REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Mouloud Mammeri de Tizi Ouzou

Faculté des Sciences

Département de Mathématiques

Projet de fin d'étude

En vue de l'obtention du Diplôme de Master

En Recherche Opérationnelle

THEME

Optimisation non linéaire discrète

Encadré par :

M[r] M.Ouanes

Présenté par :

M[elle] KESSI Karima

M[elle] SIDENNAS Djedjiga

Membres du jury :

Président :M.MORSLI , U.M.M.T.O.

Rapporteur : M.OUANES, M,C,A, U.M.M.T.O.

Examinateur :B.OUKACHA , , U.M.M.T.O.

Examinateur : M .CHEBBAH , , U.M.M.T.O.

Promotion : 2011/2012

Nous remercions dieu tout puissant d'avoir guidé nos pas vers les portes du savoir tout en illuminant notre chemin, et nous avoir donné suffisamment de courage et de persévérance pour mener notre travail à terme

Nous tenons à adresser nos plus vifs remerciements à notre promoteur Monsieur ***M.OUANES*** *qui a su nous guider et nous orienter. Nous lui exprimons notre plus grande gratitude pour l'attention et le temps qu'il nous a consacré ainsi que les précieux conseils qu'il nous a prodigués.*

Nous remercions aussi les membres de la commission qui nous ont fait l'honneur d'en faire partie et qui ont eu la patience de nous écouter,

Nous témoignons une reconnaissance particulière à l'ensemble des enseignants qui nous ont suivis pendant notre cursus,

Enfin, nous remercions tous ceux qui nous ont encouragé tout au long de notre parcours universitaire et ceux qui ont contribué de près ou de loin à notre formation.

Dédicace

Tout d'abord je remercie Dieu le tout puissant, de m'avoir aidé dans les moments les plus difficiles, de m'avoir aidé à accomplir ce travail. Qu'il soit toujours dans mon cœur et ma tête.

Je dédie ce travail à mes parents, à ma très chère mère pour ses encouragements et ses prières tout au long de mes études, à mon père pour tous ce qu'il avait fait pour avoir ce résultat.

Je le dédie à mes frères et sœurs, et spécialement à mon marie, je le remercie pour leur encouragement et leur aide.

A mon binôme Karima.

A tous mes amies sans citer les noms.

A toute personne qui mérite l'appréciation et le respect de ma part.

Djedjiga

Dédicaces

Tout d'abord je remercie Dieu le tout puissant, de m'avoir aidé dans les moments les plus difficiles, de m'avoir aidé à accomplir ce travail. Qu'il soit toujours dans mon cœur et ma tête.

Je dédie mon modeste travail :

- ***A mes très chers parents***
- ***A mes frères***
- ***A mes sœurs et mes beaux frères.***
- ***A ma grande mère***
- ***A ma tente fatma***
- ***A mes tentes maternelles et paternelles***
- ***A mes oncles maternels et paternels***
- ***A mon binôme et amie :Djedji***
- ***A tous mes amis.***
- ***A tous ceux qui m'aime***
- ***A toute la promotion de recherche opérationnelle***

Karima

Sommaire

La recherche opérationnelle (RO), aussi appelée science du management ou science de la décision, est une discipline dont l'objet est d'aider les gestionnaires à prendre des décisions en utilisant des modèles et des méthodes scientifiques adaptées.

On situe la naissance de la RO lors de la deuxième guerre mondiale, lorsque l'Etat-major britannique fit appel à des équipes de mathématiciens et de physiciens pour l'aider a analyser des questions de stratégie militaire (développement d'un réseau de radars, organisation des convois maritimes…). Après la guerre, cette approche systématique et scientifique des problèmes de décision a été transposée au monde économique et industriel ou elle a connu de nombreux succès. Depuis, de nouvelles méthodes et de nouveaux champs d'application ont vu le jour.

Les questions auxquelles s'intéresse la RO peuvent être classées en différentes catégories, selon les caractéristiques des situations visées, des modèles proposées pour les présenter et des techniques de résolution utilisées. On peut par exemples évoquer les problèmes combinatoires, aléatoires ou concurrentiels. Les problèmes combinatoires apparaissent Lorsque les réponses possibles sont trop nombreuses pour pouvoir être énumérées Complètement (ordonnancement de la production, planning du personnel…). En Environnement aléatoire, tous les paramètres du problème ne sont pas connus avec certitude (gestion des pannes, des stocks, des files d'attente, de portefeuilles financiers…). En environnement concurrentiel, le décideur doit sélectionner une stratégie sans connaitre la position qu'adoptera son adversaire (choix politique, décision d'investissement, positionnement des produit…).

Une des parties essentielles de la recherche opérationnelle est la programmation non linéaire, qui étudie l'optimisation d'une fonction objective non linéaire soumise à des contraintes non linéaires.

La programmation non linéaire a pour objet l'étude et la résolution des « programmes non linéaires » c'est-à-dire des problèmes d'optimisation dans lesquels la fonction objective ou bien les contraintes (ou la fonction objective et les contraintes) sont exprimées de manière non linéaire. Il s'agit de la technique la plus célèbre de la recherche opérationnelle, celle qui a donné à cette discipline sa respectabilité scientifique et sa notoriété. Ceci est certainement dû au fait que de nombreux problèmes concrets provenant de domaines aussi divers que l'industrie , le raffinage, les transports, l'agriculture, la gestion…peuvent être (et de fait ont été) modélisés comme des programmes non linéaires, la résolution de ces modèles ayant permis l'obtention de gains

substantiels. D'autre part la programmation non linéaire est un outil mathématique très riche qui donne une vue pénétrante des méthodes d'optimisation et qui constitue une technique fondamentale de l'optimisation combinatoire.

On distingue dans la programmation non linéaire, la programmation non linéaire continue(en nombre réels) et la programmation non linéaire en nombres entiers, comme il est possible d'avoir les deux en même temps, ce qu'on appelle la programmation non linéaire mixte.

Dans notre travail, nous nous intéressons à la modélisation et la résolution des problèmes de programmation non linéaire en nombre entiers avec les méthodes principales : la méthode de transformation des systèmes non linéaires à des systèmes linéaires(la méthode de KCG) et la méthode de séparation et évaluation (Brunch and Bound).

Notre travail contient une introduction générale et trois chapitres et une conclusion.

Le premier chapitre présente des généralités sur la programmation non linéaire continue en rappelant ses notions de base et quelques méthodes utilise pour la résolution.

Le deuxième chapitre est consacre aux notions de base da la programmation non linéaire en nombres entiers, et quelques modèles utilisant la programmation en nombres entiers ainsi leurs méthodes de résolutions

Le troisième chapitre est vérification des résultats des exemples présentés dans les deux premiers chapitres sur LINGO.

Nous terminons notre travail par une conclusion et des références.

I. Introduction :

Dans ce chapitre, nous allons présenter les problèmes classiques de l'optimisation, leurs conditions d'optimalité, et quelque définition usuelle.

Un problème d'optimisation introductif consiste à minimiser une fonction $f: E \to R$ qui est appelée fonction objective et l'ensemble E est dit domaine réalisable ou domaine des solutions réalisables.

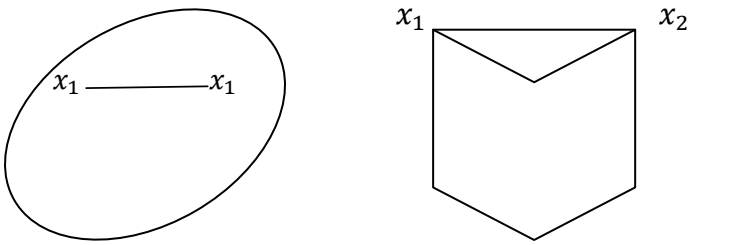
II. Convexité et optimisation :

Definition1.1 :

Un ensemble de points est dit convexe si tout segment joignant deux points quelconques de l'ensemble appartient à l'ensemble.

Un ensemble $C \subset R^n$ est convexe si :

$$\forall x_1, x_2 \in C: \alpha x_1 + (1-\alpha)x_2 \in C, \forall \alpha \in [0,1]$$



Ensemble convexe ***Ensemble non convexe***

Figure1.1 : ensembles convexe et non convexe.

Proposition1.1 :[9]

Soit $C_1, C_2, \dots, C_p$ des ensembles convexes de R^n alors :

$$C = \bigcap_{i=1}^{i=p} C_i \ \ est\ un\ ensemble\ convexe.$$

Définition1.2 :

On appelle combinaison linéaire convexe de n vecteurs: $x_1, x_2, \dots, x_n$ de R^n,de la somme :

$$\alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_n \ \ avec \quad \alpha_i \geq 0, i = 1, \dots, n, \sum_{i=1}^{i=n} \alpha_I = 1$$

Définition 1. 3 :

Soit A un ensemble de R^n on appelle enveloppe convexe de A, le plus petite ensemble convexe contenant A, noté *Conv(A)* avec : $A \subset Conv(A)$.

L'ensemble K **Conv(K)**

Figure 1.2 : enveloppe convexe.

Definition1.4:

On dit qu'une fonction f de R^n dans IR est convexe si pour tout x et tout y de R^n et $\forall \lambda \in [0,1]$ on a : $f(\lambda x + (1-\lambda) y) \leq \lambda f(x) + (1-\lambda) f(y)$. Si cette inégalité est stricte, on dit que f est strictement convexe.

Definition1.5 :

Soit : $\alpha = (\alpha_1, \alpha_2, \dots \alpha_n) \in R^n et\ \beta \ \forall \in R\ alors$

L'ensemble $H = \{x \in R^n, \alpha x = \beta\}$ est un Hyperplan de R^n.

Exemple :

$H=\{(x_1, x_2) \in R^2 \setminus \alpha_1 x_1 + \alpha_2 x_2 = \beta, \alpha_1, \alpha_2, \beta \in R\}$ est un Hyperplan de R^2 qui est une droite

Propriété 1.1 :[9]

Un hyperplan est un ensemble convexe.

Définition 1.6 :

Soient $\alpha = (\alpha_1, \alpha_2, \dots \alpha_n) \in R^n et\ \beta \in R\ on\ appelle$:

- Demi-espace positif fermé, L'ensemble $H_+ = \{x \in R^n, \alpha x \geq \beta\}$
- Demi-espace négatif fermé, L'ensemble $H_- = \{x \in R^n, \alpha x \leq \beta\}$
- Demi-espace positif ouvert, L'ensemble $H_+^\circ = \{x \in R^n, \alpha x > \beta\}$
- Demi-espace négatif ouvert, L'ensemble $H_+ = \{x \in R^n, \alpha x < \beta\}$

Définition 1.7 :

L'intersection d'un nombre fini de demi-espace fermé de R^n est appelée polytope convexe.

Définition 1.8 :

On appelle polyèdre, un polytope convexe borné.

Propriété1.2 :[9]

Un ensemble de R^n est dit compact, s'il est fermé et borné.

Exemple :

1) Les compacts convexes dans R^2 sont : les triangles, les cercles,…
2) Les compacts convexes dans R^3 sont : les sphères,…

Remarque :

Chaque contrainte d'un problème de (P.L) définit un demi-espace fermé et l'ensemble des contraintes définissent un polytope convexe.

Définition 1.9:

Soit C un ensemble convexe de R^n . x^* est appelé point extrême de C s'il n'existe pas deux points $x_1 \; et \; x_2$ dans C tels que:

$$x^* = \alpha x_1 + (1 - \alpha) x_2, \forall \alpha \in [0,1]$$

- **Caractérisation des points extrêmes :**

Considérons un problème canonique de programmation linéaire:

$$(P) \Leftrightarrow \begin{cases} Cx \to max \\ Ax = b, \\ x \geq 0, \end{cases}$$

Notation :

Désignons par :

$M = \{x \in R^n, Ax = b, x \geq 0\}$ L'ensemble des solutions admissibles du

problème (P).

Théorème 1.1 :[9]

L'ensemble M est convexe fermé.

Théorème1. 2 :[9]

soit $x = (x_1, x_2, \dots, x_n)$ un point extrême, alors :Les vecteurs colonnes de A correspondant aux composantes positives de X sont linéairement indépendants.

Théorème1. 3 :[9]

Si le système de vecteurs $\{a_1, a_2, \dots, a_n\}$ (les colonnes de A) possède m vecteurs linéairement indépendants $\{a_1, a_2, \dots, a_m\}$ alors :

La solution réalisable $x = (x_1, x_2, \dots, x_m, 0, \dots, 0)$ est un point extrême.

Conséquence1.1 :

1) Si la fonctionnelle atteint son maximum (minimum) sur un point quelconque de X,alors ce point est extrême.

2) Si le maximum (minimum) est atteint sur deux points extrêmesx_1 et x_2, alors: Tous les points du segment $[x_1, x_2]$sont des solutions optimales

Définition 1.10 :(minimum global) :

Soit $S \subset R^n$ et soit $f: S \to R$

Un point $x^* \in S \subset R^n$, est minimum global de f, si :

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

Définition 1.11 :(minimum local)

Soit $S \subset R^n$ est soit $f: S \to R^n$

Un point $x^* \in S \subset R^n$, est minimum local de f si :

$$\exists \varepsilon > 0 \; tel \; que:$$

$$f(x) \geq f(x^*), \forall \, x \in S \quad avec \quad \|x - x^*\| < \varepsilon$$

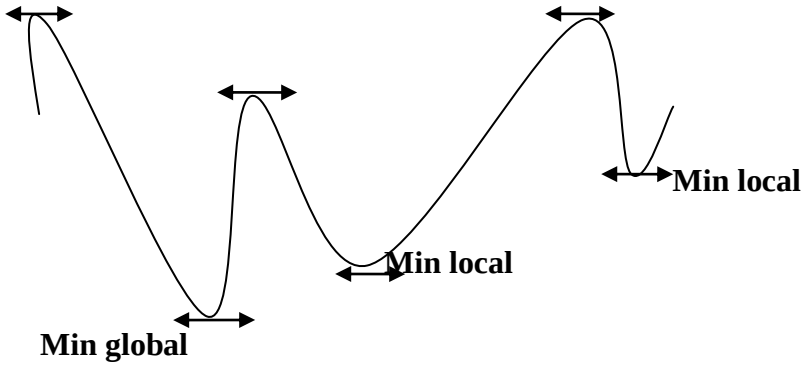


Figure 1.3 : minimum global, minimum local

Remarque :

Si x^* est un minimum global alors x^*est un minimum local, mais le contraire est faux.

Théorème1. 4 : [7]

Soit S un ensemble convexe de $R^n et \; soit \; f: S \to R^n$

Considérons le problème :

$$\begin{cases} f(x) \to min \\ x \in S, \end{cases}$$

Supposons que $x^* \in S$ est un minimum local alors :

x^* est un minimum global.

Théorème1. 5 :[7]

Supposons que $f: R^n \to R$ est différentiable au point x

Si x est un minimum local alors :$\nabla f(x) = 0$.

Théorème 1.6 :[7]

Supposons que $f: R^n \to R$ est différentiable et convexe sur R^n.

Si :$\nabla f(x^*) = 0, alors\ x^*$ est un minimum global de f :

$\Leftrightarrow f(x) \geq f(x^*), \forall x \in R^n$.

III. La programmation linéaire continue :

La programmation linéaire traite la résolution des problèmes d'optimisation pour les quelles la fonction objectif et les contraintes sont affines. **SAKAROVITCH** la considère la technique la plus célèbre de la recherche opérationnelle, celle qui a donnée à cette discipline sa notoriété.

Un problème de programmation linéaire mis sous la forme standard peut être défini comme suit :

$$m_p = \min_{x}\{C^t x:\ x \geq 0, Ax = b\} \qquad (P)$$

ou $C \in R^n, b \in R^m, x \in R^n$ *et A est une matrice* $m \times n$.

IV.La programmation linéaire en nombre entiers :

IV .1. Introduction

Dans la plupart des problèmes d'optimisation linéaire ou non linaire, les variables considérées sont supposées à valeurs réelles or dans certaines applications pratiques on peut avoir le cas ou toutes les variables sont entières, dans ce cas les variables présentent un nombre d'unité non fractionnables.

IV.2. Formulation du problème

Un programme linéaire en nombres entiers (PLNE) se présente comme suit

$$(P) \quad \begin{cases} (\min) & Max\ Z = CX \\ & A\ X\ (\leq, \geq, =)\ b \\ & x_j \in \mathbb{N}\ , j = 1 \dots n \end{cases}$$

Où $\in \mathbb{R}^n$, $A \in \mathbb{R}^{n*m}$, $b \in \mathbb{R}^m$.

Le programme relaxé de (PLNE) est donné comme suit :

$$(PL) \quad \begin{cases} (\min) & Max\ Z = CX \\ & A\ X\ (\leq, \geq, =)\ b \\ & x_j \geq 0\ , j = 1 \dots n \end{cases}$$

IV. 3. Méthodes de résolution

IV.3.1. Méthode des troncatures (algorithme de GOMORY) [6]

La méthode des troncatures appelée également « algorithme du plan sécant » ou « méthode des coupes» a été développée par Ralph. GOMORY en 1958.

IV.3.1.1. Algorithme de GOMORY :

1. Etant donné un (PLNE), résoudre le PL correspondant à l'aide d'un algorithme du simplexe. Si la solution optimale du PL est entière, elle est également une solution optimale au PL NE. La résolution est terminée.
2. Si une ou plusieurs variables de base dans la solution optimale du PL ne sont pas entières, on doit alors générer à partir d'une des lignes du tableau (celle dont la partie fractionnaire pour la variable de base correspondante est plus élevée) une contrainte supplémentaire dite coupe de GOMORY. Cette contrainte est ajoutée au tableau optimal du PL et on détermine le nouveau tableau optimal à l'aide de la méthode duale du simplexe.
3. Si les variables de bases dans le nouveau tableau optimal sont entières, nous avons obtenu la solution optimale du PLNE. La résolution est terminée.
4. Sinon, on doit générer à partir du dernier tableau optimal une nouvelle coupe de GOMORY, l'ajouter au dernier tableau optimal et trouver la solution optimale à l'aide de la méthode duale du simplexe.

Si la solution optimale obtenue est à valeurs entières, la résolution est terminée.

Sinon on répète la procédure jusqu'à l'obtention d'une solution optimale à valeurs entières,

IV.3.1.2. Règle pour générer la coupe de GOMORY

On choisira, dans le tableau optimal, la variable de base ayant la plus grande partie fractionnaire et on déduit l'équation de la coupe en exprimant toutes les variables du tableau en fonction de la valeur x_{Bi} de la variable de base.

Formellement la coupe de Gomory est générée comme suit :

$$x_{Bi} + \sum_{j\in J} a_{ij}x_j = b_i \qquad (1.1)$$

Où $J = \{\, j : j\ indice\ de\ variable\ hors\ base\}\ \ j = \overline{1,n}$

Notons par $\lfloor a \rfloor$ la partie entière inferieur à a. Dans ce cas,

$$x_{Bi} + \sum_{j\in J}\lfloor a_{ij}\rfloor x_j + \sum_{j\in J} f(a_{ij})x_j = \lfloor b_i\rfloor + f(b_i) \qquad (1.2)$$

Où $f(a) = a - \lfloor a\rfloor$. L'équation (1.2) est également égale à :

$$x_{Bi} + \sum_{j\in J}\lfloor a_{ij}\rfloor x_j = \lfloor b_i\rfloor + f(b_i) - \sum_{j\in J} f(a_{ij})x_j \qquad (1.3)$$

Nous pouvons écrire (1.3) sous forme d'une inéquation :

$$x_{Bi} + \sum_{j\in J}\lfloor a_{ij}\rfloor x_j \le \lfloor b_i\rfloor + f(b_i) \qquad (1.4)$$

Cette inégalité satisfait aussi :

$$x_{Bi} + \sum_{j\in J}\lfloor a_{ij}\rfloor x_j \le \lfloor b_i\rfloor \qquad (1.5)$$

De (1.1) nous avons :

$$x_{Bi} = b_i - \sum_{j\in J} a_{ij}x_j \qquad (1.6)$$

Nous remplaçons (1.6) dans (1.5) , nous aurons l'expression la coupe de Gomory

$$\sum_{j=1}^{n} f(a_{ij})x_j \ge f(b_i) \qquad (1.7)$$

IV.3.1.3. Exemple numérique :

Soit le programme linéaire suivant :

$$\max Z = 100x_1 + 120x_2$$
$$3x_1 + 4x_2 \leq 4100$$
$$x_1 + 3x_2 \leq 2400$$
$$2x_1 + 2x_2 \leq 2625$$
$$x_1, x_2 \in \mathbb{N}$$

En ajoutant les variables d'écart , nous obtenons le programme:

$$\max Z = 100x_1 + 120x_2$$
$$3x_1 + 4x_2 + x_3 = 4100$$
$$x_1 + 3x_2 + x_4 = 2400$$
$$2x_1 + 2x_2 + x_5 = 2625$$
$$x_j \in \mathbb{N}, \qquad j = \overline{1,5}$$

En appliquant l'algorithme du simplexe, on obtient le tableau optimal suivant :

c_j		100	120	0	0	0	
c_B	x_j de base	x_1	x_2	x_3	x_4	x_5	B
100	x_1	1	0	-1	0	2	1150
120	x_2	0	1	1	0	$-3/2$	162.5
0	x_4	0	0	-2	1	$5/2$	762.5
z_j		100	120	20	0	20	$Z = 134\,500$
$c_j - z_j$		0	0	-20	0	-20	

La solution n'est pas entière ⇒ construction de la coupe de GOMORY

Ici, comme la parie fractionnaire est identique, nous choisissons arbitrairement la 2^{eme} ligne du tableau.

La variable de base qu'on veut rendre entière est donc x_2 .

La coupe de Gomory est :

$$\frac{1}{2}\, x_5 \geq \frac{1}{2}$$

Pour appliquer l'algorithme duale du simplexe, il faut multiplier la contrainte de Gomory par (-1).

$$-\frac{1}{2}\, x_5 \leq -\frac{1}{2}$$

En introduisant la variable d'écart x_6, on obtient :

$$-\frac{1}{2}\, x_5 + x_6 = -\frac{1}{2}$$

Ajoutant cette équation au tableau optimal :

c_j		100	120	0	0	0	0	
c_B	x_j de base	x_1	x_2	x_3	x_4	x_5	x_6	b
100	x_1	1	0	-1	0	2		1150
120	x_2	0						162.5
0	x_4	0	1	1	0	$-3/2$		762.5
0	x_6	0						$-1/2$
		0	0	-2	1	5/2	0	
		0	0	0	0	$-1/2$		
		1						
z_j		100	120	20	0	20		
$c_j - z_j$		0						Z
		0	0	-20	0	-20	0	$= 134\,500$

σ_j	_ _ _ _ 40 -	

x_6 est la variable sortante. $\sigma_j = \frac{c_j - z_j}{a_{ij_0}} \quad \backslash a_{ij_0} < 0$

Où j_0 est l'indice de la variable sortante.

Le nouveau tableau après le pivotage est :

c_j		100	120	0	0	0	0	
c_B	x_j de base	x_1	x_2	x_3	x_4	x_5	x_6	b
100 120 0 0	x_1 x_2 x_4 x_5	1 0 3 0 0 2	0 1 0 0	-1 1 -2 0	0 0 1 0	1 0 0 1	4 - 5 -	1148 164 760 1
		100 40 0	120 0	20 -20	0 0	0 0	0 -40	Z $= 134\,480$

La solution optimale est : $x_1 = 1148\,, x_2 = 164 \;\; et \; Z = 134\,480$

La solution est entière donc la procédure est terminée.

e programmation linéaire peut être mis sous cette forme.

V. La programmation quadratique continue:

La programmation quadratique est la minimisation d'une fonction objective quadratique sous contraintes linéaires.

L'importance de la programmation quadratique provient de fait que plusieurs problèmes réels et académiques sont quadratiques, c'est le cas de l'optimisation de portefeuille dans la finance par exemple.

Un problème quadratique s'écrit sous la forme suivante :

$$\begin{cases} \text{Min } \frac{1}{2}\, x^T Qx + c^T x \\ a_i^t x = b_i \quad i \in I \\ a_i^t\, x \leq b_i \\ x \in R^n \end{cases}$$

Ou Q est une matrice symétrique semi définie positive, c est un vecteur de R^n

$a_I \in R^n$ et $b_i \in$ R.

VI. La programmation non linéaire continue :

VI.1. Introduction :

La programmation non linéaire regroupe un ensemble de sujets dans l'étude de problèmes d'optimisation.

La programmation non linéaire est la recherche d'un optimum d'une fonction objective non linéaire sur un sous ensemble convexe ou non convexe.

Le problème non linéaire s'écrit sous la forme

$$\begin{cases} minimiser\ f(x) \\ sous\ les\ contraintes \\ h(x) = 0 \\ g(x) \leq 0 \\ x \in R^n \end{cases}$$

$f : R^n \to R$, $h : R^n \to R^p$, $g : R^n \to R^m$ et $x \in R^n$.

Ainsi la programmation non linéaire se présente comme une généralisation de la programmation quadratique et la programmation linéaire.

Définition 1.12: (fonction linéaire)

Une fonction $f : R^n \longrightarrow R$ est dite linéaire si elle s'écrit comme suit :

$$f(x) = c^t x = \sum_{i=1}^{i=n} c_i x_i$$

c : un vecteur de R^n constant, indépendant de x

$f : R^n \longrightarrow R$ est dite linéaire si chacune de ses composantes

$f_i : R^n \longrightarrow R$, $i=1...n$, est linéaire : $f(x)= Ax$ ou $A \in Rn \times m$ est une matrice constante.

Définition1.13 :(fonction affine)

1. Une fonction $f : R^n \longrightarrow R$ est dite affine si elle s'écrit :

$f(x)=c^t x + d$

c : un vecteur de constantes de R^n

d : une constante de R

2. Une fonction $g : R^n \longrightarrow R^m$ est dite affine si chacune de ses composantes :

$g_i : R^n \longrightarrow R$, $i=1...m$, est affine :

$g_i(x) = Ax+b$

A : est une matrice de $R^{n \times m}$

b : est un vecteur de R^m

Définition 1.14:(fonction non linéaire)

Toute fonction qui n'est pas affine est dite non linéaire.

Définition1.15 :(dérives partielles)

Soit f : $R^n \longrightarrow R$ une fonction continue, la fonction notée :

$f(x) : R^n \longrightarrow R$, également notée par: $\frac{\delta f(x)}{\delta(xi)}$

est appelée ième dérivé partielle de *f*, elle est définie par :

$$\lim_{\alpha \to 0} \frac{f(x_1, \dots, x_{i+\alpha}) - f(x_1, \dots, x_i, \dots, x_n)}{\alpha}$$

Définition1.16:(Gradient)

Soit $f : R^n \longrightarrow R$ une fonction différentiable, la fonction notée :

$$f(x) : \mathrm{R}^n \longrightarrow \mathrm{R}$$

est appelée le gradient de f définie par :

$$f(x) = \begin{pmatrix} \frac{\delta f(x)}{\delta(x1)} \\ \\ \\ \\ \cdot \frac{\delta f(x)}{\delta(xn)} \end{pmatrix}$$

Définition1.1 7:(Dérivée directionnelle)

Soit $f : R^n \longrightarrow R$ une fonction continue. Si pour tous d R^n, la dérivée directionnelle de $D(x)$ de f en x dans la direction d est donnée par :

$$D(x) = \lim_{\alpha \to 0} \frac{f(x_1 + \alpha d) - f(x_1)}{\alpha}$$

si la limite existe, de plus, le gradient de f existe, alors la dérivée directionnelle est le produit scalaire entre le gradient de f et la direction d :

$$D(x) = (\quad f(x))^T d$$

Définition1.1 8:(fonction différentiable)

Soit $f : R^n \longrightarrow R$ une fonction continue. Si pour tous $d \in R^n$, la dérivée directionnelle de f dans la direction d existe, alors la fonction f est dite différentiable.

Exemple :

Soit $f(x_1, x_2, x_3) = e^{x1} + x_1^2 x_3 - x_1 x_2 x_3$

Et soit d = $\begin{pmatrix} d_1 \\ d_2 \\ d3 \end{pmatrix}$ la dérivée directionnelle de f en $x = (x_1, x_2, x_3)$ dans la direction

d est :

D(x) $= (d_1, d_2, d_3)\nabla f(x_1, x_2, x_3)$

$= d_1 ((e^{x1} + 2x_1x_3) - (x_2x_3)) - d_2x_1x_3 + d_3(x_1^2 - x_1x_2)$

Remarque :

La dérivée partielle = dérivée directionnelle dans la direction des axes de coordonnées :

$$\frac{\delta f(x)}{\delta(xi)} = \nabla f(x))^T e_i$$

Avec :

$e_{i = (0,\dots 1,\dots 0)}{}^T$: le ième vecteur de la base canonique de R^n

Définition 1.19 : (direction de descente)

Soit $f : R^n \longrightarrow R$, $x \in R^n$, $d \in R^n$ est dite direction de descente au point x si et seulement si :

$$\exists \theta > 0 : f(x + \lambda d) < f(x), \forall\, \lambda \in]0, \theta[$$

Théorème 1.7 :[13]

Soit $f : R^n \longrightarrow R$ différentiable au point $x \in R^n$ et $d \in R^n$ un vecteur qui vérifie :

$$d^T \nabla f(x) < 0$$

Alors : *d* est une direction de descente au point x.

Théorème 1.8:(convexité par le gradient)[13]

1. $f : X \subseteq R^n \longrightarrow R$ une fonction différentiable sur un ensemble convexe *X*.

f est convexe sur X si et seulement si :

$f(y) - f(x) \geq (y - x)^T \quad \nabla f(x), \qquad \forall x, y \in X$

2. $f:$ X $\subseteq$ $R^n \longrightarrow$ R une fonction strictement convexe sur X si et seulement si :

$f(y) - f(x) > (y - x)^T \quad \nabla f(x), \qquad \forall x, y \epsilon X$

Définition 1.20: (matrice Jacobienne)

Soit $f: R^n \longrightarrow R^m$.

La fonction $J(x):$ $R^n \longrightarrow R^{n\times m}$ définie par :

$$J(x) = (\nabla f(x))^T = \begin{pmatrix} \frac{\delta f_1}{\delta x_1} & \cdots & \frac{\delta f_1}{\delta x_n} \\ \vdots & \ddots & \vdots \\ \frac{\delta f_m}{\delta x_1} & \cdots & \frac{\delta f_m}{\delta x_n} \end{pmatrix}$$ est appelée matrice Jacobienne.

Définition1.21 :(matrice Hessienne)

Soit $f: R^n \longrightarrow R$ deux fois différentiable.la fonction notée

$\nabla^2 f(x)$: $R^n \longrightarrow R^{n\times m}$ définie par :

$$H = \nabla^2 f(x) = \begin{pmatrix} \frac{\delta^2 f(x)}{\delta x_1^2} & \frac{\delta^2 f(x)}{\delta x_1 \delta x_2} & \cdots & \frac{\delta^2 f(x)}{\delta x_1 \delta x_n} \\ \frac{\delta^2 f(x)}{\delta x_2 \delta x_1} & \frac{\delta^2 f(x)}{\delta x_2^2} & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\delta^2 f(x)}{\delta x_n \delta x_1} & \frac{\delta^2 f(x)}{\delta x_n \delta x_2} & \cdots & \frac{\delta^2 f(x)}{\delta x_n^2} \end{pmatrix}$$

est appelée matrice Hessienn

Exemple:

Soit $f(x_1, x_2, x_3) = e^{x1} + x_1^2 x_3 - x_1 x_2 x_3$

Le gradient de f donnée par :

$$f(x_1, x_2, x_3) = \begin{pmatrix} e^{x1} + 2x_1 x_3 - x_2 x_3 \\ -2x_1 x_3 \\ x_1^2 - x_1 x_2 \end{pmatrix}$$

Le Hessienne de f est donnée par :

$$H = \nabla^2 f(x_1, x_2, x_3) = \begin{pmatrix} e^{x1} + 2x_3 & -x_3 & 2x_1 \\ -2x_3 & 0 & -2x_1 \\ 2x_1 - x_2 & -x_1 & 0 \end{pmatrix}$$

Définition 1.22:(matrice semi définie positive)

La matrice A$\in R^{n\times m}$ est dite semi définie positive si :

$$x^T Ax \geq 0, \forall x \in R^n$$

Définition 1.23 : :(matrice définie positive)

La matrice A$\in R^{n\times m}$ est dite définie positive si :

$x \neq 0 \quad x^T Ax > 0, \forall x \in R^n$

Définition 1.24 : :(convexité par le Hessienne)

Soit :$f: X \longrightarrow R^n$ une fonction convexe, si $\nabla^2 f(x)$ est semi définie positive (resp défini positive) pour tous x ∈ X alors f est convexe (resp. strictement convexe).

Remarque :

- Toute matrice symétrique semi défini positive n'admet pas de valeurs propres négatives.

- Toute matrice symétrique définie positive admet de valeurs propres Strictement positives.

Définition 1.25 :

- On appelle le rang colonne d'une matrice, le nombre maximum de colonnes linéairement indépendantes.

- On appelle le rang ligne d'une matrice, le nombre maximum de lignes linéairement indépendantes.

Remarque :(convexité par le gradient)

Quelque soit la matrice A, le rang ligne = rang colonne et on le note par : rang(A).

Proposition 1.2:[12]

Soit A ∈ $R^{n\times m}$, si rang(A) = r ≤ n alors :

$\forall b \in R^n$ Le système : Ax = b admet une solution $x \in R^n$.

Définition 1.26: (**matrice non singuliere**)

On dit que A ∈ $R^{n\times m}$ est non singulière si:

rang (A) = n.

Définition 1.27 :(point stationnaire)

Soit f: $R^n \longrightarrow R$, un point x d∈ X est dite stationnaire ou (critique) de f si :

$\nabla f(X) = 0$.c'est donc un point ou le gradient s'annule.

On note S(f) l'ensemble des points stationnaire de f.

Proposition1.3 :[12]

Si f admet un extremum local en x, alors x est un point stationnaire.

Remarque :

Pour déterminer les extrêmes locaux, on cherche donc tout d'abord à déterminé les points stationnaires.

La matrice Hessienne permet alors de conclure, s'il s'agit bien (ou non) d'un maximum ou d'un minimum.

VI.2. L'optimisation sans contraintes

Définition1.28 :

Soit f: $R^n \longrightarrow R$, on appelle problème de minimisation sans contraintes (P) suivant :

$$(P_1) \qquad \begin{cases} \min f(X) \\ X \in R^n \end{cases}$$

f : est appelée critère ou fonction coût.

V.2.1. Condition nécessaire d'optimisation du premier ordre :

Théorème 1.9:[12]

Soit :$f: X \in R^n \longrightarrow R$ différentiable au point $x \in X$, *si* X est un minimum local de (P_1) alors :

$$\nabla f(x) = 0.$$

V.2.2 Condition nécessaire d'optimalité de second ordre :

Théorème 1.10 :[12]

Soit $f: X \longrightarrow R^n$ deux fois différentiable au point $x \in R^n$, si x est un minimum local de *(P)* alors : $\nabla f(x) = 0$. Et la matrice Hessienne de f au point x , noté par $H(x)$, est semi définie positive.

V.2.3. Condition suffisantes d'optimalité de second ordre :

Théorème 1.12 : [12]

Soit :$f: X \longrightarrow R^n$ différentiable au point $x \in X$:

- Si $\nabla f(x) = 0$ (x est un point stationnaire de *f*) et H(x) est définie positive, alors x est un minimum local de (P_1).

- Si $\nabla f(x) = 0$ et H(x) est définie négative, alors x est un maximum local de (P_1).

Exemple :

Déterminer les points stationnaires de f et la matrice Hessienne va déterminer la nature des points.

Soit $f(x) = x^3 + y^3 - 3xy$ définie sur R^2 pour déterminer les extrêmes de f il faut d'abord déterminer les points stationnaires.

➢ Calcul de $\nabla f(x,y) = \left(\frac{\delta f(x,y)}{\delta(x)}, \frac{\delta f(x,y)}{\delta(y)}\right) = (3x^2 - 3y, 3y^2 - 3x)$

$(3x^2 - 3y, 3y^2 - 3x) = (0,0)$ donc

$$\begin{cases} 3x^2 - 3y = 0 \\ 3y^2 - 3x = 0 \end{cases} \qquad \begin{cases} y = x^2 \\ x = 0,\ x = 1 \end{cases}$$

$S(f) = \{(0,0) ; (1,1)\}$ des points stationnaires sont les seuls extrêmes éventuels.

❖ **Nature du point (0 ,0) :**

$$\nabla^2 f(x,y) = \begin{pmatrix} 6x & -3 \\ -3 & 6y \end{pmatrix}$$

$H(0,0) = \nabla^2 f(0,0) = \begin{pmatrix} 0 & -3 \\ -3 & 0 \end{pmatrix}$

$H((0,0)-\lambda I) = \begin{pmatrix} 0-\lambda & -3 \\ -3 & 0-\lambda \end{pmatrix}$

$\det H((0,0)-\lambda I) = \lambda^2 - 9$

$H((0,0)-\lambda I) = 0 \Rightarrow \lambda_1 = 3$ et $\lambda_2 = -3$

H(0,0) n'est pas définie positive ni définie négative. Donc le point (0,0) n'est pas un extremum (minimum, maximum).

❖ **Nature de point stationnaire(1,1)**

$D^2f(x,y) = \begin{pmatrix} 6x & -3 \\ -3 & 6y \end{pmatrix}$

H(*1,1*) = $D^2f(1,1) = \begin{pmatrix} 6 & -3 \\ -3 & 6 \end{pmatrix}$

H((1,1)-λI) = $\begin{pmatrix} 6-\lambda & -3 \\ -3 & 6-\lambda \end{pmatrix}$

det H((1,1)-λI) =(6-λ) (6-λ) – 9

=(3-λ) (9-λ) =0

$\Rightarrow \lambda_1$=3 et λ_2= 9

$\lambda_i > 0$ donc H(*1,1*) est définie positive, alors : le point (1,1) est un minimum local.

VI.3. Optimisation avec contraintes :

Nous avons traité dans la partie précédente, la résolution d'un problème de minimisation sans contraintes.

Nous considérons maintenant le cas avec contraintes, donc l'ensemble X est une partie de R^n décrite par des contraintes fonctionnelles d'égalité ou d'inégalité.

Soit le problème suivant :

$$(P_2) \qquad \begin{cases} f(X) \to \min \\ g_i(X) \leq 0 \quad I = \{1, \dots, m\} \quad X \epsilon R^n \end{cases}$$

$f: R^n \to R$ de classe C^1.

$g_i: X \ \epsilon R^n \to R \quad \forall i = 1, \dots, m$

Définition 1.29 :(contrainte qualifiée)

Nous dirons que, les contraintes sont qualifiée au point X^* si :

- les fonctions g_i sont affines.
- Ou bien les vecteurs $\nabla g_i(X^*)$, $i \in I_0$,sont linéairement indépendants ou :

$$I_0 = \{i \in I, g_i(X^*) = 0\}$$

Proposition 1.4 : [14]

Si les contraintes sont qualifiées au point X^*, alors :$y \in R^n$ une direction admissible, si et seulement si :

$$\nabla g_i(X^*).\, y \leq 0, \forall i \in I_0$$

VI.3.1. Méthodes de résolutions d'un problème non linéaire avec contraintes :

VI.3.1.1. Méthode de multiplicateurs de lagrange :

VI.3.1.1.1. Introduction :

Pour résoudre un problème non linéaire, plusieurs méthodes sont utilisé .dans cette partie, on s'intéresse à l'une des ces méthodes : **méthodes des multiplicateurs de Lagrange** qui est une méthode d'optimisation, permettant de déterminer l'extremum local d'une fonction f sous contraintes g.

On pose I: (I) $\begin{cases} f(x,y) \to min \\ g(x,y) = 0 \end{cases}$

- $f: R^n \to R$
- $g: R^n \to R$

Pour résoudre le problème(I), on forme la fonction **L** appelée Lagrangien :

$$L(x,y,\lambda) = f(x,y) + \lambda g(x,y)$$

- λ: *une variable inconnue appelée multiplicateures de Lagrange.*

VI.3.1.1.2. Principe de la méthode :

Supposons que f et g sont des fonctions différentiables

1. Il faut chercher tous les points stationnaires du problème comme solution du système(S) obtenu en annulant les dérivées partielle de **L** par rapport à toutes les variables x ,y et λ avec :

$$\begin{cases} \dfrac{\delta L}{\delta x} = \dfrac{\delta f}{\delta x} + \dfrac{\delta g}{\delta x} = 0 \\ \dfrac{\delta L}{\delta y} = \dfrac{\delta f}{\delta y} + \dfrac{\delta g}{\delta y} = 0 \\ \dfrac{\delta L}{\delta \lambda} = \dfrac{\delta f}{\delta \lambda} + \dfrac{\delta g}{\delta \lambda} = 0 \end{cases}$$

2. Pour savoir si les points stationnaires correspondent a un maximum ou un minimum, il faut examiner les conditions suivant :

on a un maximum si :

$$\begin{cases} \left[\frac{\delta^2 L}{\delta x^2}\ \frac{\delta^2 L}{\delta y^2}\right] - \left[\frac{\delta^2 L}{\delta x \delta y}\right]^2 > 0, \\ \frac{\delta^2 L}{\delta x^2} < 0 \\ \frac{\delta L}{\delta y^2} < 0 \end{cases}$$

Ou on a un minimum si :

$$\begin{cases} \left[\frac{\delta^2 L}{\delta x^2}\ \frac{\delta^2 L}{\delta y^2}\right] - \left[\frac{\delta^2 L}{\delta x \delta y}\right]^2 > 0, \\ \frac{\delta^2 L}{\delta x^2} > 0 \\ \frac{\delta L}{\delta y^2} > 0 \end{cases}$$

Exemple :

Résoudre le problème non linéaire suivant :

$$\begin{cases} f(x,y) = 5x^2 + 6y^2 + xy \to min \\ x + 2y - 24 = 0 \end{cases}$$

- On forme le Lagrangien :

$$L(x,y,\lambda) = (5x^2 + 6y^2 + xy) + \lambda(\, x + 2y - 24)$$

$$\nabla L(x,y)=0 \Leftrightarrow \begin{cases} \dfrac{\delta L}{\delta x}=10x+y+\lambda \quad =0 \\ \dfrac{\delta L}{\delta y}=-x+12y+2 \quad =0 \\ \dfrac{\delta L}{\delta \lambda}=x+2y-24=0 \end{cases}$$

On résout le système de trois équations et trois inconnus on trouve :

$$x=6 \qquad y=9 \qquad \lambda=-51$$

On calcule :

$$\begin{cases} \dfrac{\delta^2 L}{\delta x^2}=10 \\ \dfrac{\delta^2 L}{\delta y^2}=12 \\ \dfrac{\delta^2 L}{\delta x \delta y}=-1 \end{cases}$$

On a:

$$\begin{cases} \left[\dfrac{\delta^2 L}{\delta x^2}\ \dfrac{\delta^2 L}{\delta y^2}\right]-\left[\dfrac{\delta^2 L}{\delta x \delta y}\right]^2=119>0, \\ \dfrac{\delta^2 L}{\delta x^2}=10>0 \\ \dfrac{\delta L}{\delta y^2}=12>0 \end{cases}$$

Donc le point (6 ,9) est un minimum ; alors c'est la solution du problème.

Remarque :

Si on a n variables et m contraintes, alors :

$$L: R^{n+m} \to R$$

$$(X,\lambda) \to L(X,\lambda\ \ )=f(X)+\sum_{i=1}^{m} \lambda\ \ {}_i g_i(X)$$

- On résout le système $\nabla L(X,\lambda) = 0$ pour trouver les points stationnaires de L.
- Pour déterminer la nature du point stationnaire (minimum, maximum) il faut trouver H(x) : la matrice Hessienne de L, si :

➢ H(x) est définie positive alors x est un minimum local.

➢ H(x) est définie négative alors x est un maximum local.

➢ H(x) est indéfinie alors x n'est pas un extrémum.

VI.3.1.2. Méthode de Kuhn-Tuer

Théorème 1.12 : (condition nécessaire)[14]

Soit le problème :(P_2) $\begin{cases} f(X) \to min \\ g_i(X) \leq 0 \\ X \in R \end{cases}$ $I = \{1, \dots, m\}$

- $f: R^n \to R$ *de classe* C^1.
- $g_i: X \subseteq R^n \to R$ *de classe* C^1.

On suppose que les contraintes sont qualifiées au point X^*.

Alors, une condition nécessaire pour que X^* soit une solution de (P_1), est qu'il existe des nombres positive $\lambda_1, \dots, \lambda_n$(appelés multiplicateurs de Kuhn-Tuer ou de Lagrange généralisé) tel que

$$\begin{cases} \nabla f(X^*) + \sum_{i=1}^{m} \lambda_i g_i(X^*) = 0, \\ \lambda_i g_i(X^*) = 0 \qquad \forall i \in I \end{cases}$$

Théorème 1.13: (condition suffisante)[14]

Soit le problème :(P_3) $\begin{cases} f(X) \to min \\ g_i(X) \leq 0 \end{cases}$ $I = \{1, \dots, m\}$

- $f: R^n \to R$ *de classe* C^1 *et convexe*
- $g_i: X \subseteq R^n \to R$ *de classe* C^1*. et convex*

On suppose que les contraintes sont qualifie au point X^*, alors X^* est une solution de (P_3) si et seulement si il existe des nombres positives $\lambda_1, \dots, \lambda_m$ tel que :

$$\begin{cases} \nabla f(X^*) + \sum_{i=1}^{m} \lambda_i g_i(X^*) = 0, \\ \lambda_i g_i(X^*) = 0 \qquad \forall i \in I \end{cases}$$

Exemple :

Soit le problème (P) suivant :

$$\begin{cases} \min\ x^2 + y^2 \\ x + y \leq 1 \\ x \in R, y \in R \end{cases}$$

Pour trouver la solution de (P) on résout le système suivant :

$$\begin{cases} 2x + \lambda = 0 \\ 2y + \lambda = 0 \\ \lambda(x + y - 1) = 0 \\ \lambda \geq 0 \\ x \in R, y \in R \end{cases}$$

Si $\lambda \neq 0$ on aura :

$x=y=1/2$ et $\lambda = -1$ ne vérifie pas la troisiéme condition donc
$(x, y) = \left(\frac{1}{2}, \frac{1}{2}\right)$ n'est pas une solution optimale.

Si $\lambda = 0$on aura

$x = y = 0$ et $\lambda = 0$ vérifie la troisième condition donc la solution optimale de problème (P) est $(x, y) = (0,0)$.

I. Introduction :

De très nombreux problèmes d'optimisations peuvent se modéliser comme des programmes non linéaires à variables mixtes.

Un problème de programmation non linéaire à variables mixtes est un problème dont la fonction économique ou l'une de contraintes est non linéaire, en plus certains variables de modèle sont réelles et d'autres entiers.

II. Notion de base :

II.1. Programmation non linéaire en nombres entiers pure :

Un problème de programmation non linéaire en nombre entiers est un problème non linéaire auquel on ajoute la contrainte supplémentaire que les variables ne peuvent prendre que des valeurs entières et sa formule générale est la suivante :

$$\begin{cases} \min f(x) \\ g(x) \leq 0 \\ x_i \in N \;\; i = 1, \dots, n \end{cases}$$

Où :

f est une fonction convexe.

x_i $(i = 1, \dots, n)$: représentent les variables entiers.

L'une des fonctions f, g est non linéaire (ou f, g sont toutes les deux non linéaires).

II.2. Programmation non linéaire en variables binaires :

Un problème de programmation non linéaire en variables binaires est un cas particulier de la programmation non linéaires en nombres entiers pures et sa formule générale est la suivante :

$$\begin{cases} \min f(x) \\ g(x) \leq 0 \\ x_i \in \{0,1\} \;\; i = 1, \dots, n \end{cases}$$

II.3. Programmation non linéaire en variables mixtes

Un problème de programmation non linéaire en variable mixte est un problème dont certain variables sont réelles et d'autre sont entiers :

$$\begin{cases} \min f(x) \\ g(x) \leq 0 \\ x_i \in R^n \;\; i = 1, \dots, n \\ x_j \in N \quad j \neq i \;\; i = 1, \dots, n, j = 1, \dots, n \end{cases}$$

III. Quelques exemples de la programmation non linéaire Discrète :

III.1. Problème de sac a dos :

Considérons un ensemble de *n* articles $\{A_1, A_2, \dots, A_n\}$ à placer dans un sac. Le poids de l'article A_i est égale à a_i et le poids de sac ne doit pas dépasser la valeur *b*. A chaque article c_i et à chaque paire d'articles différentes$\{A_i, A_j\}$, *i*<*j* est associe la valeur c_{ij} et on cherche à placer dans le sac un ensemble d'articles qui respecte la contrainte de poids et maximise la valeur total.

- **Formulation mathématique :**

Soit *n* articles $A_1, \dots, A_n$ des coefficients entiers positifs ou nuls $c_i, a_i \; (1 \leq i \leq n)$

c_{ij} $(1 \leq i < j \leq n$ et un coefficient entier positif *b*.

Pour tout *i*∈ $\{1, \dots, n\}$ définissons la variable bivalente x_i qui vaut 1 si et seulement si l'article A_i est retenu pour être placer dans le sac on peut alors formuler le problème de sac à dos comme suit :

$$\begin{cases} \max f(x) = \sum_{i=1}^{n} c_i x_i + \sum_{i=1}^{n-1} \sum_{j+i+1}^{n} c_{ij} x_i x_j \\ s.c \\ \quad \sum_{i=1}^{n} a_i \, x_i \leq b \\ \quad x_i \in \{0,1\} \quad i = 1, \dots, n \end{cases}$$

III.2. Problème de location et de transport :

Une entreprise de vente en gros effectue des livraisons en utilisant des camions ayant tous la même capacité Cm^3. Elle cherche à louer des entrepôts et dispose pour cela une liste d'entrepôts $E_1, E_2, \ldots, E_n$ proposés à la location, le prix mensuel de la location de l'entrepôt E_i est égale à p_i $(i = 1, \ldots, n)$. La capacité de stockage en m^3 d chaque entrepôt est égale à $c_i \times C$ $(i = 1, \ldots, n)$ ou c_i désigne un nombre entier. La région est découpée en p zones géographiques, $Z_1, Z_2, \ldots, Z_p$, et l'entreprise doit à partir de ces entrepôts, livrer de la marchandise dans ces différentes zones. La demande par mois de chaque zone est égale à $d_j \times C$, d_j etant un nombre entier. Cette entreprise a estimé le cout de livraison de chaque zone depuis chaque entrepôt. Le coût de livraison de la zone Z_j à partir de l'entrepôt E_i est égale à$l_{ij}/camion$. On fait de plus que chaque livraison consiste à acheminer un camion complètement rempli (Cm^3) d'un entrepôt E_i vers une zone géographiqueZ_j. On cherche à déterminer les entrepôts qu'il faut louer de façon à minimiser la somme des couts de transport et de location.

- **Formulation mathématique :**

Ce problème de location des entrepôts peut se formuler comme un programme en variable bivalentes et entiers. Les variables bivalentes y_i ($i = 1, \ldots, n$) correspondent au fait qu'un entrepôt est soit loue $(y_i = 1)$, soit non $(y_i = 0)$. Et la fonction économique exprime la somme des couts de location des entrepôts et des couts de livraison donc ce problème peut se formuler comme suit :

$$\begin{cases} \min \sum_{i=1}^{n} p_i y_i + \sum_{i=1}^{n} \sum_{j=1}^{p} l_{ij} x_{ij} \\ \sum_{j=1}^{p} x_{ij} \leq c_i \; i = 1, \ldots, n \\ (1 - y_i) x_{ij} = 0 \;\; i =, \ldots, n \, , j = 1, \ldots, p \\ \sum_{i=1}^{n} x_{ij} = d_j \qquad j = 1, \ldots, p \\ x_{ij} \in N \;\; i = 1, \ldots, n, \; j = 1, \ldots, p \\ y_i \in \{0,1\} \; i = 1, \ldots, n \end{cases}$$

IV. Méthodes de résolutions:

IV.1. Introduction :

Dans ce paragraphe on s'intéresse à des programmes non linéaires dont La forme générale comme suit :

$$\textbf{(PN)} \quad \begin{cases} \boldsymbol{Max\ f(x)} \\ \boldsymbol{g_i\ (x) \le b_i \quad i = 1 \dots m} \\ \boldsymbol{x \in X\ = \{\, x \in Z^n \quad L_j \le\ x_j\ \le U_j \quad j = 1 \dots n \,\}} \end{cases}$$

Où f et les g_i sont des fonctions à valeurs réelles, L_j et U_j sont des nombres entiers avec $L_j < U_j$ pour $j = 1 \dots n$.

Dans ce paragraphe nous décrirons les méthodes de résolution essentielles de (PN) : la méthode « Poly-blocs », la méthode de transformation d'un système non linéaire à un système linéaire et la méthode de Brunch and Bound.

IV.2. La méthode « Poly-blocs » pour (PN) :[5]

IV.2.1. Principe de la méthode :

Cette méthode suppose que f et les g_i sont des fonctions croissantes sur $[L_j, U_j]$ pour $j = 1 \dots n$,

Considérons la fonction

$$G(x) = \max_{i=\overline{1,m}} \{\, g_i(x) - b_i \,\} \qquad (2.1)$$

La frontière de la région réalisable peut être exprimée par $\Gamma = \{x \in X/\ G(x) = 0\}$

Soit $S = \{\, x \in X \subset \mathbb{Z}^n / \ g_i\ (x) \le b_i \quad i = 1 \dots m \,\}$

Soit (α, β) un box dans X , avec $\alpha \in S$ et $\beta \notin S$.

Supposons $G(\alpha) < 0$.

Soit x_b un point d'intersection de la droite $x = \lambda\,\alpha + (1-\lambda)\,\beta$, $0 \le \lambda \le 1$.

Puisque $G(\alpha) < 0$ et $G(\beta) > 0$, il existe un point x_b dans X qui satisfait $G(x_b) = 0$.

Notons par $\lfloor x \rfloor$ le vecteur dont chaque composante x_i est égale à la parie entière inferieure à $x_i \quad i = \overline{1,n}$.

Notons par $\lceil x \rceil$ le vecteur dont chaque composante x_i est égale à la parie entière supérieure à $x_i \quad i = \overline{1,n}$.

Soit $x^F = \lfloor x_b \rfloor$ et $x^I = \lceil x_b \rceil$.

On peut remarquer que $x^F \in S$ et $x^I \notin S$.

Considérons deux boxes entiers $\langle\, \alpha, x^F \rangle$ et $\langle\, x^I, \beta\, \rangle$.

Par la monotonie de f et des g_i , il n'y a aucun point réalisable meilleur que x^F dans $\langle \alpha, x^F \rangle$ et il n'y a aucun point réalisable dans $\langle x^I, \beta \rangle$.

Par conséquent, lorsqu'on cherche une solution optimale pour (PN), on peut supprimer les boxes $\langle \alpha, x^F \rangle$ et $\langle x^I, \beta \rangle$ de $\langle \alpha, \beta \rangle$ après avoir comparé x^F avec la solution courante.

Pour construire les boxes des nombres entiers nous utilisons les formules suivantes :

Corollaire 1.1: soit $A = \langle \alpha, \beta \rangle$, $B = \langle \alpha, \gamma \rangle$ et $C = \langle \gamma, \beta \rangle$ où $\alpha \le \gamma \le \beta$ alors $A \backslash B$ et $A \backslash C$ peut être partitionner au n nouveau subboxes

$$A \backslash B = \bigcup_{i=1}^{n} \left(\prod_{j=1}^{i-1} \langle \alpha_j, \gamma_j \rangle * \langle \gamma_i + 1,\ \beta_i \rangle * \prod_{j=i+1}^{n} \langle \alpha_j, \beta_j \rangle \right) \tag{2.2}$$

$$A \backslash C = \bigcup_{i=1}^{n} \left(\prod_{j=1}^{i-1} \langle \gamma_j, \beta_j \rangle * \langle \alpha_i,\ \gamma_i - 1 \rangle * \prod_{j=i+1}^{n} \langle \alpha_j, \beta_j \rangle \right) \tag{2.3}$$

IV.2.2. Algorithme « Poly-blocs »

Etape 0 : Initialisation

Soient $L = (l_1, l_2 \dots l_n)^T$, $U = (u_1, u_2 \dots u_n)^T$.

Si L est non réalisable, alors le problème (PN) n'a pas de solution réalisable, si U est realisable, alors U est la solution optimale pour (PN), stop

Sinon poser $x_{opt} = L$, $f_{opt} = f(x_{opt})$, $x^1 = \{\langle L, U\rangle\}$ et poser $K = 1$.

Etape 1 : sélection de box et recherche d'un point de la frontière

Sélectionner un box $\langle \alpha, \beta \rangle \in X^K$

Soit $X^K = X^K \backslash \langle \alpha, \beta \rangle$, trouvons la racine λ^* de l'équation suivante :

$$G(\lambda\, \alpha + (1 - \lambda)\beta) = 0, \qquad \lambda \in [0,1] \tag{2.4}$$

Où G est définie en (2.1).

Poser $x_b = \lambda^* \alpha + (1 - \lambda^*)\beta$

Poser $x^F = \lfloor x_b \rfloor$, $x^I = \lceil x_b \rceil$

Remarque : si $x^F = x_b$ poser $x^I = x_b + e_j$ avec $x_b + e_j \leq \beta$. (e_j est le vecteur de base canonique)

Si $f(x^F) > f_{opt}$, poser $x_{opt} = x^F$ et $f_{opt} = f(x^F)$.

Etape 2 : Partition et suppression

1. Appliquer la formule (2.2) pour partitionner l'ensemble $\Omega_1 = \langle \alpha, \beta \rangle \setminus \langle x^I, \beta \rangle$ en une union de boxes de nombres entiers.
   Soit $x^F \in \langle \tilde{\alpha}, \tilde{\beta} \rangle \in \Omega_1$. Posons $\Omega_1 := \Omega_1 \setminus \langle \tilde{\alpha}, \tilde{\beta} \rangle$.

   Soit $y^K = \Omega_1 \cup \Omega_2$.

2. Pour chaque box obtenu dans le processus de partition au dessus, effectuer les tests suivants :

a. Si β est réalisable, supprimer $\langle \alpha, \beta \rangle$ de y^K.

De plus si $f(\beta) > f_{opt}$, poser $x_{opt} = \beta$ et $f_{opt} = f(\beta)$.

b. Si α est non réalisable, supprimer $\langle \alpha, \beta \rangle$ de y^K .

c. Si $f(\beta) \leq f_{opt}$, supprimer $\langle \alpha, \beta \rangle$ de y^K .

d. Si α est réalisable, β est non réalisable et $f(\alpha) > f_{opt}$, poser $x_{opt} = \alpha$ et $f_{opt} = f(\alpha)$.

Noter par z^K l'ensemble de boxes restants à la fin du processus de suppression ci-dessus.

Etape 3 : Mise à jour des boxes

Supprimer tous les boxes $\langle \alpha, \beta \rangle$ dans X^K avec $f(\beta) \leq f_{opt}$.

Poser $X^{K+1} = X^K \cup z^K$.

Si $X^{K+1} = \emptyset$. Stop

Sinon poser $K \coloneqq K + 1$ et aller a l'étape 1.

Remarque : on peut utiliser la méthode de bissection ou bien la méthode de Newton pour trouver la racine de l'équation(4) .

IV.3. Méthode pour transformer un système non linéaire en un système linéaire (l'algorithme de KCG (Kelley et Cheney et Goldstein)[11]

IV.3.1. Introduction :

La méthode s'applique aux programmes non linéaires convexes. Un programme non linéaire peut être facilement modifié pour avoir un programme linéaire équivalent.

Le principe fondamental de cette méthode est d'approximer le domaine réalisable d'un programme non linéaire par un ensemble fini de demi-espaces fermés puis de résoudre la séquence des programmes linéaires approximés.

IV.3.2.Principe de la méthode :

Considérons le problème de programmation convexe suivant

$$(PNC) \qquad \begin{cases} Min\, f(x) = c^t x \\ g_i(x) \geq 0 \quad i = 1 \dots m \end{cases}$$

Où $x = (x_1, x_2 \dots x_n)^t \in \mathbb{R}^n$ et $g_1, g_2 \dots g_m$ sont des fonctions concaves dans $\mathbb{R}^n$,

Soit $G = \{x:\ g_i(x) \geq 0, i = \overline{1,m}\}$ et supposons que G est continu dans un ensemble compact $T \subset \mathbb{R}^n$, définie par un ensemble fini d'inégalités linéaires

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

Pour $x \in T$, soit $\partial g_i(x) \subset \mathbb{R}^n$ l'ensemble des sous gradients, ou sous différentiel de g_i

Prenant $\xi \in \partial\, g_i(x)$ alors

$$g_i(y) \leq g_i(x) + \xi^t(y - x) \qquad \text{Pour chaque } y \in \mathbb{R}^n$$

Si g_i est différentiable en x , alors $\xi = \nabla g_i(x)$, nous supposons que $\partial g_i(x) \neq 0 \quad i = \overline{1,m}$.

IV.3.3.Algorithme de KCG :

On va donner les étapes de l'algorithme de découpage (Kelley et Cheney et Goldstein) pour résoudre le problème (PNC) :

1. Résoudre le programme linéaire $Min\, f(x) = c^t x$, $x \in T$ et soit x^0 sa solution optimale.

Si x^0 est continue dans l'ensemble $G(\epsilon) = \{x : x \in T, g_i(x) \geq -\epsilon,\ i = \overline{1,m}\}$

Où $\epsilon > 0$, stop ; l'optimum de (PNC) est atteint.

Sinon poser $k = 0$ et aller a l'étape 2.

2. Soit $x^k \in T$, tel que $x^K \notin G(\epsilon)$, trouver l'indice s_k par

$$g_{s_k}(x^k) = \min\{g_i(x^k), i = \overline{1,m}\} < 0$$ et sélectionner $\xi^k \in \partial g_{s_k}(x^k)$.

Résoudre le programme linéaire suivant

$$\begin{cases} \min c^t x \\ \tilde{g}_{s_h}(x, x^h) = g_{s_h}(x^h) + (\xi^h)^t(x - x^h) \geq 0\,, \qquad h = \overline{0,k} \\ x \in T \end{cases}$$

3. Soit x^{k+1} la solution optimale du programme linéaire précédent.

Si $x^{k+1} \in G(\varepsilon)$, stop, sinon poser $k = k + 1$ et aller a 2.

IV.3.4. Exemple numérique : pour illustrer l'application de l'algorithme Considérons le problème convexe suivant :

$$(i)\begin{cases} \min f(x) = 4x_1 + 5x_2 \\ sc \\ g_1(x) = -2(x_1)^2 - 2x_1x_2 - 2(x_2)^2 + 4 \geq 0 \\ g_2(x) = -(x_1)^2 - (x_2)^2 + 4x_1 - 3 \geq 0 \\ T = \{x : x \in Z^n, 0 \leq x_1 \leq 4, -4 \leq x_2 \leq 4\} \end{cases}$$

La première itération il s'agit de résoudre le problème suivant :

$$(1) \quad \begin{cases} \min f(x) = 4x_1 + 5x_2 \\ sc \\ x \in T \end{cases}$$

La solution optimale pour (1) est $x^0 = (0, -4)$ où $f(x^0) = -20$

$g_1(x^0) = -28$, $g_2(x_0) = -19$ et $s_0 = 1$.

On voit que la contrainte $g_1(x)$ est la plus violée au point x_0, alors on construit la contrainte linéaire suivante : $g_1(x^0) + (x - x^0)\nabla g_1(x^0) \geq 0 \qquad (*)$

Après le calcul on trouve $(*) = 2x_1 + 4x_2 + 9 \geq 0$

On ajoute cette contrainte au problème (1) :

$$(2) \quad \begin{cases} \min f(x) = 4x_1 + 5x_2 \\ sc \\ 2x_1 + 4x_2 + 9 \geq 0 \\ 0 \leq x_1 \leq 4 \\ -4 \leq x_2 \leq 4 \\ x_1 \in Z^n \\ x_1 \in Z^n \end{cases}$$

Nous trouvons la solution optimale $x^1 = (0, -2)$, $(x^1) = -10$,

$g_1(x^1) = -4$, $g_2(x^1) = -7$ et $s_1 = 2$. donc on ajoute la contrainte

$$g_2(x^1) + (x - x^1)\nabla g_2(x^1) \geq 0 \qquad (**)$$

Après le calcul on trouve $(**) = 4x_1 + 4x_1 - 15 \geq 0$

On ajoute cette contrainte au problème (2) on obtient :

$$(3)\begin{cases} \min f(x) = 4x_1 + 5x_2 \\ sc \\ 2x_1 + 4x_2 + 9 \geq 0 \\ 4x_1 + 4x_1 - 15 \geq 0 \\ 0 \leq x_1 \leq 4 \\ -4 \leq x_2 \leq 4 \\ x_2 \in Z^n \\ x_2 \in Z^n \end{cases}$$

La solution optimale correspondante au problème (3) est $x_2 = (1, -1)$ avec

$f(x^0) = -1$, cette dernière solution est la solution du problème de départ (i)

Remarque : l'objectif de l'algorithme consiste à transformer un système d'équations non linéaires en un système d'équations linéaires. La contrainte non linéaire est remplacée par son hyperplan d'appui (la coupe) qui est une contrainte linéaire. Les variables de décision peuvent être continues, discrètes ou bien mixtes. Dans le cas où les variables de décision sont discrètes, on résout le problème par l'une des méthodes décrites au premier chapitre.

IV.4. La méthode Séparation et évaluation :[6]

IV.4.1. Introduction :

Un algorithme par séparation et évaluation, également appelé selon le terme anglo-saxon branch and bound, est une méthode générique de résolution de problèmes d'optimisation, et plus particulièrement d'optimisation combinatoire ou discrète. C'est une méthode d'énumération implicite: toutes les solutions possibles du problème peuvent être énumérées mais, l'analyse des propriétés du problème permet d'éviter l'énumération de larges classes de mauvaises solutions.

Dans un bon algorithme par séparation et évaluation, seules les solutions potentiellement bonnes sont donc énumérées.

IV.4.2. Principe de la méthode :

Soit S un ensemble fini mais de « grande » cardinalité qu'on appelle ensemble (ou espace) des solutions réalisables. On dispose d'une fonction f qui, pour toute solution réalisable x de S, renvoie un coût f(x). Le but du problème est de trouver la solution réalisable x de coût minimal.

D'un point de vue purement existentiel, le problème est trivial : une telle solution x existe bien car l'ensemble S est fini. En revanche, l'approche effective du problème se confronte à deux difficultés. La première est qu'il n'existe pas forcément un algorithme simple pour énumérer les éléments de S. La seconde est que le nombre de solutions réalisables est très grand, le temps de parcours est prohibitif exagéré (la complexité algorithmique est en général exponentielle). Ce qui signifie que l'énumération de toutes les solutions réalisable du problème posé en un temps raisonnable devient impossible.

D'où l'idée d'introduire des procédures .intelligentes qui évitent l'examen systématique de toutes les solutions admissible dites « procédure de séparation» et « procédure d'évaluation» La séparation permet d'obtenir une méthode générique pour énumérer toutes les solutions tandis que l'évaluation évite l'énumération systématique de toutes les solutions.

IV.4.2.1. Séparation

La phase de séparation consiste à diviser le problème en un certain nombre de sous problèmes qui ont chacun leur ensemble de solutions réalisables de telle sorte que tous ces ensembles forment un recouvrement (idéalement une partition) de l'ensemble S. Ainsi, en résolvant tous les sous problèmes et en prenant la meilleure solution trouvée, on est assuré d'avoir résolu le problème initial. Ce principe de séparation peut être appliqué de manière récursive à chacun des sous-ensembles de solutions obtenus, et ceci tant qu.il y a des ensembles contenant plusieurs solutions. Les ensembles de solutions (et leurs sous problèmes associés) ainsi construits ont une hiérarchie naturelle en arbre, souvent appelée arbre de recherche ou arbre de décision.

IV.4.2 .2. Évaluation

L'évaluation d'un nœud de l'arbre de recherche a pour but de déterminer la borne inférieure (respectivement supérieur pour un problème de maximisation) associé au nœud en question.

L'évaluation représente la valeur de la fonction objective au nœud considéré.

IV.4.2.3. Stérilisation

L'évaluation peut être au contraire une preuve, que cet ensemble ne contient pas de solution intéressante pour la résolution du problème (typiquement, qu'il n'y a pas de solution optimale lorsqu'un tel nœud est identifié dans l'arbre de recherche, il est donc inutile d'effectuer la séparation de son espace de solutions il est alors stérilisé.

Pour déterminer qu'un ensemble de solutions réalisables ne contient pas de solution optimale,

la méthode la plus générale consiste à déterminer une borne inférieure pour le coût des solutions contenues dans l'ensemble (s'il s'agit d'un problème de minimisation). Si on arrive à trouver une borne inférieure de coût supérieur au coût de la meilleure solution trouvée jusqu'à présent, on a alors l'assurance que le sous-ensemble ne contient pas l'optimum.

Cette méthode se termine quand tous les sommets sont soit terminaux ou éliminés, donc chaque sommet de l'arborescence sera classé dans un des trois états suivants :

1. Exploré : ses successeurs sont créés par séparation et sont évalués.
2. Actif : non encore exploré.
3. Eliminé son évaluation est supérieure à la valeur de la meilleure solution trouvée.

IV.4.3. Algorithme de séparation et évaluation :

Étape 1 : Résolution du PNL associe :

Résoudre le PNL associe, on calcule la valeur de Z (la valeur de la fonction objectif) qui donne la borne supérieure (Z_{BS}) au MINLP et la valeur de Z correspondante à la solution arrondie qui donne la borne inferieure ((Z_{BI}) au MINLP/

Étape 2 : séparation :

- Choisir une variable non entière pour générer deux sous problèmes ,soit x_k cette variable devant etre entiere dont la valeur optimale x_k^* est fractionnaire , représentons par $[x_k^*]$ la partie entière de x_k^*.

➢ Pour éliminer la solution non entière x_k on crée deux branches et deux sous problèmes. On obtient avec $x_k \leq [x_k^*]$ et l'autre avec $x_k \geq [x_k^*] + 1$.
Les deux sous problèmes sont obtenus en ajoutant au problème non linéaire] associé (PNL) précédant (celui d'où provient la séparation), la contrainte $x_k \leq [x_k^*]$ pour un sous problème et la contrainte $x_k \geq [x_k^*] + 1$ pour l'autre sous problème. Toutes les solutions entières réalisables sont maintenant contenues dans l'un ou l'autre sous problème.

Remarque :

- Une valeur entière réalisable ne peut être dans l'intervalle $[x_k^*] \leq x_k \leq [x_k^*] + 1$.
- Le critère de choix de variable de branchement est de prendre la variable la plus distante d'un entier.

Étape 3 : Résolution des sous problèmes et détermination de nouvelles bornes pour Z :

➢ Il s'agit de résoudre chaque sous problèmes (en ne tenant pas compte des contraintes de nombres entiers). Puisque ces deux sous problèmes comportent toutes les solutions au problème à résoudre, nous pouvons également en déduire de nouvelles bornes pour Z.
Notons par MBSD, la meilleure borne supérieure disponible. Elle correspond à la valeur maximale (minimale) de la fonction objectif des deux sous problèmes que nous venons de résoudre.

➢ Nous révisons également la borne inferieure de Z, notons cette valeur par MBID, la meilleure borne inferieure disponible elle correspond à la valeur maximale (minimale) da la fonction objective de toutes solutions entière obtenus jusqu'à présent (pour un MINLP).

Étape 4 : évaluation et stérilisation :

L'évaluation a pour but de déterminer le sous problème qu'on doit brancher (séparer), la stérilisation a pour but de déterminer si l'exploration d'une branche est terminée.
L'exploration d'une branche est terminée si l'une des conditions est satisfaite.

❖ Les sous problèmes de la branche considérée n'admettent pas de solution réalisable.

❖ La solution entière (ou mixte) obtenue pour le sous problème de la branche est réalisable mais la valeur de Z^* obtenue est plus petite ou égale à MBID ($Z^* \leq MBID$). Ainsi une branche est terminée aussitôt qu'on obtient une solution entière.

❖ Le sous problème de la branche considérée admet une solution optimale non entière mais sa valeur Z^* est moindre (ou égale) que celle d'un autre sous problème à solution entière.

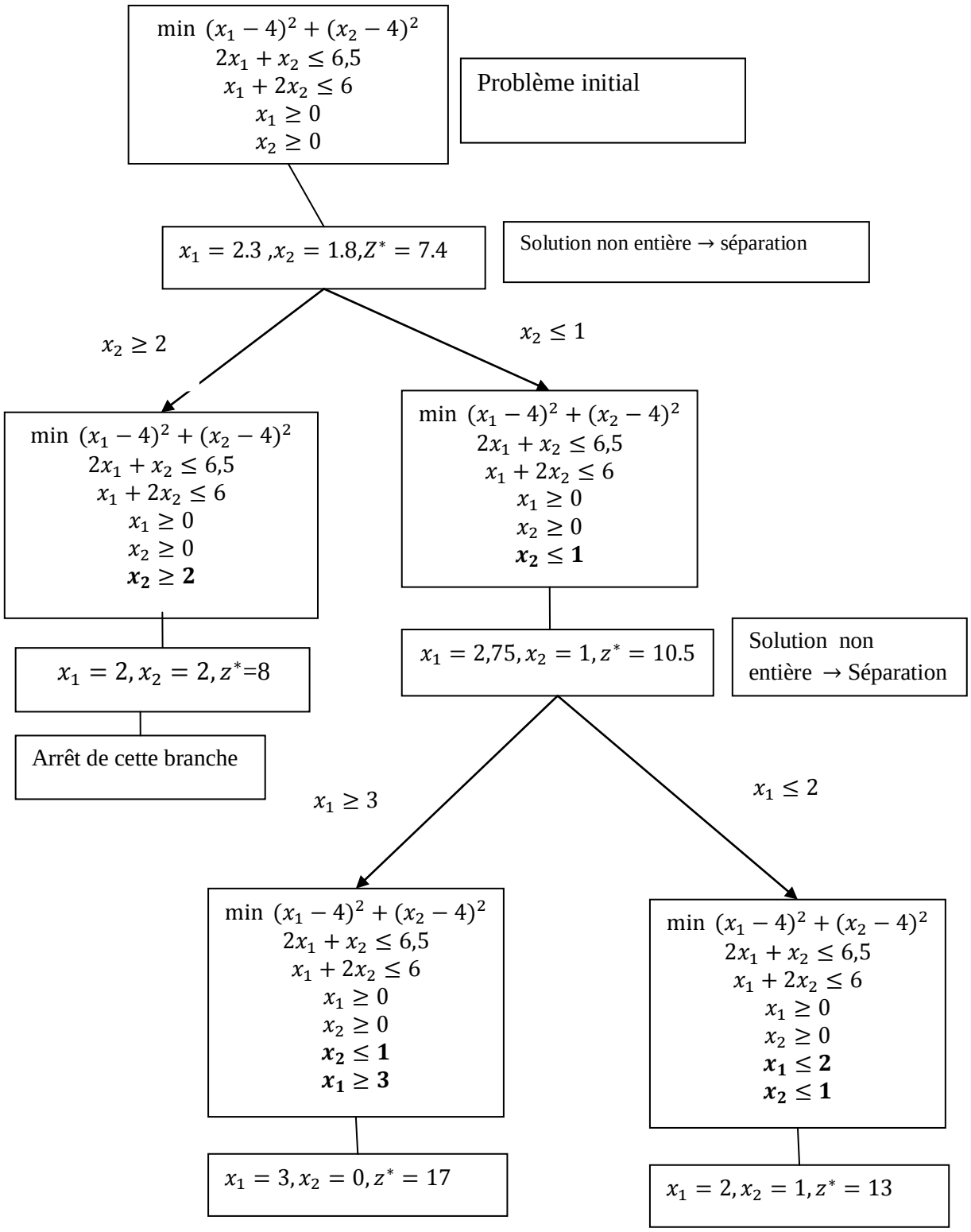
Remarque :

Si la solution optimale qu'on obtient à l'un ou l'autre des sous problèmes de chaque branche contient au moins une variable (devant être entière) ayant valeur fractionnaire, on poursuit l'étape de séparation à partir de sous problèmes ayant la valeur la plus élevé pour Z^* (c à d à partir de sous problème dont on a obtenue MBSD).

IV.4.4. Exemple numérique :

Pour illustrer cet algorithme on va résoudre le problème suivant :

$$\begin{gathered}\min\ (x_1-4)^2+(x_2-4)^2\\ 2x_1+x_2\leq 6{,}5\\ x_1+2x_2\leq 6\\ x_1\geq 0, x_2\geq 0\\ x_1, x_2\ \ entiers\end{gathered}$$



17 > 13 > 8 ⇒ **La solution entière optimale est : $x_1=2, x_2=2,\ z^*=8$**

I. Introduction :

Lingo est un logiciel utilisé pour résoudre des problèmes d'optimisation linéaires, entière et quadratique. Il aussi utilisé pour résoudre les modèles d'optimisations globales non linéaires.

Une des caractéristiques de Lingo c'est qu'il offre des outils qui peuvent aider à l'analyse des modelés utilisant la méthode de simplexe et la méthode de Brunch and Bound.

II. Description de l'logiciel :

LINGO utilise quatre solveurs pour résoudre différents types de programmes mathématiques :

- ✓ Solveur direct
- ✓ Solveur linéaire
- ✓ Solveur non linéaire
- ✓ Méthode de type séparation et évaluation

Le modèle mathématique reçu par LINGO doit respecter le langage de modélisation associé. Les données transmises au solveur peuvent se faire soit d'une manière directe, soit à travers des fichiers intermédiaires.

Durant l'exécution d'un programme, LINGO commence par l'utilisation de son solveur direct qui détermine les valeurs des différentes variables. Une fois que l'affectation des variables est terminée, il affiche un rapport à propos la solution.
S'il existe des variables indéterminées dans le modèle mathématique, LINGO détermine quels solveurs faut-il utiliser en examinant la structure du modèle à résoudre.

- Pour un modèle linéaire continu, LINGO appelle le solveur linéaire qui utilise la méthode du simplexe révisé avec une forme produit inverse.

- Si le modèle contient une ou plusieurs contraintes non linéaires, le solveur non linéaire est appelé. Ce dernier s'appuie sur les algorithmes de programmation linéaire successive et les algorithmes du gradient réduit. Dans le cas où le modèle contient des restrictions entières, un algorithme de type séparation et évaluation est invoqué.

- Les modèles entiers sont résolus en utilisant un algorithme de type séparation et évaluation ou de « découpe». Ces découpes augmentent considérablement le temps de résolution de la plupart des modèles de programmation entière.

Remarque : L'utilisateur peut choisir librement la stratégie de branchement et d'élagage ainsi que le nombre de solutions optimales qu'il souhaite obtenir. Ceci se fait en cliquant dans l'onglet LINGO/Options puis sur l'onglet « Integer-Solver » dans la boite de dialogue qui apparaît.

III. Environnement de travail de logiciel :

Le logiciel est reconnaissable grâce à l'icône de la figure 3.1.

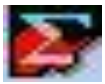


Figure 3.1

La figure 3.2 montre l'environnement de travail du logiciel.

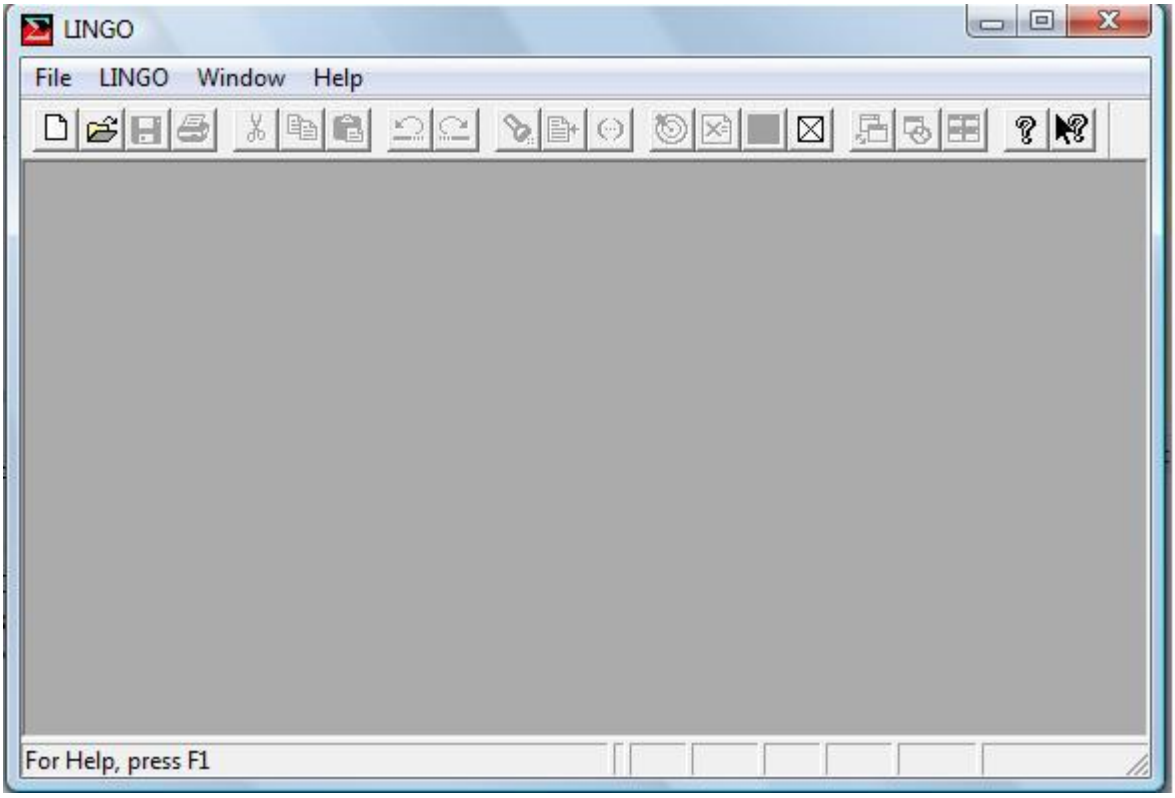


Figure 3.2 : Fenêtre principale.

IV. Programmation des exemples sur lingo :

IV.1. Programme linéaire en nombre entiers :

Exemple3.1: Soit à résoudre le problème linéaire en nombre entiers suivant

$$\max Z = 100x_1 + 120x_2$$
$$3x_1 + 4x_2 \leq 4100$$
$$x_1 + 3x_2 \leq 2400$$
$$2x_1 + 2x_2 \leq 2625$$
$$x_1, x_2 \in \mathbb{N}$$

En écrivant le modèle lingo de notre exemple sur la fenêtre principale, on aura la fenêtre représenté dans la figure 3.3 suivante :

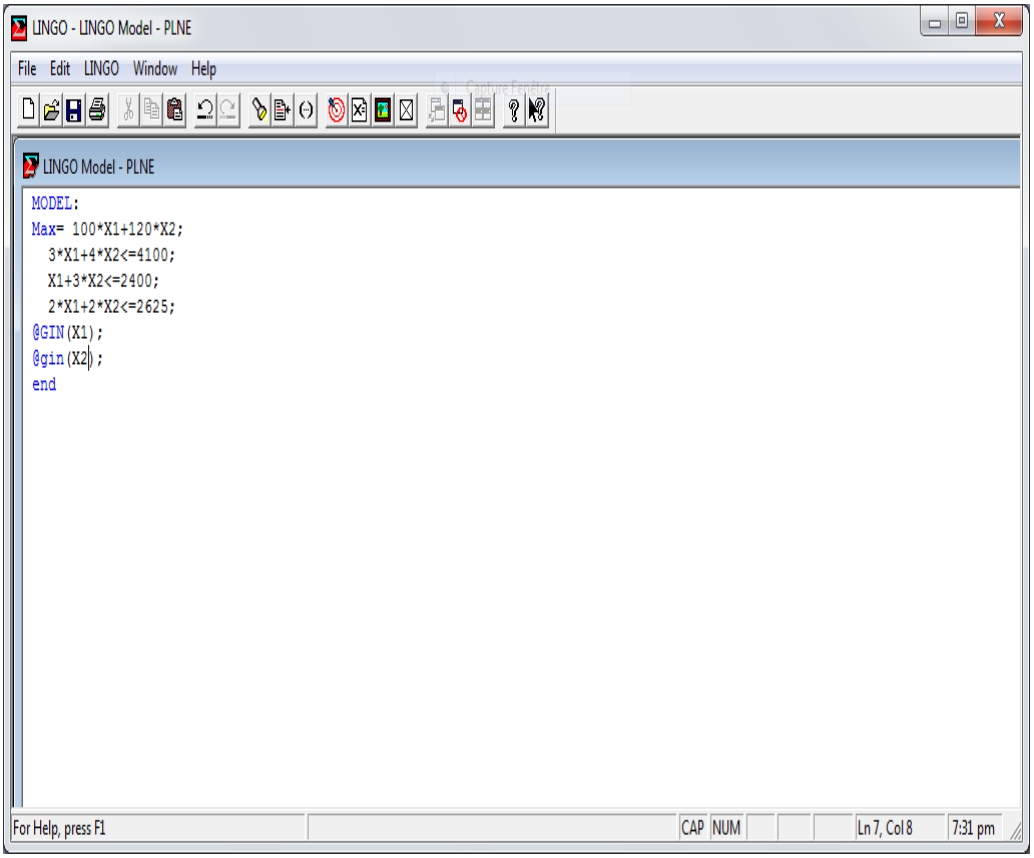


Figure 3.3

On passe à la résolution, pour résoudre notre programme il faut cliquer sur le bouton dans la barre d'outils (qui est représente dans la figure suivante) :

Figure 3.4 : bouton d'exécution.

Lingo va commencer ainsi à compiler le modèle, si un message d'erreur s'affiche c'est le programme, par exemple non borné ou non réalisable…

Lors de la compilation, on voit les deux figures suivantes :

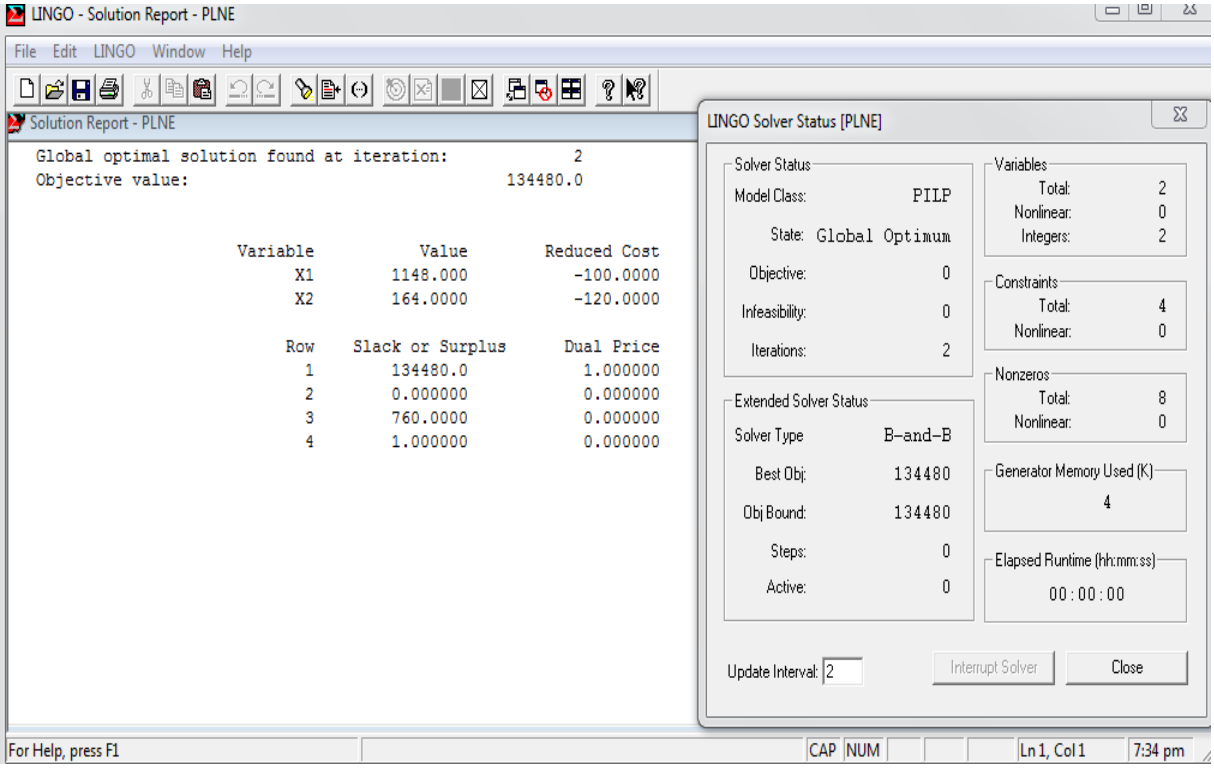


Figure 3 .5

IV.2. Programmation non linéaire continue :

Exemple3.2 : Soit le problème de programmation non linéaire continue suivant :

$$(\text{NLP}) \begin{cases} min\ x^2 + y^2 \\ x + y \leq 1 \\ x \geq 0, y \geq 0 \end{cases}$$

Le modèle d'exécution sur LINGO est le suivant:

```
MODEL:
MIN= X^2+Y^2;
     X+Y<+1;
     X>=0;
     Y>=0;
End
```

Les résultats de l'exécution de ce modèle est représenté dans la figure suivante :

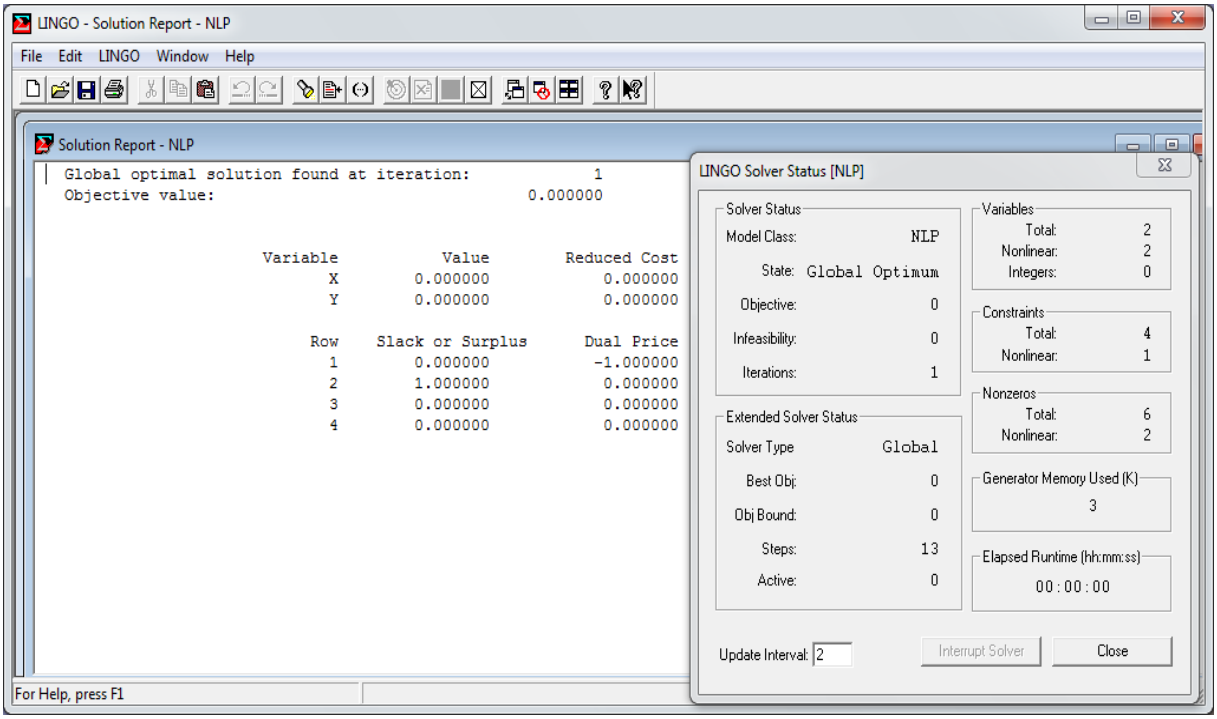


Figure3.6.

IV.3. Programme non linéaire en nombre entiers :

Exemple3.3 : On va résoudre le problème suivant **:**

$$\min f(x) = 4x_1 + 5x_2$$
$$sc$$
$$2x_1 + 4x_2 + 9 \geq 0$$
$$0 \leq x_1 \leq 4$$
$$-4 \leq x_2 \leq 4$$
$$x_1 \in Z^n$$
$$x_1 \in Z^n$$

Le modèle lingo correspondant a cet exemple est :

```
MIN= 4*X1+5*X2;
     -2*(X1)^2-2*X1*X2-2*(X2)^2+4>=0;
     -(X1)^2-(X2)^2+4*X1-3>=0;
      @gin(X1);
@bnd(0,X,4);
      @gin(X2);
 @bnd(-4,X2,4);
End
```

Après l'exécution de ce modèle on aura les résultats affiche dans la figure suivante :

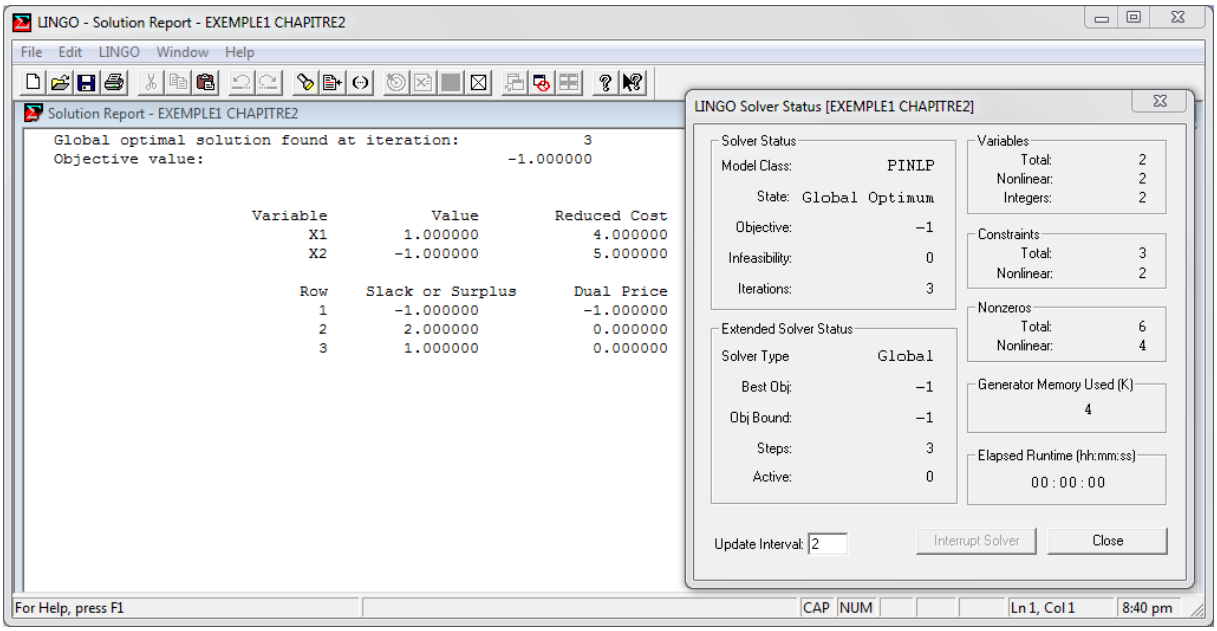


Figure3.7.

Exemple4.3 : Soit le problème suivant :

$$\min\ (x_1-4)^2+(x_2-4)^2$$
$$2x_1+x_2\le 6{,}5$$
$$x_1+2x_2\le 6$$
$$x_1\ge 0, x_2\ge 0$$
$$x_1, x_2\ \ entiers$$

Le modèle lingo correspondant à cet exemple est le suivant :

```
MODEL:
min=  (x1-4)^2+(x2-4)^2;
      2*x1+x2<=6.5;
      x1+2*x2<=6;
      x1>=0;
      x2>=0;

     @gin(x1);
     @gin(x2);
end
```

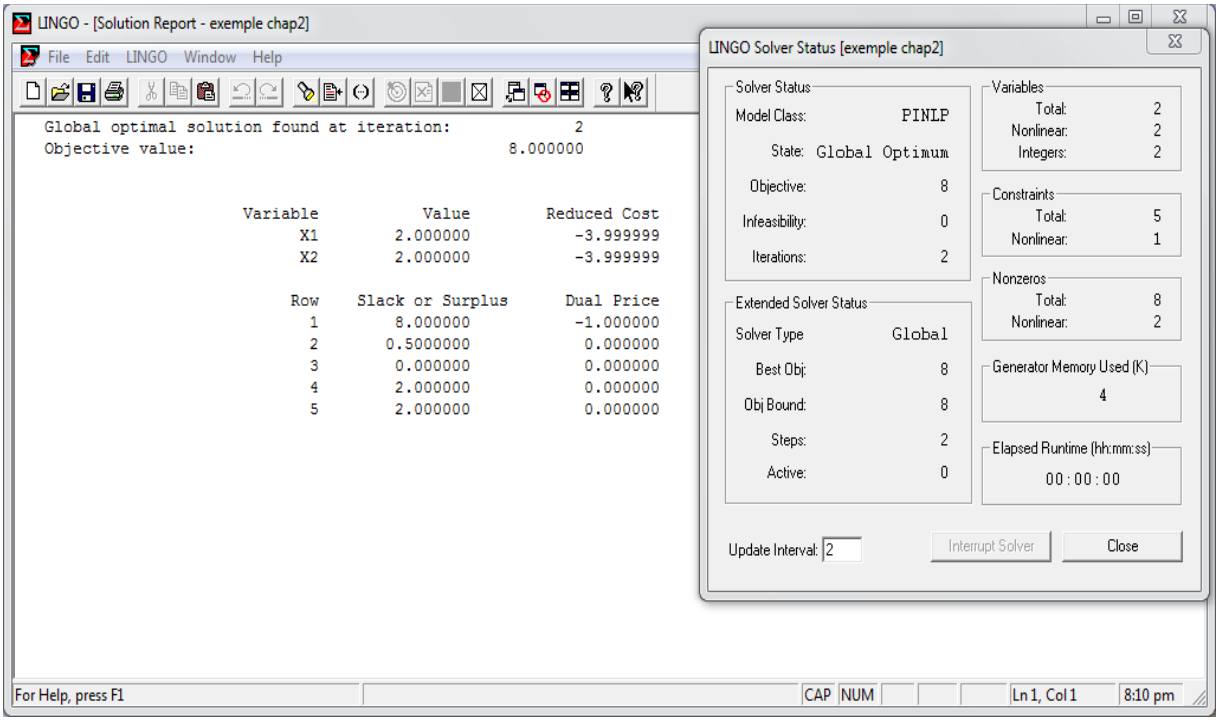

LINGO - [Solution Report - exemple chap2]
File Edit LINGO Window Help
Global optimal solution found at iteration: 2
Objective value: 8.000000
Variable Value Reduced Cost
X1 2.000000 -3.999999
X2 2.000000 -3.999999
Row Slack or Surplus Dual Price
1 8.000000 -1.000000
2 0.5000000 0.000000
3 0.000000 0.000000
4 2.000000 0.000000
5 2.000000 0.000000
LINGO Solver Status [exemple chap2]
Solver Status
Model Class: PINLP
State: Global Optimum
Objective: 8
Infeasibility: 0
Iterations: 2
Extended Solver Status
Solver Type Global
Best Obj: 8
Obj Bound: 8
Steps: 2
Active: 0
Variables
Total: 2
Nonlinear: 2
Integers: 2
Constraints
Total: 5
Nonlinear: 1
Nonzeros
Total: 8
Nonlinear: 2
Generator Memory Used (K)
4
Elapsed Runtime (hh:mm:ss)
00 : 00 : 00
Update Interval: 2
Interrupt Solver
Close
For Help, press F1
CAP NUM
Ln 1, Col 1
8:10 pm

Figure3.5.

Dans ce modeste travail, on a essayé de récapituler un peu sur l'optimisation discrète, tel que on a vu quelques algorithmes de résolution, soit dans les cas linéaire ou bien non linéaire, on peut conclure que l'optimisation discrète de manière générale est basée sur l'optimisation continue, vu l'utilisation successives des algorithmes de la méthode du simplexe, dual du simplexe et KKT, ce qui revient à augmenter les complexités des algorithmes applicables dans le cas discret.

Pour résoudre un problème discret non linéaire par l'un des algorithmes cité au chapitre II, on emploie toujours les algorithmes de résolution des problèmes discrets linéaire.

La méthode de brunch and bound et la méthode de KCG ont été utilises dans plusieurs domaines d'optimisation, comme l'optimisation combinatoire, l'optimisation semi infini, ainsi que l'optimisation globale.

Pour montrer l'efficacité des algorithmes présentés dans le chapitre I et II, on a traité quelques exemples numériques et la vérification des résultats a été faite sur le logiciel Lingo.

Pour finir, on peut dire que l'optimisation discrète d'une manière générale n'est pas une tache facile, surtout l'optimisation non linéaire et la difficulté revient à les variables qui sont discret et aussi à la contestation qu'il ya sur certains objectifs, les méthodes existantes ne résout pas ce problème à un seul coup, mais ils arrivent à déterminer l'ensemble de solutions efficaces, le choix de la solution reviendra au décideur.

Bibliographie

[1] Alain Billonnet, optimisation discrète, Dunod parid 2007.

[2] C.Guéret ,C.Prins, M.Sevaux.65 problèmes d'optimisation modélisés et résolus avec Visuale xpresse ,Programmation linéaire, éditions Eyrolles 2000.

[3] Chemini youba,rahim mohmed Optimisation des moyens logistique, mémoire d'ingénieure d'état en recherche opérationnelle 2008.

[4] F .Droebeke,M.Halline ,CL.Lefevere, Programmation linéaire par exemple.

Edition marketing 1983.

[5] FREDERICK S, HILLIER, Non linear Integer Programming, Series Editor, stanford University (2006).

[6] Gerald Baillargeon. Outil d'optimisation et l'aide à la décision, Programmation linéaire appliqué. Les éditions SMG.

[7] Horst-Pardalos,convexity,cnvex Relaxion, and globale optimisation, Ed AcadimicPublishers,handbook ,1984.

[8] Ignacio E. Grossmann, Review of Nonlinear Mixed-Integer and Disjunctive Programming Techniques, Chemical Engineering,Carnegie Mellon University,2002
[] Leslous Fadhila, contribution a l'optimisation semi infini linéaire et non linéaire, 2009.

[9]M.Aiden ,B.Oukacha ,Les manuels de l'étudiant Recherche opérationnelle, Programmation linéaires .Copright Eurl Page Bleues Interactionnelles, Maison d'édition pour l'enseignement et la formation,2005.

[10] Michelle Minoux, Programmation mathématique : théorie et algorithme,Lavoisier 2008.

[11] Mordecai Avriel, Nonlinear programming, Analysis and methods (1976), Series in Automatic compulation .

[12] Pierre Fougers, Rappele sur l'optimisation libre, Ed monréal paris 1986.

[13] Rachid ben zine, optimisation convexe, optimisation sans contraintes,Chemical Engineers journal 2007.

[14] Xavier antoine ; yannick Privat,introduction à l'optimisation :aspects theorique et algorithmes ;Ed Eyrommes ;2007.