République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université A. Mira de Bejaia

Faculté des Sciences Exactes

Département de Recherche Opérationnelle

Mémoire

En vue de l'obtention du diplôme de

Master en Mathématiques Appliquées

Option : Modélisation Mathématique et Evaluation des Performances des Réseaux

Thème

Méthode de Branch and Bound pour la résolution d'un problème de minimisation concave.

Réalisé par :

M^{elle} OUAMEUR Rima et M^{elle} ZAID Fatima

Devant le jury composé de :

Président	M^{r} Belkacem BRAHMI	M.C.A	U.A/Mira Béjaia.
Encadreur	M^{r} M.O BIBI	Professeur	U.A/Mira Béjaia.
Examinatrice	M^{elle} Zohra AOUDIA	M.A.A	U.A/Mira Béjaia.
Examinatrice	M^{elle} Karima BOUIBED	M.C.B	U.A/Mira Béjaia.

Béjaia 2020

Remerciements

Nous tenons à remercier :

Le bon Dieu de nous avoir donné la patience et la volonté pour accomplir ce travail. Nous remercions nos très chers parents pour leurs soutiens et leurs encouragements durant notre cycle d'étude.

Nos remerciements s'adressent également à :

Notre promoteur Mr Mohand Ouamer Bibi pour ses conseils, durant l'encadrement.

A Mademoiselle Aoudia Zohra pour son aide, ses orientations et pour nous avoir transmis les renseignements nécessaires à la réalisation de ce travail.

Nous remercions également :

Les membres de jury pour l'honneur qu'ils nous font en acceptant de lire et d'évaluer ce mémoire.

Enfin, nous remercions toute personne ayant contribué de près ou de loin à la réalisation de ce travail.

Dédicaces

Fatima

Je dédie ce mémoire de fin d'études

À **Mes parents** et à la mémoire des mes **grands parents**.

À Mon cher époux **Amrane Houssam** et toute sa famille.

À mes chers frères **Kamel,Lahcen,Ghiles,Tahar** et **Youcef**.

À mes chères soeurs **Nadia,Narimene,** et **Takfa**.

À mes copines de chambre **Sonia, Lynda, Hassiba** et **Chahinez**, pour leur aide, leur soutien, leur amour et surtout leur patience.

À tous ceux qui ont une relation de près ou de loin avec la réalisation de ce travail.

À toutes les personnes que j'aime.

Rima

Je dédie ce mémoire de fin d'études

À **Mes chers parents** pour leur aide, leur soutien, leur amour et surtout leur patience.

À la mémoire des mes **grands parents**.

À mon cher frère **Nabil**.

À ma chère soeur **Thoria** et ses enfants **Nassim et Ahmed**.

À Mon cher ami **O.Tarik** et toute sa famille.

À mes chères amies **Saloua** et **Lynda**.

À tous ceux qui ont une relation de près ou de loin avec la réalisation de ce travail.

À toutes les personnes que j'aime.

Table des matières

Table des figures

Introduction générale

La recherche opérationnelle (R.O) peut être définie comme l'ensemble des méthodes et techniques rationnelles orientées vers la recherche de la meilleure façon d'opérer des choix en vue d'aboutir au résultat visé ou au meilleur résultat possible.
Elle fait partie des méthodes d'aide à la décision dans la mesure où elle propose des modèles conceptuels en vue d'analyser et de maîtriser des situations complexes pour permettre aux décideurs de comprendre et d'évaluer les enjeux et d'arbitrer et/ou de faire les choix les plus efficaces.

La programmation mathématique est une branche des mathématiques appliquées ayant pour objet l'étude théorique des problèmes d'optimisation, ainsi que la conception et la mise en oeuvre des algorithmes de résolution.
La présence du terme "programmation" dans le nom donné à cette discipline peut s'expliquer historiquement par le fait que les premières recherches et les premières applications se sont développées dans le contexte de l'économie et de la recherche opérationnelle.

Une des caractéristiques de la recherche opérationnelle est qu'elle tente souvent de rechercher la meilleure solution (appelée solution optimale) pour le modèle représentant le problème auquel on est confronté. Cette «recherche d'optimalité» est un thème important dans la RO. Ce processus est résumé en un terme dit ”optimisation”.

L'objectif de ce mémoire est de présenter une méthode de résolution d'un problème de minimisation concave, qui est la méthode de " Séparation et Évaluation " (Branch and Bound Method).
Notre travail est composé de trois chapitres :

Le premier chapitre présente quelques définitions des éléments mathématiques de base qui peuvent nous permettre d'étudier la résolution des problèmes d'optimisation, tels que : fonction convexe et concave, enveloppe convexe, polyèdre, simplexe, point extrême, etc,ainsi des rappels sur les formes quadratiques et l'algèbre linéaire. Par la suite, nous présentons la méthode de Séparation et Évaluation, son principe, ses trois procédures essentielles et les trois stratégies de parcours, ainsi que l'algorithme général.

Dans le deuxième chapitre, nous nous intéressons à la résolution des problèmes de programmation linéaire en nombres entiers par la méthode de Séparation et Évaluation "Branch and Bound Method". Nous montrons d'abord, comment modéliser un problème de programmation linéaire, ensuite nous présentons ses méthodes de résolution, puis la méthode de Branch and Bound pour la résolution d'un problème de programmation linéaire en nombres entiers. À la fin, nous avons illustré cette méthode par deux exemples d'application.

Le dernier chapitre est consacré à la minimisation d'une fonction concave par la méthode de Séparation et Évaluation. Nous commençons par la représentation d'un problème de minimisation concave, ainsi que les propriétés et les résultats mathématiques essentiels dont on aura besoin pour traiter le cas de la minimisation d'une forme quadratique concave, ensuite nous présentons l'algorithme de la méthode de Branch and Bound appliquée à la minimisation concave, et nous finirons par l'appliquer sur un exemple, qui nous renseignera mieux sur l'efficacité de l'algorithme.

Chapitre 1

La méthode de séparation et d'évaluation (Branch and Bound Method)

1.1 Quelques rappels mathématiques

Définition 1.1.1 : Un ensemble $S \subset \mathbb{R}^n$ est dit convexe si

$$\forall\ x_1,\ x_2 \in S\ ,\ \forall\ \lambda \in [0,1] \Rightarrow z = \lambda\ x_1 + (\ 1 - \lambda)\ x_2 \in \mathrm{S}.$$

L'interprétation géométrique de cette définition est que pour deux points quelconques $x_1,\ x_2 \in S$, le segment de droite joignant ces deux points est entièrement inclus dans S.
En d'autres termes, l'ensemble convexe est caractérisé par la propriété qu'il contient toutes les combinaisons convexes de chaque paire de ses éléments.

Définition 1.1.2 : Soient les vecteurs $x_1,\ x_2,\ \ldots\ x_k$ de $\mathbb{R}^n$. On dira qu'un vecteur x de $\mathbb{R}^n$ est une combinaison convexe de $x_1,\ x_2,\ \ldots\ x_k$, s'il existe λ_1,$\lambda_2,\ \ldots\ \lambda_k \in \mathbb{R}$ tels que :

$$x = \sum_{j=1}^{k} \lambda_j x_j\ ,\ \sum_{j=1}^{k} \lambda_j = 1\ ,\ \lambda_j \geq 0\ ,\ j = 1...k\ .$$

Par exemple, la combinaison convexe de deux points est le segment joignant ces deux points, et la combinaison convexe de trois points est un triangle.
Un ensemble S est convexe si et seulement si toute combinaison convexe finie de points de S appartient à S.

Définitions 1.1.3 :

- **Fonction convexe :**

  Une fonction réelle f définie sur un ensemble convexe S de $\mathbb{R}^n$ est dite convexe si :

$$\forall x_1, x_2 \in S, \lambda \in [0,1], f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2).$$

Une fonction est dite strictement convexe si l'inégalité précédente est stricte, c'est à dire que :

$$\forall x_1, x_2 \in S, x_1 \neq x_2, \lambda \in]0,1[, f(\lambda x_1 + (1-\lambda)x_2) < \lambda f(x_1) + (1-\lambda) f(x_2).$$

Autrement dit, une fonction f dans $\mathbb{R}$ est convexe si et seulement si sa courbe représentative est entièrement située au-dessus de chacune de ses tangentes.

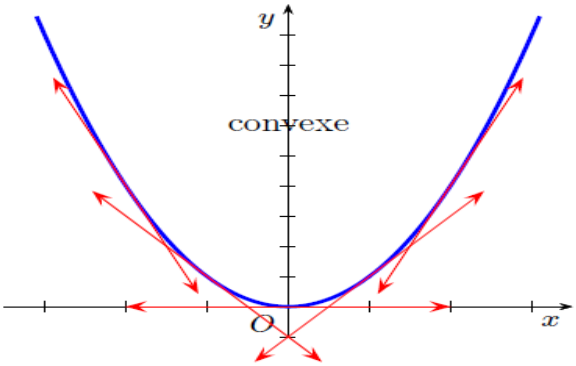


FIGURE 1.1 – fonction $x \longrightarrow x^2$.

- **Fonction concave :**

  Une fonction $f(x)$ est concave sur un ensemble convexe S si et seulement si la fonction $-f(x)$ est convexe sur S.

  Une fonction f, définie sur un ensemble convexe S de $\mathbb{R}^n$, est dite concave si :

$$\forall x_1, x_2 \in S, \lambda \in [0,1], f(\lambda x_1 + (1-\lambda)x_2) \geq \lambda f(x_1) + (1-\lambda) f(x_2).$$

Une fonction est dite strictement concave si l'inégalité précédente est stricte, c'est à dire que :

$$\forall x_1, x_2 \in S, x_1 \neq x_2, \lambda \in]0,1[, f(\lambda x_1 + (1-\lambda)x_2) > \lambda f(x_1) + (1-\lambda) f(x_2).$$

Autrement dit, une fonction f dans $\mathbb{R}$ est concave si sa courbe représentative est entièrement située au-dessous de chacune de ses tangentes.

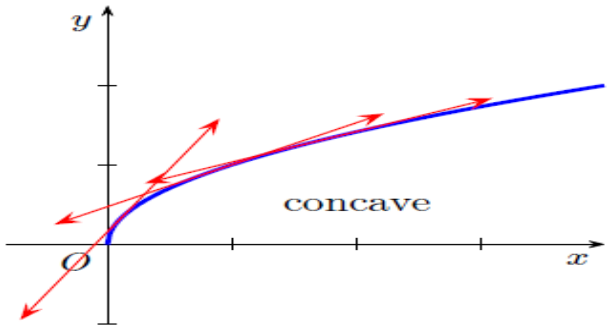


FIGURE 1.2 – fonction $x \longrightarrow \sqrt{x}$.

Lien avec la dérivée :

Soit f une fonction réelle, dérivable sur un intervalle $I \subset \mathbb{R}$, de dérivée f'. Alors :

- f est convexe sur I si et seulement si f' est croissante sur I ,
- f est concave sur I si et seulement si f' est décroissante sur I.

Conséquence :

Soit f une fonction réelle, deux fois dérivable sur un intervalle I :

- Si la dérivée seconde est positive, alors la fonction f est convexe ,
- Si la dérivée seconde est négative, alors la fonction f est concave.

x	a	x_0	b
$f''(x)$	$-$	0	$+$
f'	↘		↗
	concave		convexe

Définition 1.1.4 : Si $E = \mathbb{R}^n$ un espace vectoriel, et T une partie de E, l'enveloppe convexe de T est l'intersection de toutes les parties convexes contenant T. C'est elle-même un convexe, et c'est

le plus petit convexe contenant T. L'enveloppe convexe de T est noté $Co(T)$ ou simplement T^c.

Définition 1.1.5 : L'enveloppe convexe d'une fonction f noté $Co(f)$, prise sur un sous-ensemble non vide T de son domaine est une fonction g telle que :

1. g est une fonction convexe définie sur l'enveloppe convexe T^c de l'ensemble T,

2. $g(x) \leq f(x)$ pour tout $x \in T$,

3. Si h est une fonction convexe définie sur T^c, et si $h(x) \leq f(x)$ pour tout $x \in T$, alors $h(x) \leq g(x)$ pour tout $x \in T^c$.

Définition 1.1.6 : Soit un espace vectoriel $E = \mathbb{R}^n$ défini sur $\mathbb{R}$. Ses éléments sont des points de E.

- L'expression

$$\sum_{j=1}^{k} \lambda_j x^j$$

est dite combinaison linéaire des vecteurs $x^1, x^2, ..., x^k$, où $\lambda_j \in \mathbb{R}$.

- Les vecteurs $x^1, x^2, ..., x^k$ sont dits linéairement dépendants s'il existe k scalaires réels $\lambda_1, \lambda_2, ..., \lambda_k$, non tous nuls, tels que :

$$\lambda_1 x^1 + \lambda_2 x^2 + ... + \lambda_k x^k = 0.$$

Autrement, ils sont dits linéairement indépendants.

- La dimension de E est le nombre maximal de vecteurs linéairement indépendants dans E. Si E est de dimension n, une base de E est un ensemble de n vecteurs linéairement indépendants de E.

On appelle norme sur un espace vectoriel E défini sur $\mathbb{R}$, une fonction de E dans $\mathbb{R}^+$ qui, à tout $x \in E$, fait correspondre un scalaire non négatif, noté $\|x\|$, satisfaisant :

1. $\| x \| = 0$ si et seulement si $x = 0$,
2. $\| x + y \| \leq \| x \| + \| y \|$ pour tout $x, y \in E$,
3. $\| \alpha x \| = |\alpha| \, . \, \| x \|$, pour tout $\alpha \in \mathbb{R}$, $x \in E$.

Un espace vectoriel E muni d'une norme est dit **normé**.

Définition 1.1.7 : Dans un espace vectoriel normé $E = \mathbb{R}^n$, on appelle boule ouverte de centre x^0 et de rayon $r > 0$, l'ensemble

$$B_r(x^0) = \{x \in \mathbb{R}^n : \|x - x^0\| < r\}.$$

Dans $\mathbb{R}^2$ et $\mathbb{R}$, on emploie les mots disque ouvert et intervalle ouvert respectivement.

Définitions 1.1.8 :

- Soit S une partie de $\mathbb{R}^n$. Un point x est appelé point intérieur de S s'il existe $r > 0$ tel que $B_r(x) \subset S$. L'ensemble des points intérieurs à S est appelé intérieur de S, on le note $int(S) = S^0$.

- L'extérieur d'un sous-ensemble $S \subset E$ est l'intérieur du complément de S.

- Soit $p \in E$ et $S \subset E$. On dit que p est un point-frontière de S s'il n'est ni intérieur ni extérieur à S. Un point-frontière de S peut appartenir ou non à S.

- La frontière d'un sous-ensemble $S \subset E$ est l'ensemble des points-frontières.

Définition 1.1.9 : Soient a un vecteur non nul de $\mathbb{R}^n$ et α un scalaire réel. L'ensemble

$$H = \{x \in \mathbb{R}^n / a^t x = \alpha\}$$

est appelé un hyperplan.

Les ensembles

$$H^+ = \{x \in \mathbb{R}^n / a^t x \geq \alpha\}$$
$$H^- = \{x \in \mathbb{R}^n / a^t x \leq \alpha\}$$

sont appelés des demi-espaces, respectivement positif et négatif.

Définition 1.1.10 : Un polyèdre P est un ensemble convexe qui peut être décrit de la manière suivante :

$$P = \{x \in \mathbb{R}^n / Ax \leq b\},$$

où A est une matrice d'ordre $m \times n$ et b est un vecteur de $\mathbb{R}^m$.

Un ensemble de la forme

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

est un polyèdre sous forme standard.

Un polyèdre P est dit borné s'il existe une constante $c > 0$ telle que $\| x \| \leq c$ pour tout $x \in P$.

Un polyèdre borné est dit polytope.

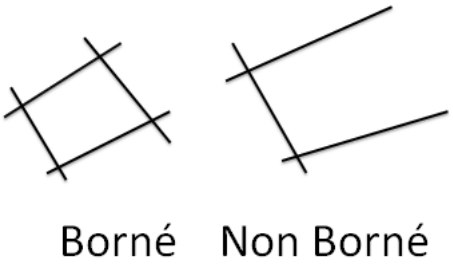


FIGURE 1.3 – Polyèdre borné (Polytope) et polyèdre non borné.

• Un point x d'un ensemble convexe S est dit extrême si x ne peut pas être écrit comme combinaison linéaire convexe de deux autres points différents de S.

Caractérisation d'un polytope

Un polytope est l'enveloppe convexe de ses points extrêmes.

Définition 1.1.11 : Soit S un ensemble compact dans $\mathbb{R}^n$, les sous-ensembles $\{S^1, S^2, ..., S^q\}$ sont dits être une partition de cet ensemble, si $\bigcup_{i=1}^{k} S^i = S$, avec $S^i \cap S^j = \emptyset$, $i \neq j$.

Définition 1.1.12 : On appelle r-simplexe (figure 1.4) un polytope de dimension r avec $(r+1)$ sommets.

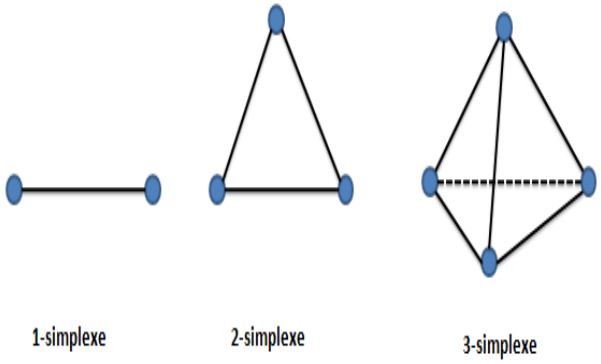


FIGURE 1.4 – Les simplexes.

Pour en savoir plus, on peut consulter les références suivantes :[1],[2].

1.2 Algèbre linéaire

1.2.1 Vecteurs et matrices

Définition 1.2.1[3] : Soit n,$m \in \mathbb{N}^*$. Une matrice d'ordre $m \times n$ à coefficients dans $\mathbb{R}$ est un tableau à deux dimensions, ayant m lignes et n colonnes, représenté sous la forme suivante :

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

où $I = \{1,2,...,m\}$ et $J = \{1,2,...,n\}$ représentent respectivement l'ensemble des indices des lignes et colonnes de A. Pour des calculs pratiques, la matrice A se note aussi :

$$A = (a_1, a_2, ..., a_j, ..., a_n) = \begin{pmatrix} A_1^t \\ A_2^t \\ . \\ A_i^t \\ . \\ A_m^t \end{pmatrix},$$

où

$$a_j = A(I, j) = \begin{pmatrix} a_{1j} \\ a_{2j} \\ . \\ a_{ij} \\ . \\ a_{mj} \end{pmatrix},$$

est un vecteur-colonne de dimension m, $A_i^t = A(i, J) = (a_{i1},\ a_{i2},\ .\ .\ .,a_{i2},\ .\ .\ .,\ a_{in}\ )$ est un vecteur-ligne de dimension n. Le symbole $(^t)$ est celui de la transposition. Chaque vecteur, noté $x = x(J) = (x_j, j \in J)$, sera ainsi considéré comme un vecteur-colonne, tandis que le vecteur-ligne sera noté x^t. La matrice transposée de A sera notée :

$$A^t = A^t(J, I) = (a_{ji} = a_{ij}, j \in J, i \in I).$$

Notons qu'un vecteur-colonne de dimension n peut être considéré comme une matrice d'ordre $(n \times 1)$, tandis qu'un vecteur-ligne de dimension n peut être considéré comme une matrice d'ordre $(1 \times n)$. La matrice A est dite carrée si on a $n = m$, de plus, si $A = A^t$, la matrice est dite symétrique. La matrice d'identité d'ordre n sera notée I_n.

1.2.2 Matrices et vecteurs partitionnés

Définition 1.2.2[3] : On peut effectuer le produit d'une matrice A et d'un vecteur x, après les avoir partitionnés judicieusement. On dit alors qu'on a effectué le produit par blocs. En effet, si l'on a :

$$A = [A_1|A_2],\ x = [\tfrac{x_1}{x_2}],$$

alors on peut écrire :

$$Ax = [A_1|A_2][\tfrac{x_1}{x_2}] = A_1x_1 + A_2x_2\ ,$$

de même, pour :

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_2 \end{pmatrix},$$

$$x = [\tfrac{x_1}{x_2}], \qquad b = [\tfrac{b_1}{b_2}],$$

l'équation $Ax = b$ peut alors s'écrire :

$$\begin{cases} A_{11}x_1 + A_{12}x_2 = b_1, \\ \\ A_{21}x_1 + A_{22}x_2 = b_2. \end{cases}$$

On peut partitionner une matrice d'une manière arbitraire. Par exemple, si A= $A(I, J)$ est une matrice d'ordre $(m \times n)$, J_B et J_N sont deux sous-ensembles quelconques de J, tels que :

$$|J_B| = m,\ J_B \cup J_N = J,\ J_B \cap J_N = \emptyset,$$

alors on peut partitionner A de la façon suivante :

$$A = (a_1, a_2, ..., a_j, ..., a_n) = [A_B|A_N],$$

avec $A_B = A(I, J_B)$= $(a_j, j \in J_B)$, et $A_N = A(I, J_N)$ =$(a_j, j \in J_N)$.

Si $x = x(J) = [\tfrac{x_B}{x_N}]$, avec $x_B = x(J_B)$, et $x_N = x(J_N)$, alors on peut écrire :

$$Ax = \sum_{j=1}^{n} a_jx_j = \sum_{j\in J_B} a_jx_j + \sum_{j\in J_N} a_jx_j = A(I, J_B)x(J_B) + A(I, J_N)x(J_N) = A_Bx_B + A_Nx_N.$$

1.3 Rappels sur les formes quadratiques

- **Propriétés des formes quadratiques [4],[5]**

  Une fonction $F : \mathbb{R}^n \longrightarrow \mathbb{R}$ est dite forme quadratique de n variables x_1, x_2, . . . , x_n, si elle s'écrit sous la forme suivante :

$$F(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_i x_j + \sum_{j=1}^{n} c_j x_j = x^t Ax + c^t x, \tag{1.1}$$

  où $x = (x_1,\ x_2,\ \ldots,\ x_n)^t$ et c sont des n-vecteurs colonnes, $A = (a_{ij}, 1 \leq i, j \leq n)$ est une matrice carrée d'ordre n. Le symbole $(^t)$ est l'opérateur de transposition vectorielle et matricielle.

  Pour $i \neq j$, le coefficient du terme $x_i x_j$ s'écrit $a_{ij} + a_{ji}$. En vertu de cela, la matrice A peut être supposée symétrique. En effet, en définissant des nouveaux coefficients

$$d_{ij} = d_{ji} = \tfrac{a_{ij}+a_{ji}}{2},\ 1 \leq i, j \leq n,$$

  on obtient une nouvelle matrice $D = (d_{ij},\ 1 \leq i, j \leq n)$ symétrique. Il est clair qu'après une redéfinition des coefficients, la valeur de la forme quadratique $F(x)$ reste inchangée :

$$F(x) = x^t Ax + c^t x = x^t Dx + c^t x, \forall x \in \mathbb{R}^n. \tag{1.2}$$

  Pour cela, il est naturel de considérer que la matrice d'une forme quadratique est toujours symétrique.

  Par exemple, écrivons la matrice symétrique associée à la forme quadratique suivante :

  $F(x) = 2x_1^2$ - $4x_2^2$ + $6x_1x_2$ + $8x_1x_3$ - $12x_2x_3$,

  donc on a :

$$D = \begin{pmatrix} 2 & 3 & 4 \\ 3 & -4 & -6 \\ 4 & -6 & 0 \end{pmatrix}.$$

- **Gradient d'une forme quadratique**

  Soit F :$\mathbb{R}^n \longrightarrow \mathbb{R}$ une forme quadratique définie par (1.2). Le gradient de F noté $\nabla F(x)$ au point x est égal à :

$$\nabla F(x) = \begin{pmatrix} \frac{\partial F}{\partial x_1} \\ \frac{\partial F}{\partial x_2} \\ \vdots \\ \frac{\partial F}{\partial x_n} \end{pmatrix} = 2Dx + c,$$

  où $\frac{\partial F}{\partial x_i}$ est la dérivée partielle de F d'ordre 1 par rapport à x_i.

  Soit f une fonction réelle de classe C^1, $f : \mathbb{R}^n \longrightarrow \mathbb{R}$. Le Hessien de la fonction f au point x est défini par :

$$\nabla^2 f(x) = (\nabla \frac{\partial f}{\partial x_1}, \nabla \frac{\partial f}{\partial x_2}, \ldots, \nabla \frac{\partial f}{\partial x_j}, \ldots, \nabla \frac{\partial f}{\partial x_n}) = \begin{pmatrix} \frac{\partial^2 f}{\partial^2 x_1} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial^2 x_2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial^2 x_n} \end{pmatrix}.$$

  Si les éléments de cette matrice sont continus, on dit alors que f est de classe C^2.

  Donc le Hessien de la forme quadratique (3.2) est $\nabla^2 F(x) = 2D$, $\forall x \in \mathbb{R}^n$.

- **Formes quadratiques définies et semi-définies positives ou négatives [4]**

  Soit $F(x) = x^t Dx$, une forme quadratique et D la matrice symétrique associée.
  $F(x)$ est dite définie positive si $x^t Dx > 0$, $\forall x \in \mathbb{R}^n$, $x \neq 0$. Elle est dite semi-définie positive si $x^t Dx \geq 0$, $\forall x \in \mathbb{R}^n$, et $\exists x \neq 0$ telque $x^t Dx = 0$. $F(x)$ est dite définie négative si $x^t Dx < 0$, $\forall x \in \mathbb{R}^n$, $x \neq 0$, Elle est dite semi-définie négative si $x^t Dx \leq 0$, $\forall x \in \mathbb{R}^n$, $\exists x \neq 0$ telque $x^t Dx = 0$.
  $F(x)$ est non-définie si $\exists x, y \in \mathbb{R}^n$, $x \neq 0$, $y \neq 0$ tels que : $x^t Dx > 0$ et $y^t Dy < 0$.
  Une matrice définie positive (semi-définie positive) se note $D > 0$ ($D \geq 0$).

- **Caractérisation des formes quadratiques**[4]

  Critère de Sylvester

  L'intérêt de ce critère est de caractériser une forme quadratique définie ou semi-définie. Pour cela, considérons la matrice symétrique suivante :

$$D = \begin{pmatrix} d_{11} & d_{12} & \dots & d_{1n} \\ d_{21} & d_{22} & \dots & d_{2n} \\ \vdots & \vdots & \dots & \vdots \\ d_{n1} & d_{n2} & \dots & d_{nn} \end{pmatrix}.$$

  Le mineur d'ordre p de la matrice D, formé des lignes i_1, i_2, . . . , i_p et des colonnes j_1, j_2, . . . , j_p, est désigné comme suit :

$$D\begin{pmatrix} i_1 & i_2 & \dots & i_p \\ j_1 & j_2 & \dots & j_p \end{pmatrix} = \begin{vmatrix} d_{i_1 j_1} & d_{i_1 j_2} & \dots & d_{i_1 j_p} \\ d_{i_2 j_1} & d_{i_2 j_2} & \dots & d_{i_2 j_p} \\ \vdots & \vdots & \dots & \vdots \\ d_{i_p j_1} & d_{i_p j_2} & \dots & d_{i_p j_p} \end{vmatrix}, 1 \leq p \leq n.$$

  On appelle mineur principal d'ordre p, le mineur formé des lignes et des colonnes de D portant le même numéro,i.e, $i_1 = j_1$, $i_2 = j_2$, . . . , $i_p = j_p$.
  On appelle mineur successif d'ordre p, le mineur formé des p premières lignes et colonnes de D.

  Le critère de Sylvester se formule comme suit :
  1) Une matrice symétrique D est définie positive si et seulement si tous ses mineurs principaux successifs sont strictement positifs.

  2) Une matrice symétrique D est semi-définie positive si et seulement si tous ses mineurs principaux sont non négatifs.

3) Une matrice symétrique D est définie négative si (-D) est définie positive.

4) Une matrice symétrique D est semi-définie négative si (-D) est semi-définie positive.

Remarque

- Le critère de Sylvester n'est valable que pour les matrices symétriques.

Théorème

Soit D une matrice symétrique d'ordre n. On note par λ_i, $i = 1, ..., n$, ses valeurs propres qui sont réelles. On a alors les équivalences suivantes :

1) $D \geq 0 \Leftrightarrow \lambda_i \geq 0$, $i = 1, ..., n$,

2) $D > 0 \Leftrightarrow \lambda_i > 0$, $i = 1, ..., n$,

3) D est non-définie si seulement si elle a au moins deux valeurs propres non nulles de signe différent.

Propriétés

Soit $F(x) = x^t Dx$ une forme quadratique

- F est convexe si et seulement si D est semi-définie positive.

- F est strictement convexe si et seulement si D est définie positive.

- F est concave si et seulement si D est semi-définie négative.

1.4 Optimisation locale et globale

L'optimisation mathématique consiste à rechercher dans un domaine admissible une solution qui optimise une fonction objectif.

Pour un domaine continu, on distingue classiquement deux types d'optimisation [5][6] :

1.4.1 Optimisation locale

Recherche une solution qui est la meilleure localement, c'est-à-dire que dans son voisinage aucune solution n'est meilleure qu'elle. Cette solution est appelée un optimum local.
Rappelons qu'un voisinage $V(x^*, r)$ d'un point x^* de $\mathbb{R}^n$, est défini comme une boule ouverte.

1.4.2 Optimisation globale

Recherche la meilleure solution du domaine admissible, c'est-à-dire que dans tout ce domaine il n'existe aucune solution qui lui soit meilleure. Cette solution est appelée l'optimum global. Par définition, l'optimum global est aussi une solution locale. L'optimisation globale a un intérêt plus important par rapport à l'optimisation locale, elle garantit le fait qu'on ne peut trouver une solution meilleure que celle trouvée.

La figure suivante (Figure 1.5) représente une courbe représentant les optimums locaux et les optimums globaux d'une fonction objectif.

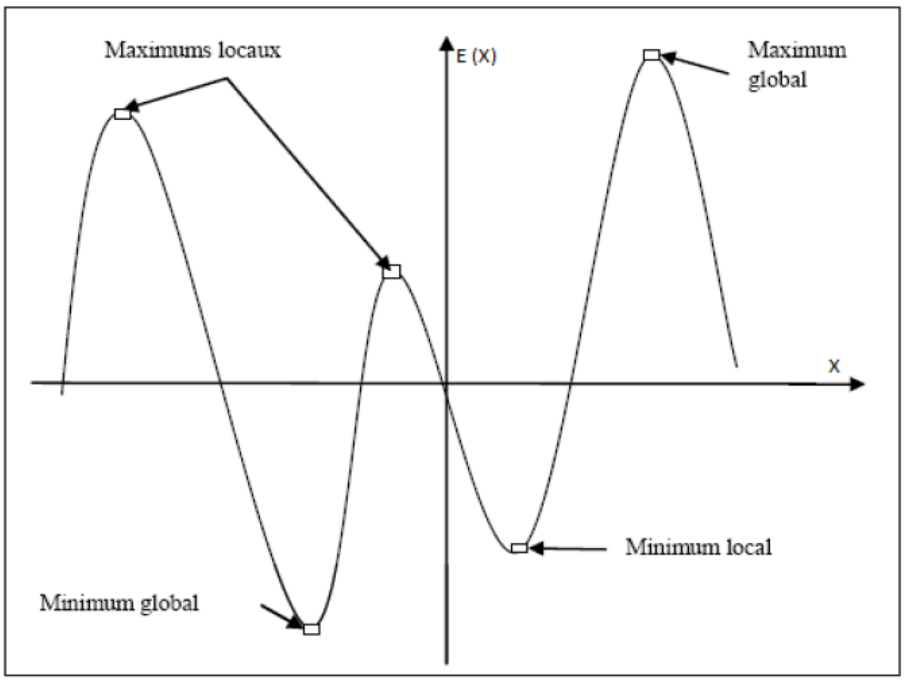


FIGURE 1.5 – Courbe représentant les optimums locaux et les optimums globaux.

1.5 La méthode de Branch and Bound

Pour plusieurs problèmes d'optimisation, en particulier les problèmes combinatoires, l'espace des solutions réalisables est fini (dénombrable). Il est donc possible en principe d'énumérer toutes les solutions et ensuite de prendre la meilleure. L'inconvénient majeur de cette approche est le temps de calcul qui est en général énorme, lorsque le nombre de ces solutions est exponentiel.
Nous allons montrer comment construire des approches efficaces qui font beaucoup mieux que l'énumération exhaustive de toutes les solutions réalisables. C'est le cas de la méthode de séparation et d'évaluation (Branch and Bound Method)[7].

La méthode de Branch and Bound est un algorithme assez général qui joue un rôle très important dans la théorie de l'optimisation combinatoire et globale. L'idée générale de la méthode est de décomposer le problème primaire en sous-problèmes parallèles (Branching) qui peuvent être graduellement plus faciles à résoudre, puis évaluer les bornes inférieures et supérieures (Bounding) des valeurs des solutions optimales sur ces problèmes secondaires.

L'énumération des solutions du problème P consiste à construire un arbre Branch and Bound dont les noeuds sont des sous-ensembles de solutions du problème P, et les branches sont les nouvelles contraintes à respecter. La taille de l'arbre dépend de la stratégie utilisée pour la construire. Pour ce faire, cette méthode se dote d'une fonction qui permet de mettre une borne inférieure (en cas de minimisation) ou une borne supérieure(en cas de maximisation) sur certaines solutions pour, soit les exclure, soit les maintenir comme des solutions réalisables.

Notons que la méthode de Branch and Bound a été à l'origine développée pour résoudre des problèmes de programmation linéaire en nombres entiers. Plus tard, cet algorithme a été appliqué avec succès dans des problèmes très difficiles en optimisation globale comme la minimisation concave et la minimisation de la différence de deux fonctions convexes[8].

1.5.1 Arborescence des solutions réalisables

Soit S l'ensemble de solutions réalisables du problème initial P. On le suppose discret, fini mais très grand. On énumère tous les éléments de S en le scindant en sous-ensembles non vides S^i (sous-problèmes P_i) contenant chacun une partie, de taille variable, des éléments de S (on n'impose pas toujours que ces sous-ensembles soient disjoints, mais c'est en général le cas). On peut recommencer avec chaque sous-ensemble qui contient plus d'un élément, et ainsi de suite jusqu'à ce que tous les ensembles ne contiennent plus qu'un seul élément. Cette énumération peut se représenter par un arbre de la façon suivante : la racine de l'arbre représente le problème initial P, les noeuds-fils représentent les sous-problèmes P_i créés à partir de la première partition de P, et ainsi de suite.

Les feuilles de l'arbre représentent les éléments de l'ensemble, c'est-à-dire les solutions au problème P. Si on a effectué une partition des ensembles à chaque fois, chaque ensemble n'est présent qu'une seule fois.

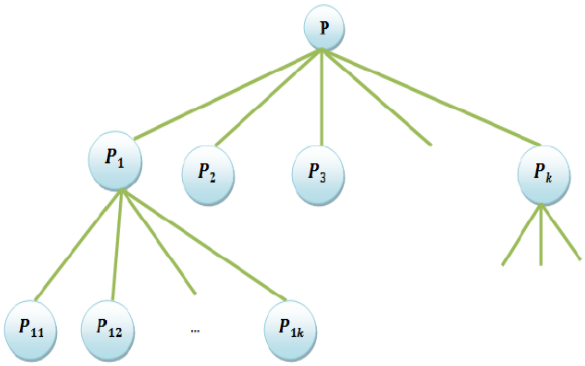


FIGURE 1.6 – Représentation par un arbre d'une énumération totale des solutions du problème.

1.5.2 Principe de la méthode

La méthode de séparation et évaluation (Branch and Bound Method) consiste à énumérer implicitement toutes les solutions dans S (l'espace de solutions) en examinant les sous-ensembles de S[9].

Il s'agit essentiellement de diviser l'ensemble de toutes les solutions réalisables S (problème initial) en sous-ensembles plus petits S^i, $i = 1...k$ (sous problèmes) en veillant à ce que :

$$\bigcup_{i=1}^{k} S^i = S,$$

c'est la phase « séparation » (Branching). Puis divers critères sont utilisés pour identifier les sous-ensembles qui peuvent contenir la solution optimale et les sous-ensembles qui ne doivent pas être explorés plus à fond, car ils ne peuvent pas contenir la solution optimale, c'est la phase « évaluation » (Bounding).

Soit (P) le problème d'optimisation globale :

$$(P) \begin{cases} minf(x), \\ x \in S, \end{cases}$$

où S est un compact de $\mathbb{R}^n$,

$f : S \longrightarrow \mathbb{R}$ $(S \subseteq \mathbb{R}^n)$, f continue et non convexe.

L'algorithme de Branch and Bound consiste à engendrer deux suites convergentes $\{UB_k\}$ et $\{LB_k\}$ des bornes supérieure et inférieure respectivement de la valeur minimale de la fonction f du problème (P), avec les abréviations

UB : Upper bound, LB : Lower bound

Une relaxation initiale R de l'ensemble réalisable S sera définie telle que : $S \subset R \subset \mathbb{R}^n$, où R est convexe, et peut être un simplexe, un rectangle, etc. À chaque itération k, les problèmes des bornes inférieure et supérieure seront résolus sur un nombre fini de sous-ensembles de R. On notera ces sous-ensembles $R_{ki} \in I_k$, où I_k est l'ensemble des sous-ensembles actifs à l'itération k. Sur chaque sous-ensemble R_{ki}, les bornes inférieure et supérieure LB_k et UB_k seront calculées par la relaxation de f sur R_{ki} et la relaxation de minimum de f localement sur le sous-ensemble réalisable $R_{ki} \cap S$ respectivement.

En effet, les bornes inférieure et supérieure finales pour l'itération k seront données par :

$$\begin{cases} LB_k = min\{LB_{ki}\} \\ UB_k = min\{UB_{ki}\} \end{cases}$$

respectivement, et tout sous-ensemble sur lequel la borne inférieure dépasse la borne supérieure sera éliminé, car le minimum de f ne peut être atteint sur un tel sous-ensemble.

En effet, cette méthode peut se représenter schématiquement par une arborescence qui a pour racine l'ensemble R, et pour sommets les sous-ensembles R_{ki} qui s'obtiennent par les subdivisions successives, et à chaque niveau de l'arborescence créé, les bornes inférieure et supérieure seront obtenues par l'application d'une recherche locale[10],[11].

Notons x^* la solution optimale du problème (P) pour ce qui suit.

Une méthode de Branch and Bound est basée sur trois axes principaux :

• La procédure de séparation ou bien le branchement, qui consiste à partitionner un ensemble de solutions en sous-ensembles (branching rule),

• La procédure d'évaluation, permettant le calcul d'une borne pour un ensemble de solutions (evaluation function),

• La stratégie de parcours ou procédure de cheminement d'exploration de l'arborescence de recherche.

1. **La procédure de séparation**

   La séparation consiste à diviser le problème en un certain nombre de sous-problèmes qui ont chacun leur ensemble de solutions réalisables. En résolvant tous les sous-problèmes et en prenant la meilleure solution trouvée, on est assuré d'avoir résolu le problème initial. Cela revient à décrire comment construire l'arbre permettant d'énumérer toutes les solutions.

   L'ensemble de noeuds de l'arbre qu'il reste encore à parcourir comme étant susceptibles de contenir une solution optimale, c'est-à-dire encore à diviser, est appelé ensemble des noeuds actifs.

   Son principe doit satisfaire aux trois règles suivantes :

   a)La règle de finitude :le nombre total de noeuds engendrés doit être fini.

   b)La règle de conservation : aucune solution d'un sous-problème ne peut être éliminée par la séparation, c'est-à-dire :

$$\bigcup_{k=1}^{p} S^{(ik)} = S^i,$$

   et qu'il vérifie également $S^{(ik)} \bigcap S^{(il)} = \emptyset$, $k \neq l$, où $S^{(ik)}$, $k = 1...p$, représentent les sous-ensembles (enfants) du sous-ensemble (parent) $S^{(i)}$.

c)La Règle d'arrêt : Un noeud ou sous-ensemble terminal de l'arborescence noté $S^{(t)}$ est défini comme un noeud qu'il n'est plus possible de séparer, lorsque :

- Soit $S^{(t)} = \emptyset$, où S^t est un singleton.

- Soit qu'il est possible de déterminer une solution optimale du problème P_i qui est le sous-problème réduit de (P) et qui est défini comme suit :

$$P_i \begin{cases} minf(x), \\ x \in S^{(i)}. \end{cases}$$

Exemple

Si $S = \{0,1\}^3$, on peut construire l'énumération suivante : d'abord on divise S en deux ensembles :

$S_0 = \{x \in S\,;\, x_1 = 0\}$, $S_1 = \{x \in S\,;\, x_1 = 1\}$, ensuite

$S_{00} = \{x \in S_0\,;\, x_2 = 0\}$, $S_{01} = \{x \in S_0\,;\, x_2 = 1\}$ et ainsi de suite...(figure 1.7)[12].

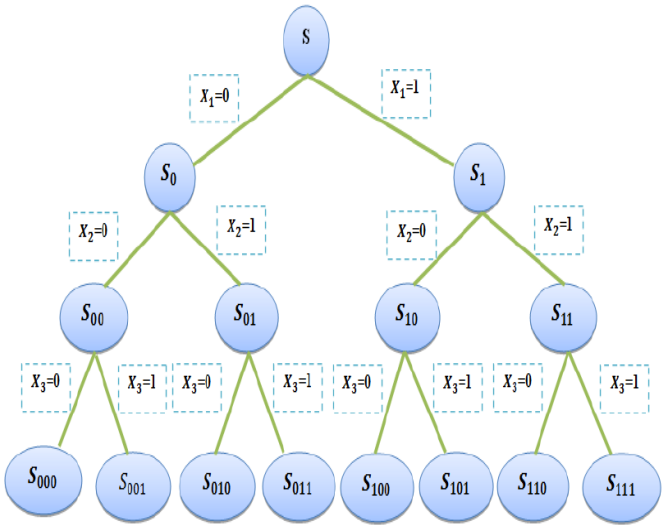


FIGURE 1.7 – Décomposition d'un problème binaire.

2. **La procédure d'évaluation :**

L'évaluation d'un sous-problème (noeud) (P_i) de (P) consiste à évaluer la valeur optimale de sa fonction de coût, c'est-à-dire à déterminer l'optimum de l'ensemble des solutions réalisables associées au noeud ou, au contraire, de prouver mathématiquement que cet ensemble

ne contient pas de solution intéressante pour la résolution du problème. Lorsqu'un tel noeud est identifié dans l'arbre de recherche, il est inutile d'effectuer la séparation de son espace de solutions.

On peut distinguer pendant le déroulement de l'algorithme trois types de noeuds dans l'arbre de recherche : le noeud courant qui est le noeud en cours d'évaluation, des noeuds actifs qui sont dans la liste des noeuds qui doivent être traités, et des noeuds inactifs qui ont été élagués au cours du calcul.

La procédure d'évaluation consiste à analyser un noeud (sous-problème), cette analyse vise à évaluer la valeur optimale de la fonction objectif du sous-problème, plus précisément, on doit déterminer une borne inférieure puisque la fonction de coût est à minimiser. L'évaluation permet de réduire l'espace de recherche en éliminant quelques sous-ensembles qui ne contiennent pas la solution optimale, et elle a aussi pour but de déterminer le sous-problème suivant qu'on doit séparer.

L'exploration d'une branche est éliminée si :

a) Le sommet de l'arborescence ne peut être séparé.

b) Le sous-problème n'admet pas de solution.

c) La valeur de Z_i (la valeur de la fonction objectif associée au sous-problème (P_i)) est supérieur à $Z^{(opt)}$ (une borne inférieure du sous-problème (P_i)), si le problème est à minimiser[7].

3. **La procédure de cheminement :**

   Cette procédure indique le sous-ensemble à analyser et dans quel ordre.

   Bien évidemment, il est souhaitable d'examiner le moins possible de sous-problèmes selon la stratégie choisie, certains d'entre eux pourront ne pas être séparés car, par exemple, leur analyse mettra en évidence qu'ils ne contiennent pas de solutions meilleures que celles déjà trouvées, nous dirons qu'un tel sous-ensemble ou le noeud correspondant de l'arborescence est sondé. C'est parce que certains sous-ensembles de solutions ne devront pas être examinés explicitement, que la méthode Branch and Bound est parfois appelée méthode d'énumération implicite.

   Lorsqu'un noeud de l'arborescence est sondé, il conviendra de remonter dans l'arborescence vers un autre noeud situé à un niveau inférieur ou égal[7].

Il est évident d'examiner la totalité des sommets de l'arborescence pour réaliser une énumération implicite efficace. Pour savoir quel sommet doit-on séparer, on utilise des stratégies qui sont au nombre de trois[7] :

- **La largeur d'abord :** Cette stratégie favorise les sommets les plus proches de la racine (noeud père) en faisant moins de séparations du problème initial. Mais elle est moins efficace que les deux autres stratégies.

- **La profondeur d'abord :** Cette stratégie avantage les sommets les plus éloignés de la racine (de profondeur la plus élevée) en appliquant plus de séparations au problème initial. Cette voie mène rapidement à une solution optimale en économisant la mémoire.

- **Le meilleur d'abord :** Cette stratégie consiste à explorer les sous-problèmes possédant la meilleure borne. Elle permet ainssi d'éviter l'exploration de tous les sous-problèmes qui possèdent une mauvaise évaluation par rapport à la valeur optimale.

1.5.3 L'algorithme de base de la méthode Branch and Bound

On peut résumer la procédure précédente par les étapes suivantes [13] :

1. Construire l'ensemble R tel que : $S \subset R$,

2. Posons : $k = 1$, $I_k = R$, fixer $\varepsilon > 0$,

3. Construire les problèmes des bornes inférieure et supérieure de min $f(x)$ sur R. Soient LB_k, UB_k les solutions obtenues respectivement,

4. Si : UB_k - $LB_k \leq \varepsilon$, donc on s'arrête et on pose :
min $f(x) = UB_k$ et $x^* = x^k \in \{\ x : f(x) = UB_k,\ x \in S \cap R\ \}$,

5. Sinon, subdiviser I_k en deux sous-ensembles (ou en un nombre fini de sous-ensembles) R_{k1} et R_{k2} tels que :

$$\bigsqcup_{i=1}^{i=2} R_{ki} = R \text{ et } R_{k1}^0 \cap R_{k2}^0 = \emptyset,$$

où R^0 est l'intérieur de R.

6. Construire les problèmes des bornes inférieure et supérieure de min $f(x)$ sur $S \cap R_{ki}$, $i = 1, 2$. Soient LB_{k1} , UB_{k1} et LB_{k2} , UB_{k2} les solutions obtenues,

7. Posons :

$$\begin{cases} UB_{k+1} = min\{UB_{k1}, UB_{k2}\} = UB_{k^*}, \\ LB_{k+1} = min\{LB_{k1}, LB_{k2}\} = LB_{k^*}, \end{cases}$$

8. Posons : $I_k = \{R_{k1}, R_{k2}\}$,

9. Eliminer de I_k tout sous-ensemble R_{kj} , $j = 1, 2$, tel que :

$$LB_{kj} > UB_{k+1}, \text{ où } S \cap R_{kj} = \emptyset,$$

et posons : $I_{k+1} = R_{ki^*}$,

10. Posons : $k = k + 1$ et revenons à **4 - 10**.

Notation

R_k : Le sous-ensemble actuel ; LB_k : La borne inférieure (à la $k^{ième}$ itération) ; UB_k : La borne supérieure (à la $k^{ième}$ itération) ; et x^k : La solution trouvée (à la $k^{ième}$ itération).

Chapitre 2

Application de la méthode Branch and Bound à la Programmation Linéaire en Nombres Entiers (PLNE)

La programmation linéaire (PL) est un outil auquel on fait souvent appel dans de nombreux problèmes théoriques et pratiques. Dans le présent travail, nous présentons la programmation linéaire en nombres entiers qui sera abrégée en PLNE dans la suite.

De nombreux problèmes peuvent être modélisés sous forme d'un problème d'optimisation dans lequel on cherche les valeurs des variables de décision pour lesquelles la fonction objectif prend une valeur maximale (ou minimale), ces variables étant soumises à un ensemble de contraintes. En pratique, on est constamment amené à traiter des problèmes de PL, où certaines variables sont astreintes à prendre des valeurs entières, on parle alors de programmation linéaire mixte. Si toutes les variables sont à valeurs entières, on a un problème de programmation linéaire en nombres entiers (PLNE). En particulier, les variables peuvent être simplement booléennes, c'est-à-dire ne prendre que les valeurs 0 ou 1. De nombreuses contraintes, en apparence non linéaires, peuvent être linéarisées grâce à des variables entières . Ces possibilités augmentent énormément le champ d'application de la programmation linéaire. Même si les programmes obtenus sont souvent difficiles à résoudre, la PLNE est déjà très utile comme outil de modélisation [14].

Ce chapitre est consacré à la programmation linéaire en nombres entiers. Nous mettons également l'accent sur une des principales méthodes de résolution de ces problèmes, plus précisément

la méthode de Branch-and-Bound, appelée en français : méthode de séparation et d'évaluation.

2.1 Modélisation d'un programme linéaire

La formalisation d'un programme est une tâche délicate mais essentielle, car elle conditionne la découverte ultérieure de la bonne solution. Elle comporte les mêmes phases quelles que soient les techniques requises ultérieurement pour le traitement [15].
Pour résoudre un problème d'optimisation, il faut assurer :
✓ La définition de l'ensemble des solutions réalisables, noté généralement par $S(x)$,

✓ L'expression de l'objectif à optimiser (maximiser ou minimiser une fonction, généralement notée par $f(x)$),
✓ Le choix de la méthode d'optimisation à utiliser (méthode de résolution de notre problème).

Les deux premiers points relèvent de la modélisation du problème, le troisième point de sa résolution. Afin de définir l'ensemble des solutions réalisables, il est nécessaire d'exprimer l'ensemble des contraintes du problème. Ceci ne peut être fait qu'avec une bonne connaissance du problème sous étude et de son domaine d'application. Lors de la modélisation d'un problème de programmation linéaire, on est amené à déterminer trois éléments importants [7] :

1. Les variables de décision :
C'est la première étape dans le processus de modélisation qui consiste à identifier correctement toutes les variables de décision (inconnues) du problème à modéliser. Ces variables doivent correspondre exactement aux préoccupations du responsable de la décision.

2. Les contraintes :
Dans la problématique posée, il faut être en mesure d'identifier tout genre de restriction qui peut limiter les valeurs que peuvent prendre les variables de décision.

3. La fonction objectif :
On associe à chaque variable de décision du modèle correspondant, un coefficient économique in-

diquant la contribution unitaire de la variable correspondante à l'objectif poursuivi. Par la suite, on pourra en déduire la fonction objectif que l'on veut optimiser (soit à maximiser, soit à minimiser), autrement dit, la formulation de la fonction économique (ou fonction objectif) traduisant les préférences du décideur exprimées sous la forme d'une fonction des variables identifiées.

2.2 La programmation linéaire

En mathématiques, les problèmes de programmation linéaire (PL) sont des problèmes d'optimisation, où la fonction objectif et les contraintes sont toutes linéaires. La programmation linéaire désigne également la manière de résoudre les problèmes de (PL). Le terme programmation linéaire suppose que les solutions à trouver doivent être représentées en variables réelles, ces variables étant appelées variables de décision [16].
La forme la plus générale d'un problème de (PL) est :

$$(PL)\begin{cases} optZ = Max(Min)Z = \sum_{j=1}^{n} c_j x_j, & (1.1)\\ \sum_{j=1}^{n} a_{ij}x_j \leq b_i, \quad i \in I \subset \{1,...,m\}, & (1.2)\\ \sum_{j=1}^{n} a_{lj}x_j = b_l, \quad l \in L \subset \{1,...,m\}, & (1.3)\\ \sum_{j=1}^{n} a_{kj}x_j \geq b_k, \quad k \in K \subset \{1,...,m\}, & (1.4)\\ x_j \in \mathbb{R}^+, \qquad j \in \{1,...,n\}, & (1.5)\\ I \cap L \cap K = \emptyset, \\ I \cup L \cup K = \{1,...,m\}, \end{cases}$$

avec $c \in \mathbb{R}^n$ vecteur de coûts, $b \in \mathbb{R}^m$ vecteur des termes constants des contraintes et $A \in \mathbb{R}^{m\times n}$ matrice des coefficients des contraintes, où n est le nombre de variables et m est le nombre de contraintes.
Le terme (1.1) représente la fonction économique ou fonction objectif.
Les termes (1.2),(1.3) et (1.4) représentent les contraintes principales .
Le terme (1.5) représente la contrainte de non-négativité.

Remarques

- Toute inégalité $\geq$ (resp $\leq$) peut être transformée en égalité, en soustrayant (resp en rajoutant) des variables d'écart.
- Toute inégalité $\geq$ est équivalente à une inégalité $\leq$ en multipliant ses termes par (-1).
- L'égalité (1.3) est équivalente à deux inégalités $\leq$ telles que :

$$b_l \leq \sum_{j=1}^{n} a_{lj}x_j \leq b_l, \quad l \in L \subset \{1, ..., m\},$$

c'est à dire :

$$\begin{array}{lll} \sum_{j=1}^{n} a_{lj}x_j \leq b_l & \text{et} & \sum_{j=1}^{n} a_{lj}x_j \geq b_l, \quad l \in L \subset \{1, ..., m\}, \\ \sum_{j=1}^{n} a_{lj}x_j \leq b_l & \text{et} & -\sum_{j=1}^{n} a_{lj}x_j \leq -b_l, \quad l \in L \subset \{1, ..., m\}. \end{array}$$

- Remarquons que $minZ = -max(-Z)$.
- Un programme linéaire est dit sous forme canonique si toutes les inégalités sont dans le même sens ($\leq$), les variables de décision sont positives ou nulles et les contraintes d'égalité en sont absentes.
- Tout programme linéaire sous forme canonique peut s'écrire sous forme standard, en ajoutant pour chaque contrainte i une variable s_i appelée variable d'écart, le programme linéaire s'écrivant alors de la manière suivante :

$$(PLS) \begin{cases} maxz(x) = c^t x, \\ s.c \quad (A + I)(x, x_s)^t = b, \\ \qquad x \in \mathbb{R}^n_+, x_s \in \mathbb{R}^m_+. \end{cases}$$

2.3 Formulation d'un problème de PLNE

Un problème de programmation linéaire (PL) est un problème d'optimisation consistant à maximiser une fonction linéaire dite fonction objectif sous contraintes linéaires exprimées sous forme d'équations ou d'inéquations. Un problème de PLNE est un problème de PL dans lequel les

variables sont astreintes à prendre des valeurs entières.

Un tel programme peut être représenté sous forme matricielle suivante :

$$(PLNE)\begin{cases} maxz(x) = c^t x, \\ s.c \quad Ax \leq b, \\ \quad x \in \mathbb{N}^n, \end{cases}$$

avec les notations suivantes :

- max : maximiser,
- n : nombre de variables,
- m : nombre de contraintes,
- $A = (a_{ij},\ 1 \leq i \leq m,\ 1 \leq j \leq n)$: matrice réelle des contraintes, d'ordre $m \times n$,
- $c = (c_1, ..., c_n)^t$: vecteur des coûts, $c \in \mathbb{R}^n$,
- $b = (b_1, ..., b_m)^t$: vecteur-colonne des seconds mombres, $b \in \mathbb{R}^m$,
- $x = (x_1, ..., x_n)^t$: vecteur des variables de décision.

Dans le cas où les variables entières sont astreintes à ne prendre que les valeurs 0 ou 1, on parle alors de programmation linéaire en 0-1, d'où le problème (PLB) suivant :

$$(PLB)\begin{cases} maxz(x) = c^t x, \\ s.c \quad Ax \leq b, \\ \quad x \in \{0,1\}^n. \end{cases}$$

2.3.1 Relaxation linéaire

Avant de passer à la résolution d'un problème de PLNE, on doit écrire sous forme d'un programme linéaire continu le problème relaxé (obtenu à partir de PLNE), en relaxant les contraintes d'intégrité [17] :

$$(PLR)\begin{cases} maxz(x) = c^t x, \\ s.c \quad Ax \leq b, \\ \quad x_j \geq 0, \forall j = 1, ..., n. \end{cases}$$

L'objectif de la relaxation linéaire est de transformer le PLNE en un PLR beaucoup plus facile à résoudre, et dont la solution optimale constitue une borne supérieure de la solution du PLNE. Si la solution optimale du PLR est entière, elle est alors la solution optimale du PLNE. Dans le cas où le domaine réalisable du PLR est vide, le domaine réalisable du PLNE l'est aussi.
Nous notons par x^* la solution optimale de la relaxation linéaire et par Z^* sa valeur optimale.

2.3.2 La complexité théorique d'un problème

Le problème de PLNE est du type combinatoire. Cette notion de problème combinatoire est formellement caractérisée par la théorie de la complexité qui propose une classification des problèmes en fonction de la complexité de leur résolution.

On entend ici par « complexité d'un problème » une estimation du nombre d'instructions à exécuter pour résoudre les instances de ce problème, cette estimation étant un ordre de grandeur par rapport à la taille de l'instance. Il s'agit d'une estimation dans le pire des cas, dans le sens où la complexité d'un problème est définie en considérant son instance la plus difficile [18].

Classes de complexité

L'expérience montre que certains problèmes sont plus faciles à résoudre que d'autres, dans le sens où la meilleure solution peut être obtenue rapidement. La théorie de la complexité a été développée pour permettre à classer mathématiquement les problèmes selon leur difficulté [19]. On définit les trois grandes classes **P, NP** et **NP-complet**. Le lecteur intéressé par de plus amples informations pourra consulter différents ouvrages dédiés à la complexité, dont les livres de Garey et Johnson [20] ou Papadimitriou[21], et plus récemment celui de Goldreich [22].

- **La classe P (Polynomial time)**

  Elle contient l'ensemble des problèmes polynomiaux, i.e, pouvant être résolus par un algorithme de complexité polynomiale. Cette classe caractérise l'ensemble des problèmes que l'on peut résoudre «efficacement». Les problèmes appartenant à cette classe étant traitables, ils peuvent être résolus par une machine de Turing déterministe pendant un temps de calcul polynomial.

- **La classe NP (Non deterministic Polynomial time)**

C'est la classe des problèmes de décision pour lesquels il est possible de vérifier en temps polynomial qu'une solution donnée est réalisable, c'est-à-dire, on peut construire un algorithme polynomial qui est capable de vérifier si pour une solution donnée, la réponse au problème de décision est OUI ou NON. Ainsi, tout problème de décision qui peut être résolu par un algorithme polynomial, donc appartenant à la classe P, appartient également à la classe NP, d'où P $\subseteq$ NP.

- **La classe NP-complet**

  Certains problèmes NP apparaissent plus difficiles à résoudre dans le sens où l'on ne trouve pas d'algorithme polynomial pour le résoudre avec une machine de Turing déterministe. Les plus difficiles de NP définissent la classe des problèmes NP-complets : un problème de NP est NP-complet s'il est au moins aussi difficile à résoudre que n'importe quel autre problème de NP, i.e , que l'examen de chaque cas puisse être réalisé efficacement par une procédure polynomiale. Si on enlève cette contrainte d'appartenance à la classe NP, on obtient la classe plus générale des problèmes NP-difficiles, contenant l'ensemble des problèmes qui sont au moins difficiles que n'importe quel problème de NP.

2.4 La liaison entre la programmation concave et le problème de programmation linéaire entière

Considérons le problème de programmation linéaire en nombres 0-1 :

$$PLB : minZ(x) = c^t x,$$

$$(s.c) \begin{cases} Ax \leq b, \\ x \in B_2^n, \end{cases}$$

où $B_2 = \{$ 0,1$\}$, $A = \{a_{ij}, 1 \leq i \leq m, 1 \leq j \leq n\}$ est une matrice d'ordre $m \times n$, $x = (x_1, x_2, ..., x_n)^t$ et b est un m-vecteur, avec $b = (b_1, b_2, ..., b_m)^t$.

De nombreux algorithmes existent pour résoudre le PLB, comme la méthode de Branch and Bound. Toutefois, lorsque le problème est grand, le temps d'exécution augmente considérablement.

La liaison entre la programmation concave et le problème de programmation linéaire entière est évidente sous des hypothèses convenables. Le lien s'obtient par une relaxation continue en remplaçant $x \in B_2^n$ par $x \in I_2^n$, avec $I_2 = [0, 1]$.
Par conséquent, le PLB est équivalent à la minimisation du problème concave CMP, qui s'écrit sous la forme suivante :

$$CMP : minF_\mu = Z(x) + \mu x^t(e - x),$$

$$(s.c) \begin{cases} Ax \leq b, \\ x \in I_2^n, \\ \mu \in \mathbb{R}^+, \end{cases}$$

où e est le vecteur des uns. Pour assurer l'équivalence entre PLB et CMP, nous devons choisir un μ approprié tel que F_μ devient concave. Cette équivalence représente deux difficultés majeures : D'une part, la minimisation de F_μ (qui est non-convexe) est un problème NP-difficile même si elle porte sur des variables continues, deuxièment, un bon paramètre μ qui garantisse un minimum global ou local, lorsque les contraintes fonctionnelles sont présentes, est difficile à obtenir. À notre connaissance, il n'existe pas de méthode efficace pour la détermination d'une valeur optimale pour le paramètre μ dans ce cas.

2.5 Méthodes de résolution

2.5.1 La méthode graphique

La méthode graphique permet la résolution des problèmes linéaires simples.
Cette méthode est limitée aux problèmes à deux ou trois variables de décision puisqu'il n'est pas possible d'illustrer graphiquement plus de trois dimensions. Bien qu'en général on puisse difficilement trouver des problèmes avec seulement deux ou trois variables de décision, cette méthodologie de résolution est cependant très utile, avec la reproduction graphique de situations possibles, telles que :

- L'existence d'une solution optimale unique,

- La non-existence de solution et l'absence de partie bornée,

La méthode graphique nous offre donc une aide visuelle pour interpréter et comprendre l'algorithme de la méthode utilisée pour résoudre le problème.
Les étapes de résolution des problèmes par la méthode graphique sont les suivantes :

1) Créer un système de coordonnées cartésiennes, dans lequel chaque variable de décision est représentée sur un axe,

2) Pour chacun des axes, établir une échelle de mesure appropriée à sa variable associée,

3) Dessiner dans le système de coordonnées les contraintes du problème, y compris celles des variables de décision,

4) Remarquer qu'une inéquation précise une région qui sera le demi-plan limité par la ligne droite, celle-ci représente la contrainte qu'on considère comme une contrainte d'égalité alors que si une équation linéaire détermine une région c'est la ligne droite, elle-même,

5) L'intersection de toutes les régions détermine la région ou l'espace réalisable (qui est un ensemble convexe),
Si non, s'il n'y a pas de point qui satisfait toutes les contraintes simultanément, alors le problème est impossible, car n'ayant pas de solution réalisable,

6) Déterminer les points extrêmes ou les sommets du polyèdre qui forme la région réalisable,

7) Ces points seront les candidats à la solution optimale,

8) Évaluer la fonction objectif à chaque sommet et celui qui maximise la fonction objectif définit la solution optimale (pour un problème de maximisation)[23].

2.5.2 Méthode du Simplexe

Dans la plupart des problèmes réels, on a plus de deux variables à déterminer .
Une procédure algébrique pour résoudre les programmes linéaires avec plus de deux variables fera l'objet de cette section. C'est la méthode du simplexe.
La méthode du simplexe est une méthode itérative, elle démarre d'un point réalisable (sommet de départ) et passe de ce point à un autre en augmentant la valeur de la fonction objectif dans le cas de maximisation.
On arrête le déroulement de l'algorithme lorsqu'il n'est plus possible d'augmenter la valeur de la fonction objectif[7].

L'algorithme du simplexe

Soit un problème de maximisation suivant :

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

pour ce problème, le principe de résolution nécessite un nombre d'étapes qui sont :

1). Écrire le système sous la forme standard,

2). Le point de départ de l'algorithme est l'existence d'une solution réalisable soit :
$x = (x_B, x_N) = (x_B, 0)$: la solution réalisable basique, et A_B^{-1} : la matrice inversible de $A_B = A(I, J_B)$, où $x_B = x(J_B) = A_B^{-1}b \geq 0$ et $x_N = x(J_N) = 0$, $J_B \cup J_N = J$, $J_B \cap J_N = \emptyset$, $|J_B| = |I| = m$,

3). Construire le premier tableau correspondant à la forme standard,

4). Calculer les estimations : $E_j = u^t a_j$ - c_j, $j \in J_N$ (critère d'optimalité), où :

$u^t = c_B^t A_b^{-1}$ est le vecteur des potentiels,

5). Si tous les $E_j \geq 0$, $\forall j \in J_N$, la solution $(x_B, 0)$ est optimale.

Si non :

-Si $\exists k \in J_N$ tel que : $E_k < 0$ et $A_B^{-1} a_k < 0$, le (PL) n'admet pas de maximum fini (non borné), arrêt.

-Sinon si $\exists k \in J_N$ tel que : $E_k < 0$ et $A_B^{-1} a_k > 0$, alors aller à 6),

6). Choisir le vecteur-colonne a_k qui entre dans la base tel que : $E_k = \min_{j \in J_N} E_j$, $E_j < 0$,

7). Choisir le vecteur a_l qui sort de la base tel que :

$\theta_l = \min \{ \frac{b_i}{a_{ik}}, a_{ik} > 0, 1 \leq i \leq m \}$, où θ_l est le pas de la direction choisie,

8). Encadrer le pivot a_l,

9). Multiplier la ligne du pivot par le rapport : $\frac{1}{a_{lk}}$,

10). Calculer les valeurs a'_{ij} des autres lignes :

$a'_{ij} = a_{ij}$ - $(\frac{a_{ik} * a_{lj}}{a_{lk}})$,

aller à 4).

2.5.3 Méthode duale du simplexe

Pour éviter l'utilisation des variables d'écart, quand on a un problème avec des contraintes sous forme :

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i,\ I = \{1, ..., m\},\ m > n,$$

on utilise l'algorithme dual du simplexe afin de réduire la taille du (PL) et le nombre d'itérations[24].

Principe de la dualité

On suppose que A est une matrice d'ordre $m \times n$ et $b \in \mathbb{R}^m$.

À chaque problème d'optimisation linéaire, nous allons définir un nouveau problème appellé le dual. Le problème original est le primal. Le principe de la méthode duale est de garder le critère d'optimalité satisfait à chaque itération et de rendre positifs certains b_i qui sont négatifs.

• Un programme linéaire primal :

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

On associe le programme linéaire dual :

$$(PLD) \begin{cases} minL = b^t y, \\ A^t y \geq c, \\ y \geq 0. \end{cases}$$

On notera que, pour le problème primal, on a $x \in \mathbb{R}^n$ tandis que $y \in \mathbb{R}^m$ pour le dual[25].

2.6 Méthodes exactes pour la résolution des problèmes de PLNE

Il existe plusieurs méthodes exactes (algorithmes) de résolution des problèmes de programmation linéaire en nombres entiers, qui permettent l'obtention d'au moins une solution optimale du problème à résoudre. Une fois une solution réalisable est obtenue, l'algorithme doit également être capable d'en prouver l'optimalité. Ce qui est parfois tout aussi difficile à faire que d'obtenir cette solution. Citons quelques approches exactes classiques utilisées en recherche opérationnelle que nous définirons juste après :

- Méthode de Branch and Bound (Séparation et Evaluation)[26],
- Méthode des coupes [27],[28],

- Méthode de programmation dynamique [2],[29],
- Méthode de support [3].

2.7 La méthode de Branch and Bound

L'algorithme de Branch and Bound est une méthode qui permet d'énumérer implicitement l'ensemble des solutions réalisables d'un problème de programmation linéaire en nombres entiers. Il suit le principe de « diviser pour régner » en décomposant récursivement un problème en plusieurs sous-problèmes plus simples à résoudre. La solution optimale du problème est retrouvée en prenant la solution du meilleur sous-problème, c'est-à-dire le sous-problème qui permet d'identifier la solution réalisable ayant la plus grande borne pour le cas de maximisation.
L'algorithme génère et parcourt un arbre de recherche, où chaque noeud de l'arbre de recherche correspond à un sous-domaine du domaine réalisable du problème original. La racine de l'arbre de recherche correspond au domaine réalisable du problème, tandis que les noeuds représentent les domaines réalisables des sous-problèmes.
À chaque itération, l'algorithme de Branch and Bound résout la relaxation du sous-problème courant. Cette dernière fournit une borne supérieure (cas de maximisation) de la valeur optimale du sous-problème courant.
Différents cas peuvent être rencontrés :

✓ Le sous-problème n'est pas réalisable, cela signifie qu'il n'y a pas de solution dans cette branche. Elle est exclue.

✓ La borne supérieure du noeud est inférieure à la meilleure solution réalisable obtenue (cas de maximisation) par l'algorithme de Branch and Bound : on peut alors affirmer que la solution optimale globale ne peut pas être contenue dans le sous-ensemble de solutions représenté par ce noeud. On l'élague.

✓ La solution de la relaxation continue du noeud devient la meilleure solution réalisable du problème initial si elle est entière, et que sa valeur optimale est supérieure à celle obtenue par l'algorithme de Branch and Bound (cas de maximisation) jusqu'à présent.

✓ Finalement, si la borne supérieure obtenue est meilleure que la dernière valeur obtenue par

l'algorithme de Branch and Bound pour un cas de maximisation, mais sa solution n'est pas entière, le problème sera divisé en sous-problèmes.

La méthode sera appliquée récursivement au reste des noeuds, et elle s'arrête lorsqu'il n'y a plus de noeud à évaluer.

La méthode de Branch and Bound est résumée comme suit[26] :

Entrée :

problème de maximisaton (P) .

Sortie :

x^* solution optimale du problème (P),

Z^* valeur optimale.

Noté par :

(P) : le problème initial,

Γ : l'ensemble des problèmes,

(P_i) : un sous-problème à chaque itération,

$(P_i)_{RL}$: problème ralaxé de (P_i),

x^R : la solution optimale de $(P_i)_{RL}$

Z^R_{RL} : valeur optimale de $(P_i)_{RL}$.

1) Résolution du problème relaxé (PR) :

Si x^* est entier alors : fin.

Sinon, aller à 2.

2) Initialisation :

$Z^* = -\infty$, $\Gamma = \{P\}$,

3) Répéter jusqu'à $\Gamma = \emptyset$,

4) Sélectionner un noeud $(P_i) \in \Gamma, \Gamma \leftarrow \Gamma \backslash (P_i)$,

5) Évaluer (P_i) :

Résoudre la relaxation $(P_i)_{RL}$ de (P_i),

• Si $(P_i)_{RL}$ est non réalisable, supprimer (P_i) de Γ et aller à l'étape 3,

• Sinon soit : x^R la solution optimale de $(P_i)_{RL}$, et Z^R_{RL} sa valeur optimale,

6) Élaguer :

• Si $Z^R_{RL} < Z^*$, supprimer $(P_i)_{RL}$ de Γ et retourner à l'étape 3.

• Sinon

- Si x^R n'est pas entière, aller à l'étape 7,

- Sinon (x^R est entière) $Z^* = Z^R_{RL}$, $x^* = x^R$. Aller à l'étape 3,

7) Brancher :

Choisir une variable non entière x_l, créer deux branches, on obtient deux sous-problèmes sous la forme $(P_i) = (P_{i+1}) \cup (P_{i+2})$:

$$\begin{cases} P_{i+1} : P_i + \text{la contrainte } x_l \geq [x_l] + 1, \\ P_{i+2} : P_i + \text{la contrainte } x_l \leq [x_l], \end{cases}$$

avec $[x_l]$: la partie entière de x_l,

retourner à l'étape 3.

Exemple 1[30]

Soit le problème suivant, noté $(\overline{P})$:

$$(\overline{P}) \begin{cases} \text{Min } \overline{Z} = \ x_1 - 2x_2, \\ \quad -4x_1 + 6x_2 \leq 9, \\ \quad x_1 + x_2 \leq 4, \\ \quad x_1, x_2 \geq 0, \\ \quad x_1, x_2 \in \mathbb{N}. \end{cases}$$

Le problème $\overline{P}$, nous le traitons sous la forme suivante :

$$(P)\begin{cases} \text{Max } Z = \ -x_1 + 2x_2, \\ \quad -4x_1 + 6x_2 \leq 9, \\ \quad x_1 + x_2 \leq 4, \\ \quad x_1, x_2 \geq 0, \\ \quad x_1, x_2 \in \mathbb{N}. \end{cases}$$

Résoudre le problème relaxé, noté par $R(P)$:

$$R(P)\begin{cases} \text{Max } Z = \ -x_1 + 2x_2, \\ \quad -4x_1 + 6x_2 \leq 9, \\ \quad x_1 + x_2 \leq 4, \\ \quad x_1, x_2 \geq 0. \end{cases}$$

Si la solution de la relaxation est entière, pas besoin de partitionner le sous-problème. Sinon, on choisit une composante non entière x_i, et on crée deux sous-problèmes en ajoutant les contraintes :

$$x_i \geq [x_i] + 1 \text{ et } x_i \leq [x_i].$$

En utilisant la méthode graphique, le domaine admissible et la fonction objectif sont représentés sur la figure suivante :

La solution optimale réelle obtenue par le graphe (Figure 2.1) est :

$$x^0 = (1.5,\ 2.5),$$

et

$$Z^0 = 3.5,$$

puisque x^0 n'est pas entier, alors on divise le problème initial (P) en deux sous-problèmes.

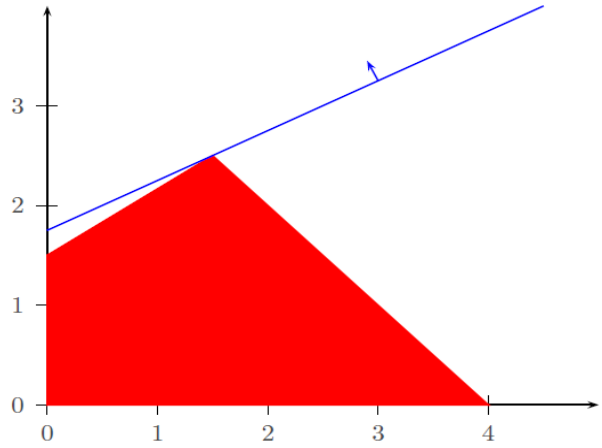
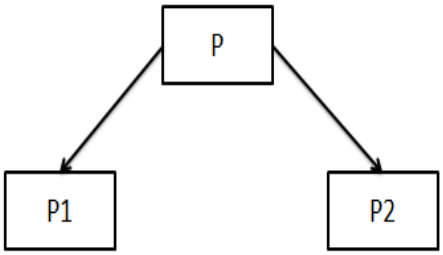


FIGURE 2.1 – Problème initial relaxé actif $\{R(P)\}$.



La forme du problème (P_1) :

$$(P_1)\begin{cases} \text{Max } Z = -x_1 + 2x_2, \\ -4x_1 + 6x_2 \leq 9, \\ x_1 + x_2 \leq 4, \\ x_2 \geq 3, \\ x_1, x_2 \geq 0. \end{cases}$$

La forme du problème (P_2) :

$$(P_2)\begin{cases} \text{Max } Z = -x_1 + 2x_2, \\ -4x_1 + 6x_2 \leq 9, \\ x_1 + x_2 \leq 4, \\ x_2 \leq 2, \\ x_1, x_2 \geq 0. \end{cases}$$

On résoud les deux problèmes par la méthode graphique (Figure 2.2) :

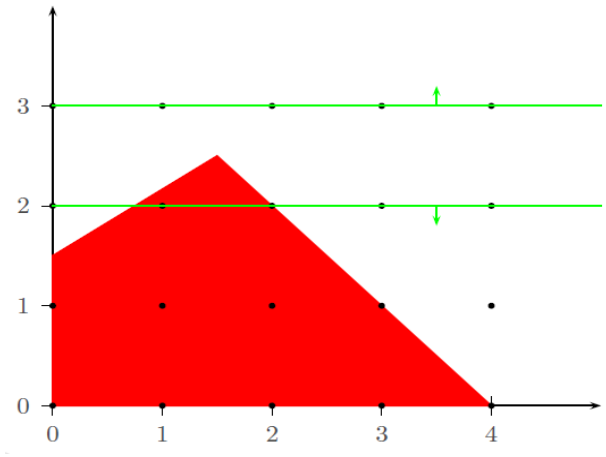
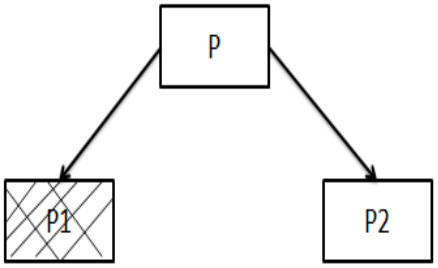


FIGURE 2.2 – Sous-problèmes actifs $\{P_1 , P_2\}$.

On observe que :

le problème P_1 n'est pas réalisable.



Résolution du sous-problème (P_2) graphiquement (Figure 2.3) :

La solution optimale réelle de (P_2) relaxé est :

$$x^2 = (0.75,2),$$

et

$$Z^2 = 3.25.$$

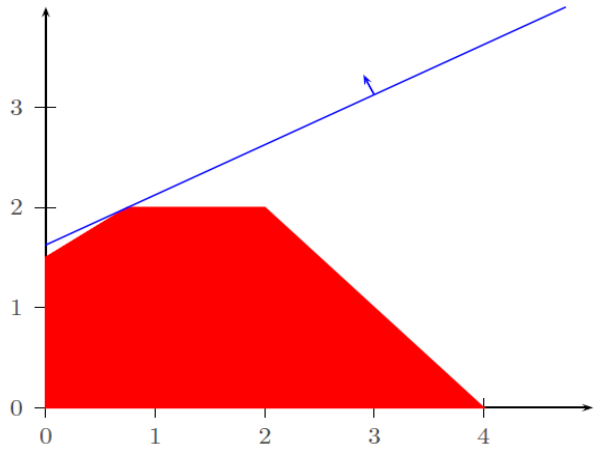
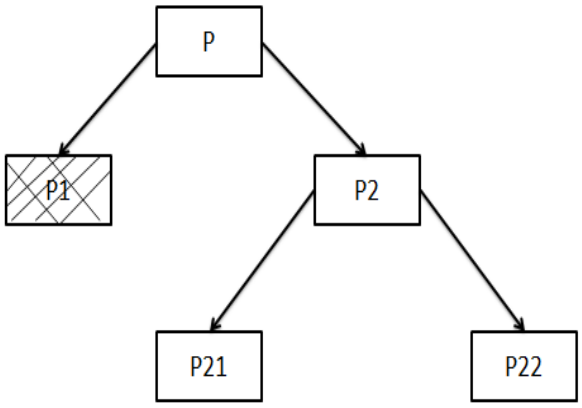


FIGURE 2.3 – Sous-problème actif { P_2}.

La solution x^2 n'est pas entière, alors on divise le sous-problème (P_2) en deux sous-problèmes, (P_{21}) et (P_{22}) .



La forme du problème (P_{21}) :

$$(P_{21}) \begin{cases} \text{Max } Z = \ -x_1 + 2x_2, \\ -4x_1 + 6x_2 \leq 9, \\ x_1 + x_2 \leq 4, \\ x_2 \leq 2, \\ x_1 \geq 1, \\ x_1, x_2 \geq 0. \end{cases}$$

La forme du problème (P_{22}) :

$$(P_{22}) \begin{cases} \text{Max } Z = \ -x_1 + 2x_2, \\ -4x_1 + 6x_2 \leq 9, \\ x_1 + x_2 \leq 4, \\ x_2 \leq 2, \\ x_1 \leq 0, \\ x_1, x_2 \geq 0. \end{cases}$$

On résoud les deux problèmes par la méthode graphique (Figure 2.4) :

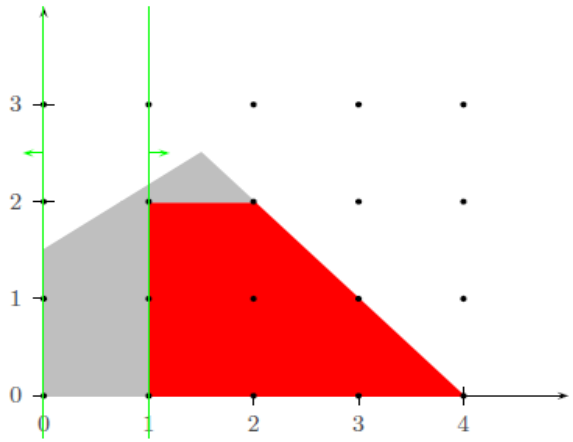


FIGURE 2.4 – Sous-problèmes actifs $\{P_{21}\ P_{22}\}$.

On observe que :

La solution optimale de (P_{21}) relaxé est entière :

$$x^{21} = (1,2),$$

et,

$$Z^{21} = 3.$$

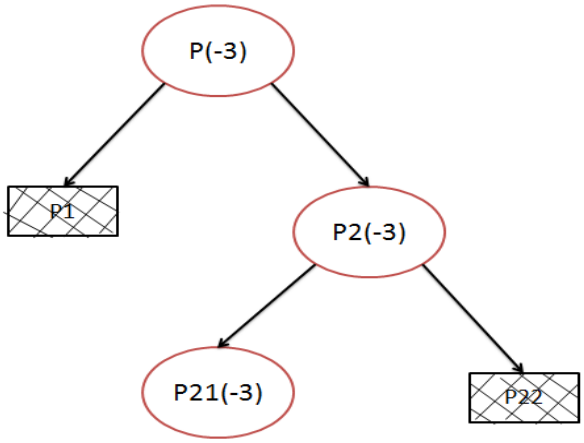
La solution optimale de (P_{22}) est :

$$x^{22} = (0,1.5),$$

et,

$$Z^{22} = 3.$$

Cette solution n'étant pas entière, alors la borne supérieure pour (P_{22}) est inférieure ou égale à 3, donc la solution optimale du problème initial (P) est $x^* = x^{21} = (1,2)$, avec $Z^* = \overline{Z} =$ - $Z_{21} =$ -3.



Exemple 2

Soit le problème suivant, noté (P) :

$$(P)\begin{cases}(\text{PR})\begin{cases}\text{Max } Z = \ 3x_1+4x_2,\\ 2x_1+x_2\leq 6,\\ 2x_1+3x_2\leq 9,\\ x_2\leq 1,\\ x_1,x_2\geq 0,\end{cases}\\ x_1,x_2\in\mathbb{N}.\end{cases}$$

Résolution du problème (PR) :

- **Sa forme standard :**

$$(PR)\begin{cases}\text{Max } Z = \ 3x_1+4x_2,\\ 2x_1+x_2+x_3=6,\\ 2x_1+3x_2+x_4=9,\\ x_2+x_5=1,\\ x_i\geq 0, i=\overline{1,5}\end{cases}$$

- Résoudre le (PR) par la méthode du simplexe :

		c_j	3	4	0	0	0	
c_B	Base	$b\backslash a_j$	a_1	a_2	a_3	a_4	a_5	θ
0	a_3	6	2	1	1	0	0	6
0	a_4	9	2	3	0	1	0	3
0	a_5	1	0	**1**	0	0	1	1
	$Z = 0$	E_j	-3	-4	0	0	0	

- a_2 entre dans la base.

- a_5 sort de la base.

		c_j	3	4	0	0	0	
c_B	Base	$b\backslash a_j$	a_1	a_2	a_3	a_4	a_5	θ
0	a_3	5	**2**	0	1	0	-1	$\frac{5}{2}$
0	a_4	6	2	0	0	1	-3	3
4	a_2	1	0	1	0	0	1	/
	$Z = 4$	E_j	-3	0	0	0	4	

- a_1 entre dans la base.

- a_3 sort de la base.

		c_j	3	4	0	0	0	
c_B	Base	$b\backslash a_j$	a_1	a_2	a_3	a_4	a_5	θ
0	a_1	$\frac{5}{2}$	1	0	$\frac{1}{2}$	0	-$\frac{1}{2}$	/
0	a_4	1	0	0	-1	1	-2	/
4	a_2	1	0	1	0	0	1	/
	$Z = 11.5$	E_j	0	0	$\frac{3}{2}$	0	$\frac{5}{2}$	

- Tous les $E_j \geq 0 \Rightarrow$ la solution optimale est $x^* = (\ 2.5\ ,\ 1\ )$, avec $Z^* = 11.5$,

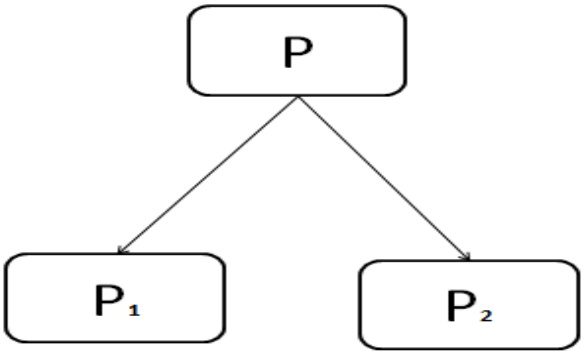
Z^* est la borne supérieure de (P).

Soit (P) le problème initial (le sommet initial de l'arborescence) tel que :

$$Z^* = 11.5,$$
$$x_1^* = 2.5,$$
$$x_2^* = 1.$$

Cette solution n'est pas entière, alors on divise le problème (P) en deux sous-problèmes, (P_1) et (P_2), sous la forme :

$$\begin{cases} P_1 : P + \text{la contrainte } x_1 \geq 3. \\ P_2 : P + \text{la contrainte } x_1 \leq 2. \end{cases}$$



Résolution des sous-problèmes :

La forme standard du problème (P_1) :

$$(P_1) \begin{cases} \text{Max} Z = \ 3x_1 + 4x_2, \\ \qquad 2x_1 + x_2 + x_3 = 6, \\ \qquad 2x_1 + 3x_2 + x_4 = 9, \\ \qquad x_2 + x_5 = 1, \\ \qquad x_1 - x_6 = 3, \qquad x_i \geq 0, i = \overline{1,6}. \end{cases}$$

À partir du dernier tableau, on a $x_1 + \frac{1}{2}x_3 - \frac{1}{2}x_5 = \frac{5}{2}$, et on a $x_1 - x_6 = 3$, on résoud et on obtient la contrainte suivante : $\frac{1}{2}x_3 - \frac{1}{2}x_5 + x_6 = -\frac{1}{2}$.

On rajoute cette contrainte au dernier tableau de (PR), et on applique le simplexe :

		c_j	3	4	0	0	0	0	
c_B	Base	$b\backslash a_j$	a_1	a_2	a_3	a_4	a_5	a_6	θ
3	a_1	$\frac{5}{2}$	1	0	$\frac{1}{2}$	0	-$\frac{1}{2}$	0	/
0	a_4	1	0	0	-1	1	-2	0	/
4	a_2	1	0	1	0	0	1	0	/
0	a_6	-$\frac{1}{2}$	0	0	$\frac{1}{2}$	0	-$\frac{1}{2}$	1	/
	$Z = 11.5$	E_j	0	0	$\frac{3}{2}$	0	$\frac{5}{2}$	0	

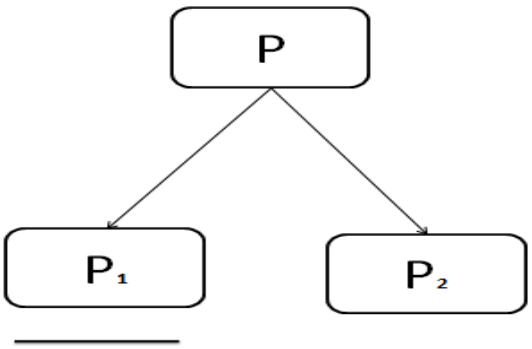
La solution du tableau précédent étant non optimale, on résout avec l'algorithme dual du simplexe et on obtient :

- a_6 sort de la base.

- a_5 entre dans la base.

		c_j	3	4	0	0	0	0	
c_B	Base	$b\backslash a_j$	a_1	a_2	a_3	a_4	a_5	a_6	θ
3	a_1	3	1	0	0	0	0	-1	/
0	a_4	3	0	0	-3	1	0	-4	/
4	a_2	0	0	1	1	0	0	2	/
0	a_5	1	0	0	-1	0	1	-2	/
	$Z = 9$	E_j	0	0	4	0	0	5	

La solution optimale est $x^* = (3,0)$, avec $Z^* = Z^1 = 9$, cette solution est entière, on coupe ce sommet.



La forme standard du problème (P_2) :

$$(P_2)\begin{cases} \text{Max}Z = \ 3x_1 + 4x_2, \\ \qquad 2x_1 + x_2 + x_3 = 6, \\ \qquad 2x_1 + 3x_2 + x_4 = 9, \\ \qquad x_2 + x_5 = 1, \\ \qquad x_1 + x_7 = 2, \qquad x_i \geq 0, i = \overline{1,7}. \end{cases}$$

À partir du dernier tableau, on a $x_1 + \frac{1}{2}x_3$ - $\frac{1}{2}x_5 = \frac{5}{2}$, et on a $x_1 + x_7 = 2$, on résoud et on obtient la contrainte suivante : -$\frac{1}{2}x_3 + \frac{1}{2}x_5 + x_7 =$ -$\frac{1}{2}$.

On rajoute cette contrainte au dernier tableau de (PR), et on applique le simplexe :

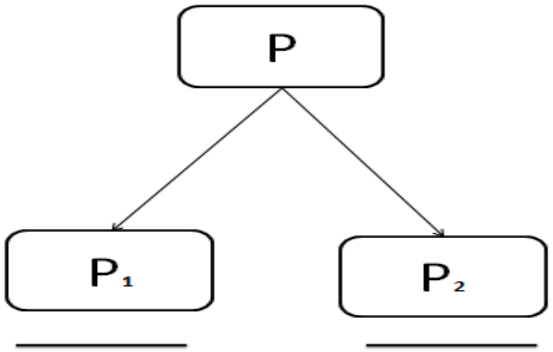
		c_j	3	4	0	0	0	0	
c_B	Base	$b\backslash a_j$	a_1	a_2	a_3	a_4	a_5	a_7	θ
3	a_1	$\frac{5}{2}$	1	0	$\frac{1}{2}$	0	$-\frac{1}{2}$	0	/
0	a_4	1	0	0	-1	1	-2	0	/
4	a_2	1	0	1	0	0	1	0	/
0	a_7	$-\frac{1}{2}$	0	0	$-\mathbf{\frac{1}{2}}$	0	$\frac{1}{2}$	1	/
	$Z = 11.5$	E_j	0	0	$\frac{3}{2}$	0	$\frac{5}{2}$	0	

La solution du tableau précédent étant non optimale, on résout avec l'algorithme dual du simplexe et on obtient :

- a_7 sort de la base.
- a_3 entre dans la base.

		c_j	3	4	0	0	0	0	
c_B	Base	$b\backslash a_j$	a_1	a_2	a_3	a_4	a_5	a_7	θ
3	a_1	2	1	0	0	0	0	1	/
0	a_4	2	0	0	0	1	-3	-2	/
4	a_2	1	0	1	0	0	1	0	/
0	a_3	1	0	0	1	0	-1	-2	/
	$Z = 10$	E_j	0	0	0	4	4	3	

La solution optimale est x^* =(2,1), avec $Z^* = Z^2 = 10$, cette solution est entière, on coupe ce sommet.



Tous les sommets sont coupés, le critère d'arrêt est vérifié.

On a $Z^2 > Z^1$, alors $Z^*= Z^2$, d'où la solution optimale est : x^* =(2,1), avec $Z^* = 10$.

Chapitre 3

La méthode de Branch and Bound appliquée à la minimisation concave

Le problème de minimisation concave consiste à minimiser une fonction concave et continue sur un certain domaine convexe, c'est-à-dire à trouver un point de ce domaine de plus basse valeur. Il s'agit d'un problème difficile, qui peut avoir de nombreux minimums locaux. Cependant, il possède une propriété intéressante qui le rend plus facile à résoudre qu'un problème d'optimisation générale, à savoir que le minimum d'une fonction concave sur un ensemble convexe compact est atteint en un de ses points extrêmes. En particulier, si le domaine est polyédral, cela permet de réduire le problème de minimisation concave au calcul d'un nombre fini (mais éventuellement très grand) de points.

3.1 Problème de minimisation concave

Les problèmes de minimisation concave sont des problèmes qui consistent à minimiser une fonction concave sur un ensemble convexe S.
On considère que S est compact et défini par des inégalités linéaires de la forme

$$Ax - b \leq 0,$$

i.e, $A_i^t x - b_i \leq 0$, $(i = 1,\ldots,m)$, où A est une matrice d'ordre $m \times n$, $m > n$, avec

$$A = (a_1, a_2, ..., a_n) = \begin{pmatrix} A_1^t \\ A_2^t \\ \vdots \\ A_m^t \end{pmatrix},$$

où $a_j = (a_{1j}, \ldots a_{ij}, \ldots, a_{mj})^t$ est la j-ème colonne de A, et $A_i^t = (a_{i1}, \ldots, a_{ij}, \ldots, a_{in})^t$ est la i-ème ligne de A .

La convexité de l'ensemble réalisable S est bien exploitée dans la conception des algorithmes. De plus, la propriété la plus intéressante est que la fonction concave atteint son minimum global en un point extrême de S.

• Points extrêmes

Soit S un polyèdre, $x \in S$ est un point extrême de S si on ne peut pas trouver deux vecteurs y et z dans S, différents de x, et un scalaire $\lambda \in]0,1[$ tels que

$$x = \lambda\ y + (1 - \lambda)\ z.$$

Autrement dit, $x \in S$ est un point extrême de S si on ne peut pas l'exprimer comme combinaison convexe de deux autres points de S.

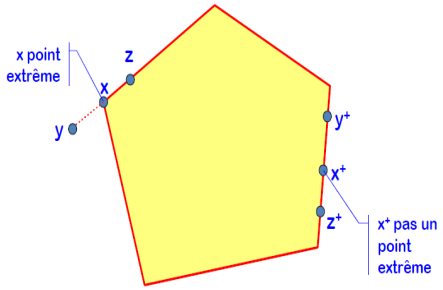


FIGURE 3.1 – Points extrêmes.

Propriétés 3.2.1[4]

Soit S un sous-ensemble polyédrique de $\mathbb{R}^n$ de la forme

$$S = \{\ x \mid A_i^t x \leq b_i,\ i = \overline{1,m}\ \},$$

où $m \geq n$, les A_i et b_i sont des vecteurs et des scalaires, respectivement. Alors $v \in S$ est un point extrême de S si et seulement si l'ensemble

$$A_v = \{\ A_i \mid A_i^t v = b_i,\ i = \overline{1,m}\},$$

contient n vecteurs linéairement indépendants.

Le résultat le plus important en minimisation concave est sans aucun doute le suivant :

Propriétés 3.2.2[4]

Soit $S \subset \mathbb{R}^n$, un ensemble polyédral et compact, et $F : S \longrightarrow \mathbb{R}$ une fonction concave et continue sur S. Alors F atteint son minimum global en un point extrême de S.

Preuve :

Pour chaque point $x \in S$, on a la représentation suivante :

$$x = \sum_{i=1}^{p} \lambda_i v^i\ ,\ \sum_{i=1}^{p} \lambda_i = 1, \lambda_i \geq 0,\ i = \overline{1,p}$$

où v^i, $1 \leq i \leq p$, sont les points extrêmes de S. La concavité de F implique :

$$F(x) \geq \sum_{i=1}^{p} \lambda_i F(v^i) \geq \sum_{i=1}^{p} \lambda_i min\{F(v^i) : i = 1, ..., p\} = min\{F(v^i) : i = 1, ..., p\},\ \forall\ x \in S.$$

Donc, il existe v^{i_0} tel que $F(x) \geq F(v^{i_0}),\ \forall\ x \in S$.

D'où le point extrême $v^{i_0} \in S$ est un minimum global de F sur S.

3.1.1 Minimisation d'une fonction quadratique concave

Soit F une fonction quadratique concave : $\exists\ D \leq 0$ (une matrice carrée semi-définie négative de dimension n), $\exists\ c \in \mathbb{R}^n$ tels que :

$$F(x) = \frac{1}{2} x^t D x + c^t x.$$

Le problème traité dans ce chapitre est la minimisation de $F(x)$ sur le polytope S de $\mathbb{R}^n$. Le modèle sera donc :

$$(P)\begin{cases} minF(x)= \frac{1}{2}x^tDx + c^tx, \\ s.c \qquad Ax - b \leq 0, \\ \qquad\qquad x \in \mathbb{R}^n, \end{cases}$$

où $D^t = D \leq 0$ (semi-définie négative), et l'ensemble admissible $S = \{x \in \mathbb{R}^n : Ax - b \leq 0\}$ est supposé être borné, c'est-à-dire un polytope.

Pour résoudre un problème de minimisation concave, on remplace la fonction objectif $min\{F(x), x \in S\}$ par son enveloppe convexe g sur un polyèdre X^0 où $X^0 = S$ et on résoud le problème suivant :

$$\begin{aligned} &\min\ g(x), \\ &\text{sc } x \in X^0. \end{aligned}$$

•Enveloppe convexe d'une fonction concave

L'enveloppe convexe g de F est une fonction convexe qui sous-estime F sur un polyèdre X^0, c'est-à-dire une fonction affine qui se confond avec F aux $(n+1)$ sommets d'un simplexe S^{01}. Puisque S^{01} est donné par les sommets $\{v_1, v_2, ..., v_{n+1}\}$, alors nous pouvons déterminer la fonction $g(x) = a^t+\alpha$ de F en tout point $x \in S^{01}$ par la résolution des $(n + 1)$ équations linéaires suivantes[4] :

$$\langle a, v^i\rangle + \alpha = F(v^i), i \in \{1, 2, ..., n + 1\}. \tag{3.1}$$

Nous calculons les inconnues $a \in \mathbb{R}^n$ et $\alpha \in \mathbb{R}$, où v^i, $i = \{1,2, \ldots ,n+1\}$, sont les sommets de S^{01}.

3.2 La méthode de Branch and Bound

Un algorithme est présenté pour résoudre le problème de minimisation concave sur un polyèdre[31] :

$$\begin{aligned} P : \min\ F(x) =& \frac{1}{2}x^tDx + c^tx, \\ \text{sc} \qquad & Ax \leq b, \\ & x \in \mathbb{R}^n. \end{aligned}$$

Le problème P peut avoir plusieurs solutions localement optimales qui ne sont pas globalement optimales. Bien qu'il est connu qu'une solution globalement optimale pour le problème P existe, qui est un point extrême de S, S peut avoir un nombre très grand de points extrêmes. Soit S le domaine admissible du problème P et $\overline{m}$ la valeur optimale de la fonction objectif du problème P.

3.2.1 Algorithme

Etape 0

0.1. Choisir $\epsilon \geq 0$, trouver un point p tel que $Ax - b < 0$ au point p. Soit $UB = F(p)$ et soit $x^0 = p$. Soit $E \neq \emptyset$ tel que E est l'ensemble des parties éliminées. Soit X^0 tel que $X^0 = S$. Trouver le simplexe S^{01} de sorte que $X^0 \subseteq S^{01}$ et que S^{01} est décrit par les sommets v_0^1, v_0^2, $\ldots$ $v_0^{n+1} \in \mathbb{R}^n$.

0.2. Trouver l'enveloppe convexe $g_{01} : S^{01} \longrightarrow \mathbb{R}$ de F prise sur S^{01}, par la résolution du système d'équations linéaires (3.1), avec $v^i = v_0^i$, $i = 1, ..., n+1$.

0.3. Trouver le point extrême x^{01}, solution optimale du programme linéaire :

$$P_{01} : min\ g_{01}(x),$$
$$x \in X^0.$$

0.4. Soit $LB = g_{01}(x^{01})$, et $x^0 = x^{01}$. Soit $k = 1$ et aller à l'étape 1.

Pour tout $k \geq 1$.

Etape k

Supposons sans perte de généralité que $x^{k-1} \in S^{k-1,k}$. Supposons également que $v_{k-1}^i, i = 1, \ldots, n+1$, sont les sommets de $S^{k-1,k}$.

k.1. Si $x^{k-1} \notin X^0$, passer à l'étape **k.3**, sinon, si $F(x^{k-1}) < UB$, soit $UB = F(x^{k-1})$ et soit $x^0 = x^{k-1}$, et aller à l'étape **k.2**.

k.2. Si UB - $LB \leq \epsilon$, conclure que x^0 est une solution ϵ-optimale pour le problème P et arrêter. Sinon, continuer.

k.3. cas1 : $x^{k-1} \in X^0$, soit $X^k = X^{k-1}$, avec $X^k = X^{k-1} \cap \{x \in \mathbb{R}^n / g_{kj}(x) \leq UB$, j=1,2$\}$, aller à l'étape **k.4**.

cas2 :$x^{k-1} \notin X^0$, Trouver le point $z^{k-1} \in X^0$ sur le segment de ligne $[p,\ x^{k-1}]$ tel que $g(z^{k-1}) = 0$. Si $F(z^{k-1}) \geq UB$, aller à l'étape **k.4**, sinon, soit $UB = F(z^{k-1})$, soit $x^0 = z^{k-1}$, et, si UB - $LB \leq \epsilon$, conclure que x^0 est la solution ϵ-optimale du problème P et arrêter. Si UB - $LB > \epsilon$, continuer.

k.4. Utiliser la bissection, pour la partition $S^{k,1}$, $S^{k,2}$ de $S^{k-1,k}$, où $S^{k,1}$ et $S^{k,2}$ sont des simplexes.

k.5. Pour tout $j = 1, 2$, trouver l'enveloppe convexe de $g_{kj} : S^{kj} \longrightarrow \mathbb{R}$, prise sur S^{kj} par la résolution du système d'équations linéaire (3.1).

k.6. Pour tout $j = 1, 2$, trouver la solution optimale x^{kj} pour le problème de programmation linéaire

$$P_{kj} : min\ g_{kj}(x),$$
$$x \in X^k \cap S^{kj}.$$

k.7. Pour tout $j = 1, 2$ tel que $S^{kj} \notin$ E, si $g_{kj}(x^{kj}) > UB$, ajouter S^{kj} à E.

k.8. Calculer $LB = min\{g_{kj}(x^{kj}) \mid j \in \{1, 2\}$ et $S^{kj} \notin$ E $\}$. Soit $x^k = x^{kj^*}$ et $g_k^* = g_{kj^*}$, où $LB = g_{kj^*}(x^{kj^*})$, poser k = k+1 et aller à l'étape k.

3.2.2 Exemple illustratif

On considère le problème de minimisation concave suivant :

$$(P)\begin{cases} \text{Min } F(x) = -x_1^2 - x_2^2 + 3x_1 - x_2, \\ x_1 + 3x_2 \leq 7, \\ 3x_1 + x_2 \leq 7, \\ x_1 - x_2 \leq 3, \\ -4x_1 + x_2 \leq 5, \\ x_1 \geq 0, x_2 \geq 0, \end{cases}$$

Pour résoudre ce problème par le nouvel algorithme, nous avons légèrement modifié le critère d'arrêt de sorte que nous sommes arrêtés lorsque nous avions la garantie que $F(x^0)$ était à moins de 0.01 de $\overline{m}$.

Initialiser $UB = +\infty$ et $LB = -\infty$.

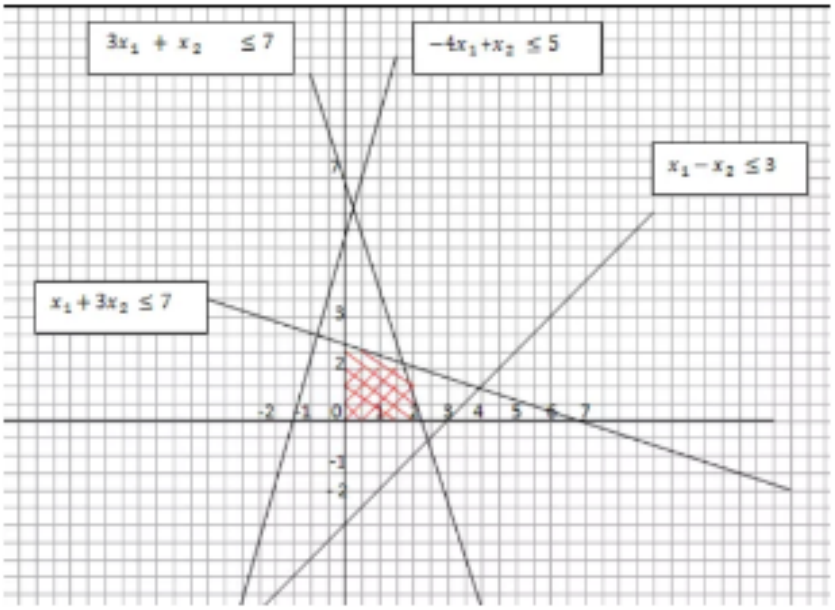


FIGURE 3.2 – Ensemble réalisable pour le problème P .

étape 0

p a été choisi égal à (0.0), et le polyèdre X^0 est décrit par l'ensemble S, avec S=$\{(x_1, x_2)/0 \leq x_1 \leq \frac{7}{3}$, $0 \leq x_2 \leq \frac{7}{3}$, $x_1 + 3x_2 \leq 7$, $3x_1 + x_2 \leq 7$, $-4x_1 + x_2 \leq 5$ $\}$, et S^{01} est choisi par les sommets

$v_0^0 = (0,0)$, $v_0^1 = (0,7)$, et $v_0^2 = (7,0)$. L'enveloppe convexe g_{01} (calculé par (3.1)) de F prise sur S^{01} est $g_{01}(x_1, x_2) = \frac{2}{3}x_1 - \frac{10}{3}x_2$ pour tout $(x_1, x_2) \in S^{01}$. Une solution optimale pour le problème P_{01}, c'est le point extrême $x^{01} = (0, \frac{7}{3})$, avec $g_{01}(x^{01}) = -\frac{70}{9}$.

Donc à la fin de l'étape 0, $x^0 = (0, \frac{7}{3})$ et $LB = -\frac{70}{9}$.

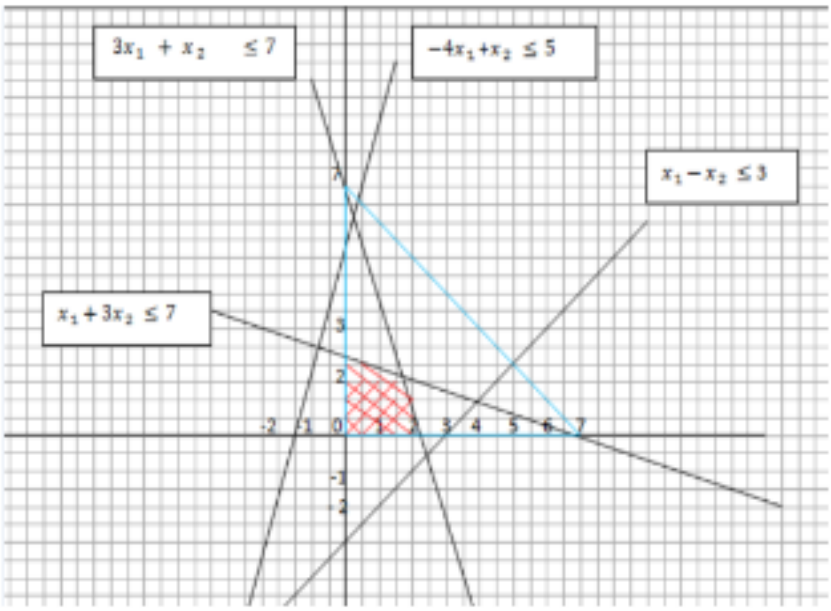


FIGURE 3.3 – Fin de l'étape 0.

étape 1.1

Puisque $x^0 = (0, \frac{7}{3}) \in S$ et $F(0, \frac{7}{3}) = -\frac{70}{9} < UB$, UB est égale à $-\frac{70}{9}$ et x^0 devient $(0, \frac{7}{3})$, aller à l'étape 1.2.

étape 1.2

$UB - LB = -\frac{7}{3} + \frac{7}{3} = 0 \leq \epsilon$, conclure que $x^0 = (0, \frac{7}{3})$ est la solution ϵ-optimale pour le problème P et arrêter.

Conclusion générale

Dans ce mémoire, nous avons étudié une méthode de résolution d'un problème de minimisation d'une forme quadratique concave avec des contraintes linéaires. C'est la méthode de Branch and Bound, où nous avons appliqué l'algorithme de cette méthode pour trouver une solution optimale globale. Il est basé sur deux étapes très importantes :

- La subdivision de l'ensembles réalisable et les ensembles des partitions.
- Le calcul des bornes inférieures et supérieures de la fonction F sur les sous-ensembles des partitions.

Notre but était de trouver une solution pour les problèmes multi-extrêmaux, c'est à dire, d'éviter les minimums locaux et de trouver une solution optimale globale.
La méthode de Branch and Bound nous assure d'arriver à un minimum global, mais elle peut engendrer un nombre d'itérations assez conséquent, ce qui rend son utilisation compliquée quand l'ensemble des points extrêmes est très grand.

Bibliographie

[1] *P.Fouilhoux,Programmation mathématique discrète et modèle linéaire ,Université Pierre et Marie Curie ,2012*.

[2] *M.Truchon, Théorie de l'Optimisation Statistique et Différentiable, Gaetan morin, Canada, 1987*.

[3] *M.O.Bibi. Cours de Programmation Linéaire et Quadratique. Master 1, Université de Béjaia, 2019*.

[4] *Aizel Boualem, Ouazib A.Malek. Minimisation d'une Forme Quadratique Concave Soumise à des Contraintes Linéaires de Type Inégalités. Mémoire de Master en Recherche Opérationnelle . Université de Béjaia, 2014*.

[5] *Farid Imakhloufen. Minimisation d'une Fonction Quadratique Concave Soumise à des Contraintes Linéaires. Mémoire de Master en Recherche Opérationnelle, Université de Béjaia, 2011*.

[6] *Djordan Ninin, Optimisation Globale basée sur l'analyse d'intervalles, Thèse de Doctorat, Université Toulouse, 2010..*

[7] *Bernard Mans, Contribution à l'algorithmique non numérique parallèle , Thèse de Doctorat , Université Paris 6, 1992..*

[8] *Pierre Hansen, Brigitte Jaumard, Christophe Meyer, Hoang Tuy. Best Simplicial and Double-Simplicial Bound for Concave Minimization. Simplicial Branch and Bound of concave minimisation, 1999*.

[9] *Larachiche Imene et Remadelia Amina. Algorithme Branch and Bound Appliqué au Problème de Sac à Dos. Mémoire de Master en Informatique,Université Djilali Bounaamâ-Khmis Miliana, 2018*.

[10] *Brian Borchirs and John E.Mitchell. A Computational Comparison of Branch and Bound and outer Approximation Algorithms 0-1 Mixed Integer Nonlinear Programs. Preprint Submitted To Elsevier , 1996*.

[11] *Nikolaos V. Sahinidis. A General Purpose Global Optimization Software Package. Journal of Global Optimization 8 : 201-205, 1996*.

[12] *L.A.Wolsey. Integer Programming. Wiley-Interscience,New Jersy,1998*.

[13] *Jaumard B, Ellaia R. and Gourdin, E. Global Optimization of Holder Function. Journal of Global Optimization, 8(4) :323 348, 1996*.

[14] *Stanislaw Walukiewicz. Integer Programming. Mathematics and its Applications. Springer, Varsovie, 1990*.

[15] *Laurent Smouch. La programmation Linéaire - Méthode Graphique. Recherche Opérationnelle, Master2 LT, MPM, MIR Université du Littoral, 2013*.

[16] *Dantzig. Linear Programming and Extensions. Princeton University Press, Princeton, 1963*.

[17] *Jaques Teghem,Recherche Opérationnelle Tome 1,Ellipses, Paris, 2012*.

[18] *Moisdon, Jean-Claude et Nakhla Michel. Méthode d'Optimisation en Gestion. Recherche Opérationnelle.Presses des MINES, Paris, 2010*.

[19] *Lilia Zaourar. Recherche Opérationnelle et Optimisation pour la Conception Testable de Circuits Intégrées Complexes. Thèse de Doctorat, Université de Grenoble, 2010*.

[20] *M.R.Garey and D.S. Johnson. A Guide to the Theory of NP-Completeness. Computers and Intractability. A Series of Books in the Mathematical Sciences.WH Freeman and Company, San Francisco, Clif, 1979*.

[21] *Ch.Papadimitriou. Combinatorial Optimization : Algorithms and Copmlexity. Prentice-Hall, New York, 1982*.

[22] *O.Goldreich. Computanional Complexity : a Conceptual Perspective. Cambridg University Press, 2008*.

[23] *Yves Nobert , Roch Ouellet et Régis Parent. La Recherche Opérationnelle, G.Morin, 1995.*

[24] *Lemke C.E. The dual method for solving the linear programming problem. Naval Research Logistic Quarterly 1, 36-47, 1954.*

[25] *Aidene M. Oukacha B. Programmation Linéaire. Les Pages Bleues internationales, Maison d'Edition pour l'enseignement et la formation, Université de Tizi-Ouzou, 2005.*

[26] *Sameh Grainia. Développement d'un Algorithme de Branch-and-Price-and-Cut pour le Problème de Conception de Réseau avec Coûts Fixes et Capacités. Mémoire Présenté à la Faculté des Arts et des Sciences en vue de l'Obtention du Grade de Maître ès Sciences en Informatique, Université de Montréal, 2015.*

[27] *Egon Balas, Matteo Fishetti et Arrigo Zanette. A Hard Integer Program Made Easy by Lexicography. Mathematical Programming, Vol 135, pp.509-514, 2011.*

[28] *L.A. Wolsey. Integer Programming. John Wiley and Sons. New York, 1998.*

[29] *Richard Bellman. On the theory of Dynamic Programming. Proceedings of the National Academy of Sciences of the United States of America, 1952.*

[30] *Michel Bierlaire. Optimisation en Nombres Entiers, Cours Recherche Opérationnelle, Ecole Polythechnique Fédérale de Lausanne, 2010.*

[31] *Harold P Benson, Reiner Horst. A Branch and Bound-Outer Approximation Algorithm for Concave Minimization over a Convex Set. Computers Math-Applic. Vol 617, pp.67-76, 1991.*

Résumé

Dans ce mémoire, l'objectif est de présenter une méthode de résolution d'un problème de minimisation concave qui est la méthode de Séparation et d'Évaluation (Branch and Bound method). On a considéré le cas où la fonction objectif est quadratique concave et les contraintes linéaires. Pour ce faire, nous avons commencé par présenter quelques définitions des éléments de base pour la résolution des problèmes d'optimisation, et nous avons aussi présenté la méthode de Branch and Bound. Ensuite, nous nous sommes intéressons à la résolution des problèmes de programmation linéaire en nombres entiers par la méthode de Séparation et d'Évaluation. La dernière partie est consacrée à la minimisation d'une fonction quadratique concave sur un polytope. Enfin, on a présenté l'algorithme de la méthode de Branch and Bound, appliqué à la minimisation concave et nous avons fini par l'appliquer sur un exemple illustratif.

Abstract

The aim of this work is to present a method of solving the problem of concave minimization, the Branch and Bound Method. We consider the case where the objective function is quadratic and the linear constraints. To do this, we begin by presenting some definitions of the basic elements for solving optimization problems, and we also present the Branch and Bound Method. Next, we are interested in solving integer linear programming problems by the method of Branch and Bound. The last part is devoted to the minimization of a concave quadratic function over a polytope. Finally, we present the algorithm of the Branch and Bound applied to concave minimization, and we end up applying it on one illustrative example.