Faculté des Sciences et des Sciences Appliquées
Département de Mathématiques

Mémoire de fin d'étude

En vue de l'obtention du diplôme de Master en :
Filière : Mathématiques
Option : Recherche Opérationnelle

THÈME

Résolution d'un problème de programmation linéaire avec la méthode DC

Présenté par :

M^r Sid Ali HAMDACHE

Soutenu le 29 Novembre 2018 devant le jury composé de :

Président	M^r Abderrahmane AKKOUCHE	M.C.B	U. A/M/Oulhadj Bouira.
Encadreur	M^r L'hadi BOUGHANI	M.A.A	U. A/M/Oulhadj Bouira.
Examinateur	M^r Karim HAMID	M.A.A	U. A/M/Oulhadj Bouira.
Examinateur	M^r Sofiane DJOUDER	M.A.A	U. A/M/Oulhadj Bouira.
Invité	M^r Mohamed-Ahmed BOUDREF	M.C.B	U. A/M/Oulhadj Bouira.

BOUIRA 2018

Remerciements

En premier lieu, je tiens à remercier le Bon **DIEU** de m'avoir donné le courage, la santé et la volonté pour continuer et achever ce travail.

Je remercie vivement mon promoteur **Mr L'HADI BOUGHANI** pour avoir accepté de m'encadrer et aussi pour l'effort fournis, ses encouragements, et ses précieux conseils tout au long de la réalisation de ce mémoire.

Je tiens aussi à remercier également **Mr KARIM HAMID** et **SOFIANE DJOUDER** qui m'ont fait l'honneur d'accepter d'évaluer ce travail, ainsi que **ABDERRAHMANE AKKOUCHE** pour l'honneur qu'il me fait en présidant le jury de ce mémoire de master.

Un grand merci pour **Mr MOHAMED-AHMED BOUDREF** d'accepter d'être parmi nous, et qui donne une valeur à mon modeste travail.

Mes sincères remerciements vont à **Mme SABRINA BENGRAB**, la secrétaire de département Mathématiques, pour sa remarquable gentillesse.

Un grand merci à mes **PARENTS** et ma fiancée **KENZA AIS**.

Je remercie tous mes **ENSEIGNANTS** et tous mes **AMIS**

Je remercie enfin tous ceux qui d'une manière ou d'une autre ont contribué à la réussite de ce travail et qui n'ont pas pu être cités ici.

S.Hamdache

Dédicaces

Je dédie ce modeste travail :

A mes très chères parents ;

A ma chère fiancée Kenza ;

A ma petite sœur Ryma, mon frère Billal et toute ma famille ;

A ma grand-mère, mes oncles, mes tantes, cousins et cousines ;

A tous mes amis qui sont en Algérie et à l'étranger ;

A tous ceux qui me sont chère(e)s ;

A tous mes enseignants ;

A tous mes camarades et collègues d'études ;

A ceux qui me connaissent ;

A tous ceux qui m'ont aidé de près ou de loin .

S. Hamdache

Table des matières

Table des figures

Abréviations & notations

Symboles	Désignation
DC	Déférence de deux fonctions convexes.
DCA	Algorithme **DC**.
s.c.i	Semi-continue inférieurement.
PL	Programme linéaire.
$Co(A)$	L'enveloppe convexe de l'ensemble A.
$dom\ f$	Le domaine effectif de la fonction f.
$epi(f)$	L'épigraphe de la fonction f.
$\Gamma_0(\mathbb{R}^n)$	L'ensemble de toutes les fonctions semi-continues inférieurement, propres et convexes sur $\mathbb{R}^n$.
χ_C	La fonction indicatrice de l'ensemble C.
f^*	La fonction conjuguée de f.
f^{**}	La bi-conjuguée de f.
$\partial f(x_0)$	Sous-différentiel de f au point x_0.
$Conv(C)$	L'ensemble des fonctions convexes sur C à valeur dans $\mathbb{R}$.
DC(C)	L'ensemble des fonctions **DC** sur C.
L_{C^2}	L'ensemble des fonctions de classe C^2 sur C.

Introduction générale

L'optimisation est à la fois une science et un outil largement utilisé dans divers domaines scientifiques, en ingénierie comme en industrie, qui nous aide à prendre la meilleure décision pour un grand nombre de décisions possibles. Elle s'efforce à la fois de construire des méthodes de calcul pour trouver des solutions optimales, d'explorer les propriétés théoriques et d'étudier l'efficacité numérique des algorithmes.
Il ne serait pas exagéré de dire que chacun utilise l'optimisation dans sa vie d'une façon ou d'une autre. Parmi les éminentes applications d'optimisation nous pouvons citer la gestion de la planification de production, les réseaux informatiques, différentes filières d'ingénierie, etc. Pour résoudre ces problèmes, l'optimisation offre un cadre algorithmique très riche. Dans ce cadre d'étude, il nous faut d'abord distinguer deux filières d'optimisation :

- les modèles d'optimisation stochastique,
- les modèles d'optimisation déterministe.

Ce mémoire se situe dans le contexte d'optimisation déterministe.

La programmation linéaire est une branche de la programmation mathématique qui s'occupe de problèmes d'optimisation linéaire. On dispose au moins deux méthodes efficaces et exactes de résolution : la méthode du simplexe, découverte par Dantzig en 1947 et la méthode adaptée a été proposée par Gabassov et Kirrillov durant les années 70. Cette discipline a longtemps fait l'objet de communications dont les apports se situaient essentiellement au niveau de la performance des algorithmes proposés. Depuis la fin des années quatre-vingt, des approches innovantes d'utilisation de ces techniques sont disponibles et, combinées aux progrès spectaculaires réalisés en micro-informatique, rendent accessibles ces approches au traitement de problèmes complexes, dans des conditions raisonnables de temps et de coût de traitement.

Position du problème

L'algorithme du simplexe est de complexité exponentiel (non polynomial) ce qui rend la résolution informatique des programmes linéaire de grandes échelles non supportée par le matériel informatique. L'obtention d'une solution est pourtant plus que nécessaire, c'est pour cela que l'on accepte des solutions approchées de l'optimum. C'est l'objet de plusieurs travaux de recherche algorithmique dans l'optimisation, d'où l'idée de proposer des algorithmes à solutions approchées pour ces problèmes complexes de programmation linéaire.

Ce mémoire consiste à résoudre un problème de programmation linéaire par une méthode d'optimisation relativement nouvelle, appelée programmation **DC**. La programmation **DC** et **DCA** (Algorithme **DC**) sont introduits par Pham Dinh Tao en 1986 à l'état préliminaire. Ces outils théoriques et algorithmiques sont intensivement développés à partir de 1993 par Le Thi Hoai An et Pham Dinh Tao pour devenir maintenant classiques et de plus en plus populaires.

La programmation **DC** fait partie du domaine de l'optimisation globale, elle traite de la minimisation d'une différence de deux fonctions convexes, étant capables de traiter des problèmes (différentiables ou non) de très grande dimension. Elle a été appliquée à plusieurs problèmes (résolution d'un jeu bi-matriciel, résolution de problème de Bin paking et résolution d'un problème de décision bi-niveau linéaire...[21][9][8]).
DCA semble être le seul algorithme disponible, à l'heure actuelle adopté pour l'optimisation non convexe, c'est une méthode de descente sans recherche linéaire basée sur les conditions d'optimalité locales et la dualité **DC**, on y travaille non pas avec la fonction **DC** f elle même mais avec les deux fonctions convexes g et h (composantes **DC** de f) telles que : $f = g - h$.

La forme standard d'un programme **DC** est donné par :

$$inf\{f(x) = g(x) - h(x)/x \in \mathbb{R}^n\}$$

où $g, h : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ sont convexes semi-continues inférieurement et propres.
La programmation **DC** est une extension de la Programmation Convexe : cette extension est assez large pour couvrir la quasi-totalité des programmes non convexes. **DCA** est une approche locale qui utilise les deux fonctions convexes g et h (dont la différence est la fonction objectif elle-même du programme **DC**) et non avec la fonction objectif f.
Puisqu'une fonction **DC** admet une infinité de décompositions **DC**, il y a une infinité de **DCA** appliqués a un programme **DC**, et les impacts de ces décompositions **DC** sur les qualités des **DCA** correspondants (rapidité, robustesse, globalité, ...) sont importants.

La résolution d'un problème concret par **DCA** devrait répondre aux deux questions cruciales :

- La recherche d'une bonne décomposition **DC** : cette question est largement ouverte. En pratique on cherche des décompositions **DC** bien adaptées a la structure des problèmes traités.
- Les techniques de reformulation sont souvent utilisées et très efficaces pour l'obtention des décompositions **DC** intéressantes.

Ce mémoire s'articule autour de trois chapitres principaux :
Le premier chapitre est consacré à la méthode **DC** dont nous exposerons la théorie.
Le deuxième chapitre, résume les notions de base de la programmation linéaire.
Le troisième chapitre est consacré à l'application de la méthode **DC** sur un problème de programmation linéaire.
Puis enfin une conclusion générale sur le travail effectué, et quelques perspectives.

CHAPITRE 1

Généralité sur la méthode DC

1.1 Introduction

La programmation **DC** est introduite par Pham Dinh Tao en 1986 (à l'état préliminaire). Cet outil théorique est intensivement développé à partir de 1993 par Le Thi Hoai An et Pham Dinh Tao pour devenir maintenant classique et de plus en plus populaire. Dans le domaine de la programmation non convexe, la programmation **DC** joue un rôle très important, grâce à son aspect théorique qu'encore ses applications pratiques assez larges. Une fonction est dite **DC** si elle peut être représentée comme la différence de deux fonctions convexes. Les problèmes de programmation Mathématique traitant avec des fonctions **DC** sont dits problèmes de programmation **DC**. On présentera dans ce qui suit les principaux résultats de cette théorie, ses applications, et la méthode de résolution dans le sens de l'optimisation globale. On se restreint aux approches déterministes et à la classe des programmes **DC** traitants avec les fonctions **DC**.

1.2 Éléments d'analyse convexe et quelques rappels élémentaires

Dans la suite, $\mathbb{R}^n$ est l'espace euclidien muni du produit scalaire usuel noté $< ., . >$ et de la norme euclidienne associée $|| \ x \ ||_2 = \sqrt{< x, x >}$. On notera $\overline{\mathbb{R}} = \mathbb{R} \cup \{\pm\infty\}$ l'espace muni de la structure algébrique déduite de celle de $\mathbb{R}$. On adoptera par la suite la convention : $(+\infty) - (+\infty) = +\infty$.

Dans cette section, nous allons rappeler quelques définitions et théorèmes d'analyse convexe qui fondent la base de la programmation **DC**.

1.2.1 Ensemble convexe

Définition 1.1
Un ensemble $C \in \mathbb{R}^n$ est convexe si $\forall a, b \in C$, $\alpha a + (1-\alpha)b \in C$, $\forall \alpha \in [0,1]$.
Ou encore : C est convexe si et seulement si le segment reliant tout couple de points de C est inclus dans C (voir la figure 1.1).

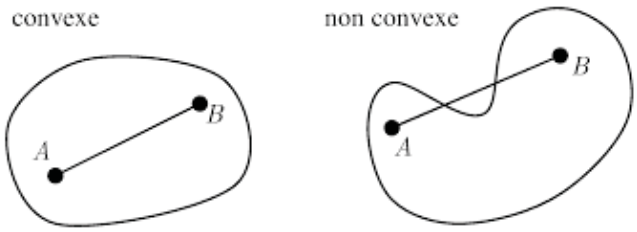


FIGURE 1.1 – *Ensemble convexe et ensemble non convexe*

Propriété 1.1 *[1]*
Soient S_1 et S_2 deux ensembles convexes de $\mathbb{R}^n$ alors :
$S_1 + S_2 = \{x \in \mathbb{R}^n / x = c + d, c \in S_1, d \in S_2\}$ est un ensemble convexe.

Propriété 1.2 *[1]*
Soit $S \subset \mathbb{R}^n$ convexe alors : $\alpha S = \{\alpha x / x \in S\}$ est convexe $\forall \alpha \in \mathbb{R}$.

Propriété 1.3 *[1]*
Soient $S_1, S_2, ..., S_p$ des ensembles convexes $\subset \mathbb{R}^n$ alors :

$$S = \bigcap_{i=1}^{p} S_i \text{ est un ensemble convexe.}$$

1.2.2 Combinaison convexe

Définition 1.2
On appelle une combinaison linéaire convexe de n-vecteurs $x_1, x_2, ..., x_n \in \mathbb{R}^n$, la somme :

$$\alpha_1 x_1 + \alpha_2 x_2 + ... + \alpha_n x_n, \sum_{i=1}^{n} \alpha_i = 1, \alpha_i \geq 0, i = \overline{1,n}.$$

Théorème 1.1 *[5]*
Un ensemble $S \subset \mathbb{R}^n$ convexe si et seulement si toute combinaison convexe de m points de S appartient à S.

1.2.3 Enveloppe convexe

Définition 1.3
Soit $A \subset \mathbb{R}^n (A \neq \varnothing)$, on appelle enveloppe convexe de A, le plus petit ensemble convexe contenant A, noté $Co(A)$, $A \subset Co(A)$ (voir la figure 1.2).

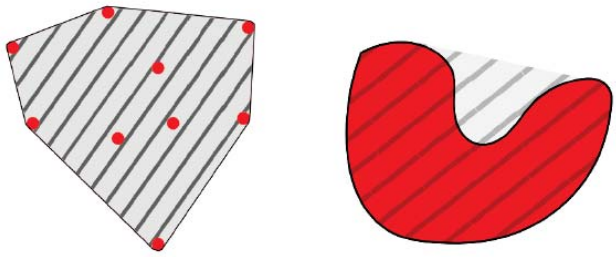


FIGURE 1.2 – *Enveloppe convexe d'un ensemble discret et d'un ensemble continu*

Théorème 1.2 *[5]*
L'enveloppe convexe d'un ensemble $S \subset \mathbb{R}^n$ est l'ensemble de toutes les combinaisons convexes de points de S.

1.2.4 Ensemble polyèdre convexe

Définition 1.4
Un sous-ensemble C de $\mathbb{R}^n$ est dit polyèdre convexe s'il est l'intersection d'un nombre fini de demi-espaces de $\mathbb{R}^n$.

1.2.5 Fonction convexe

Définition 1.5
Soit f une fonction définie sur un convexe $S \subset \mathbb{R}^n (f : S \longrightarrow \mathbb{R})$
f est dite convexe si : $\forall x, y \in S, \forall t \in [0,1]$:
$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y)$.
La fonction f est dite strictement convexe sur S si l'inégalité est stricte pour $x \neq y$.
Cette définition est aussi valable pour $f : S \longrightarrow \mathbb{R} \cup \{+\infty\})$.
Géométriquement, la fonction $f : S \longrightarrow \mathbb{R} \cup \{+\infty\})$ est convexe si le segment entre les points $(x, f(x))$ et $(y, f(y))$ se trouve au dessus du graphe de f (voir la figure 1.3).

Définition 1.6
Une fonction f est dite concave si $-f$ est convexe c'est à dire :
$f(tx + (1-t)y) \geq tf(x) + (1-t)f(y), \forall t \in [0,1]$.

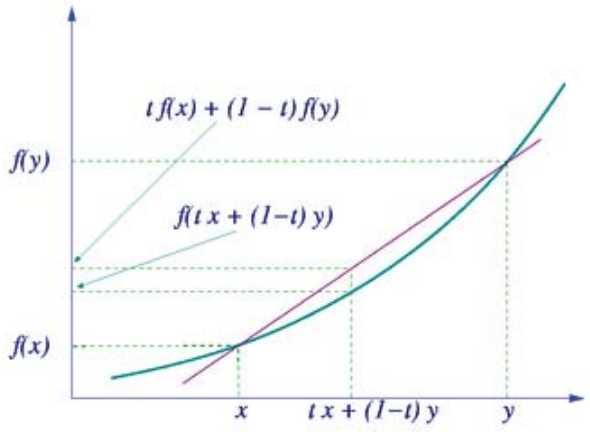


FIGURE 1.3 – *Fonction convexe*

Remarque 1.1
Les fonctions linéaires sont convexes et concaves au même temps.

Proposition 1.1 *[5]*
Soit S un ensemble convexe de $\mathbb{R}^n$ et soient f_1 et f_2 deux fonctions convexes sur S alors les fonctions suivantes sont aussi convexes :

- $f_1 + f_2$.
- $max\{f_1, f_2\}$.
- $\lambda f,\ \lambda \geq 0$.

1.2.6 Le domaine effectif d'une fonction

Définition 1.7
Le domaine effectif d'une fonction $f : C \subset \mathbb{R}^n \to \overline{\mathbb{R}}$, noté $dom(f)$, est défini par :

$$domf = \{x \in C / f(x) < +\infty\}.$$

1.2.7 Fonction propre

Définition 1.8
Une fonction $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ est dite propre si elle ne prend jamais la valeur $-\infty$ et n'est pas identiquement égale à $+\infty$.

1.2.8 Épigraphe d'une fonction

Définition 1.9
Soit $S \subset \mathbb{R}^n$ et soit $f : S \to \mathbb{R}$. On appelle un épigraphe de f noté $epi(f)$, le sous ensemble de $\mathbb{R}^{n+1}$ donné par :

$$\{(x, \alpha) / x \in S, \alpha \in \mathbb{R} : f(x) \leq \alpha\} \text{ (voir la figure 1.4)}.$$

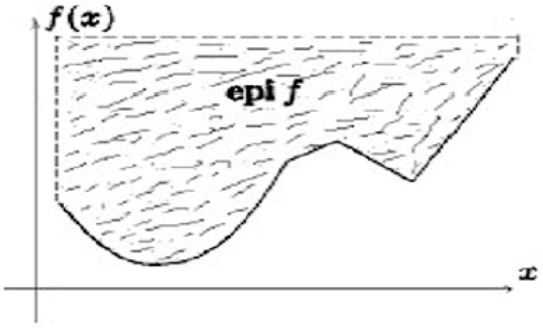


FIGURE 1.4 – *Épigraphe d'une fonction*

Remarque 1.2
La fonction f est convexe si et seulement si $epi(f)$ est un sous-ensemble convexe.
On dit que f est une fonction convexe propre, si son épigraphe est non vide.

1.2.9 Fonction semi-continue inférieurement

Définition 1.10
Soit $f : \mathbb{R}^n \to \mathbb{R}$ et $x_0 \in \mathbb{R}^n$, f est dite semi-continue inférieurement (s.c.i) en x_0 si et seulement si :

$$\liminf_{x \to 0} f(x) \geq f(x_0).$$

Proposition 1.2 *[3]*
Une fonction f est semi-continue inférieurement si et seulement si son épigraphe est fermé.

Notons par $\Gamma_0(\mathbb{R}^n)$ l'ensemble de toutes les fonctions semi-continues inférieurement, propres et convexes sur $\mathbb{R}^n$.

1.2.10 La fonction indicatrice

Définition 1.11
Soit C un sous ensemble de $\mathbb{R}^n$. La fonction indicatrice de C notée χ_C est définie par :

$$\chi_C(x) = \begin{cases} 0 & si\ x \in C, \\ +\infty & sinon. \end{cases}$$

Remarque 1.3
La fonction indicatrice χ_C est une fonction convexe si et seulement si C est un ensemble convexe.

1.2.11 Fonction affine

Définition 1.12
Une fonction est dite affine si elle est la somme d'une fonction linéaire et une constante.

1.2.12 Fonction conjuguée

Définition 1.13
Soit $f \in \Gamma_0(\mathbb{R}^n)$, La fonction conjuguée de f, notée f^*, est définie par :

$$f^*(y) = sup\{< x, y > -f(x)/x \in \mathbb{R}^n\}.$$

La fonction $f^{**} = (f^*)^*$ est appelée la bi-conjuguée de f, elle est donnée par :

$$f^{**}(x) = sup\{< x, y > -f^*(y)/y \in \mathbb{R}^n\}.$$

Propriété 1.4 *[2]*
- *f^* est toujours convexe.*
- *Si f prend la valeur $-\infty$ alors f^* est égale à $+\infty$.*
- *On a $f^{**}(x) \leq f(x), \forall x \in \mathbb{R}^n$. Si de plus $f \in \Gamma_0(\mathbb{R}^n)$, alors $f^{**} = f$.*
- *Soit χ_C la fonction indicatrice de l'ensemble C, sa conjuguée est donnée par :*

$$\chi_C^*(y) = \sup_{x \in C} < y, x > .$$

1.2.13 Fonction convexe polyédrale

Définition 1.14
Une fonction f est dite convexe polyédrale, si son épigraphe est un ensemble polyédrale convexe. En d'autres termes, f est convexe polyédrale si et seulement si elle peut être exprimée sous la forme :

$$f(x) = sup\{< a_i, x > -b_i, \overline{1, m}\} + \chi_C(x),$$

avec $a_i \in \mathbb{R}^n$, $b_i \in \mathbb{R}$ pour $\overline{1, m}$ et C est un sous ensemble non vide et convexe de $\mathbb{R}$ et χ_C est la fonction indicatrice de l'ensemble C.

Proposition 1.3 *[13]*
Les fonctions convexes polyédrales possèdent les propriétés suivantes :
- *Si f_1 et f_2 sont convexes polyédrales, alors $f_1 + f_2$, $max\{f_1, f_2\}$ sont convexes polyédrales.*
- *Si f est convexe polyédrale alors f^* l'est aussi.*

Remarque 1.4
Les fonctions affines et la fonction indicatrice d'un ensemble convexe, sont des fonctions convexes polyédrales.

1.3 Sous-différentiabilité

1.3.1 Sous-gradient

Définition 1.15

Soit f une fonction convexe et propre sur $\mathbb{R}^n$ et $x_0 \in dom(f)$. On appelle sous-gradient de f au point x_0 tout vecteur $y \in \mathbb{R}^n$ vérifiant :

$$f(x) \geq f(x_0) + < y, x - x_0 >, \forall x \in \mathbb{R}^n.$$

Cette relation possède une interprétation géométrique : elle signifie que le graphe de la fonction affine $\varphi(x) = f(x_0) + < y, x - x_0 >$ est un hyperplan de support non vertical à l'ensemble convexe $epi(f)$ au point $(x_0; f(x_0))$. Pour une fonction différentiable, cet hyperplan est vertical au point x_0 (voir les figures 1.5 et 1.6).

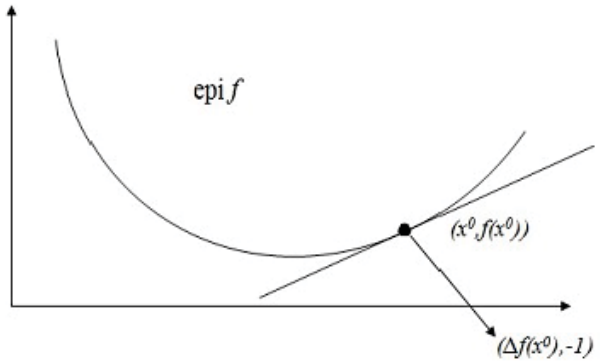


FIGURE 1.5 – *fonction différentiable*

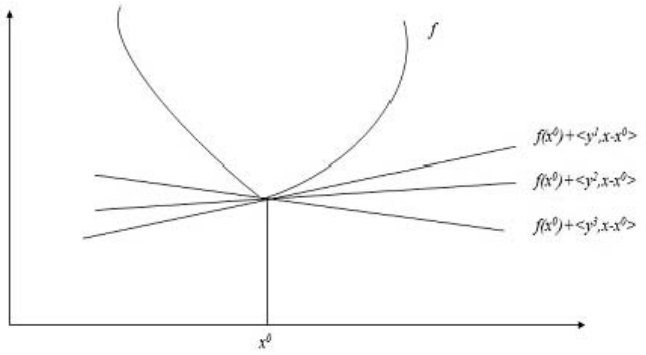


FIGURE 1.6 – *fonction non différentiable*

Les points y^1, y^2 *et* y^3 sont des sous-gradients de f au point x_0.

1.3.2 Sous-différentiel

Définition 1.16
L'ensemble de tous les sous-gradients de f en x_0 est appelé sous-différentiel de f au point x_0. On le note $\partial f(x_0)$.

$$\partial f(x_0) = \{y \in \mathbb{R}^n / f(x) \geq f(x_0) + < y, x - x_0 >, \forall x \in \mathbb{R}^n\}.$$

Le domaine du sous-différentiel de f, noté $dom(\partial f)$, est défini par :

$$dom(\partial f) = \{x \in \mathbb{R}^n / \partial f(x) \neq \varnothing\}.$$

Remarque 1.5
Pour une fonction convexe f sur $\mathbb{R}^n$ et $x \in dom(f)$, $\partial f(x)$ est un sous-ensemble convexe fermé de $\mathbb{R}^n$.

Propriété 1.5 *[13]*
Soit f une fonction convexe et propre et $x_0 \in int(dom\ f)$. Nous avons :
1. $f \in \Gamma_0(\mathbb{R}^n)$ est différentiable en x si et seulement si $\partial f(x)$ se réduit au singleton $\{\nabla f(x)\}$.
2. $y \in \partial f(x) \Leftrightarrow f(x) + f^*(y) =< x, y >$.
3. Si f est (s.c.i), alors $\partial f(x_0) \neq \varnothing$.
4. Si $f \in \Gamma_0(\mathbb{R}^n)$, alors $y \in \partial f(x) \Leftrightarrow x \in \partial f^*(y)$.
5. $x_0 \in argmin\{f(x)/x \in \mathbb{R}^n\} \Leftrightarrow 0 \in \partial f(x_0)$.

1.3.3 (ε-sous-gradient et ε-sous-différentiel)

Soit ε un réel strictement positif, f une fonction convexe sur $\mathbb{R}^n$ et $x_0 \in dom(f)$.
Un vecteur $y \in \mathbb{R}^n$ est appelé un ε-sous-gradient de f au point x_0 si :

$$f(x) \geq (f(x_0 - \varepsilon)) + < y, x - x_0 >, \forall x \in \mathbb{R}^n.$$

L'ensemble de tous les ε-sous-gradients de f en x_0 est appelé ε-sous-différentiel de f au point x_0. On le note $\partial_\varepsilon f(x_0)$.

1.3.4 Calcul du sous-différentiel

Soient $f, g : C \subset \mathbb{R}^n \to \overline{\mathbb{R}}$ et $\lambda > 0$. De la définition du sous-différentiel, nous avons les règles suivantes :

$$\partial(\lambda f(x)) = \lambda \partial f(x),$$

$$\partial f(x) + \partial g(x) \subset \partial(f + g)(x).$$

En général $\partial f(x) + \partial g(x) \neq \partial(f + g)(x)$. La proposition suivante nous donne une condition suffisante pour l'égalité.

Proposition 1.4 *[11]*
Soient $f, g : C \subset \mathbb{R}^n \to \overline{\mathbb{R}}$ deux fonctions convexes. S'il existe $x_0 \in \partial f(x) \cap \partial g(x)$ où f est continue, alors pour tout $x \in C$:

$$\partial f(x) + \partial g(x) = \partial(f + g)(x). \tag{1.1}$$

Remarque 1.6
Dans le cas où les fonctions f et g sont convexes et continues, l'égalité dans (1.1) reste toujours vraie [16].

1.4 Optimisation DC

1.4.1 Fonctions DC

Afin d'étendre la programmation convexe à la résolution de la plupart des problèmes d'optimisation non convexe de la vie courante tout en continuant à utiliser son arsenal théorique et numérique, une nouvelle classe de fonctions fut introduite, la classe des fonctions **DC**.

Soit C un ensemble convexe de $\mathbb{R}^n$. On note $Conv(C)$ l'ensemble des fonctions convexes sur C à valeur dans $\mathbb{R} \cup \{+\infty\}$.

Définition 1.17
Une fonction $f : C \subset \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ est dite ***DC*** *sur C, s'il existe deux fonctions $g, h : C \to \mathbb{R} \cup \{+\infty\}$ telles que :*

$$f(x) = g(x) - h(x), \forall x \in C, \tag{1.2}$$

où g et h sont des fonctions de $\Gamma_0(C)$.

Remarque 1.7
• *La forme (1.2) est dite une décomposition* ***DC*** *de f.*
• *Les fonctions g et h sont dites des composantes* ***DC*** *de f.*
• *Si $C = \mathbb{R}^n$, alors f est simplement dite fonction* ***DC*** *.*
• *Si f est une fonction* ***DC*** *sur un ensemble convexe C et si f admet une décomposition* ***DC*** *comme $f = g - h$ alors pour toute fonction φ convexe finie sur C, $(g + \varphi) - (h + \varphi)$ fournit aussi une décomposition* ***DC*** *de f. Ainsi, toute fonction* ***DC*** *admet une infinité de décompositions* ***DC****.*
• *On note* ***DC(C)*** *l'ensemble des fonctions* ***DC*** *sur C, c'est l'espace vectoriel engendré par $Conv(C)$.*

Proposition 1.5 *[7]*
Soient f et f_i, $i = \overline{1,m}$ des fonctions ***DC****, alors les fonctions suivantes sont aussi* ***DC*** *:*

- $\sum_{i=1}^{m} \lambda_i f_i(x),\ \lambda_i \in \mathbb{R},\ i = \overline{1,m}.$
- $\max_{i=1,\dots,m} f_i(x).$
- $\min_{i=1,\dots,m} f_i(x).$
- $\mid f(x) \mid.$
- $f^+ = max\{0, f(x)\}.$
- $f^- = min\{0, f(x)\}.$
- $\prod_{i=1}^{m} f_i(x).$

Proposition 1.6 *[2]*
- *Chaque fonction $f : \mathbb{R}^n \to \mathbb{R}$ dont les dérivées partielles sont continues est* ***DC****.*
- *Soit $f : \mathbb{R}^n \to \mathbb{R}$* ***DC*** *et $g : \mathbb{R} \to \mathbb{R}$ convexe alors la fonction $g \circ f$ est* ***DC****.*

Définition 1.18
Soit C un ouvert convexe. On dit qu'une fonction f est localement ***DC*** *sur C, si pour chaque x_0 de C, $\exists$ un $\varepsilon > 0$ tel que f est* ***DC*** *sur la boule :*

$$B(x_0, \varepsilon) = \{x \in \mathbb{R}^n : \| x - x_0 \| \leq \varepsilon\}.$$

On note par L_{C^2}, l'ensemble des fonctions de classe C^2 sur C.

Théorème 1.3 *[2]*
Soit C un ouvert convexe de $\mathbb{R}^n$. Toute fonction localement ***DC*** *sur C est* ***DC*** *sur C. En particulier toute fonction de classe C^2 sur C est* ***DC*** *sur C. On a la chaîne d'inclusions suivante :*

$$Conv(C) \subset L_{C^2} \subset DC(C).$$

1.4.2 Les problèmes de la programmation DC

Un programme DC est un problème d'optimisation de la forme :

$$inf\{f(x) = g(x) - h(x) / x \in \mathbb{R}^n\} \tag{1.3}$$

où g et h sont des fonctions appartenant à $\Gamma_0(\mathbb{R}^n)$.

Considérons les problèmes suivant :

$$sup\{f(x)/x \in C\} \tag{1.4}$$

où f est une fonction convexe sur C un ensemble convexe de $\mathbb{R}^n$.

$$inf\{g_0(x) - h_0(x)/x \in C, g_i(x) - f_i(x) \leq 0\} \tag{1.5}$$

où $g_i, h_i,\ i = \overline{0,m}$ sont des fonctions convexes sur C un ensemble convexe de $\mathbb{R}^n$.
On a alors :
• Le problème (1.4) est un cas particulier du problème (1.3) avec $g = \chi_C$ et $h = f$.
• Le problème (1.5) est appelé problème **DC** canonique généralisé et peut être transformé d'une façon équivalente au problème (1.3) en utilisant les méthodes de pénalités [8] [10].
• On observe que les problèmes d'optimisation convexes peuvent s'écrire comme des problèmes d'optimisation **DC** et être résolus en utilisant les techniques d'optimisation **DC**.
En effet,
le problème d'optimisation suivant :

$$inf\{f(x)/x \in \mathbb{R}^n\},\ avec\ f\ convexe$$

est équivalent au faux problème d'optimisation **DC** suivant :

$$inf\{(f+g)(x) - g(x) \in \mathbb{R}^n\}$$

où g est une fonction convexe.

1.4.3 Dualité en optimisation DC

Considérons le problème **DC** suivant :

$$(P_{DC}) : \alpha = inf\{f(x) = g(x) - h(x)/x \in \mathbb{R}^n\}$$

avec $g\ et\ h \in \Gamma_0(\mathbb{R}^n)$.
Le programme dual associé à (P_{DC}) est donné par :

$$(D_{DC}) : \alpha_D = inf\{h^*(y) - g^*(y)/y \in \mathbb{R}^n\}$$

En effet :
Puisque $h \in \Gamma_0(\mathbb{R}^n)$ donc on peut écrire :

$$h(x) = sup\{< x, y > -h^*(y)/y \in \mathbb{R}^n\},$$

Donc on aura :

$$\alpha = inf\{g(x) - sup\{< x, y > -h^*(y)/y \in \mathbb{R}^n\}/x \in \mathbb{R}^n\} = inf\{\alpha(y)/y \in \mathbb{R}^n\},$$

où

$$\alpha(y) = inf\{g(x) - [< x, y > -h^*(y)]/x \in \mathbb{R}^n\} \quad (P_y).$$

Il est clair que (P_y) est un programme convexe et

$$\alpha(y) = \begin{cases} h^*(y) - g^*(y) \ si \ y \in dom(h^*), \\ +\infty \quad sinon \end{cases}$$

Par suite

$$\alpha = inf\{h^*(y) - g^*(y)/y \in dom(h^*)\}$$

Finalement on obtient, avec la convention $(+\infty) - (+\infty) = +\infty$:

$$\alpha = \alpha_D = inf\{h^*(y) - g^*(y)/y \in \mathbb{R}^n\}.$$

Remarque 1.8
(D_{DC}) est aussi un programme DC car h^* et g^* sont convexes. De plus, (P_{DC}) et (D_{DC}) ont la même valeur optimale et on peut observer la parfaite symétrie entre ces deux problèmes, le dual de (D_{DC}) est exactement (P_{DC}).

Les résultats suivants donnent quelques propriétés concernant les solutions de (P_{DC}) et (D_{DC}).

Théorème 1.4 *[11]*
Soient $g, h \in \Gamma_0(\mathbb{R}^n)$.
(i) x^* est une solution optimale globale de (P_{DC}) si et seulement si

$$\alpha = (g - h)(x^*) \leq (h^* - g^*)(y), \ \forall y \in \mathbb{R}^n.$$

(ii) y^* est une solution optimale globale de (D_{DC}) si et seulement si

$$\alpha = (h^* - g^*)(y^*) \leq (g - h)(x), \ \forall x \in \mathbb{R}^n.$$

Théorème 1.5 *[11]*
Soient $g, h \in \Gamma_0(\mathbb{R}^n)$.
(i) $inf\{g(x) - h(x)/x \in dom(g)\} = inf\{h^*(y) - g^*(y)/y \in dom(h^*)\}$.
(ii) Si y_0 est un minimum de $h^* - g^*$ alors chaque $x_0 \in \partial g^*(y_0)$ est un minimum de $g - h$.
(iii) Si x_0 est un minimum de $g - h$ alors chaque $y_0 \in \partial h(x_0)$ est un minimum de $h^* - g^*$.

Ce dernier théorème montre que la résolution de l'un des deux problèmes (P_{DC}) et (D_{DC}) implique celle de l'autre.

1.4.4 Conditions d'optimalité en programmation DC

Il est bien connu en optimisation convexe que $x_0 \in \mathbb{R}^n$ minimise une fonction convexe $f \in \Gamma_0(\mathbb{R}^n)$ si et seulement si : $0 \in \partial f(x_0)$.
En programmation **DC**, la condition nécessaire et suffisante d'optimalité globale est formulée à l'aide des ε-sous-différentiels de g et h.

1.4.4.1 Condition d'optimalité globale

Théorème 1.6 *[2]*
Soient $g, h \in \Gamma_0(\mathbb{R}^n)$ et $x_0 \in \mathbb{R}^n$.
x_0 est un minimum globale du $g - h$ si et seulement si

$$\partial_\varepsilon h(x_0) \subset \partial_\varepsilon g(x_0), \forall \varepsilon > 0. \tag{1.6}$$

1.4.4.2 Condition nécessaire d'optimalité locale

Point critique

Définition 1.19
Un point $x^* \in \mathbb{R}^n$ est dit point critique ou point KKT généralisé de $g - h$ si

$$\partial g(x^*) \cap \partial h(x^*) \neq \varnothing.$$

Minimum local

Définition 1.20
Soient $g, h \in \Gamma_0(\mathbb{R}^n)$. Un point $x^* \in dom(g) \cap dom(h)$ est un minimum locale de $g - h$ s'il existe un voisinage $V(x^*)$ de x^* tel que :

$$(g - h)(x^*) \leq (g - h)(x), \forall x \in V(x^*).$$

Théorème 1.7 *[3]*
Si x^* est un minimum local de $g - h$ alors $\partial h(x^*) \subset \partial g(x^*)$. Cette condition est suffisante si h est polyédrale. De plus si f est localement convexe en x^*, en particulier si h est polyédrale et différentiable en x^*, alors x^* est une solution locale.

Corollaire 1.1 *[22]*
Si $h \in \Gamma_0(\mathbb{R}^n)$ est polyédrale alors une condition nécessaire et suffisante pour que x^* soit un minimum local de $g - h$ est :

$$\partial h(x^*) \subset int(\partial g(x^*)).$$

1.4.4.3 Condition suffisante d'optimalité locale

Théorème 1.8 *[3]*
Si $x^* \in dom(g) \cap dom(h)$ admet un voisinage V tel que :
$\partial h(x) \cap \partial g(x^*) \neq \varnothing, \forall x \in V \cap dom(g)$ alors x^* est un minimum local de $g - h$.

Théorème 1.9 *[4]*
Soit $x^* \in dom(\partial h)$ un minimum local de $g - h$ et $y^* \in \partial h(x^*)$. Supposons que $V(x^*)$ est un voisinage de x^* vérifiant $(g - h)(x) \geq (g - h)(x^*), \forall x \in V(x^*) \cap dom(g)$.
Si $x^* \in int(dom(h))$, $y^* \in int(dom(g^*))$ et $\partial g^*(y^*) \subset V(x^*)$ alors y^* est un minimum local de $h^* - g^*$.

1.5 Algorithme d'optimisation DC (DCA)

1.5.1 Construction de l'algorithme

DCA (DC Algorithm) est une méthode itérative d'optimisation locale basée sur l'optimalité locale et la dualité en programmation **DC**. Il existe deux formes de **DCA** : la forme complète et la forme simplifiée. En pratique, l'utilisation de la forme complète de **DCA** est une tâche difficile et coûteuse. Elle est donc remplacée par la forme simplifiée qu'on utilisera dans la suite. Cette approche algorithmique permet de construire deux suites x^k et y^k (candidats supposés pour des solutions optimales des programmes **DC** primal et dual respectivement), telles que les suites $(g-h)(x^k)$ et $(h^*-g^*)(y^k)$ soient décroissantes et tendent vers la même limite ou bien solution $\beta = (g-h)(x^*) = (h^*-g^*)(y^*)$. Ces suites sont améliorées à chaque itération de façon à vérifier les conditions suivantes :
1) Les suites $g(x^k) - h(x^k)$ et $h^*(y^k) - g^*(y^k)$ décroissent et tendent vers la même limite β qui est supérieure ou égale à la valeur optimale globale α.
2) Si $(g - h)(x^{k+1}) = (g - h)(x^k)$ l'algorithme s'arrête à l'itération $k + 1$, et le point x^k (resp. y^k) est un point critique de $g - h$ (resp. $h^* - g^*$).
3)Si la valeur optimale du problème (P_{DC}) est finie et si les suites x^k et y^k sont bornées, alors toute valeur d'adhérence x^* de la suite x^k (resp. y^* de la suite y^k) est un point critique de $g - h$ (resp. $h^* - g^*$).

Construction des suites x^k et y^k :

Pour construire les deux suites x^k et y^k, on définit deux programmes convexes (P_k) et (D_k) comme :

(D_k) $\quad y^k \in arg\ min\{h^*(y) - [g^*(y^{k-1}) + < y - y^{k-1}, x^k >] : y \in \mathbb{R}^n\}.$
(P_k) $\quad x^{k+1} \in arg\ min\{g(x) - [h(x^k) + < x - x^k, y^k >] : x \in \mathbb{R}^n\}.$

On peut facilement comprendre que (P_k) (resp. (D_k)) est obtenu en remplaçant h (resp.g^*) de (P_{DC}) (resp.(D_{DC})) par sa minorante affine $h_k(x) = h(x^k) + < x - x^k, y^k >$ au voisinage de x^k avec $y^k \in \partial h(x^k)$ (resp. $g_k^*(y) = g_k^*(y^{k-1}) + < y - y^{k-1}, x^k >$ au voisinage de y^{k-1} avec $x^k \in \partial g^*(y^{k-1})$).

On a le schéma simple suivant pour décrire **DCA** :

$$x^k \longrightarrow y^k \in \partial h(x^k)$$
$$\swarrow$$
$$x^k \in \partial g^*(y^{k-1}) \longrightarrow y^{k+1} \in \partial h(x^{k+1})$$

DCA est donc un schéma de point fixe $x^{k+1} \in (\partial g^* \circ \partial h)(x^k)$.
L'interprétation de **DCA** est simple : à chaque itération k, la seconde composante du problème primal (P_{DC}) (resp. problème dual (D_{DC})) est remplacée par sa minorante affine $h_k(x) = h(x^k) + < x - x^k, y^k >$ au voisinage de x^k avec $y^k \in \partial h(x^k)$
(resp. $g_k^*(y) = g_k^*(y^{k-1}) + < y - y^{k-1}, x^k >$ au voisinage de y^{k-1} avec $x^k \in \partial g^*(y^{k-1})$).
D'après les conditions d'optimalité, **DCA** s'arrête si au moins l'une des suites $(g-h)(x^k)$, $(h^* - g^*)(y^k)$, x^k, y^k converge. En pratique, nous utilisons souvent les conditions d'arrêt suivantes :
- $| (g-h)(x^{k+1}) - (g-h)(x^k) | \leq \varepsilon$.
- $|| x^{k+1} - x^k || \leq \varepsilon$.

1.5.2 Algorithme DCA simplifié

Algorithme DCA simplifié

Étape 0 : x^0 donné, $k = 0$.
Étape 1 : On calcule $y^k \in \partial h(x^k)$.
Étape 2 : On détermine $x^{k+1} \in \partial g^*(y^k)$.
Étape 3 : Si les conditions d'arrêt sont vérifiées alors on termine DCA ; Sinon $k = k + 1$ et on répète l'Étape 1.

FIGURE 1.7 – *Algorithme* ***DCA***

1.5.3 Convergence de DCA

Pour un programme **DC**, si **DCA** peut effectivement construire les deux suites $\{x^k\}$ et $\{y^k\}$ à partir d'un point initial arbitraire $x_0 \in \mathbb{R}^n$, alors les deux suites sont dites bien définies.

Lemme 1.1 *(Existence des suites [6])*
Les propositions suivantes sont équivalentes :
1. Les suites $\{x^k\}$ et $\{y^k\}$ sont bien définies.
2. $dom(\partial g) \subset dom(\partial h)$ et $dom(\partial h^*) \subset dom(\partial g^*)$.

Le lemme suivant établit les conditions de bornitude pour les suites générées par **DCA**.

Lemme 1.2 *(bornitude des suites [6])*
Si $g - h$ est coercive (i.e. $\lim_{x\to\infty}(g-h)(x) = +\infty$), alors
1. la suite $\{x^k\}$ est bornée.
2. si $\{x^k\} \subset int(dom(h))$ alors la suite $\{y^k\}$ est aussi bornée.
Par dualité, si $h^* - g^*$ est coercive alors
1. la suite $\{y^k\}$ est bornée.
2. si $\{y^k\} \subset int(dom(g^*))$ alors la suite $\{x^k\}$ est aussi bornée.

La convergence de l'algorithme est assurée par les résultats suivants :

Théorème 1.10 *(Convergence de* ***DCA****[2])*
Supposons que les suites $\{x^k\}$ et $\{y^k\}$ sont bien définies. Alors ***DCA*** *est une méthode de descente sans recherche linéaire mais avec une convergence globale et possède les caractéristiques suivantes :*
1. Les suites $\{(g-h)(x^k)\}$ et $\{(h^*-g^*)(y^k)\}$ sont décroissantes et
• *$(g-h)(x^{k+1}) = (g-h)(x^k)$ si et seulement si*

$$\begin{cases} y^k \in \partial g(x^k) \cap \partial h(x^k), \\ y^k \in \partial g(x^{k+1}) \cap \partial h(x^{k+1}), \end{cases}$$

De plus, si g et h sont strictement convexes sur $\mathbb{R}^n$ alors $x^{k+1} = x^k$. Dans ce cas, ***DCA*** *se termine à la $(k+1)^{eme}$ itération (convergence finie de* ***DCA****).*
• *$(h^*-g^*)(y^{k+1}) = (h^*-g^*)(y^k)$ si et seulement si*

$$\begin{cases} x^{k+1} \in \partial h^*(y^k) \cap \partial g^*(y^k), \\ x^{k+1} \in \partial h^*(y^{k+1}) \cap \partial g^*(y^{k+1}), \end{cases}$$

De plus, si h^* et g^* sont strictement convexes sur $\mathbb{R}^n$ alors $y^{k+1} = y^k$. Dans ce cas, ***DCA*** *se termine à la $(k+1)^{eme}$ itération (convergence finie de* ***DCA****).*

2. Si la valeur optimale du problème primal (P_{DC}) est finie alors
• *Les suites décroissantes $\{(g-h)(x^k)\}$ et $\{(h^*-g^*)(y^k)\}$ convergent vers la même limite $\beta \geq \alpha$.*
• *Si de plus, les suites $\{x^k\}$ et $\{y^k\}$ sont bornées alors pour toute valeur d'adhérence $\tilde{x}$ de $\{x^k\}$ il existe une valeur d'adhérence $\tilde{y}$ de $\{y^k\}$ telle que*

$$\tilde{y} \in \partial g(\tilde{x}) \cap \partial h(\tilde{x}) \text{ et } g(\tilde{x}) - h(\tilde{x}) = \beta.$$

De même pour la suite $\{y^k\}$.
3. ***DCA*** *a une convergence linéaire pour un programme* ***DC*** *dans le cas général. Dans le cas polyédral, cette convergence est finie.*

Remarque 1.9
• *Il est clair que* ***DCA*** *s'applique aux fonctions convexes g et h, et non à la fonction f elle-même. On voit ainsi comment le mécanisme de* ***DCA*** *fonctionne pour les programmes DC non différentiables (f est une fonction DC non différentiable). Et puisqu'une fonction DC admet une infinité de décompositions* ***DC****, il en aura autant de* ***DCA****. Le choix d'une décomposition* ***DC*** *appropriée est crucial car il conditionne les qualités essentielles (rapidité, robustesse, globalité des solutions calculées) du* ***DCA*** *résultant. Théoriquement, le problème de décomposition* ***DC*** *optimale reste à définir et à étudier. En pratique on cherche des décompositions* ***DC*** *bien adaptées à la structure spécifique du problème traitée afin que les deux suites $\{x^k\}$ et $\{y^k\}$ soient obtenues à moindre coût en temps de calcul, si elles ne sont pas explicites.*
Ici, peut être plus qu'ailleurs, les techniques de reformulation sont omniprésentes. Il est important de noter qu'avec des décompositions ***DC*** *appropriées,* ***DCA*** *permet de retrouver, comme cas particuliers, la plupart des méthodes standard en programmation convexe/non convexe. D'autre part un programme convexe est un programme* ***DC*** *pour lequel* ***DCA*** *converge vers une solution (locale qui est aussi globale) : de cette manière* ***DCA*** *permet de construire une infinité d'algorithmes pour la programmation convexe, qui pourraient être plus performants que les méthodes existantes.*

• *Il est important de noter qu'avec des décompositions* ***DC*** *appropriées,* ***DCA*** *permet de retrouver, comme cas particuliers, la plupart des méthodes standards en programmation convexe/non convexe. A notre connaissance,* ***DCA*** *est actuellement parmi les rares algorithmes de la programmation non convexe, capables de traiter des problèmes de très grande taille et reste le seul algorithme performant pour des programmes non convexes non différentiables.*

1.6 Programmation DC polyédrale

Dans cette partie, on présente une classe de problèmes **DC** qui se rencontre fréquemment dans la pratique et possède des propriétés importantes, tant sur le plan théorique que sur le plan algorithmique. Cette classe de problèmes **DC** s'appelle **DC** polyédrale.

Définition 1.21
*Un programme **DC** est dit polyédral si l'une des composantes convexes (g et h) de la fonction $f = g - h$ est une fonction convexe polyédrale.*

Il a été montré que, pour résoudre un programme **DC** polyédrale, **DCA** a une convergence finie. (voir[12][17][20]).

1.7 Conclusion

Ce chapitre résume les notions essentielles de la programmation **DC** utiles dans la suite de notre mémoire. Le chapitre suivant discutera les notions fondamentales de la programmation linéaire nécessaires à la suite de notre travail.

CHAPITRE 2

Rappels sur la programmation linéaire

2.1 Introduction

La programmation linéaire est une technique qui permet de résoudre un grand nombre de problème de gestion. Cette technique consiste à modéliser le problème posé sous forme d'un modèle linéaire, ensuite, elle assure la résolution du problème. Enfin, l'interprétation des résultats obtenus et ainsi que l'analyse de sensibilité qui demeure nécessaire dans un environnement turbulent, constituent les sources d'information pour le décideur de l'entreprise.

En effet, la modélisation linéaire est généralement liée à des contraintes d'allocations des ressources limitées et à la meilleure façon possible d'exploiter, afin de maximiser le profit ou de minimiser les coûts de revient. Donc, le problème que l'on se pose est de réaliser les objectifs d'une entreprise sujette à des restrictions des ressources.

La programmation linéaire a été introduite par le russe Kontrovitch et la première résolution a été faite par l'américain G.B. Dantzig en 1951.

Dans ce chapitre, on rappellera les notions élémentaires de la programmation linéaire et les outils de résolution d'un programme linéaire.

2.2 Formulation d'un problème d'optimisation

La formulation d'un problème d'optimisation comporte toujours trois étapes :
• choix des variables du modèle,
• formulation de l'objectif,
• formulation des contraintes.

2.3 Formulation d'un problème de programmation linéaire sous forme standard

Tout problème de programmation linéaire peut se former de la manière suivante : Trouver les valeurs des variables $x = (x_j, j = 1, ..., n)$ qui maximisent ou minimisent la fonction linéaire suivante :

$$Z = Z(x_1, x_2, ..., x_n) = c_1x_1 + c_2x_2 + ... + c_jx_j + ... + c_nx_n = \sum_{j=1}^{n} c_jx_j \longrightarrow max(min), \quad (2.1)$$

Sous les contraintes suivantes :

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + ... + a_{1j}x_j + ... + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + ... + a_{2j}x_j + ... + a_{2n}x_n = b_2 \\ \vdots \quad\quad \vdots \quad\quad\quad\quad \vdots \quad\quad\quad\quad \vdots \quad\quad \vdots \\ a_{i1}x_1 + a_{i2}x_2 + ... + a_{ij}x_j + ... + a_{in}x_n = b_i \\ \vdots \quad\quad \vdots \quad\quad\quad\quad \vdots \quad\quad\quad\quad \vdots \quad\quad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + ... + a_{mj}x_j + ... + a_{mn}x_n = b_m \end{cases} \quad (2.2)$$

$$x_j \geq 0, j = 1, ..., n. \quad (2.3)$$

où $c_j, j = 1, ..., n$, représentent les coûts des différents produits.
Les coefficients c_j et $a_{ij}(i = 1, ..., m, j = 1, ..., n)$ sont supposés être des nombres réels, en plus on considère que l'entier m est inférieur ou égal à n, tous les nombres $b_i(i = 1, ..., m)$ sont tous positifs ou nuls et le rang du système est inférieur ou égal à m.

Définition 2.1
La fonction Z (2.1) est appelée fonction objectif, fonction de but, fonction économique ou critère de qualité.
Les contraintes (2.2) sont appelées contraintes principales, ou essentielles.
les contraintes (2.3) sont dites directes.

Les problèmes de programmation linéaire peuvent naturellement se présenter sous une forme différente de celle du programme (2.1)-(2.3) qui est dite standard. Cependant de tels cas peuvent toujours se ramener à la forme (2.1)-(2.3), en se refrant au paragraphe suivant.

2.4 Réduction d'un problème de programmation linéaire à la forme standard

— Si l'objectif consiste à minimiser une fonction linéaire :
$Z = Z(x_1, x_2, ..., x_n) = c_1x_1 + c_2x_2 + ... + c_jx_j + ... + c_nx_n \longrightarrow min.$
Alors on maximisera la fonction linéaire opposée :
$\overline{Z} = -Z(x_1, x_2, ..., x_n) = -c_1x_1 - c_2x_2 - ... - c_jx_j - ... - c_nx_n \longrightarrow max.$
Car min Z=-max $\overline{Z}$ =max(-Z).

— Si une certaine variable x_j n'est soumise à aucune condition de signe ($x_j \in \mathbb{R}$) , on la remplacera alors par deux variables x_j^1 et x_j^2 telles que :
$x_j = x_j^1 - x_j^2,\ x_j^1 \geq 0,\ x_j^2 \geq 0.$

— Lorsque dans une certaine équation on a un $b_i < 0$, alors il suffit de multiplier les deux membres de cette équation par (-1) pour avoir le second membre positif.

— Supposons qu'on a une contrainte principale sous forme d'inégalité :

$$\alpha_1x_1 + \alpha_2x_2 + ... + \alpha_jx_j + ... + \alpha_nx_n \leq \beta, \tag{2.4}$$

De là en ajoutant une variable x_{n+1} positive dite variable d'écart, au premier membre de l'inégalité, on obtient une équation équivalente :
$\alpha_1x_1 + \alpha_2x_2 + ... + \alpha_jx_j + ... + \alpha_nx_n + x_{n+1} = \beta$

Dans ce cas où l'inégalité (2.4) est dans le sens inverse, c'est à dire :
$\alpha_1x_1 + \alpha_2x_2 + ... + \alpha_jx_j + ... + \alpha_nx_n \geq \beta$
L'équation équivalente d'écrira alors sous la forme suivante :
$\alpha_1x_1 + \alpha_2x_2 + ... + \alpha_jx_j + ... + \alpha_nx_n - x_{n+1} = \beta,$
où $x_{n+1} \geq 0.$

2.5 Écriture matricielle d'un problème de programmation linéaire

Pour écrire le problème de la programmation linéaire (2.1)-(2.3) sous forme compacte, on utilise la forme matricielle (vectorielle). Pour ce faire on introduit les notions suivantes :
Soient $I = \{1, 2, ..., i, ..., m\}$ l'ensemble des indices des lignes et $J = \{1, ..., j, ..., n\}$ l'ensembles des indices des colonnes.
Ainsi l'ensembles des variables $x_1, x_2, ..., x_n$ s'écrira sous forme vectorielle :
$x = x(J) = (x_j, j \in J).$

De manière analogue on aura :
$c = c(J) = (c_j, j \in J), b = (b_i, i \in I)$.
L'ensemble des coefficients $a_{ij}, i \in I, j \in J$ sera représenté sous forme d'une matrice A d'ordre $(m \times n)$:

$$A = (I, J) = (a_{ij}, i \in I, j \in J) = \begin{pmatrix} a_{11} & a_{12} & ... & a_{1j} & ... & a_{1n} \\ a_{21} & a_{22} & ... & a_{2j} & ... & a_{2n} \\ \vdots & \vdots & & \vdots & & \vdots \\ a_{i1} & a_{i2} & ... & a_{ij} & ... & a_{in} \\ \vdots & \vdots & & \vdots & & \vdots \\ a_{m1} & a_{m2} & ... & a_{mj} & ... & a_{mn} \end{pmatrix}$$

On écrit souvent A de la manière suivante :
$A = (a_1, a_2, ..., a_j, ..., a_n)$, où a_j le vecteur colonne :

$$a_j = A(I, J) = \begin{pmatrix} a_{ij} \\ a_{2j} \\ \vdots \\ a_{ij} \\ \vdots \\ a_{mj} \end{pmatrix}$$

Dans ce qui suit on note par vecteur $x \geq 0$ c'est à dire, toute les composantes de ce vecteur sont positives.
Avec ces nouvelles notions le problème (2.1)-(2.3) peut être écrit sous la forme matricielle suivante :

$$\begin{cases} Z = Z(x) = c^{'} x \longrightarrow max \\ Ax = b \\ x \geq 0 \end{cases}$$

Ici $c^{'}$ est le transposé de c, la matrice A est en général la matrice de condition du problème.

Remarque 2.1

La forme $\begin{cases} Z = Z(x) = c^{'} x \longrightarrow max \\ Ax \leq b \\ x \geq 0 \end{cases}$ *est dite la forme canonique.*

2.6 Hyperplan et demi-espace

Définition 2.2

Soit $a = \{a_1, a_2, ..., a_n\} \in \mathbb{R}^n$ *et* $\beta \in \mathbb{R}$ *alors l'ensemble*
$H = \{x \in \mathbb{R}^n / ax = \beta\}$ *est un Hyperplan de* $\mathbb{R}^n$.

Propriété 2.1 *[1]*
Un Hyperplan est un ensemble convexe.

Définition 2.3
Soit $a = (a_1, a_2, ..., a_n) \in \mathbb{R}^n$ *et* $\beta \in \mathbb{R}$ *on appelle :*
Demi-espace fermé, l'ensemble $\{x \in \mathbb{R}^n / ax \leq \beta$ *ou* $ax \geq \beta\}$.
Et on appelle demi-espace ouvert, l'ensemble $\{x \in \mathbb{R}^n / ax < \beta$ *ou* $ax > \beta\}$.
(voir la figure 2.1).

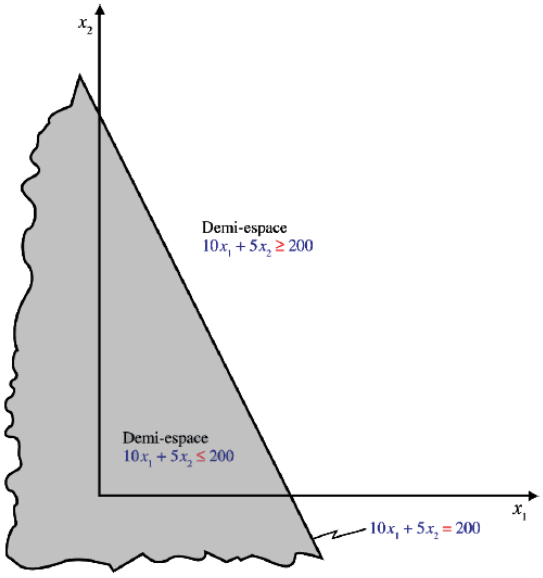


FIGURE 2.1 – *Demi-espace*

2.7 Polytope et polyèdre

Définition 2.4
L'intersection d'un nombre fini de demi-espace fermé de $\mathbb{R}^n$ *est appelée Polytope convexe.*

Définition 2.5
On appelle polyèdre un polytope convexe borné.(c'est un compact convexe de $\mathbb{R}^n$*).*

Remarque 2.2
Un ensemble est dit compact, s'il est fermé et borné.

Remarque 2.3
Soit le programme linéaire suivant :

$$(PL)\begin{cases} MaxZ = c_1x_1 + c_2x_2 + ... + c_jx_j + ...c_nx_n \\ sous\ les\ contraintes : \\ \begin{array}{ccccccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & ... & +a_{1j}x_j & + & ... & + & a_{1n}x_n & \leq & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & ... & +a_{2j}x_j & + & ... & + & a_{2n}x_n & \leq & b_2 \\ \vdots & & \vdots & & & \vdots & & & & \vdots & & \vdots \\ a_{i1}x_1 & + & a_{i2}x_2 & + & ... & +a_{ij}x_j & + & ... & + & a_{in}x_n & \leq & b_i \\ \vdots & & \vdots & & & \vdots & & & & \vdots & & \vdots \\ a_{m1}x_1 & + & a_{m2}x_2 & + & ... & +a_{mj}x_j & + & ... & + & a_{mn}x_n & \leq & b_m \end{array} \\ x_j \geq 0, j = \overline{1,n}, i = \overline{1,m}. \end{cases}$$

Les contraintes de (PL) définissent un polytope convexe et chaque contrainte définit un demi-espace fermé.

2.8 Points extrêmes et leurs caractérisation

Définition 2.6
Soit C un ensemble convexe de $\mathbb{R}^n$ et soit x_0 un élément de C.
x_0 est appelé point extrême de C s'il n'existe pas deux points x_1 et x_2 tels que :
$x_0 = \alpha x_1 + (1-\alpha)x_2 \in C, \forall \alpha \in [0,1]$.

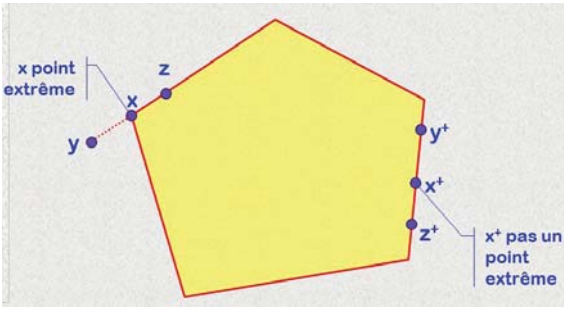


FIGURE 2.2 – *Point extrême*

Caractérisation des points extrêmes

Considérons le problème standard suivant :

$$(P)\begin{cases} Z = c^{'}x \longrightarrow max \\ Ax = b \\ x \geq 0 \end{cases}$$

Désignons par $X = \{x \in \mathbb{R}^n / Ax = b, x \geq 0\}$ l'ensemble des solutions admissibles (réalisables) du problème (P).

Théorème 2.1 *[1]*
L'ensemble X est convexe fermé.

Théorème 2.2 *[1]*
Soit $x = (x_1, x_2, ..., x_n)$ un point extrême, alors les vecteurs colonnes de A correspondants aux composantes positives de X sont linéairement indépendants.

Théorème 2.3 *[1]*
Si le système de vecteur $a_1, a_2, ..., a_n$ (les colonnes de A) possède
m-vecteurs linéairement indépendants (par exemple $a_1, a_2, ..., a_m$) alors la solution réalisable $x = (x_1, x_2, ..., x_m, 0, ..., 0)$ est un point extrême.

Corollaire 2.1 *[1]*
Le nombre de points extrêmes est fini.

Théorème 2.4 *[1]*
Si la fonctionnelle atteint son maximum(minimum) sur un point quelconque de X, alors ce point est extrême.
Si le maximum(minimum) est atteint sur deux points extrêmes x_1 et x_2 alors tous les points du segment $[x_1, x_2]$ sont des solutions optimales.

Corollaire 2.2 *[1]*
L'une des propriétés, les plus importantes des problèmes de programmation linéaire est que le maximum(minimum) d'une fonction linéaire est atteint toujours sur un ou plusieurs points extrêmes.

2.9 Résolution d'un programme linéaire avec la méthode du simplexe

Soit le programme linéaire suivant :

$$c_1x_1 + c_2x_2 + ... + c_nx_n \longrightarrow max \tag{2.5}$$

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + ... + a_{1j}x_j + ... + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + ... + a_{2j}x_j + ... + a_{2n}x_n = b_2 \\ \vdots \quad\quad \vdots \quad\quad\quad\quad \vdots \quad\quad\quad\quad \vdots \quad\quad \vdots \\ a_{i1}x_1 + a_{i2}x_2 + ... + a_{ij}x_j + ... + a_{in}x_n = b_i \\ \vdots \quad\quad \vdots \quad\quad\quad\quad \vdots \quad\quad\quad\quad \vdots \quad\quad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + ... + a_{mj}x_j + ... + a_{mn}x_n = b_m \end{cases} \tag{2.6}$$

$$x_j \geq 0, j = \overline{1,n} \tag{2.7}$$

Ici on suppose $b_i \geq 0, i = \overline{1,m}$ et rang $A = m \leq n$.

La méthode du simplexe est une méthode itérative. Elle démarre d'un point extrême (sommet de départ) et passe au sommet voisin, et ceci constitue une itération de l'algorithme du simplexe. Pour cela, on définit le point extrême de départ et le test d'arrêt.

2.9.1 Solution de base

Définition 2.7
Tout vecteur x vérifiant les contraintes (2.6) et (2.7) est appelé réalisable (admissible) du problème (2.5)-(2.7).

Définition 2.8
Une solution réalisable x^* est optimale si $c^{'}x^* = max(c^{'}x)$, pour toute solution réalisable x.

Définition 2.9
Une solution réalisable x est dite de base si $(n-m)$ de ses composantes sont nulles et aux autres $x_{j1}, x_{j2}, ..., x_{jm}$, correspondants m vecteurs $a_{j1}, a_{j2}, ..., a_{jm}$ de la matrice de condition A linéairement indépendants.
L'ensemble $j_B = \{j_1, j_2, ..., j_m\}$ est appelé ensemble des indices de base, $J_H = J \setminus J_B$ ensemble des indice hors base.
Autrement, une solution réalisable $x = x(J)$ est solution de base si $x_H = x(J_H) = 0$, $det\ A_B \neq 0$, où $A_B = A(I, J_B)$.
La matrice A_B est appelée matrice de base ; $x_j, j \in J_B$ les composantes de base ; $x_j, j \in J_H$ les composantes hors base.

Définition 2.10
Une solution de base x est dite non-dégénérée si $x_j > 0, j \in J_B$.

2.9.2 Formule d'accroissement de la fonction objectif

Soit $x(x_1, x_2, ..., x_n)$ une solution réalisable de base avec la matrice de base $A_B = A(I, J_B), J_H = J \setminus J_B$.
Effectuons la partition suivante :
$A = [A_B \mid A_H], A_H = A(I, J_H), x = [\frac{x_B}{x_H}], x_B = x(J_B), x_H = x(J_H), c = [\frac{c_B}{c_H}], c_B = c(J_B)$, $c_H = c(J_H)$.
Considérons une solution réalisable quelconque $\overline{x} = x + \delta x$.
L'accroissement de la fonction objectif Z est donc égal à :

$$\Delta Z = \overline{Z}(x) - Z(x) = c^{'}\overline{x} - c^{'}x = c^{'}\Delta x. \tag{2.8}$$

Comme x et $\overline{x}$ sont réalisables alors : $A\overline{x} = Ax = b \Longrightarrow A(\overline{x} - x) = A\Delta x = 0$.
Comme $\Delta x = [\dfrac{\Delta x_B}{\Delta x_H}], d'ou\ A\Delta x = A_B\Delta x_B + A_H\Delta x_H = 0 \Longrightarrow \Delta x_B = -A_B^{-1}A_H\Delta x_H$
et en vertu de la relation (2.8) on obtient :
$\Delta Z = c_B^{'}\Delta x_B + c_H^{'}\Delta x_H = c_B^{'}(-A_B^{-1}A_H\Delta x_H) + c_H^{'}\Delta x_H$
$\Longrightarrow \Delta Z = -(c_B^{'}A_B^{-1}A_H - c_H^{'})\Delta x_H$
Construisons le m-vecteur $y = y(I)$ dit des potentiels :

$$y^{'} = c_B^{'}A_B^{-1}. \tag{2.9}$$

Et le vecteur $\Delta = \triangle(J) = (\Delta_j,\ j \in J)$ dit des estimations :

$$\begin{cases} \Delta^{'} = y^{'}A - c^{'} \\ \Delta_j = y^{'}a_j - c_j,\ j \in J \end{cases} \tag{2.10}$$

Remarque 2.4
$\Delta_B^{'} = 0$ *par construction.*
En utilisant (2.9) et (2.10), l'accroissement de la fonctionnelle prend la forme suivante :

$$\Delta Z = -\Delta_H^{'}\Delta x_H = -\sum_{j \in J_H} \Delta_j \Delta x_j \tag{2.11}$$

Comme $\overline{x}_j \geq 0, \forall j et x_j = 0,\ \forall j \in J_H$ *donc :*
$\overline{x}_j = x_j + \Delta x_j = \triangle x_j \geq 0,\ j \in J_H$.
En utilisant cette dernière inégalité et la relation (2.11) on déduit le critère d'optimalité.

2.9.3 Critère d'optimalité

Théorème 2.5 *[1]*
Soit $\{x, A_B\}$ *une solution réalisable de départ.*
L'inégalité $\Delta_H = \Delta(J_H) \geq 0$ *est suffisante et dans le cas de la non dégénérescence elle est nécessaire pour l'optimalité de* $\{x, A_B\}$.

2.9.4 Itération de l'algorithme du simplexe

Soit $\{x, A_B\}$ une solution réalisable de base de départ et supposons le critère d'optimalité n'est pas vérifié, c'est à dire l'inégalité $\Delta_j \geq 0,\ j \in J_H$ n'est pas vérifiée.
Choisissons l'indice $j_0 \in J_H / \Delta_{j_0} = \min\limits_{\Delta_j < 0,\ j \in J} \Delta_j$.
Le but de l'itération est de faire rentrer cet indice j_0 dans la base (autrement dit la colonne a_{j_0} va renter dans la base).
Donc il faut trouver un indice $j_1 \in J_B$, qui sortira de la base(à cet indice correspond la colonne $a_{j_1} \in A_B$). Et ceci constitue l'itération, qui procède au passage de la solution de base (pont extrême) $\{x, A_B\}$ à la solution $\{\overline{x}, \overline{A}_B\}$ (sommet voisin) et tel que $Z(\overline{x}) \geq Z(x)$.

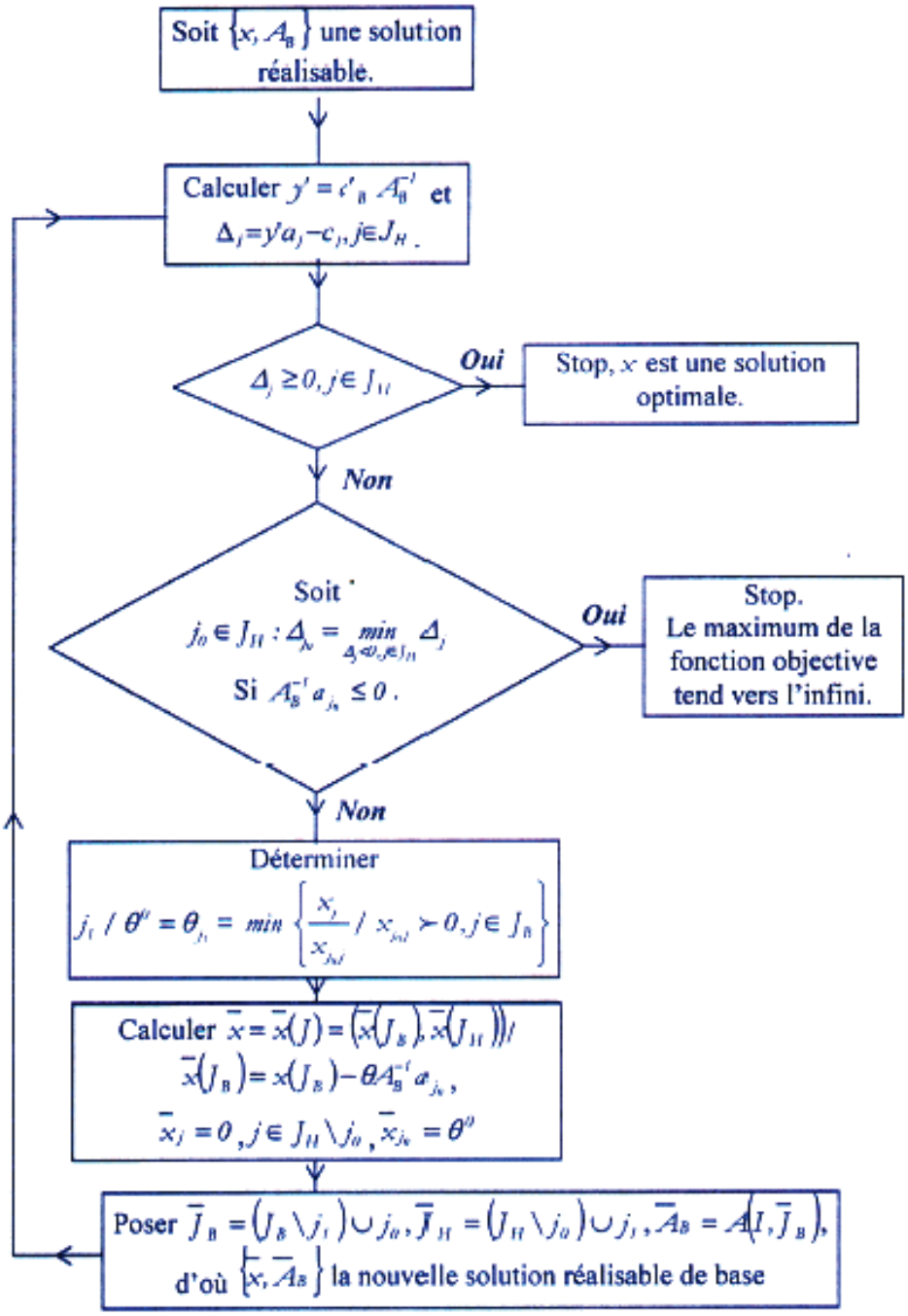
La nouvelle solution de base $\overline{x}$ sera trouvée de la manière suivante :
$\overline{x} = x + \theta l$, où l est la direction de changement de x et θ le pas de cette direction.
Construisons la direction l de la manière suivante :
Sur j_H posons : $l_j = \begin{cases} 0, j \in J_H \setminus J_0 \\ 1, j = j_0 \end{cases}$
Sur J_B : $\overline{x}$ doit être réalisable, donc elle doit vérifier $A\overline{x} = b$ et comme Ax=b donc $\theta Al = 0$, c'est à dire Al=0.
De cette dernière relation on obtient :
$l_B = l(J_B) = -A_B^{-1} A_H l_H$.
De là $\overline{x}_H = x_H + \theta l_H = \theta l_H \geq 0$ et $\overline{x}_B = x_B + \theta l_B \Longrightarrow \overline{x}_j = x_j - \theta A_B^{-1} a_{j_0}$.
Si les composantes du vecteur $A_B^{-1} a_{j_0} \leq 0$, alors $\overline{x}_j \geq 0, \forall \theta \geq 0$, donc on peut prendre $\theta = \infty$ et on aura une solution infinie.
Si parmi les composantes de vecteur $A_B^{-1} a_{j_0}$, il existent celles qui sont négatives, donc en augmentant θ certaines composantes seront négatives.
Pour avoir $\overline{x}_B \geq 0$, il faut prendre un pas maximal θ^0 :
$\theta^0 = \min\limits_{j \in J_B} \theta_j = min\{\frac{x_j}{x_{j_0 j}} / x_{j_0 j} > 0, j \in J_B\} = \theta_{j1}, j_1 \in J_B$,où $x_{j_0 j}$ est la j^{me} composante de $A_B^{-1} a_{j_0}$.
La nouvelle base sera :
$\overline{J}_B = (J_B \setminus J_0) \cup J_1$ *et* $\overline{A}_B = (A_B \setminus a_{j_0}) \cup a_{j_1}$.

2.9.5 Tableau du simplexe

c			c_1	c_2	c_3	...	c_m	c_{m+1}	...	c_j	...	c_n	
c_B	Base	b	a_1	a_2	a_3	...	a_m	a_{m+1}	...	a_j	...	a_n	θ_j
c_1	a_1	$b_1 = x_1$	1	0	0	...	0	$x_{m,m+1}$	...	x_{1j}	...	x_{1n}	θ_1
c_2	a_2	$b_2 = x_2$	0	1	0	...	0	$x_{2,m+1}$	...	x_{2j}	...	x_{2n}	θ_2
c_3	a_3	$b_3 = x_3$	0	0	1	...	0	$x_{3,m+1}$	...	x_{3j}	...	x_{3n}	θ_3
⋮	⋮	⋮	⋮	⋮	⋮		⋮	⋮		⋮		⋮	⋮
c_m	a_m	$b_m = x_m$	0	0	0	...	1	$x_{m,m+1}$	...	x_{mj}	...	x_{mn}	θ_m
Z=$c_B^{'} x_B$		$\triangle_j$	$\triangle_1$= 0	$\triangle_2$= 0	$\triangle_3$= 0	...	$\triangle_m$= 0	$\triangle_{m+1}$	...	$\triangle_j$	...	$\triangle_m$	

Remarque 2.5
L'initialisation de l'algorithme du simplexe est discutée dans l'annexe A.

2.9.6 Organigramme de l'algorithme du simplexe



2.10 Conclusion

Dans ce chapitre on a résumé les notions de base de la programmation linéaire. dans le chapitre qui suit on proposera un nouveau algorithme pour la résolution d'un programme linéaire en se basant sur le schéma général de l'algorithme **DCA**.

CHAPITRE 3

Résolution d'un programme linéaire avec la méthode DC

3.1 Introduction

Ce chapitre est consacré à notre application. Après avoir abordé dans les chapitres précédents quelques notions de la programmation linéaire et la programmation DC, nous introduisons une nouvelle approche pour la résolution d'un programme linéaire. Cela passe par la décomposition de notre programme linéaire sous forme d'un programme **DC**.

3.2 Décomposition d'un programme linéaire sous forme DC

Soit le programme linéaire suivant sous la forme canonique :

$$\begin{cases} max\ Z = c^{'}x \\ Ax \leq b \\ x \geq 0 \end{cases} \tag{3.1}$$

Notons (3.1) par PL.
Dans cette partie, nous allons construire une formulation **DC** pour le PL.
Soit l'ensemble $C = \{Ax \leq b,\ x \geq 0\}$.
Il est clair que C est un ensemble convexe (voir le théorème 2.1).
Donc notre PL s'écrira sous la forme suivante :

$$sup\{Z(x)/x \in C\}. \tag{3.2}$$

La fonction Z est une fonction convexe puisque elle est linéaire, et on a aussi l'ensemble C est convexe, donc le problème (3.2) est de la forme (1.4).

Donc on peut transformer le problème (3.2) à un problème sans contraintes c'est à dire de la forme (1.3) d'une manière équivalente. En effet :
On pose : $g = \chi_C$ et $h = Z$, avec χ_C est la fonction indicatrice de C.
Et donc le problème (3.2) s'écrira sous cette forme :

$$inf\{g(x) - h(x)/x \in \mathbb{R}^n\}. \tag{3.3}$$

La forme (3.3) est une décomposition **DC** de notre PL puisque, la fonction g (la fonction indicatrice de C) est une fonction convexe (voir la remarque 1.3) et h aussi est une fonction convexe (fonction linéaire). et donc notre PL est un problème **DC**.

3.3 Développement de l'algorithme DCA sur le PL décomposé

Dans cette section, on essaie d'appliquer l'algorithme **DC** sur le PL qu'on a déjà décomposé sous la forme **DC** c'est à dire le problème sous La forme (3.3) .
Soit le problème (3.3) :

$$inf\{g(x) - h(x)/x \in \mathbb{R}^n\}.$$

avec $g(x) = \chi_C(x)$ et $h = Z$ et $C = \{Ax \leq b,\ x \geq 0\}$.

Propriété 3.1 *[14]*
Si h est une fonction homogène(c'est à dire de classe C^∞) alors $\partial h(x^k) = \{\bigtriangledown h(x^k)\}$.

DCA appliqué au problème (3.3) :

Étape initiale :
Soit x^0 un point initial donné , $k = 0$, $\varepsilon > 0$.
Étape 1 :
Calculons $y^k \in \partial h(x^k)$.
Dans notre cas $\partial h(x^k) = \{\bigtriangledown h(x^k)\} = \{c^{'}\}$ d'après la propriété (3.1).
Étape 2 :
On détermine $x^{k+1} \in \partial g^*(y^k)$ c'est à dire :
$x^{k+1} \in argmax\{< y, x >, x \in C\}$.
Étape 3 :
Si les conditions d'arrêt sont vérifiées ($|| x^{k+1} - x^k || \leq \varepsilon$), alors on termine DCA, $x^* = x^{k+1}$ est la solution optimale, sinon $k = k + 1$ et on répète l'Étape 1.

L'inconvénient de cette méthode est la nécessite de résoudre un problème de programmation linéaire à chaque itération, ceci reste loin d'être l'idéal, cet inconvénient disparaît avec la version suivante.

On propose la décomposition suivante :

$R(x) = \frac{\rho}{2} \parallel x \parallel^2 - c^{'}x + \chi_C(x)$ et $T(x) = \frac{\rho}{2} \parallel x \parallel^2$, avec ρ est un paramètre strictement positif on peut le considérer comme régulateur.

il est clair que ce problème

$$inf\{R(x) - T(x)/x \in \mathbb{R}^n\} \tag{3.4}$$

est équivalent au problème de départ (PL) (3.1).

Montrons que $R(x)$ et $T(x)$ sont des fonctions convexes :
Montrons que la fonction $\psi(x)$=$\parallel x \parallel^2$ est une fonction convexe.
On va montrer que $\psi(\lambda x + (1-\lambda)y) \leq \lambda\psi(x) + (1-\lambda)\psi(y)$, $\forall\ \lambda \in [0,1]$, $\forall\ x, y\ \in \mathbb{R}^n$.
$\parallel \lambda x + (1-\lambda)y \parallel^2 \leq \parallel \lambda x \parallel^2 + \parallel (1-\lambda)y \parallel^2$ (l'inégalité triangulaires)
$\leq\ \lambda^2 \parallel x \parallel^2 + (1-\lambda)^2 \parallel y \parallel^2$ (homogénéité)
$\leq\ \lambda \parallel x \parallel^2 + (1-\lambda) \parallel y \parallel^2$ (puisque $\lambda \in [0,1]$).
D'ou le résultat.
La fonction $T(x) = \frac{\rho}{2}\ \psi(x)$ est aussi convexe (voir la proposition 1.1).
La fonction $(-c^{'}x)$ est aussi convexe (linéaire), et même pour la fonction indicatrice donc la fonction $R(x)$ est une fonction convexe d 'après la proposition (1.1).
Donc le problème (3.4) est un problème **DC**.

DCA appliqué au problème (3.4) :

Étape initiale :
Soit x^0 un point initial est donné , $k = 0$, $\varepsilon > 0$.
Étape 1 :
$y^k = \nabla T(x^k) = \rho x^k$(d'après la propriété (3.1)).
Étape 2 :
$x^{k+1} \in argmin\{\frac{\rho}{2} \parallel x \parallel^2 - x^{'}(y^k + c)/x \in C\}$
$\Leftrightarrow x^{k+1} \in argmin\{\frac{2}{\rho}(\frac{\rho}{2} \parallel x \parallel^2 - x^{'}(y^k + c))/x \in C\}$
$\Leftrightarrow x^{k+1} \in argmin\{\parallel x \parallel^2 - 2x^{'}(\frac{y^k + c}{\rho})/x \in C\}$.
Étape 3 :
Si les conditions d'arrêt sont vérifiées ($\parallel x^{k+1} - x^k \parallel \leq \varepsilon$), alors on termine DCA, $x^* = x^{k+1}$ est la solution optimale, sinon $k = k + 1$ et on répète l'Étape 1.

Proposition 3.1

En posant $z^k = \dfrac{y^k + c}{\rho}$, *on remarque que* x^{k+1} *n'est autre que la projection de* z^k *sur l'ensemble* C. *Cette projection existe toujours car l'ensemble* C *est fermé convexe (voir le théorème 2.1).*

Preuve :

Soit C un convexe fermé et z^k un point fixé dans $\mathbb{R}^n$ un espace de Hilbert.
Montrons que la solution optimale du problème $\{|| \ x \ ||^2 - 2 < x, z^k >)/x \in C\}$ est la projection de z^k sur C :
D'une manière générale on a :
$\min_{x \in C}$ f(x) $\Leftrightarrow$ $\min_{x \in C}$ (f(x)+a)
où a une constante réelle quelconque fixée.
D'où
$min\{|| \ x \ ||^2 - 2 < x, z^k >)/x \in C\} \Leftrightarrow min\{|| \ x \ ||^2 - 2 < x, z^k > + || \ z \ ||^2)/x \in C\}$
$\Leftrightarrow min\{< x - z^k, x - z^k > /x \in C\}$
$\Leftrightarrow min\{|| \ x - z^k \ ||^2 \ /x \in C\}$.
D'où le résultat. Ce problème est un problème de minimisation non linéaire avec contraintes linéaires qui aussi convexe, ce qui implique l'unicité de la solution.
Géométriquement, on cherche à minimiser le rayon de la sphère de centre z^k pour qu' elle soit tangentielle à l'ensemble C (voir la figure 3.1).

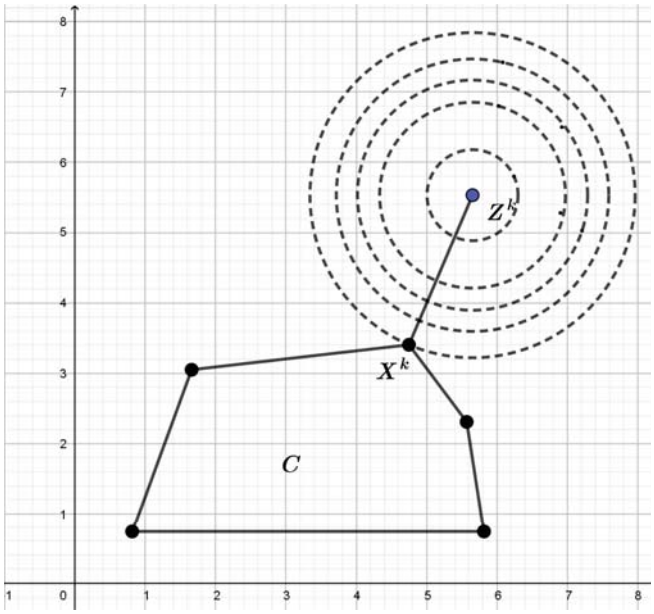


FIGURE 3.1 – *La projection d'un point fixé sur un un polyèdre*

On peut déterminer le point x^k par le système de KKT [10][15].

Remarque 3.1
Calcul de ∇T *:*
$T(x) = \frac{\rho}{2} \mid\mid x \mid\mid^2$, $x \in \mathbb{R}^n$ *donc* x *s'écrit comme suit* $x(x_1, x_2, ..., x_n)$, *et on a*
$\mid\mid x \mid\mid = < x, x >^{1/2} = \sqrt{x_1^2 + x_2^2 + ... + x_n^2} \Rightarrow \mid\mid x \mid\mid^2 = x_1^2 + x_2^2 + ... + x_n^2$.
Donc $T(x) = \frac{\rho}{2}(x_1^2 + x_2^2 + ... + x_n^2) \Rightarrow \nabla T(x_1, x_2, ..., x_n) = \rho(x_1, x_2, ..., x_n)^t$
D'où le résultat.

3.4 Exemple d'application

$$\begin{cases} Max\ Z(x,y) = -x + 6y \\ s.c : \\ -x + y \leq 3 \\ x + y \leq 7 \\ y \leq 4 \\ x, y \geq 0. \end{cases} \tag{3.5}$$

Les composantes **DC** sont :
$R(x) = \frac{\rho}{2} \mid\mid x \mid\mid^2 + x - 6y + \chi_C(x)$ et $T(x) = \frac{\rho}{2} \mid\mid x \mid\mid^2$
avec $C = \{x \in \mathbb{R}^2 : Ax \leq b;\ x, y \geq 0\}$

où $A = \begin{pmatrix} -1 & 1 \\ 1 & 1 \\ 0 & 1 \end{pmatrix}$ et $b = \begin{pmatrix} 3 \\ 7 \\ 4 \end{pmatrix}$ et on aussi $c = \begin{pmatrix} -1 \\ 6 \end{pmatrix}$

On applique **DCA**, avec $\rho = 1$:
Pour $k = 0$ on a :
$x^0 = (0, 0)$, $\varepsilon = 10^{-3}$, $y^0 = \rho x^0 = (0.0)$, x^1 est la projection de z^0 avec
$z^0 = y^0 + c = (0, 0) + (-1, 6) = (-1, 6) \Longrightarrow x^1 = (1, 4)$
Test d'arrêt n'est pas vérifié puisque $\mid\mid x^1 - x^0 \mid\mid > \varepsilon$, posons k=k+1 et on répète.
Pour $k = 1$ on a :
$y^1 = (1.4)$, $z^1 = (0, 10) \Longrightarrow x^2 = (1, 4)$, $\mid\mid x^2 - x^1 \mid\mid \leq \varepsilon$.
Donc x^2 est une solution optimale.
(Voir l'interprétation géométrique dans la figure (3.2)).

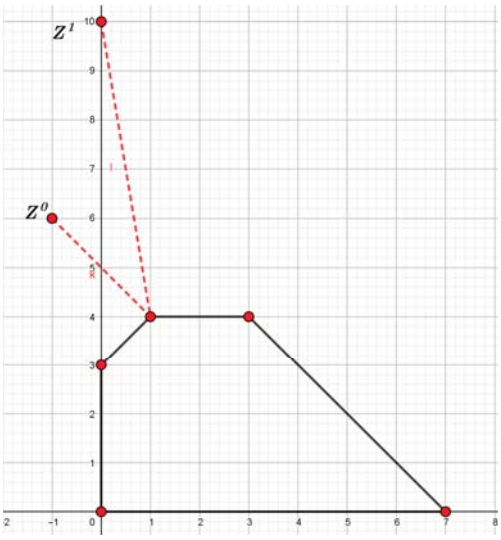


Figure 3.2 -

On refait **DCA**, avec $\rho = 2$. On aura les résultats suivants :

$\varepsilon = 10^{-3},\ \rho = 2$	x^k	y^k	z^k	x^{k+1}	Test d'arrêt
$k = 0$	(0,0)	(0,0)	(-0.5,3)	(0,3)	$\|\| x^1 - x^0 \|\| > \varepsilon$
$k = 1$	(0,3)	(0,6)	(-0.5,6)	(1,4)	$\|\| x^2 - x^1 \|\| > \varepsilon$
$k = 2$	(1,4)	(2,8)	(0.5,7)	(1,4)	$\|\| x^3 - x^2 \|\| \leq \varepsilon$

D'où $x^* = (1, 4)$ (Voir l'interprétation géométrique dans la figure (3.3)).

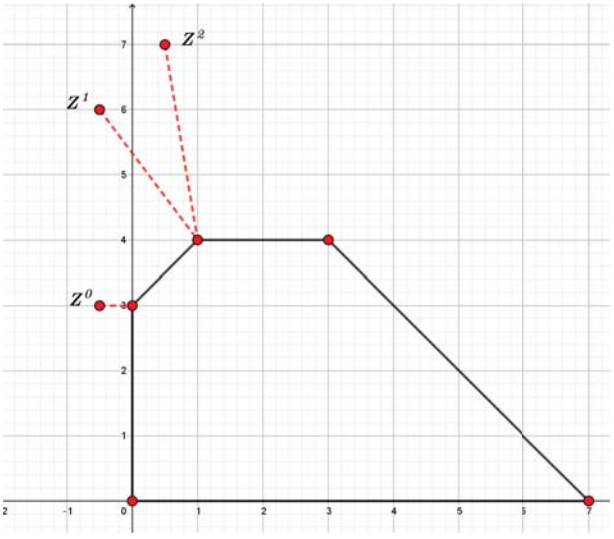


Figure 3.3 -

De même, avec $\rho = 4$. On aura les résultats suivants :

$\varepsilon = 10^{-3},\ \rho = 4$	x^k	y^k	z^k	x^{k+1}	Test d'arrêt
$k = 0$	(0,0)	(0,0)	(-0.25,1.5)	(0,1.5)	$\| x^1 - x^0 \| > \varepsilon$
$k = 1$	(0,1.5)	(0,6)	(-0.25,3)	(0,3)	$\| x^2 - x^1 \| > \varepsilon$
$k = 2$	(0,3)	(0,12)	(-0.25,4.5)	(0.75,3.75)	$\| x^3 - x^2 \| > \varepsilon$
$k = 3$	(0.75,3.75)	(3,15)	(0.5,5.25)	(1,4)	$\| x^4 - x^3 \| > \varepsilon$
$k = 4$	(1,4)	(4,16)	(0.75,8)	(1,4)	$\| x^5 - x^4 \| \leq \varepsilon$

D'où $x^* = (1, 4)$ (Voir l'interprétation géométrique dans la figure (3.4)).

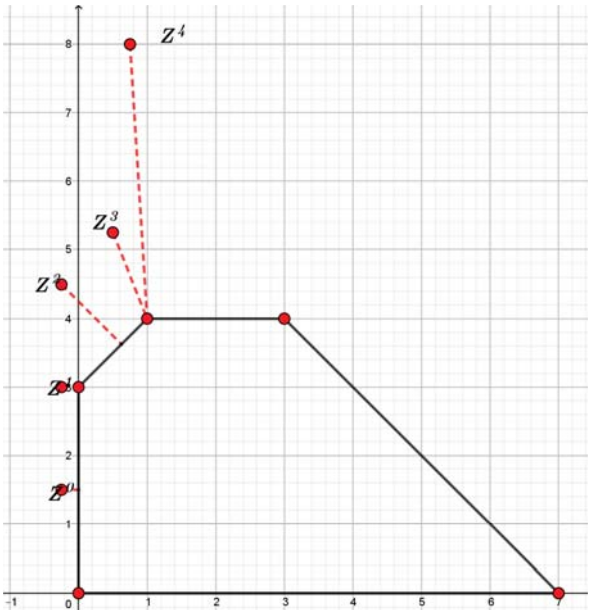


Figure 3.4 -

Remarque 3.2

On remarque que le point de départ à chaque fois dans l'exemples précédent est un sommet (x^0), dans l'exemple suivant on choisira un point à l'intérieur de l'ensemble C, et on applique ***DCA*** *sur cet exemple.*

Appliquons **DCA** sur l'exemple précédent avec $x^0 = (1, 1)$ comme point de départ. On aura les résultats suivants :

$\varepsilon = 10^{-3},\ \rho = 1$	x^k	y^k	z^k	x^{k+1}	Test d'arrêt
$k = 0$	(1,1)	(1,1)	(0,7)	(1,4)	$\| x^1 - x^0 \| > \varepsilon$
$k = 1$	(1,4)	(0,7)	(-1,13)	(1,4)	$\| x^2 - x^1 \| \leq \varepsilon$

D'où $x^* = (1, 4)$ (Voir l'interprétation géométrique dans la figure (3.5)).

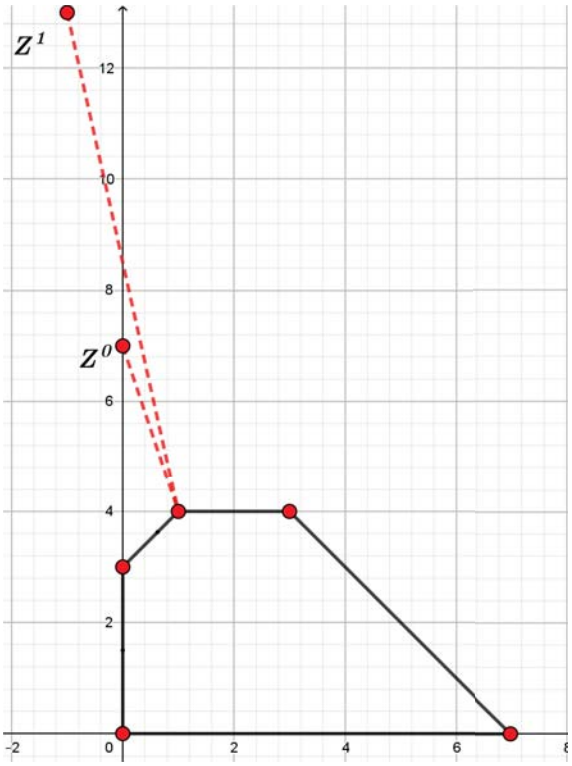


Figure 3.5 -

De même, avec $\rho = 2$. On aura les résultats suivants :

$\varepsilon = 10^{-3},\ \rho = 2$	x^k	y^k	z^k	x^{k+1}	Test d'arrêt
$k = 0$	(1,1)	(2,2)	(0.5,4)	(0.75,3.75)	$\| x^1 - x^0 \| > \varepsilon$
$k = 1$	(0.75,3.75)	(1.5,7.5)	(0.25,6.75)	(1.4)	$\| x^2 - x^1 \| > \varepsilon$
$k = 2$	(1,4)	(2,8)	(0.5,7)	(1,4)	$\| x^3 - x^2 \| \leq \varepsilon$

D'où $x^* = (1, 4)$ (Voir l'interprétation géométrique dans la figure (3.6)).

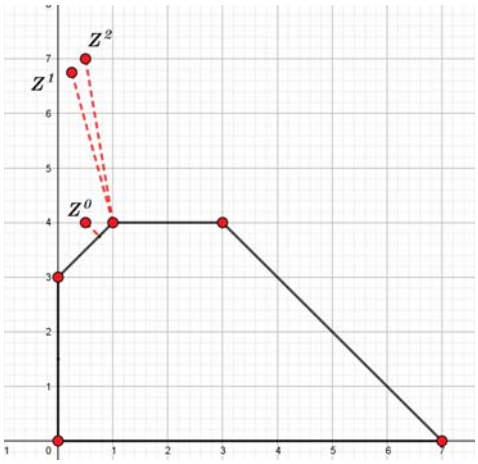


Figure 3.6 -

Commentaire :
On remarque que à chaque fois la valeur de ρ augmente, le nombre d'itérations augmente.

On essaie d'appliquer l'algorithme **DCA** avec un point à l'extérieur de l'ensemble C (point n'est pas réalisable), on prend le point $(-1, 0)$ et on aura les résultats suivants :

$\varepsilon = 10^{-3},\ \rho = 1$	x^k	y^k	z^k	x^{k+1}	Test d'arrêt
$k = 0$	(-1,0)	(-1,0)	(-2,6)	(0.5,3,5)	$\| x^1 - x^0 \| > \varepsilon$
$k = 1$	(0.5,3.5)	(0.5,8.5)	(-0.5,9.5)	(1,4)	$\| x^2 - x^1 \| > \varepsilon$
$k = 2$	(1,4)	(1,4)	(0,10)	(1,4)	$\| x^3 - x^2 \| \leq \varepsilon$

D'où $x^* = (1, 4)$ (Voir l'interprétation géométrique dans la figure (3.7)).

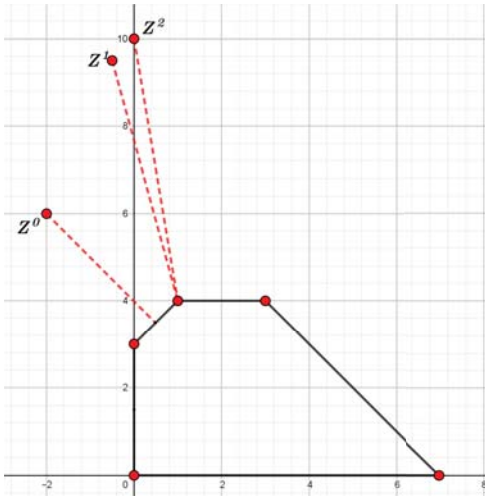


Figure 3.7 -

3.5 Résolution d'un PL avec la méthode simplexe

On résout le problème (3.5) avec la méthode du simplexe.
Le problème (3.5) est équivalent au problème suivant :

$$\begin{cases} Max\ Z(x,y) = -x + 6y \\ s.c : \\ -x + y + e_1 = 3 \\ x + y + e_2 = 7 \\ y + e_3 = 4 \\ x, y, e_i \geq 0, i = \overline{1,3}. \end{cases}$$

où les e_i sont des variables d'écarts.

- Dressons le premier tableau du simplexe :

Base	x	y	e_1	e_2	e_3	b	θ
e_1	-1	1	1	0	0	3	3
e_2	1	1	0	1	0	7	7
e_3	0	1	0	0	1	4	4
Δ_j	1	-6	0	0	0	Z=0	

- Dressons le deuxième tableau du simplexe :

Base	x	y	e_1	e_2	e_3	b	θ
y	-1	1	1	0	0	3	/
e_2	2	0	-1	1	0	4	2
e_3	1	0	-1	0	1	1	1
Δ_j	-5	0	6	0	0	Z=18	

- Dressons le dernier tableau du simplexe :

Base	x	y	e_1	e_2	e_3	b	θ
y	0	1	0	0	1	4	
e_2	0	0	1	1	-2	2	
x	1	0	1	0	1	1	
Δ_j	0	0	0	0	5	Z=23	

3.6 Conclusion

Dans ce chapitre, nous avons vu que l'optimisation **DC** peut être utilisée pour la résolution d'un problème de programmation linéaire.

Conclusion générale & perspectives

La classe des problèmes de programmation linéaire est une classe d'optimisation très importante, car elle trouve son application dans plusieurs domaines. On peut citer à titre d'exemple, les sciences économiques.

La recherche de la solution optimale d'un problème de programmation linéaire a été résolu par des méthodes exactes, la méthode du simplexe et la méthode adaptée.

Dans ce mémoire, on a concentré sur la résolution d'un problème de programmation linéaire qui donne une solution approchée. Pour se faire, nous avons abordé le problème de la par une nouvelle approche itérative qui est la programmation **DC** (Différence of convex function) et **DCA** (**DC** Algorithm). Cette méthode est basée sur la dualité **DC**.

Dans un premier temps, on a décomposé le problème de programmation linéaire sous forme d'un problème **DC** et puis on a adapté **DCA** sur ce problème décomposé.

L'application a montrée que la vitesse de convergence de l'algorithme dépend du paramètre ρ et du choix du point initial.

Comme perspectives de travail, on se propose :
• D'implémenter sur ordinateur l'algorithme **DC** appliqué sur le PL décomposé, et l'exécuter sur des problèmes plus compliqués .
• De comparer cette approche avec les méthodes exactes.
• De chercher un algorithme simple qui nous permet de trouver la projection d'un point sur un ensemble convexe pour faciliter les calcules.
• Enfin essayer de chercher d'autres décompositions **DC** de notre problème.

Annexes

3.7 Annexe A : Vue générale sur les méthodes de résolution d'un PL

La méthode graphique

Considérons le problème de la programmation linéaire suivant :

$$(P)\begin{cases} Z = c^{'}x \longrightarrow max \\ Ax \leq b \\ x_j \geq 0, j = \overline{1,n} \end{cases}$$

où A est une matrice d'ordre $m \times n$, $x \in \mathbb{R}^n$, $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$. Si le nombre de variables est supérieur d'une unité ou deux unités par rapport au nombre d'équations, alors la résolution d'un tel problème peut se faire simplement par la méthode graphique pour les deux cas.

— 1^{er} **cas :** $n - m = 1$

On fixe une variable, par exemple x_1 et on calcule les autres variables $(x_2, x_3, ..., x_n)$ en fonction de x_1 dans l'équation $Ax = b$, on obtient :

$$x_2 = \alpha_{21}x_1 + \alpha_2, x_1 = \alpha_{31} + x_1 + \alpha_3, ..., x_n = \alpha_{n1}x_1 + \alpha_n$$

On remplace ces variables par leurs valeurs dans la fonctionnelle qui devient :
$Z = \alpha_1 x_1 + \beta$, une fonction d'une seule variable, dont le maximum sera trouvé en fonction de signe de α_1 et des bornes de x_1.

— 2^{me} **cas :** $n - m = 2$

On fixe deux variables par exemple x_1 et x_2 et on calcule $x_3, ..., x_n$ en fonction de x_1 et x_2 :

$x_3 = \alpha_{31}x_1 + \alpha_{32}x_2 + \alpha_3$

$\vdots \quad \vdots \quad \vdots \quad \vdots$

$x_n = \alpha_{n1}x_1 + \alpha_{n2}x_2 + \alpha_n,$

On remplace $x_3, ..., x_n$ dans la fonctionnelle Z qui devient :
$Z = \gamma_1 x_1 + \gamma_2 x_2 + \gamma$.
Ensuite dans un plan $(x_1 o x_2)$ on trace tous les demi-plans $x_j \geq 0$, qui donneront le domaine des solutions admissibles. Par suite on tracera la droite $Z = \gamma = 0$ qui correspond à $\gamma_1 x_1 + \gamma_2 x_2 = 0$.
De là suivant le signe de γ_1 et γ_2, on trouvera le point extrÃłme qui correspond au maximum(minimum) désiré.

Exemple 3.1

$$\begin{cases} Z = x_1 + 2x_2 - x_3 \longrightarrow max \\ x_1 + x_2 + x_3 = 10 \\ 2x_1 - x_2 + x_3 = 20 \\ x_j \geq 0, j = \overline{1,3} \end{cases}$$

Ici $n = 3$, $m = 2$, $n - m = 1$, donc on fixe par exemple x_1 et on calcule les autres en fonction de x_1 :

$$\begin{cases} x_3 = -\frac{3}{2}x_1 + 15 \\ x_2 = \frac{1}{2}x_1 - 5 \end{cases}$$

Donc $Z = \frac{7}{2}x_1 - 25$.
On sait que $x_j \geq 0, \overline{1,3}$, c'est à dire, $-\frac{3}{2}x_1 + 15 \geq 0, \frac{1}{2}x_1 - 5 \geq 0$ et $x_1 \geq 0$,
De là $x_1 \leq 10$, $x_1 \geq 0$ et finalement $0 \leq x_1 \leq 10$, donc le max Z= $\max\limits_{0 \leq x_1 \leq 10} (\frac{7}{2}x_1 - 25)$ est atteint pour $x_1 = 10$. En suite on trouve $x_2 = x_3 = 0$ et $Z = 10$.

Exemple 3.2

$$\begin{cases} Z = 2x_1 + x_2 \longrightarrow max \\ x_1 + x_2 \leq 5 \\ -2x_1 + x_2 \leq 4 \\ 2x_1 - 4x_2 \leq 4 \\ x_j \geq 0, j = \overline{1,2} \end{cases}$$

L'intersection des cinq demis plans représentent les contraintes, forme le polygone $OABCD$, qui est l'ensemble des solutions admissibles.
La fonction objectif Z est représentée par l'équation $2x_1 + x_2 = Z_0(\triangle_{Z_0})$.
Pour $Z_0 = 2$ on aura $2x_1 + x_2 = 2$.
Si l'on déplace la droite $(\triangle_{Z_0})$ dans le sens de la flèche, qui le sens de croissance du bénéfice, l'optimum sera atteint au point extrÃłme $C = \triangle_1 \cap \triangle_2 = (4, 1)$.
D'ou la solution optimale $x^0 = (4, 1)$ avec $Z^0 = 9$.

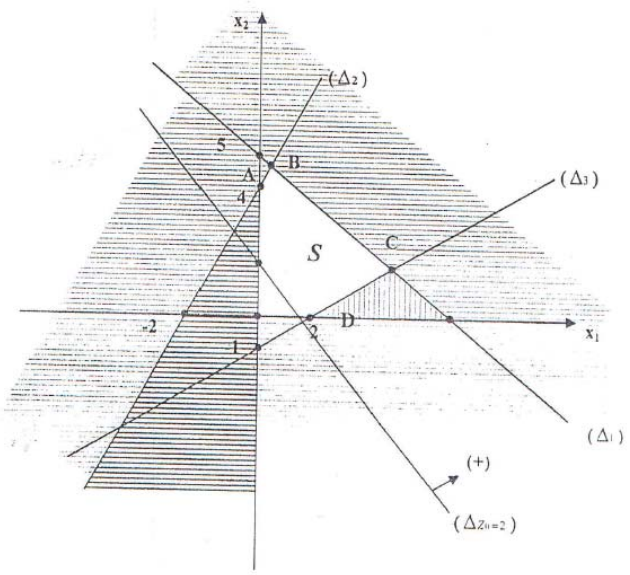


Figure 3.8 - *Résolution graphique d'un programme linéaire*

La méthode des deux phases

Considérons le problème de la programmation linéaire suivant :

$$(P)\begin{cases} Z = c^{'}x \longrightarrow max \\ Ax \leq b \\ x_j \geq 0, j = \overline{1,n} \end{cases}$$

• Première phase :

La première phase de résolution du problème (p) consiste à déterminer une solution de base réalisable de (P). Pour cela, on construit le problème suivant :

$$(P^{'})\begin{cases} -\sum_{i=1}^{m} x_{n+i} \rightarrow max, \\ [Ax]_i + x_{n+i} = b_i, i = \overline{1,m}, \\ x \geq 0, x_{n+i} \geq 0, i = \overline{1,m}, \end{cases}$$

Où les x_{n+i} sont appelés des variables artificielles.

Le problème $(P^{'})$ posède $n+m$ variables et m equations. Le vecteur $X(0,...,0,b,...,b_m)$ est réalisable pour $(P^{'})$ donc l'ensemble des solutions admissibles de $(P^{'})$ est non vide.

D'un autre côté la fonction objectif $(-\sum_{i=1}^{m} x_{n+i} \leq 0)$, donc le problème $(P^{'})$ admet une solution optimale $X^0(x^0, x_a^0)$.

Théorème 3.1 *[1]*

Soit $\{x^0, x_a^0\}$ une solution optimale de $(P^{'})$.

Si x_a^0 est différent de zéro alors les contraintes du problème de départ (P) sont contradictoires.

• Deuxième phase :
Soit $\{x^0, x_a^0\}$ une solution optimale de $(P^{'})$ avec des variables artificielles nulles, alors on utilise la solution x^0 avec sa matrice A_B^0, comme solution de base de départ du problème (P), et ceci constitue la deuxième phase.
Si $x_{n+i}^0 = 0$, $\forall i$ et $\exists$ un indice $i_0/a_{i0} \in A_B$, alors pour revenir à la deuxième phase, il faut exclure cette colonne de A_B.

La M-méthode
Le mathématicien américain Tcharness a proposé une méthode en rassemblant les deux phases en une seule.
Du problème (P), on construit un problème $(P^{''})$ de la maniéré suivante :

$$\begin{cases} \overline{Z} = c^{'}x - M\sum_{i=1}^{m} x_{n+i} \rightarrow max, \\ [Ax]_i + x_{n+i} = b_i, i = \overline{1,m}, \\ x_j \geq 0, j = \overline{1,n+m}, \end{cases}$$

Où $M > 0$ (un nombre positif très grand) et x_{n+i}, $i = \overline{1,m}$, des variables artificielles.
Le vecteur $X = (0,b) = (x = 0, x_{n+i} = b_i, i\overline{1,m})$ est une solution de base réalisable de $(P^{''})$.
Ici on choisit M suffisamment grand de telle sorte à avoir $c^{'}x - M\sum_{i=1}^{m} x^{n+i} \leq 0, \forall x \geq 0, \forall x_{n+i} \geq 0$ et ceci dans le but de prouver le problème $(P^{''})$ admet une solution optimale car c'est le maximum d'une fonction borné sur un ensemble des contraintes non vide.
On résout le problème $(P^{''})$ par la méthode du simplexe avec une solution de base de départ $(0,b)$ et on obtient une solution optimale X^0.

• Interprétation de la solution de $(P^{''})$:
a. S'il existe au moins un indice i_0 tel que $x_{n+i_0}^0 \neq 0$, alors les contraintes de $(P^{''})$ sont contradictoires.
b. si $x_{n+i_0}^0 = 0$, $\forall i = \overline{1,m}$, alors x^0 est une solution optimale de (P).

La dualité
Considérons les problèmes suivants :

$$\begin{cases} Z = c^{'}x \longrightarrow max \\ Ax \leq b \\ x_j \geq 0, j = \overline{1,n} \end{cases} \tag{3.6}$$

$$\begin{cases} b^{'}y \longrightarrow min \\ Ay \geq c \\ y \in \mathbb{R}^m \end{cases} \tag{3.7}$$

Le problème (2.5) est dit problème primal et le problème (2.6) est appelé problème dual du primal.

Remarque 3.3
Le dual est un problème linéaire et le dual du dual est le primale.

Méthode de construction du dual :
Les différentes transformations sont résumées dans le tableau suivant :

Primal	**Dual**
$c^{'}x \to max$	$b^{'}y \to min$
$(Ax)_i = b_i, i = \overline{1, m_1}$	$(Ay)_j \geq c_j, j = \overline{1, n_1}$
$(Ax)_i \leq b_i, i = \overline{m_1 + 1, m_2}$	$(Ay)_j \leq c_j, j = \overline{n_1 + 1, n_2}$
$(Ax)_i geq b_i, i = \overline{m_2 + 1, m}$	$(Ay)_j = c_j, j = \overline{n_2 + 1, n}$
$x_j \geq 0, j = \overline{1, n_1}$	$y_i \in \mathbb{R}, i = \overline{1, m_1}$
$x_j \leq 0, j = \overline{n_1 + 1, n_2}$	$y_i \geq 0, i = \overline{m_1 + 1, m_2}$
$x_j \in \mathbb{R}, j = \overline{n_2 + 1, n}$	$y_i \leq 0, i = \overline{m_2 + 1, m}$

Propriétés de la dualité :
Etant donné un problème primal (2.5) et son dual (2.6).

Théorème 3.2 *[1]*
Si x et y sont deux solutions réalisables du primal et du dual respectivement, alors $c^{'}x \leq b^{'}y$.

Théorème 3.3 *[1]*
Soient x^0 et y^0 deux solutions réalisables du primal et du dual respectivement.
Si $c^{'}x^0 = b^{'}y^0$ alors x^0 et y^0 sont des solutions optimales respectivement du primal et du dual.

Théorème 3.4 *(**Ecarts complémentaires**)[1]*
Deux solutions réalisables x^0 et y^0 respectivement du primal et du dual sont optimales si et seulement si :

$$(\sum_{i=1}^{m} a_{ij}y_i^0 - c_j)x_j^0 = 0, j = \overline{1, n}$$

$$(-\sum_{j=1}^{n} a_{ij}x_j^0 - b_i)y_i^0 = 0, i = \overline{1, m}$$

Bibliographie

[1] AIDENE. M ET OUKACHA. B. *Programmation Linéaire*. Pages Bleus. 2007.

[2] AKOA. F. B. *Approches de Points Intérieurs Et De La Programmation Dc En Optimisation Non Convexe*. Codes et Simulations Numériques Industrielles. Thèse préparée au Laboratoire de Mathématiques de l'Institut National des Sciences Appliquées de Rouen. France. 2005.

[3] EKLAND. L AND TEMAM. R. *Analyse Convexe Et Problèmes Variationnels*. Dunod Paris. 1974.

[4] HARTMAN. P. *On fonctions representable as a difference of convex functions*. Picafic J.Math 1959.

[5] HIRIART-URRUTY. J.B *Optimisation et analyse convexe*. Edp Sciences. 2009.

[6] HIRIART-URRUTY. J. B AND LEMARECHAL. C. *Convex Analysis and Minimization Algorithms*. Springer. Berlin. 1993.

[7] HORST. R AND THOAI. N. V. *Dc programming : Overview. Journal Of Optimization Theory And Applications*. 103 (Octobre 1999). 1-43.

[8] HOUACINE. M ET HADDADOU. S. *Développement de la méthode DC pour la résolution d'un problème de décision bi-niveau linéaire*. Mémoire d'Ingénieur. Université de béjaïa. 2007.

[9] LENOUAR. H ET KASRI. R. *Programmation DC pour la Résolution d'un J eu Bi-matriciel*. Mémoire de Master. Université de béjaïa. 2015.

[10] MINOUX. M. *Programmation Mathématiques, théorie et algorithmes*. DUNOD. 2^{eme} édition. 2008.

[11] NIU. Y. S *Programmation DC et DCA en Optimisation Combinatoire et Optimisation Polynomiale via les Techniques de SDP Codes et Simulations Numériques*. Thèse préparée au Laboratoire de Mathématiques de l'Institut National des Sciences Appliquées de Rouen. France. 2010.

[12] PHAM DINH. T. *Algorithms for solving a class of non convex optimization problems*. Methods of subgradients. Elsevier Science Publishers. B.V. North-Holland. 1986.

[13] ROCKAFELLAR. R. T *Convex Analysis*. Princeton University Press. 1970.

[14] SCHÜLE. T, SCHNÖRR. C, WEBER. S AND HORNEGGER, J. *Discrete tomography by convex-concave regularization and D.C. programming. Discrete Applied Mathematics*. 2005.

[15] SINHA. S. M. *Mathematical programming : Theory and Methods* . Elsevier. 2006.

[16] STERN. R. J, CLARKE. F. H, LEDYAEV. Y. S AND WOLANSKI. P. R *Norsmooth analysis and control theory*. Springer-Verlag. New York. 998.

[17] THI. H. A AND T. PHAM DINH. T. *DC Programming : Theory, Algorithms and Applications*. The State of the Art. Proceedings of The First International Workshop on Global Constrained Optimization and Constraint Satisfaction. Valbonne-Sophia Antipolis. France. 2002.

[18] THI. H. A AND PHAM DINH. T. *On solving Linear Complementarity Problems by DC programming and DCA*. Springer Science+Business Media. LLC. 2011.

[19] THI. H. A AND PHAM DINH. T. *Stabilité de la dualité lagrangienne en optimisation DC. (différence de deux fonctions convexes)*. C.R.Acad. Paris. t.318,Série I. 1994.

[20] THI. H. A AND PHAM DINH. T. *The DC (diference of convex functions) Programming and DCA revisited with DC models of real world nonconvex optimization problems*. Annals of Operations Research. 2005.

[21] TOUATI. S. *Résolution de Problèmes de Bin Packing à une dimension par la programmation DC*. Mémoire de Magister. Université de béjaïa. 2014.

[22] TOLAND. J. F. *Duality in Non convex Optimization*. Mathematical Analysis and Applications. 1978.

Résumé

Dans ce travail, on traite la résolution d'un problème de programmation linéaire par une méthode approchée, qui est la programmation ***DC*** *(classe de problèmes d'optimisation non convexe traitant avec la différence de deux fonctions convexes) et* ***DCA*** *(Algorithme* ***DC****).*

Une décomposition ***DC*** *a été proposée par nous pour le programme linéaire, sur laquelle on a appliqué* ***DCA****. On a pu proposé un algorithme* ***PLDCA*** *(****DCA*** *appliqué à un programme linéaire) pour la résolution des programmes linéaires.*

Mots clés : Programmation linéaire, Programmation **DC**, Décomposition **DC**, Fonctions **DC**, **DCA**.

Abstract

In this work we treat the solution of a linear programming problem by ***DC*** *programming method (a class of optimisation problems dealing with the difference between two convex functions) and* ***DCA*** *algorithm.*

A ***DC*** *decomposition has been proposed to a linear programming problem and then* ***DCA*** *was applied. Thus we contructed* ***PLDCA*** *algorithm for solving the linear programming problem.*

Key words : Linear programming, **DC** programming, **DC** decompostion, **DC** functions, **DCA**.