X_0=(1,1,....,1)\in R^{(2m+2n+1)}$

Le point $a=(X_0,1)^t$ *a est une solution réalisable évidente de (2). La transformation projective* T_a *nous permet de ramener (2) à la forme convenable (p.l.r) et appliquer alors l'algorithme de KARMAKAR pour trouver une solution (X, λ) telle que* $l\leq e$ *,e étant une précision donnée. X sera alors la solution du problème (1) qui comprend les solutions primales et duales du problème de départ.*

Remarques:

Cette méthode est la plus simple à mettre en œuvre, on n'a besoin ni de résoudre le problème de faisabilité ni de connaître à priori la valeur optimale exacte, cependant on optimise un problème de 2(m+n+1) variables et (m+n+1) contraintes, ou m et n sont respectivement le nombre de contraintes et le nombre de variables du problème de départ.

Exemple II.3.2.1

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

ou

$$A=\begin{bmatrix}2&1&1&0&0\\1&2&0&1&0\\0&1&0&0&1\end{bmatrix},\ b=\begin{bmatrix}8\\7\\3\end{bmatrix}\quad et\ \ c=(-4,-5,0,0,0)^t$$

En considérant le teste d'arrête $\left(c^t x^k \leq 10^{-5}\right)$ *, nous avons obtenu au bout de (k=22) itérations les solutions suivantes:*

Solution primale: (2.999981; 1.999995; 0.000012; 0.000006; 0.999997)^t

Solution duale: (-0.999993; -1.999998; -0.000005)^t

Les solutions exactes étant: $x^* = (3,2,0,0,1)^t$, $y^* = (-1, -2, 0)^t$

II.3.2.2 Sliding objective function method:

Cette méthode est une variante de l'algorithme de base, elle s'applique dans le cas ou la valeur optimale de l'objectif n'est pas connue d'avance et consiste à localiser cette valeur dans un intervalle qu'on améliore au cours des itérations.

On s'arrêtera une fois l'intervalle obtenu est aussi petit que l'on veut: la borne supérieure du dernier intervalle obtenu sera la valeur optimale approchée et le point réalisable correspondant à cette borne sera la solution optimale approchée.

Description de la méthode:

Soit [l, u] un intervalle contenant la valeur optimale z^* *(supposée inconnue), l'algorithme n'étant applicable qu'à la connaissance de la valeur optimale de l'objectif l'idée mise en œuvre ici est de supposer cette valeur.*

Au départ on prend $u=c^t a$, *où a est la solution réalisable initiale de (p.l.r) et* $l=l_0$.

Si $(u-l) \leq e$ *terminer le point a est une solution optimale et u une valeur optimale approchée.*

Sinon (u-l> e) on teste la validité des bornes:

$l' = l+(u-l) / 3$

$u' = l+2(u-l) / 3 = u-(u-l)/ 3.$

En prétendant qu'à l'optimum l'objectif vaut l' et en appliquant une ilération de l'algorithme de KARMAKAR avec l'objectif $c'^t x = c^t x - l'$ *ou* $c' = c - l' e_n$, *on obtient le point:*

$$x = T_a^{-1}(y) = \frac{D_a y}{e_n^t D_a y} \quad ou\ D_a = diag\{a\}\ et\ y = a^0 - \alpha r d_a$$

$a^0 = (e_n / n)$ *étant le centre du simplexe* S_n *et* d_a *la projection de* $c'_a = D_a c'$ *sur le noyau de la matrice* B_a *ou*

$$B_a = \begin{bmatrix} AD_a \\ e_n^t \end{bmatrix}$$

Notons que $c^t a = c^t a - l' = u - l' = 2(u-l) / 3 > 0$.

Exemple numérique:

Reprenons l'exemple II.3.2.1 avec le teste d'arrête ($u-l \leq 10^{-5}$).

Les résultats sont donnés dans le tableau (II.3.2.1).

Itération	*Borne inférieure*	*Borne supérieure*
0	*- 35.0000*	*- 16.7125*
5	*- 23.1732*	*- 21.8396*
10	*- 22.1031*	*- 21.9933*
15	*- 22.0077*	*- 21.9933*
20	*- 22.0008*	*- 21.9999*

Tableau II.2.1

Sliding objective function method avec $a = 0.99$

Solution trouvée (au bout de 20 itérations)

$X^{20} = (2.999992; 1.999994; 0.000018; 0.000017; 0.000005)$.

II.4 Calcul de la projection:

Dans les algorithmes de point intérieur du type KARMAKAR proposés, le coût d'une itération est dominé par le calcul de la direction de déplacement $-d_k$ *ou*

$$d_k = P_k D_k c - c^t x^k / n, \quad \text{avec} \quad A_k = AD_k,\ D_k = diag\{x^k\},\ A \in IR^{mxn} \quad \text{et} \quad P_k = \{I - A_k{}^t (A_k A_k{}^t)^{-1}) A_k\} D_k c$$

On peut calculer P_k *de plusieurs façons, en particulier en résolvant le problème de moindre carré suivant:*

(M.C) $\quad Min \{ \| D_k c - A_k{}^t u \|,\ u \in IR^{mxn} \}$

dont la solution satisfait les équations normales:

(1) $A_k A_k^t u = AD_k^2 A^t u = AD_k^2 c$.

P_k est alors le résidu de (M.C): $P_k = D_k c - D_k^t u$

Le succès de l'implémentation de l'algorithme de KARMAKAR dépend donc de l'efficacité du procédé de résolution du système linéaire (1).

Les méthodes proposées jusqu'à présent sont de deux types:

*(1)- **Les méthodes directes** dans lesquelles on utilise la factorisation de la matrice du système (par exemple la méthode d'élimination de GAUSS).*

*(2)- **Les méthodes itératives** qui, produisent une suite de solutions approchées du système et n'utilisent en générale que des multiplications du type matrice vecteur telles que la méthode de JACOBY, GAUSS-SEIDEL et la méthode du gradient conjugué.*

Ceci étant, le système (1) possède deux caractéristiques

(1)- $A_k A_k^t$ est symétrique définie positive: Il suffit de voir que pour

$< A_k A_k^t x, x> = <A_k^t x, A_k^t x> = \| A_k^t x \|^2$.

La méthode fréquemment utilisée dans ce cas est la factorisation de CHOLESKY qui est un cas particulier de la méthode d'élimination de GAUSS.

(2)- Au cours de l'exécution de l'algorithme seuls les éléments de la matrice D_k peuvent changer d'une itération à l'autre.

Factorisation de CHOLESKY:

On appelle factorisation (ou décomposition) de CHOLESKY, la représentation d'une matrice symétrique définie positive $M \in IR^{mxn}$ sous la forme $M = LL^t$ ou L est une matrice triangulaire inférieure de IR^{mxm}

Exemple:

Pour $M = \begin{bmatrix} 2 & -2 \\ -2 & 5 \end{bmatrix}$ *on a* $L = \begin{bmatrix} \sqrt{2} & 0 \\ -\sqrt{2} & \sqrt{3} \end{bmatrix}$

Le calcul de la matrice L peut se faire par identification des coefficients dans l'équation:

$M = LL^t$

i.e par les formules suivantes: $M_{i,j} = \sum_{k=1}^{j} L_{i,k} . L_{j.k}$ *pour* $i \geq j$

compte tenu de la forme triangulaire de L qui entraîne $L_{j,k} = 0$ pour $k > j$.

Il existe plusieurs façons de programmer cette factorisation qui ne différent que par l'ordre dans lequel on effectue les opérations, sachant que dans tous les cas ce sont les mêmes opérations qui sont effectuées.

On peut par exemple choisir l'ordre ligne par ligne, ou colonne par colonne. L'efficacité des différents programmes dépend du mode de rangement des coefficients de la matrice dans la mémoire de l'ordinateur.

L'algorithme (version colonne):

Ayant calculé les (j - 1) premières colonnes, on calcule la $j^{ème}$ en écrivant:

$$L_{j.j} = \sqrt{M_{j,j} - \sum_{k=1}^{j-1} L^2{}_{j,k}}$$

$$L_{i,j} = (M_{i,j} - \sum_{k=1}^{j-1} L_{i,k} . L_{j,k}) / L_{j.j} \quad pour \quad i \geq j+1$$

Cet algorithme nécessite $0(n^3/6)$ opérations élémentaires (aditions, multiplications, etc…)

Ecrivons le programmes traduisant les formules ci-dessus en rangeant $L_{i,j}$ à la place de $M_{i,j}$.

PROGRAMME CHOLESKY:

```
POUR  j=1 à m FAIRE
    POUR i=j à m  FAIRE
        POUR k=1 à (j-1)  FAIRE
            M[i, j] = M[i, j] – M[i,k]*M[j,k]
        FIN de boucle k
      SI (i = j)  ALORS M[i,i]=SQRT(M[i,i])
            SINON     M[i, j] = M[i, j]/ M[j, j]
      FIN SI
   FIN de boucle i
 FIN de boucle j
```

Résolution de systèmes linéaires triangulaires:

1- Système triangulaire inférieur:

Considérons le système linéaire Ly=b , ou L est une matrice triangulaire inférieure de IR^{mxm} *et b un élément de* IR^{m}.

Ce système s'écrit encore (compte tenu de $L_{i,j}$*= 0 pour j>i)*

$$\sum_{j=1}^{i-1} L_{i,j}\, y_j + y_i L_{i,i} = b_i, i = 1 \grave{a}\, m \Rightarrow$$

$$y_i = (b_i - \sum_{j=1}^{i-1} L_{i,j} y_j) / L_{i,i} \quad i = 1 \grave{a} \quad m$$

Puisque b_i *ne sert plus après le calcule de* y_i*, on peut ranger* y_i *à la place de* b_i*. On obtient le programme suivant:*

PROGRAMME 1:

POUR i=1 à m FAIRE

POUR j=1 à (i-1) FAIRE

$b[i] = b[i] - L[i, j] * b[j]$

FIN de boucle j

$b[i] = b[i] / L[i, i]$

FIN de boucle i.

2- Système linéaire supérieur:

On peut donner un algorithme analogue pour un système linéaire triangulaire supérieur:

Ux=y

PROGRAMME 2:

POUR i=m à 1 FAIRE

POUR j=(i+1) à m FAIRE

$x[i] = x[i] - U[i, j] * x[j]$

FIN de boucle j

$x[i] = x[i] / U[i, j]$

FIN de boucle i

Application au système (1):

On factorise la matrice du système (programme CHOLESKY):

$A_k A_k^t = R_k R_k^t$ *(2)*

où R_k *est une matrice triangulaire inférieure de* IR^{mxm}

En utilisant (2) on a: $(R_k R_k^t)u = AD_k^2 c$ *(3)*

En posont $y = R_k^t u$ *, u peut être déterminé en résolvant d'abord le système linéaire triangulaire inférieur (algorithme1):*

$R_k y = AD_k^2 c$ *(4)*

Puis le système linéaire triangulaire supérieur (algorithme2):

$R_k^t u = y$ *(5)*

CHAPITRE -III-

Présentation générale de la nouvelle approche

III.1. Introduction

Comme nous l'avons signalé auparavant, pour pouvoir appliquer l'algorithme du simplexe, il est nécessaire que le programme linéaire soit écrit sous forme canonique relativement à une base réalisable. Si ce n'est pas le cas, on applique l'algorithme au problème auxiliaire associé afin de déterminer une base réalisable ou aboutir à la vacuité de l'ensemble des solutions réalisables.

Or, en pratique on ne dispose pas toujours d'une telle base et bien souvent le programme linéaire est présenté sous forme canonique relativement à une base mais non réalisable ce qui m'a incité à présenter une approche simplicaile n'exigeant pas la réalisabilité de base, si on en connaît une, et puis l'adapter aux problèmes linéaires présentés sous forme standard sans toutefois passer aux problèmes auxiliaires associés.

Nous commençons par résoudre un exemple simple par les deux méthodes.

A fin d'avoir une représentation plane, on prend l'exemple suivant, tiré de: LINEAR PROGRAMMINGAND NET WORK FLOWS, dont les auteurs sont M.S BAZARAA, J.J JARVIS, H.D SHERAL.

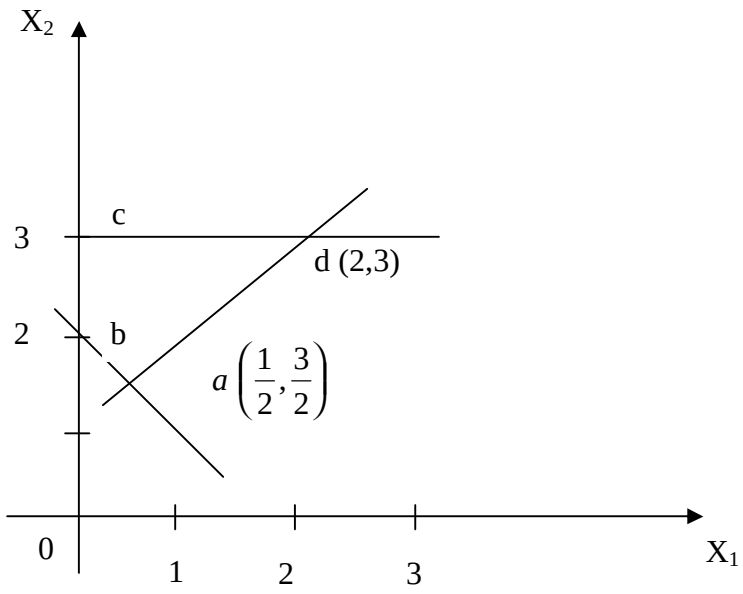
III- Résolution d'un exemple:

Exemple

$$(P) \qquad \begin{cases} MaxZ = -x_1 + 2x_2 \\ x_1 + x_2 \geq 2 \\ -x_1 + x_2 \geq 1 \\ x_2 \leq 3 \end{cases} \qquad x_1, x_2 \geq 0$$

il s'agit de trouver le point de l'ensemble des points réalisables D qui donne à la fonctionnelle $Z = -x_1 + 2x_2$ *sa plus grande valeur.*

On trace dans le place (x_1, x_2) le domaine D:



C'est le polygone convexe a b c d

III-1.1. par l'algorithme du simplexe

Après transformation des inéquations en équations par addition de variables d'écart x_3, x_4, x_5, *le programme linéaire précédent s'écrit :*

$$(P) \qquad \begin{cases} MaxZ = -x_1 + 2x_2 \\ x_1 + x_2 - x_3 = 2 \\ -x_1 + x_2 - x_4 = 1 \\ x_2 + x_5 = 3 \end{cases} \qquad x_i \geq 0$$

Phase 1

On associe à (P) le programme linéaire auxiliaire (PA) après addition de deux variables artificielles y_1 et y_2 à la première et la deuxième équation respectivement on obtient

$$(PA) \qquad \begin{cases} Maxy = 2x_2 - x_3 - x_4 - 3 \\ x_1 + x_2 - x_3 + y_1 = 2 \\ -x_1 + x_2 - x_4 + y_2 = 1 \\ x_2 + x_5 = 3 \end{cases} \qquad x_i \geq 0, y_i \geq 0$$

Où Ψ est le «fonction objectif» du problème auxiliaire (PA) qui est donc présenté sous forme canonique relativement à la base réalisable dont les variables sont y_1, y_2 et x_5. La solution associée est

$$\begin{cases} x_1 = x_2 = x_3 = x_4 = 0 \\ x_5 = 3 \\ y_1 = 2 \\ y_2 = 1 \end{cases}$$

1ère itération

La variable x_2 entre en base et la variable artificielle y_2 en sort définitivement.

Ecrivons (PA) sous forme canonique par rapport à la base des variables y_1, x_2, x_5.

$$(PA) \qquad \begin{cases} Maxy = 2x_1 - x_3 + x_4 - 1 \\ 2x_1 - x_3 - x_4 + y_1 = 1 \\ -x_1 + x_2 - x_4 = 1 \\ x_1 + x_4 + x_5 = 2 \end{cases} \qquad x \geq 0, y_1 \geq 0$$

La solution de base correspondante est:

$$\begin{cases} x_1 = x_3 = x_4 = 0 \\ x_2 = 1 \\ x_5 = 2 \\ y_1 = 1 \end{cases}$$

2ème itération

La variable x_1 entre en base et la variable artificielle y_1 sort en écrivant le programmaire linéaire auxiliaire (PA) sous forme canonique relativement à la base des variables x_1,x_2, et x_5 on obtient.

$$(PA) \qquad \begin{cases} Maxy = 0 \\ x_1 - \frac{1}{2}x_3 + \frac{1}{2}x_{41} = \frac{1}{2} \\ x_2 - \frac{1}{2}x_3 - \frac{1}{2}x_4 = \frac{3}{2} \\ \frac{1}{2}x_3 + \frac{1}{2}x_4 + x_5 = \frac{3}{2} \end{cases} \qquad x \geq 0$$

Cette solution est optimale car le vecteur coût de la fonctionnelle y est nul, elle correspond au point de coordonnées $(x_1, x_2) = \left(\frac{1}{2}, \frac{3}{2}\right)$ qui est un point extrémal (sommet) du polygone admissible du programme.

On a déterminé une base réalisable du programme linéaire initial (P), on peut donc passer à la deuxième phase.

Phase 2

On reprend le problème initial mis sous forme canonique par rapport à la base trouvée, il s'écrit :

$$(P)\qquad \begin{cases} MaxZ = \frac{1}{2}x_3 + \frac{3}{2}x_4 + \frac{5}{2} \\ x_1 - \frac{1}{2}x_3 + \frac{1}{2}x_4 = \frac{1}{2} \\ x_2 - \frac{1}{2}x_3 - \frac{1}{2}x_4 = \frac{3}{2} \\ \frac{1}{2}x_3 + \frac{1}{2}x_4 + x_5 = \frac{3}{2} \end{cases} \qquad x \geq 0$$

1ère itération :

La variable x_4 entre dans la base et x_1 en sort et (P) devient :

$$(P)\begin{cases} MaxZ = -3x_1 + 2x_3 + 4 \\ 2x_1 - x_3 + \frac{1}{2}x_{41} = 1 \\ x_1 + x_2 - x_3 = 2 \\ -x_1 + x_3 + x_5 = 1 \end{cases} \qquad x \geq 0$$

Avec comme solution de base

$$\begin{cases} x_1 = x_3 = 0 \\ x_2 = 2 \\ x_4 = x_5 = 1 \end{cases}$$

La solution n 'est pas optimale puisque le coefficient de x_3 est strictement positif.

2ème itération

la variable x_3 entre dans la base et x_s en sort et (P) s 'écrit:

$$(PA)\qquad \begin{cases} MaxZ = -x_1 - 2x_5 + 6 \\ x_1 + x_4 + x_5 = 2 \\ x_2 + x_5 = 3 \\ -x_1 + x_3 + x_5 = 1 \end{cases} \qquad x \geq 0$$

Le vecteur coût c de Z est négatif par conséquent la base courante dont les variables sont x_2, x_3 et x_4 est une base optimale et la solution de base est :

$$\begin{cases} x_1 = x_5 = 0 \\ x_2 = 3 \\ x_3 = 1 \\ x_4 = 2 \end{cases}$$

qui correspond au sommet de coordonnées (x_1,x_2)=(0,3) du polygone admissible.

Résumons les différentes étapes de la procédure

1) En appliquant l'algorithme du simplexe au programme auxiliaire (PA)

avec comme point de départ l'origine on passe au point (0,1) qui n 'est pas réalisable; en l'appliquant une deuxième fois on passe au

point réalisable $\left(\frac{1}{2},\frac{3}{2}\right)$.

2) On reprend le programme linéaire initial (P); en appliquant l'algorithme à celui-ci avec comme point de départ le sommet du polygone trouvé précédemment de coordonnées $\left(\frac{1}{2},\frac{3}{2}\right)$ *on passe au point (0,2), puis en l'appliquant une autre fois on aboutit à la solution optimale qui est le sommet du polygone de coordonnées (0,3).*

Le nombre des itérations est donc (4).

III. 1.2 Par la nouvelle approche

Après addition des variables d'écart x_3,x_4 et x_5 et en multipliant les deux premières équations par -1, le programme linéaire s'écrit :

$$(P) \qquad \begin{cases} MaxZ = -x_1 + 2x_2 \\ -x_1 - x_2 + x_5 = -2 \\ x_1 - x_2 + x_4 = -1 \\ x_2 + x_5 = 3 \end{cases} \quad x_i \geq 0$$

Il est clair que le programme linéaire (P) est mis sous forme canonique par rapport à la base des variables x_3, x_4 et x_5. Cependant cette base est non réalisable puisque la solution de base est:

$$\begin{cases} x_1 = x_2 = 0 \\ x_3 = -2 \\ x_4 = -1 \\ x_5 = 3 \end{cases}$$

Mais on peut la prendre comme base de départ pour un processus cherchant à déterminer l'optimum du programme linéaire (P).

***1ère itération**:*

Comme le seul coefficient (coût réduit) strictement positif de la fonction objectif est celui de la variable x_2, celle-ci entre dans la base, et on choisit x_5 comme variable de sortie (nous aurons à justifier ce choix par la suite).

La nouvelle base des variables x_2, x_3, x_4, qui est optimale, est la même que celle trouvée auparavant par l'algorithme du simplexe.

Conclusion :

On est arrivé à résoudre le programme linéaire donné en une seule itération contre quatre avec l'algorithme du simplexe.

Nous allons alors étendre au cas général les principes de la procédure précédente.

III. 2. Description de la méthode

Pour son utilité on rappelle le lemme de Minkowski Farkas, connu également sous le nom de thèrème des alternatives.

III 2.1 Lemme de Minkowski Farkas

L'un et seulement l'un des systèmes de contraintes suivants a une solution :

$$(1)\begin{cases} Ax = b \\ x \geq 0 \end{cases} \qquad (2)\begin{cases} yA \leq 0 \\ yb > 0 \end{cases}$$

III.2.2 Théorème

Soit le programme linéaire:

$$(P) \qquad \begin{cases} MaxZ = cx \\ Ax = b \\ x \geq 0 \end{cases}$$

écrit sous forme canonique relativement à une base quelconque B c 'est à dire réalisable ou non, J désigne l 'ensemble des indices de ses colonnes. S 'il existe un indice $s \in \bar{J} = N - J; N = \{1,2,...,n\}$ *tel que la colonne* $A^s < 0$ *et le coût réduit* $c_s > 0$ *alors le programme linéaire (P) n 'a pas de solution optimale et on a :*

l) ou bien le système des contraintes est incompatible .

2) Ou bien l 'optimum de la fonction objectif est non borné.

Preuve :

Puisque la base B est (à une permutation prés) la matrice unité U, en permutant les colonnes de A, on suppose pour simplifier la notation que $A = (U,R)$ *avec* $U=B=A^J$ *et* $A^{\bar{J}}$ *et* $A^{\bar{J}} = R$ *est la sous matrice formée par les colonnes de A qui ne sont pas dans la base B.*

Notons que toute solution du programme linéaire (P) vérifie Ax=b par suite on a $x_j + Rx_{\bar{J}} = b$ *et donc* $x_J = b - Rx_{\bar{J}}$

Considérons le point x défini par :

$$(1) \qquad \begin{cases} x_s = q \quad , \quad q \in IR \\ x_j = 0 \quad , \quad j \in \bar{J} - \{s\}, \bar{J} = \{1,2,...,n\} - J \\ x_J = b - Rx_{\bar{J}} = b - A^s x_s = b - qA^s \end{cases}$$

On distingue trois cas :

Cas a :

La colonne, $A^s < 0$, *c'est à dire que* $a_{is} < 0$ *pour tout* $i \in \{1,2,...,n\}$

Dans ce cas le point x défini par la relation (1) est réalisable si et seulement si $q \geq 0$ *et* $b - qA^s \geq 0.$

C'est à dire que $q \geq 0$ *et* $b_i - qa_{is} \geq 0$, *pour tout i, comme* $a_{is} < 0$ *on a :*,

$$\begin{cases} q \geq 0 \\ b_i - q\, a_{is} \geq 0, \forall_i \end{cases} \Leftrightarrow \begin{cases} q \geq 0 \\ q \geq \dfrac{b_i}{a_{is}} \forall i \end{cases}$$

Posons $q_0 = \underset{i}{Max}\left\{\dfrac{b_i}{a_{is}}\right\}$, *alors le point x est réalisable pour tout*

$q \geq Sup(0, q_0)$

On a défini ainsi un ensemble de points réalisables sur lequel la «fonction objectif» Z (Z = $c_s q$ *) augmente indéfiniment avec q et par conséquent le programme linéaire (P) n'admet pas de solution finie.*

Cas b :

La colonne, $A^s = 0$, *c'est à dire* $a_{is} = 0$ *pour tout i de {1,2,... .n}.*

Ce cas se subdivise lui-même en deux cas :

Cas b_1

L'ensemble des solutions réalisables est vide auquel cas le programme linéaire (P) n'a pas de solution.

Cas b_2

L'ensemble réalisable n'est pas vide autrement dit, il existe une base B ' qui soit réalisable.

Désignons par J' l'ensemble des indices de colonnes de B'. Comme $A^s = 0$_ *alors,* $s \notin J'$.

Montons à présent qu'en écrivant le programme linéaire (P) sous forme canonique par rapport à la nouvelle base B ', le coût réduit c_s *et la colonne* A^s *restent inchangés.*

Puisque $A^s = 0$ *on alors*

$$A'^s = (B')^{-1} A^s = 0 \text{ et } \quad c_s^{'} = c_s - c_{J'}(B')^{-1} A^s = c_{s.}$$

La solution associée à la base B' est $\begin{cases} x_{J'} = (B')^{-1} b \geq 0 \\ x_j = 0,\ j \notin J' \end{cases}$

Considérons x défini par

$$\begin{cases} x_s = q, q \geq 0 \\ x_j = 0,\ j \in J' \cup \{s\} \\ x_{J'} = (B')^{-1} b \geq 0 \end{cases}$$

x est réalisable car d'une part on a $x \geq 0$ *et d'autre part*

$$Ax = B' x_{J'} + qA^s = B' x_{J'} = b$$

La valeur de la «fonction objectif» en x est

$$Z = c_{\bar{J}} x_{\bar{J}} = c_{J'} x_{J'} + c_s q = c_{J'} (B')^{-1} b + c_s q$$

qui tend vers l'infini avec q *et par conséquent l'optimum est non borné.*

Cas c :

La colonne $A^s \neq 0 et \exists i (1 \leq i \leq m) : a_{is} = 0$.

Posons $I = \{i : a_{is} = 0\}$ *et supposons pour simplifier la notation que A est sous la forme*

$$A = \begin{bmatrix} A_{\bar{I}} \\ A_I \end{bmatrix} \quad ou \quad \bar{I} = \{1,2,...,m\} - I$$

On a alors $A_I^s = 0$ *et* $A_{\bar{I}}^s < 0.$

Rappelons que la base B est sous la forme $B = \begin{pmatrix} U_1 & 0 \\ 0 & U_2 \end{pmatrix}$

Où U_1 *et* U_2 *sont les matrices unités de rang respectivement card* $(\bar{I})$ *et card (I) leurs ensembles d'indices de colonnes sont respectivement* J_1 *et* J_2 *.*

Et le système de contraintes s'écrit :

$$\begin{cases} A_{\bar{I}} x = b_{\bar{I}} \\ A_I x = b_1 \\ x \geq 0 \end{cases}$$

Ce cas se subdivise lui-même en deux cas.

Cas* C_1 *:

Le système $\begin{cases} A_I x = b_I \\ x \geq 0 \end{cases}$

N'admet pas de solution.

Dans ce cas on sait d'après le lemme de Minkowski qu'il existe un card (I) - vecteur ligne vérifiant $yA_I \geq 0$ *et* $yb_I < 0.$

Définissons alors le n-vecteur par

$$z_i = \begin{cases} y_i, i \in I \\ 0, i \in \bar{I} \end{cases}$$

On a d'une part $ZA = yA_I + 0A_{\bar{I}} = yA_I < 0$ *et d'autre part* $zb = yb_I < 0$

Donc le système $\begin{cases} Ax = b \\ x \geq 0 \end{cases}$ *n a pas de solution*

Et par conséquent le programme linéaire (P) n'a pas de solution (vacuité de l'ensemble réalisable).

Cas c_2 :

Le système $\begin{cases} A_I x = b_I \\ x \geq 0 \end{cases}$ *admet une solution.*

Dans ce cas, il existe une sous matrice carrée B_0 de rang = card(I). Désignons par J_0 l'ensemble d'indices de colonnes de A qui constituent B_0, on a alors $x_{J_0} = B_0^{-1} b_I \geq 0.$

Notons que :

(1) $s \notin J_0$ *car* $a_{is} = 0$ *pour tout* $i \in I$

(2) Aucun élément de J_1 ne peut appartenir à J_0 car $a_{ij} = 0,$ *pour tout*

$i \in I$ *et* $j \in J_1$

Par conséquent la sous matrice $B' = \begin{pmatrix} U_1 & M \\ 0 & B_0 \end{pmatrix}$ *dont l'ensemble des indices de colonnes* $J' = J_1 \cup J_0$ *est une base du programme linéaire (P). Montons qu'en écrivant (P) relativement à la base B ' la colonne A^s et le coût réduit c_s restent inchangés.*

En effet:

$c' = c - c_{J'}(B')^{-1} A$ *ce qui donne*

$$c_s^{'} = c_s - c_{J'}(B')^{-1} A^s = c_s - (c_{J_1}, c_{J_0}) \begin{pmatrix} U_1 - MB_0^{-1} \\ 0 \quad\quad B_0^{-1} \end{pmatrix} \begin{pmatrix} A_{\bar{I}}^s \\ 0 \end{pmatrix}$$

$$= c_s - (0, c_{J_0}) \begin{pmatrix} A_{\bar{I}}^s \\ 0 \end{pmatrix} = c_s .$$

$$A^{'s} = (B')^{-1} A^s = \begin{pmatrix} U_1 - MB_0^{-1} \\ 0 \quad\quad B_0^{-1} \end{pmatrix} \begin{pmatrix} A_{\bar{I}}^s \\ 0 \end{pmatrix} = A^s$$

La solution de base est

$$x_{J'} = (B')^{-1} b = \begin{pmatrix} U_1 - UB_0^{-1} \\ 0 \quad\quad B_0^{-1} \end{pmatrix} \begin{pmatrix} b_{\bar{I}} \\ b_1 \end{pmatrix} = \begin{pmatrix} b'_{\bar{I}} \\ b_1 \end{pmatrix}$$

Où on a posé $b'_{\bar{I}} = b_{\bar{I}} - MB_0^{-1}b_I$

Considérons le point x défini par :

$$\begin{cases} x_s = q \\ x_j = 0,\ j \in \bar{J}' - \{s\},\ \bar{J} = \{1,2,...,n\} - J' \\ x_{J'} = (B')^{-1}b - (B')^{-1}A^s x_s = \begin{pmatrix} b'_{\bar{I}} \\ B_0^{-1}b_I \end{pmatrix} - q\begin{pmatrix} A^s_{\bar{I}} \\ 0 \end{pmatrix} \\ \qquad\qquad = \begin{pmatrix} b'_{\bar{I}} - qA^s_{\bar{I}} \\ B_0^{-1} \quad b_I \end{pmatrix} \end{cases}$$

Puisque $B_0^{-1}b_I \geq 0$ *alors le point x est réalisable si et seulement si*

$$\begin{matrix} q \geq 0 \\ b'_i - q\,a_{is} \geq 0 \end{matrix}, \text{ pour tout } i \in \bar{I}^{\mathbf{s}}$$

Comme $a_{is} < 0$ *alors x est réalisable pour* $q \geq Sup(0, q_0), q_0 = Max\left\{\frac{b'_i}{a_{is}}\right\}$,

la valeur de la fonction objectif $Z = c_{J_0}x_{J_0} + c_s x_s = c_{J_0}B_0^{-1}b_1 + c_s q$ *augmente indéfiniment avec* q *et par conséquent l'optimum est non borné.*

L'intérêt du théorème précèdent vient du fait que l'approche proposée en découle directement.

III.2.3 l'algorithme

On suppose que l'on dispose d'une base quelconque B, désignons par J l'ensemble d'indices de colonnes de A qui constitue la base B à l'itération k, ceux ci sont rangés dans un m-vecteur ligne noté col i.e J={col(l),...,col(n)} les différentes étapes de l'algorithmes sont les suivants :

Etape1*: Calculer l'inverse de la matrice B :*B^{-1}

Etape2: *Calculer* $p = c_j B^{-1}$

Etape3: *Calculer le nouveau vecteur réduit hors base par* $c_{\bar{j}} = c_{\bar{j}} - pR$

Etape4: *Déterminer le plus petit indice* $s \in \bar{J}$ *, tel que :*

$$c_s = \underset{j \in \bar{J}}{Max}\{c_j\}$$

Etape5: *Calculer b ' par la formule* $b' = B^{-1}b$

Si $c_s > 0$ *passer à l'étape (10) sinon (C 'est à dire* $c_s \leq 0$*)*

Etape6: *Déterminer le plus petit indice r tel que :*

$b'_r = \underset{i}{Min}\{b'_i\}$ *si* $b'_i \geq 0$ *; arrêt : la base courante est optimale.*

Sinon (c'est-à-dire $b'_r < 0$ *)*

Etape7: *Calculer la ligne* A'_r *par* $a'_{rj} = \sum_{k=1}^{n} b_{rk}^{-1} a_{kj}, j \in \bar{J}$

Etape8: *Déterminer* $L : L = \{j \in \bar{J} / a'_{rj} < 0\}$ *si* $L = f$ *;Arrêt : l'ensemble réalisable est vide.*

Sinon (c'est à dire $L \neq f$ *)*

Etape9: *Déterminer le plus petit indice s vérifiant*

$\frac{c'_s}{a'_{rs}} = \underset{j \in L}{Min} \left\{ \frac{c'_j}{a'_{rj}} \right\}$ *et passer à l'étape (13)*

Etape10: *Calculer la colonne* A'^s *par la formelle* $A'^s = B^{-1} A^s$

Etape11: *Déterminer* $I : I\{i / a'_{is} > 0\}$

Si $I = f$ *; Arrêt : le programme n 'a pas de solution (optimum infini ou vacuité de l'ensemble réalisable) sinon (c'est-à-dire* $I \neq f$ *)*

Etape12: *Déterminer le plus petit indice*

$$r : \frac{b'_r}{a'_{rs}} = \underset{i \in I}{Min}\left\{\frac{b'_i}{a_{is}}\right\}$$

Etape13: *Mise à jour de la base :* $J' = J \cup \{s\} - col(r), col(r) = s$

Etape14: *retourner à l'étape(1)*

III. 3 Initialisation de l'algorithme

Dans ce paragraphe on décrit la démarche à suivre pour résoudre un programme linéaire (P) si l'on ne dispose pas de base (quelconque) de départ.

On commence par introduire la notion de base partielle et donner deux corollaires du théorème du paragraphe précédent.

III. 3.1 Corollaire 1

Considérons le programme linéaire (P) écrit sous forme standard

$$(P) \quad \begin{cases} MaxZ = cx \\ Ax = b \\ X \geq 0 \end{cases}$$

S'il existe un indice s tel que $c_s>0$ et $A^s=0$ alors le problème (P) n'admet pas de solution.

Ou bien l'optimum est non borné ou bien le système des contrantes est incompatible.

Démonstration

En remplaçant $Ax=b$ par ($Ax \leq 0 et \; -Ax \leq -b$)

le programme linéaire (P) prend la forme :

$$(P) \quad \begin{cases} MaxZ = cx \\ Ax \leq x \\ -Ax \leq -b \\ x \geq 0 \end{cases}$$

qui peut être, donc considéré mis sous forme canonique relativement à la base dont l'ensemble des indices de ses colonnes est
$J'=\{n+1,\ldots,n+2m\}$ avec $c_s>0$ et $A^s=0$, les conditions d'application du théorème sont réunies et le corollaire est démontré.

III. 3.2 Définition d'une base incomplète

Etant donné un programme linéaire sous forme standard

$$(P) \quad \begin{cases} MaxZ = cz \\ Ax = b \\ x \geq 0 \end{cases}$$

tel que rang(A)=m, on appelle base incomplète (ou partielle) du programme linéaire (P) toute sous-matrice carrée régulière extraite de A de rang strictement inférieure à m.

Si B est alors une base incomplète de (P), il existent un ensemble d'indices de lignes $I \subset \{1,2,...,m\}$ et un ensemble d'indices de colonnes $J \subset \{1,2,...,m\}$ de la matrice

A tels que $B = (A_I^J) = (a_{ij}),\ i \in I,\ j \in J$ *et l'hypothèse suivant laquelle le programme linéaire (P) est écrit sous la forme canonique par rapport à B s'exprime par :*

$B = A_I^J$ *est, à une permutation près des colonnes, la matrice unité.*

$C^{J=0}$

$A_{\bar{I}}^J = 0$, ou $\bar{I} = \{1,2,...,m\} - I$.

III. 3.3 Corollaire 2

Soit le programme linéaire

$$(P) \quad \begin{cases} MaxZ = cx \\ Ax \\ x \geq 0 \end{cases}$$

écrit sous forme canonique par rapport à la base incomplète $B = A_I^J$ *s'il existe un indice* $s \in \bar{J} = \{1,2,...,n\} - J$ *tel que :*

(1) $C^s > 0$

(2) $a_{is} \leq 0, i \in I$

(3) $a_{is} = 0, i \in \bar{I} = \{1,2,...,n\} - I$

Alors le programme linéaire (P) n'admet pas de solution.

Démonstration :

En décomposant la matrice A, le problème (P) peut s'écrire sous la forme:

$$(P) \quad \begin{cases} MaxZ = cx \\ A_I x = b_I \\ A_{\bar{I}} x = b_{\bar{I}} \\ x \geq 0 \end{cases}$$

Soit encore

$$(P) \quad \begin{cases} MaxZ = cx \\ A_I x = b_I \\ A_{\bar{I}} x \le b_{\bar{I}} \\ -A_{\bar{I}} x \le -b_{\bar{I}} \\ x \ge 0 \end{cases}$$

donc il est écrit sous forme canonique par rapport à la base dont l'ensemble des indices des colonnes est

$$J' = J \cup \left\{ n+1, n+2, ..., n+2card(\bar{I}) \right\}$$

avec les conditions C_s>0 et $A^s \le 0$ et par conséquent le corollaire et démontré d'après le théorème du paragraphe précédent.

III. 3.4 initialisation de l'algorithme (phase 1)

Supposons que le programme linéaire est écrit sous forme standard et que l'on ne dispose d'aucune base.

Sans perte de généralité on suppose que le problème se présente sous la forme

$$(P) \quad \begin{cases} MaxZ = cx \\ A_I x = b_I \\ A_{\bar{I}} x = b_{\bar{I}} \\ x \ge 0 \end{cases} :$$

Ou (A_I^J) est une base incomplète sans toutefois exclure le cas ou $I = f$.

Le programme linéaire (P) est équivalent alors au problème:

$$(P') \quad \begin{cases} MaxZ = cx \\ A_I x = b_I \\ A_{\bar{I}} x \le b_{\bar{I}} \\ -A_{\bar{I}} x \le -b_{\bar{I}} \\ x \ge 0 \end{cases}$$

On peut appliquer l'algorithme de l'approche simpliciale à (P').

Remarquons que la taille de la matrice des contraintes ne change pas.

III . 4. Simulation numérique

III. 4.1.Introduction

Sachant, d'après les travaux munis par plusieurs chercheurs, que la comparaison entre l'algorithmes de KAR MAKAR et celui du simplexe est en faveur de ce dernier (les performances de l'algorithmes de KAR MARKAR seraient à apprécier dans les programmes à grande dimension), nous nous intéressons uniquement à la comparaison entre la méthode du simplexe et la nouvelle approche simpliciale proposée.

L'objet de ce bref ce chapitre est la comparaison en aspect théorique et surtout en terme d'efficacité numérique de l'algorithme du simplexe et celui que nous proposons. Il est consacré essentiellement aux tests numériques.

Notre étude comparative est organisée de la manière suivante: sachant que les deux algorithmes coïncident lorsque la base de départ du problème en question est réalisable on ne s'intéressera alors qu'aux:

(1) Programmes linéaires dont on dispose d'une base de départ mais celle-ci est non réalisable.

(2) Programmes linéaires sous forme standard sans connaître toutefois de base.

III.4.2. résolution de quelques exemples

Exemple1 : (P.tolla)

Problème sous fourme standard sans base de départ.

$$MaxZ = -3x_1 + x_2 - x_3$$

$$\begin{cases} 2x_1 + x_2 - x_4 = 0 \\ x_3 + x_5 - x_6 = 0 \\ x_1 + x_2 + x_3 + x_4 + x_5 + x_6 = 1 \end{cases}$$

$$x_1 \geq 0, i = 1,2,..,6$$

Exemple2

$$MaxZ = 6x_1 + 5x_2 + 4x_3$$

$$\begin{cases} x_1 + x_2 + x_3 \le 7 \\ 3x_1 + x_3 \ge 6 \\ 4x_2 + 3x_3 \ge 12 \\ x_1 + 2x_3 \ge 2 \\ 2x_1 + x_2 + 2x_3 \le 24 \\ x_i \ge 0, i = 1,2,3 \end{cases}$$

Exemple3

$$MaxZ = 2x_1 - x_2$$

$$\begin{cases} -x_1 + x_2 \le 1 \\ x_1 + x_2 \ge 3 \\ x_2 = 3 \end{cases}$$

$$x_i \ge 0, i = 1,2$$

Exemple4 (BAZARAA)

$$MaxZ = -x_1 + 2x_2$$

$$\begin{cases} x_1 + x_2 \ge 2 \\ -x_1 + x_2 \ge 1 \\ x_2 \le 3 \\ x_1, x_2 \ge 0 \end{cases}$$

Les tableaux ci-dessous représentent les résultats obtenus par les deux méthodes

Exemple1

Méthode	***Phase 1***	***Phase 2***
Simplexe	***3 itérations***	***2itérations***
Approche	***1***	***2***

Le point optimum $x^* = \left(0, \frac{1}{2}, 0, \frac{1}{2}, 0, 0\right)$ *et la fonction objectif vaut* $Z^* = \frac{1}{2}$

Exemple2

Méthode	***Phase 1***	***Phase 2***
Simplexe	***3 itérations***	***2itérations***
Approche	***0***	***3***

Avec x*=(2 ,4,1,0,1,7,2,0,0)

Z=36*

Exemple 3:

Méthode	***Phase 1***	***Phase 2***
Simplexe	***3 itérations***	***0***
Approche	***0***	***0***

Le programme linéaire n'admet pas de solution finie.

Exemple 4:

Méthode	***Phase1***	***Phase2***
Simplexe	***2 itérations***	***2 itérations***
Approche	***0***	***1***

avec comme optimum x=(0,3)*

et Z=6*

III.4.3 Conclusion finale:

Signalons que le temps nécessaire pour l'exécution d'une itération est le même pour les deux algorithmes.

En ce qui concerne la comparaison du point de vue théorique nous pouvons dire que la différence principale est que pour la méthode du simplexe on est obligé de passer par le problème auxiliaire dont la fonction objectif est totalement différente de

celle du programme initiale dans le but de déterminer une base réalisable à qui demande par fois un nombre important d'itérations. Par contre pour la méthode que nous proposons on évite ce passage ce qui permet de réduire le nombre d'itérations et par conséquent le temps d'exécution.

En terme d'efficacité numérique, les expérimentations réalisées montrent bien que la méthode proposée est de 2 à 4 fois plus rapide que la méthode du simplexe.

Cette nouvelle approche simpliciale va, probablement, enrichir le domaine de la programmation linéaire aussi bien sur le plan théorique que sur le plan pratique.

REFERENCES

[1] *ILAN ADLER*
An implementation of KARMAKAR'S algorithme for linear programming.
Mathematical programming (1995)

[2] *M.S BAZARAA, C.M.SHETTY*
Linear programme and net work fmows.

[3] *J.H Haeberley and M.Overton*
Primal-dual interior-point methods for semi definite programming SIAM J.Optim, 8 (1998) pp.746-768

[4] *VG Karmanov*
Mathematical, Moscow 1989

[5] *M.Minoux*
Programmation mathématique:
Théorie et algorithmes
Tome 1 Dunod, Paris 1983

[6] *M.Sakarovitch*
Programmation linéaire
E.N.S.I.M.A.G, Grenoble, Novembre 1983

[7] *M.Shida, Shindoh, and M.Kojima*
Existence of search directions in interior-point algorithmes for the SDP and the monotone SDLCP. SIAM, 8(1998) pp 387-396

[8] *M.Simonnard*
Programmations linéaire Tome1
Dunod, Paris

[9] *M.J TODD*
The Many facets of linear programming
Math. Programming Serie B (2001)

[10] *P.Tolla*
Elaboration de logiciels efficaces utilisant l'algorithme de Karmakar
Lansade, Février 1988, Université de Paris, Dauphine

TABLE DES MATIERES

Introduction général *……..P01*

Chapitre I : Fondements théoriques et présentation générale de la méthode du simplexe *…….P04 .*
Introduction.……..P05.
Notions fondamentales.……..P05.
- définition d'un problem d' optimisation mathématique.……P05.
- Définition d'un programme linéaire.……..P05.
- Forme CANONIQUE et forme STANDARD d'un PROGRAMME LINEAIRE.…..P06.
- Bases, bases réalisables, solution de base.……..P07.
- Caractérisation des bases et des solutions de bases optimales.……..P10.
- L'algorithme du simplexe.……..P11.
- Résolution d'un exemple.……..P12.
- Initialisation de l'algorithme du simplexe.……..P14.
- Dualité.……..P17.
- L'algorithme dual du simplexe. ……..P17.

Chapitre II: Présentation générale de la méthode de KARMAKAR et description de son algorithme *……..P20.*
- Introduction.……..P21.
- Préliminaires.……..P21.
- Description de la méthode.……..P25.
- Extension de l'algorithme de KARMAKAR.……..P28.

Chapitre III:Présentation générale de la nouvelle approche *…….P39.*
-Introduction.……..P40.
- Résolution d'un exemple *……..P40.*
- Description de la méthode.……..P46.
-l'algorithme.……..P51.
- Initialisation de l'algorithme.……..P52.
- Simulation numériques ……..P56.
- Résolution de quelques exemples.……..P56.
- Commentaires et conclusion finale: ……..P.58

مختصـــر:

إن الهدف من هذا العمل هو عرض صيغة بسيطية (Simpliciale) جديـــدة لحـــل و معالجة مشاكل البرمجة الخطية دون اشتراط معرفة قاعدة (أساس) مسبقا عند البدايـــة. ثم المقارنة نظريا و خاصة من حيث الفعالية بين الصيغة المقترحة و الطريقـــة الـــشهيرة المعروفة بـــ: Méthode du simplexe.

Résumé:

L'objet de ce travail est la présentation d'une nouvelle approche simpliciale pour la résolution d'un problème de programmation linéaire qui élimine l'hypothèse de connaître à priori une base réalisable et la comparaison théorique et surtout en terme d'efficacité numérique entre celle-ci et la fameuse méthode du simplexe.