RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE.
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE.

Université Mouloud Mammeri, Tizi-Ouzou
Faculté des Sciences
Département de Mathématiques

Mémoire de Master en Recherche Opérationnelle

Option : *Méthodes et Modèles de Décision*

Thème :

Algorithme simplicial Branch and Bound pour la minimisation concave

Présenté par :

Mansour Malika
Zaidi Fetta

Devant le jury d'examen composé de :

Mr M.Aiden	Président
Mr M.Ouanes	Rapporteur
Mme L.Goumeziane	Examinatrice
Mr B.Oukacha	Examinateur

Soutenu le 10 / 10 / 2012

Remerciements

Nous tenons à exprimer nos vifs remerciements à Mr Ouanes Mohand qui a proposé et accepté de diriger ce travail.
Nous tenons à exprimer nos gratitudes à Mr Aiden Mahamed qui nous a fait l'honneur de présider le jury.

Nous remercions vivement l'ensemble des membres du jury : Mme L.Goumeziane, Mr B.Oukacha pour l'honneur qu'ils nous ont fait en acceptant de juger ce travail.

Nos remerciements chaleureux s'adressent également à Fatiha et Fazia qui nous ont beaucoup aidé dans ce travail.

Merci à tous les enseignents de l'université Mouloud MAMMERI en général et au membre du département de mathématiques en particulier qui ont contribué à notre formation.

Nous remercions tous ceux qui ont contribué de près ou de loin à la réalisation de ce modeste mémoire.

Dédicaces

Fetta

Je dédie ce mémoire de fin d'études

À

Ma mère

À

Mes chèrs frères
Arezki, Mustapha, Abdelkrim, Lhadi et Aissa.
Mes chères sœurs
Tassadit, Nouara, Farida, Saliha et Fatiha
pour leur affection, compréhension et patience.

À

Ouali, Saddek, Hakima, Hakim, Samir, Souhila, Leticia, Fouad, Athmane, Amel et Brahim.

À

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

À

A toutes les personnes que j'aime.

Malika

Je dédie ce mémoire de fin d'études

À

Mon grand frère Rafik qui m'a beaucoup soutenue.

À

Mes frère Rachid et Farid.

À

Mes soeurs Fazia et Nacera.

À

Mon fiancé Youssef et sa famille.

À

Mon beau frère Kamel et sa famille.

À

Mes amies Chahra, Sabrina, Nadia, Zina, Radia, Amina ainsi mes copines de chambre.

À

Fetta et sa famille.

À

La famille Mansour et Alliouane.

À

Toute les personnes que j'aime.

Table des matières

Table des figures

Introduction

La minimisation concave est une branche de l'optimisation globale, où l'ensemble de leurs solutions est fini (en tous cas, il est dénombrable). Il est donc possible, en principe, d'énumérer toutes ces solutions, et ensuite de prendre la meilleur. L'inconvénient majeur de cette approche est le nombre prohibitif de solutions : il n'est guère évident d'effectuer cette énumération.
Le problème de minimisation concave $\min\{f(x)/x \in P\}$ consiste à trouver un minimum global x^* pour la fonction concave f sur un polyèdre $P = \{x \in \Re^n / Ax \leq b\,; x \geq 0\}$. où A une matrice de $\Re^{m\times n}$ et b un vecteur de $\Re^m$. Ces dernières années, la minimisation concave et la minimisation particulièrement concave sur un polyèdre ont reçu beaucoup d'attention(le livre de Horst and Tuy). D'un point de vue mathématique, la minimisation concave est une classe difficile de problème, sa difficulté principale est l'existence des minimums locaux. Beaucoup d'outils sont développés pour la minimisation convexe pour résoudre cette difficulté mais sont insuffisantes. Cette difficulté est aussi éxprimée par le fait que la minimisation concave est un NP Complet même pour le cas simple tel que minimiser une fonction quadratique sur un hypercube.
On rencontre les problèmes d'optimisation concave dans plusieurs applications telle que :

- Les problèmes d'économie et d'èconométrie.
- Les dessins de l'ingenieur.

Pour la résolution de ces problèmes, il éxiste deux approches : détérministe et stochastique. Ici on considère seulement l'approche détérministe.
La plus part des algorithmes détérministes populaires sont en général en trois classes :

- Méthodes de l'énumération.
- Méthodes d'approximation successive.
- Méthodes de Branch and Bound.

Les algorithmes de classes 1 et 2 peuvent trouver la solution optimale aprés un nombre fini d'itérations dans le mouvais cas, ce qui n'est pas vrai pour les méthodes de Branch and Bound, les méthodes de Branch and Bound garantient seulement le calcul d'une solution $\epsilon - optimale$.
Quelques algorithmes éxacts finis dans le champ des méthodes de Branch and Bound pour l'optimisation concave ont été paru dans la littérature par exemple le processus de la fin pour les algorithmes coniques de Branch and Bound, dans le champ des algorithmes réctangulaires pour les fonctions concaves séparables et dans le champ des algorithmes

simplicials Branch and Bound.
Notre étude se porte sur le dérnier champ.

Chapitre 1

Généralités

1.1 Quelques définitions

Définition 1.1.1.
soient les vecteurs $x_1, x_2, ..., x_m$ dans un espace $\Re^n$ et soient $\lambda_i \geq 0$, $i = 1, ..., m$ avec $\sum_{i=1}^{m} \lambda_i = 1$, $\lambda_1 x_1 + \lambda_2 x_2 + ... + \lambda_m x_m$ est dite combinaison convexe de ces poits. 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.

Définition 1.1.2.
Un ensemble $S \subset \Re^n$ est dit convexe si

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

L'interpritation géométrique de cette définition est que pour deux points quelconques $x_1, x_2 \in S$: le segment de droite joignant ces deux points c'est à dire
$[x_1, x_2] = \{(1 - \lambda) x_1 + \lambda x_2 / 0 \leq \lambda \leq 1\}$ est entierement inclus dans S.
En d'autres termes, l'ensemble convexe est caractérisé par la propriété qu'il contient toutes les combinaison convexes de chaque pair de ses éléments.

Définition 1.1.3.
On appelle enveloppe convexe d'un ensemble S, le plus petit ensemble convexe contenant S et noté $conv(S)$.

Définition 1.1.4.
Soit S une partie de $\Re^n$. Un point x est appellé point intérieur à S s'il éxiste $r > 0$ tel que $B_r(x) \subset S$. L'ensemble des points intérieurs à S est appelé intérieur de S, on le note $\text{int}(S)$

Définition 1.1.5.
Un poit $x \in S$ est appelé point extrême ou bien un sommet de S, s'ils n'existent pas deux points distincts $x_1, x_2 \in S$ tel que $x = \lambda x_1 + (1 - \lambda) x_2$ pour tout $\lambda \in]0, 1[$

Définition 1.1.6.
Soit la fonction $f : S \subset \Re^n \to \Re$, $x_0 \in S$.
f est continue en x_0 si

$$\forall \epsilon > 0,\ \exists\ \delta \text{ tel que } \|x - x_0\| < \delta \Rightarrow |f(x) - f(x_0)| < \epsilon$$

Définition 1.1.7.
On définit la dérivée partielle de $f : \Re^n \to \Re$ par rapport à $x = (x_1, ..., x_n)$ par la limite

$$\lim_{h \to 0} \frac{f(x_1, ..., x_i + h, ..x_n) - f(x_1, ..., x_n)}{h}.$$

quand elle existe, on la note

$$\frac{\partial f(x)}{\partial x_i}.$$

Définition 1.1.8.
On définit le gradient de $f : \Re^n \to \Re$ par

$$\nabla f(x) = \left(\frac{\partial f(x)}{\partial x_1}, ..., \frac{\partial f(x)}{\partial x_n} \right).$$

Définition 1.1.9.
On appelle epigraphe d'une fonction $f : S \subset \Re^n \to \Re$ l'ensemble

$$epi(f) : \{(x, r) \in S \times \Re \text{ tel que } f(x) \leq r\}.$$

avec S : ensemble convexe.

Définition 1.1.10.
Une fonction f définie sur un ensemble convexe S dans $\Re^n$ est dite convexe si

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

Elle dite concave si

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

Définition 1.1.11.
L'enveloppe convexe d'une fonction est la plus grande fonction convexe de $f(x)$ qui soit inférieure ou égal à f sur l'ensemble de définition.

Définition 1.1.12.
Une fonction f est dite affine sur S si $f(x)$ est finie, convexe et concave. Une fonction affine sur $\Re^n$ a la forme :

$$f(x) = < a, x > + \alpha \, avec \, a \in \Re^n,\ \alpha \in \Re.$$

Définition 1.1.13.

Soit $x_1, x_2, ..., x_{m+1}$ un nombre fini de points dans $\Re^n$, si $x_2 - x_1, x_3 - x_1, ..., x_{m+1} - x_1$ sont linéairement indépendants alors l'enveloppe convexe de $x_1, x_2, ..., x_{m+1}$, c'est à dire l'ensemble de toutes les combinaisons convexes de $x_1, x_2, ..., x_{m+1}$ sont appelées un m-simpexe dans $\Re^n$ avec les sommets $x_1, x_2, ..., x_{m+1}$. Un 0-simplexe est un point, un 1-simplexe est un segment, un 2-simplexe est un triangle.

Définition 1.1.14.

Un ensemble convexe $P \subset \Re^n$ est un polyèdre s'il est l'intersection d'une famille finie ou infinie de demi-espace fermé. En d'autres termes, un polyèdre est un ensemble de solutions d'un système fini d'inégalités linéaires de la forme

$$< a^i, x > \leq b, i = 1, ..., m.$$

où sous forme matricielle

$$Ax \leq b.$$

où A est une matrice $m \times n$, $b \in \Re^n$, m et n deux entiers positifs.

1.2 Propriétés des fonctions concaves

1. Une fonction $f(x)$ est concave sur un ensemble convexe si et seulement si la fonction $-f(x)$ est convexe sur l'ensemble.
2. Une fonction différentiable f est concave sur un intervalle si sa fonction dérivée $f^{'}$ est monotone qui diminue sur cet intervalle : une fonction concave a une inclinaison décroissante.
3. Pour une fonction deux fois dérivable. Si la dérivée second $f^{"}$ est positive (ou l'accélération est positive) alors le graphe est convexe ; Si $f^{"}$ est négative, alors le graphe est concave. points où les changements de la concavité sont des points de l'inflexion.
4. Si $f(x)$ est deux fois dérivable, alors $f(x)$ est concave si et seulement si $f^{"}(x) < 0$.
5. Si une fonction f est concave et $f(0) = 0$ alors f est subadditive.
   Preuve :
   puisque f est concave et $y = 0$

$$\begin{aligned} f(tx) &= f(tx + (1-t)0) \\ &\geq tf(x) + (1-t)f(0) \\ &= tf(x) \end{aligned}$$

$$\begin{aligned} f(a) + f(b) &= f((a+b)\frac{a}{(a+b)}) + f((a+b)\frac{b}{(a+b)}) \\ &= \frac{a}{(a+b)}f(a+b) + \frac{b}{a+b}f(a+b) \\ &= f(a+b) \end{aligned}$$

1.3 Problèmes d'optimisation globale

1.3.1 Problème d'optimisation

On définit un problème d'optimisation par :

$$\min f(x)$$

$$sc$$

$$g_i(x) \leq 0,\ i = 1, ..., m$$

$$h_j(x) = 0,\ j = 1, ..., p$$

$$x \in \Re^n$$

où x est appelé variable d'optimisation, la fonction f est appelée fonction objectif. les inégalités $g_i(x) \leq 0, i = 1, ..., m$ sont appelées les contraintes inégalités. les équations $h_j(x) = 0, j = 1, ..., p$ sont appelées les contraintes égalités. On définit le domaine d'un problème d'optimisation par :

$$D = dom f \cap_{i=1}^{m} dom\, g_i(x) \cap_{j=1}^{p} dom\, h_j(x).$$

avec *dom* :ensemble de définition des fonctions
Un point x est admissible s'il satisfait les contraintes $g_i(x) \leq 0,\ i = 1, ..., m$ et
$h_j(x) = 0,\ j = 1, ..., p$. pour tout point admissible x, l'ensemble des contraintes actives est noté par :

$$A(x) = \{j / g_i(x) = 0\}.$$

Si $j \neq A(x)$, on dit que la contrainte est inactive en x.

1.3.2 Problème d'optimisation globale

On définit un problème d'optimisation globale par :

$$\min f(x)$$

$$sc$$

$$g_i(x) \leq 0,\ i = 1, ..., m$$

$$x \in X$$

où X est un ensemble convexe fermé dans $\Re^n$, $f : \Omega \to \Re$ et $g_i : \Omega \to \Re$ sont des fonctions définis sur un ensemble Ω dans $\Re^n$ contenant X. Notons par S l'ensemble admissible défini comme suit :

$$S = \{x \in X / g_i(x) \leq 0, i = 1, ..., m\}.$$

Un point $x^* \in S$ tel que

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

est appelé une solution optimale globale (minimum global) du problème.
un poit $x^{'} \in S$ tel qu'il existe un voisinage V de $x^{'}$ qui satisfait $f(x^{'}) \leq f(x), \forall x \in S \cap V$, est appelé une solution optimale locale.
Quand le problème est convexe, c'est à dire, toutes les fonctions $f(x)$
et $g_i(x) \leq 0,\ i = 1, ..., m$ sont convexes, tout minimum local est global.
Un problème est dit multi-extrémal quand il possède plusieurs minimums locaux avec des valeurs objectifs correspondantes différentes (de sorte qu'un minimum local peut ne pas être un minimum global).

Exemple 1.3.1.

La fonction $f(x, y) = x^4 + y^4 - (x - y)^2$ admet deux minimums locaux qui sont (-1,1) et (1,-1) sur l'intervalle $[-2, 2] \times [-2, 2]$.

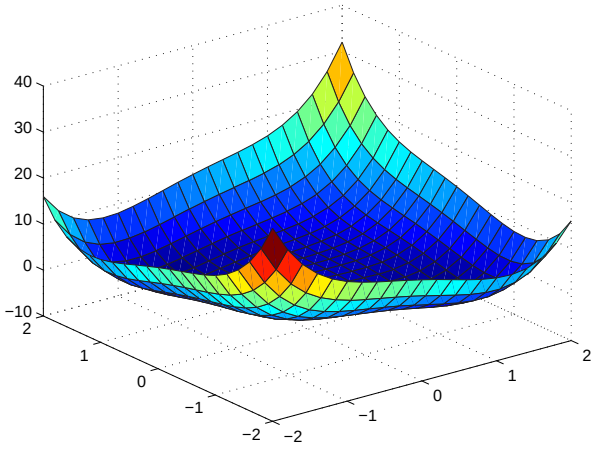


FIG. 1.1 – Fonction avec deux minimums locaux

Remarque 1.3.1.

Dans la pratique, calculer une solution optimale exacte peut être trop couteux, pendant que plus souvent nous avons besoin seulement d'une solution $\epsilon - optimale$.
Donc le problème peut être résolu si un point $x^{'}$ a été trouvé :

$$f(x^{'}) - \epsilon \leq f(x), \forall x \in X.$$

Chapitre 2

La méthode de Branch and Bound de base

Branch and Bound est un algorithme assez général qui joue un rôle trés important dans la théorie d'optimisation globale. L'idée générale de la méthode est de décomposer le problème primaire en sous problèmes parallèlles (Branching) qui peut être graduellement plus facile à résoudre, puis évaluer les bornes inférieures et supérieures (Bounding) des valeurs des solutions optimales sur ces problémes secondaires. Par conséquent, le procédé Branch and Bound peut être représenté sous forme d'un arbre : le probléme initial est situé comme racine de cet arbre et les branches sont les sous problèmes hiérarchiquement construits par l'algorithme. Cette construction est régie par la stratégie de la recherche qui déterminera la suite de solutions des problèmes secondaires. La séquence de la décomposition et la recherche de la solution continue jusqu' à ce qu'il puisse vérifier que l'un ou l'autre sur la branche indiquée ne peut pas apporter une meilleure solution que la solution du condidat sortant (trouvé déja par le procédé Branch and Bound).
Notons que Branch and Bound a été à l'origine développée pour résoudre des problème de la programation entiére. Plus tard, cet algorithme a été appliqué avec succés dans des problèmes trés difficiles en optimisation globale comme : la minimisation des fonctions lipchitziennes, la minimisation de la différence de deux fonctions convexes, et aussi la minimisation concave qui sera traité en détail dans les chapitres suivants.

2.1 Le procédé de l'algorithme Branch and Bound :

Définition 2.1.1.
Considérons le problème d'optimisation globale :

$$\min_{x \in D} f(x) \tag{2.1}$$

où $f : A \to \Re,\ D \subset A \subset \Re^n$.

Définition 2.1.2.

Soit P un polyèdre dans $\Re^n$, tel que $int(P) \neq \emptyset$, et I un ensemble fini d'indices, la famille $\{P_i : i \in I\}$ de sous-polyèdres ($int(P_i) \neq \emptyset$) est dite partition de P, si

$$P = \cup_{i \in I} P_i \, et \, P_i \cap P_j = \partial P_i \cap \partial P_j, \quad \forall i, j \in I, \; i \neq j$$

où ∂P_i est la frontière de P_i.

Définition 2.1.3.

Soit M un élément de la partition d'un polyèdre $P(D \subset P)$

* Si $M \cap D = \emptyset$ M est dit infaisable
* Si $M \cap D \neq \emptyset$ M est dit faisable
* Sinon M est dit incertain

2.1.1 L'algorithme Branch and Bound de base

– Etape 0 : Initialisation :
Choisir

$$P_0 \supseteq D$$

$$S_{P_0} \subset D$$

$$-\infty < \beta_0 \leq \min f(D)$$

P_0 est le plus petit polyèdre qui contient D si ce dernier n'est pas un polyèdre.
S_{P_0} est un ensemble de points choisis dans l'ensemble faisable.
Poser

$$\rho_0 = \{P_0\}$$

$$\alpha_0 = \min f(S_{P_0})$$

$$\beta_0 = \beta(P_0)$$

– Si $\alpha_0 < +\infty$, alors choisir

$$x^0 \in arg \min f(S_{P_0}) \, i.e : \; f(x^0) = \alpha_0.$$

– Si $\alpha_0 - \beta_0 \leq \epsilon$, alors arrêter et poser $\alpha_0 = \beta_0 = \min f(D)$
– Sinon, aller à l'étape 1

– Etape k : $(k = 1, 2, ...)$
Au début de chaque étape k, on a la partition actuelle ρ_{k-1} de sous-ensembles de P_0 qui reste aprés élimination. De plus, pour tout $P \in \rho_{k-1}$, on a un ensemble de points $S_P \subseteq D \cap P$ et des bornes $\beta(P), \alpha(P)$ vérifiant :

$$\begin{cases} \beta(P) \leq inf f(P \cap D) \leq \alpha(D), \; si \; P \; est \; faisable \\ \beta(P) \leq inf f(D), \; si \; P \; est \; incertain \end{cases} .$$

On a aussi, les bornes β_{k-1}, α_{k-1} qui vérifient :

$$\beta_{k-1} \leq inf f(D) \leq \alpha_{k-1}.$$

Si $\alpha_{k-1} < +\infty$, alors on aura un point $x^{k-1} \in D$ tel que :

$$f(x^{k-1}) = \alpha_{k-1}$$

– k.1.Eliminer tout sous ensemble $P \in \rho_{k-1}$ qui vérifie :

$$\beta(P) \geq \alpha_{k-1}$$

Considérer ℓ_k la classe des sous ensembles restants de ρ_{k-1}.

– k.2.Sélectionner une classe d'ensembles non vide $\Im_k \subset \ell_k$ et construire une nouvelle partition avec chaque élément de $\Im_k$.
Soit $\Im_k^{'}$ la classe de tous les nouveaux éléments de la nouvelle partition.

– k.3.Eliminer chaque $P \in \Im_k^{'}$ qui vérifie :

$$P \cap X = \emptyset$$

Poser $\rho_k^{'}$ la classe de tous les ensembles restants de $\Im_k^{'}$.

– k.4.Déterminer pour chaque $P \in \rho_k^{'}$ un ensemble $S_P \subseteq P \cap D$ et un nombre $\beta(P)$ tels que :
$\begin{cases} \beta(P) \leq inf f(P \cap D) \leq \alpha(P), \ si\ P\ est\ faisable \\ \beta(P) \leq inf f(P), \ si\ P \cap D\ est\ incertain \end{cases}$.
De plus il faut avoir :

$$S_P \supset S_{P'} \cap P\ et\ \beta(P) \geq \beta(P^{'})$$

pour tout ensemble $P^{'} \supset P$ de ρ_{k-1}.
Poser

$$\alpha(P) = \min f(S_P)$$

– k.5.Soit

$$\rho_k = \{\ell_k \setminus \Im_k\} \cup \rho_k^{'}$$

Calculer

$$\alpha_k = \min\{\alpha(P) : P \in \rho_k\}$$

$$\beta_k = \min\{\beta(P) : P \in \rho_k\}$$

Si $\alpha_k < +\infty$, alors déterminer $x^k \in D$ tel que :

$$f(x^k) = \alpha_k$$

– k.6.
  – Si $\alpha_k - \beta_k \leq \epsilon$ alors arrêter et poser $\alpha_k = \beta_k = \min f(D)$ (x^k est la solution optimale)
  – Sinon, poser $k = k + 1$
  Aller à l'Etape k.

Remarque 2.1.1.

1. Pour qu'un élément P de la partition ρ_k, sera supprimé à l'étape k il faut qu'il soit un élément "sondé", c-à-d il doit vérifier la condition $\beta(P) \geq \alpha_{k-1}$. Alors le critère d'arrêt $\alpha_k = \beta_k$ signifie que tous les éléments de la partition sont sondés.
2. L'étape k est exécutée seulement s'il reste des éléments de la partition ℓ_k ; alors il suffit d'exiger que $\ell_k \subset \Im$, c-à-d ; chaque élément sondé de la partition peut être encore divisé. En général, l'opération de la subdivition est définie seulement sur une certaine famille $\Im$ de sous-ensemble de $\Re^n$ (par exemple, rectangles, cônes, simplexes).
3. Dans l'étape k.4 on peut évidemment remplacer n'importe quel $P \in \rho_k^{'}$ par un plus petit ensemble $\widetilde{P} \subset P$ tel que $\widetilde{P} \in \Im$,$\widetilde{P} \cap D = M \cap D$.
4. Pour chaque partition P, S_P est une collection de points faisables incluse dans P, elle peut être rénovée au long de l'exécution de l'algorithme.
   Au cour de l'algorithme , l'ensemble S_P est toujours fini. Alors le procédé ci-dessus est également défini dans le cas où les ensemble S_P sont vides et on peut avoir $\alpha_k = \infty$ pour tous k.
   D'autre part il faut imposer des conditions sur S_P et $\beta(P)$ pour que $\{\alpha_k\} = \{f(x^k)\}$ soit une suite décroissante, $\{\beta_k\}$ une suite croissante, et $\alpha_k \geq \min f(D) \geq \beta_k$, afin que la différence $\alpha_k - \beta_k$ peut mesurer approximativement la meilleure solution optimale x^k à l'étape k.
   Pour une tolérance donnée $\epsilon > 0$, l'algorithme va s'arrêter dés que $\alpha_k - \beta_k < \epsilon$. Et puisque $\{\alpha_k\}$ a une monotonie décroissante et $\{\beta_k\}$ une monotonie croissante, alors les limites $\alpha = \lim\limits_{k\to\infty} \alpha_k$ et $\beta = \lim\limits_{k\to\infty} \beta_k$ vont exister, et par récurrence, il vont satisfaire $\alpha_k \geq \min f(D) \geq \beta_k$.
   L'algorithme est dit fini si $\alpha_k = \beta_k$ à une certaine étape, tandis qu'il converge si $\alpha_k - \beta_k \to 0$, c-à-d $\alpha = \lim\limits_{k\to\infty} f(x^k) = \beta = \min f(D)$.

2.2 Condition d'arrêt et de convergence

Si le procédé s'achève à l'itération k, alors évidemment, x^k est la solution optimale et α_k est la valeur optimale de la fonction objectif. Cependant, on ne peut pas garantir l'arrêt de l'algorithme en un nombre fini d'itérations, on doit donc établir des conditions qui assurent que tous point d'accumulation de la suite $\{x^k\}$ est une solution optimale du problème (2.1). Notons d'abord que si l'algorithme est infini, alors il doit générer au moins une suite "extraite" $\{P_{k_q}\}$ de l'ensembles P_{k_q} issus des partitions successivement raffinées et vérifiant $P_{k_q} \supset P_{k_{q+1}}$. Ceci est une conséquence immédiate du fait que, à chaque itération k, la classe ρ_k contient seulement un nombre fini d'ensembles issus de la partition.Dans l'arbre qui représente la procédure Branch and Bound la suite infinie extraite correspond à une branche dont les noeuds sont P_{k_q} $(q = 1, 2, ...)$.

Théorème 2.2.1.

Si pour toute suite infinie $\{P_{k_q}\}$, $P_{k_q} \supset P_{k_{q+1}}$ $(q = 1, 2, ...)$ *d'ensembles de parttitions successivement raffinées, et toutes les bornes à l'itération* k_q *on a :*

$$\lim_{q\to\infty}(\alpha_{k_q} - \beta_{k_q}) = \lim_{q\to\infty}(\alpha_{k_q} - \beta(P_{k_q})) = 0, \tag{2.2}$$

alors

$$\beta = \lim_{k\to\infty}\beta_k = \lim_{k\to\infty} f(x^k) = \lim_{k\to\infty}\alpha_k = \alpha, \tag{2.3}$$

et chaque point d'accumulation x^* *de la suite* $\{x^k\}$ *est une solution optomale de* $\min\{f(x) : x \in D\}$

Preuve :

Supposons que $\{x^k\}$ une suite infinie, puisque D est un compact, et $\{x^k\} \subset D$, alors la suite $\{x^k\}$ possède un point d'accumulation.
D'autre part soit x^* un point d'accumulation de $\{x^k\}$, donc il existe une sous-suite appartenant à $\{x^k\}$ qui converge vers x^*. Par construction des ensembles P_k, cette sous-suite doit contenir une sous-suite infinie $\{x^{k_q}\}$ telle que $P_{k_q} \supset P_{k_{q+1}}$ $(q = 1, 2, ...)$.
On a alors d'aprés les hypothèse du théorème, et par la continuité de f sur le compact D on a :

$$\lim_{q\to\infty} f(x^{k_q}) = f(x^*).$$

Notons $f^* = \{f(x) : x \in D\}$, la suite des bornes inférieures $\{\beta_k\}$ satisfait les conditions $\beta_{k+1} > \beta_k$ et $\beta_k < f^*$ alors sa limite existe, posons $\beta = \lim_{k\to\infty}\beta_k$. En plus, la suite des bornes supérieures $\{\alpha_k\}$ satisfait $\alpha_{k+1} < \alpha_k$ et $\alpha_k > f^*$ alors elle converge vers une limite finie $\alpha(\alpha = \lim_{k\to\infty}\alpha_k)$. On déduit donc que :

$$\beta \le f^* \le \lim_{k\to\infty} f(x^k) = f(x^*) = \alpha,$$

mais avec la condition (2.3) qui s'écrit aussi sous forme

$$\lim_{q\to\infty}\alpha_{k_q} = \lim_{q\to\infty}\beta_{k_q},$$

et

$$\begin{cases} \lim_{q\to\infty}\alpha_{k_q} = \lim_{k\to\infty}\alpha_k, \ car\ \{\alpha_{k_q}\}\ est\ une\ sous-suite\ de\ \{\alpha_k\} \\ et \\ \lim_{q\to\infty}\beta_{k_q} = \lim_{k\to\infty}\beta_k, \ car\ \{\beta_{k_q}\}\ est\ une\ sous-suite\ de\ \{\beta_k\} \end{cases}$$

on déduit que :

$$\lim_{k\to\infty}\alpha_k = \lim_{k\to\infty}\beta_k.$$

Remarque 2.2.1.

En regardant les différentes étapes de l'algorithme, on voit qu'il se base sur deux étapes trés importantes :

– La subdivision de l'ensembles faisable et les ensembles de partitions.
– Le calcul des bornes inférieures et supérieures de la fonction f sur les sous ensembles de partitions.

Chapitre 3

Algorithme simplicial Branch and Bound

On considère un problème de minimisation général avec une fonction objectif concave sur un polyédre $P \subset \Re^n$. Dans la plupart des application, P est convexe et compact défini par une famille d'inégalités $g_i(x) \leq 0\,(i = 1...m)$, où les fonctions g_i sont convexe sur un ensemble convenable $A \subseteq \Re^n$.
La plupart des problème de minimisation concave sont issus des problèmes de la recherche opérationnelle et des sciences économiques, il existe même plusieurs modèles de problèmes important dont la fonction objectif n'est pas nécessairement concave qui peut être transformer en fonction concave (programmation bilinéaire, programmation multiplicative).
Une des propriétés les plus intéressantes dans la minimisation concave c'est quand la fonction objectif est concave sur un domaine faisable P convexe ; alors elle atteint toujours son minimum global sur la frontière de P comme l'indique le théorème suivant :

Théorème 3.0.2.
Soit $S \subset \Re^n$, un ensemble compact et convexe, et $f : S \to \Re$ une fonction concave. Alors f atteint son minimum global en un point extrémal de S.

preuve :

Pour chaque point $x \in S$, on a la représentation suivante :
$x = \sum_{i=1}^{n} \lambda_i v^i,\ \sum_{i=1}^{n} \lambda_i \geq 0\ (i = 1...n), n \leq dim(S) + 1,\ \{v^i\},\ 1 \leq i \leq n$ sont des points extrémaux de S. La concavité de f implique :
$f(x) \geq \sum_{i=1}^{n} \lambda_i f(v^i) \geq \sum_{i=1}^{n} \lambda_i \min\{f(v^i) : i = 1, ..., n\} = \min\{f(v^i) : i = 1, ..., n\}.$

3.1 Paramètres de problème

$f : \Re^n \to \Re$ une fonction concave, le problème est définie comme suit :

$$\begin{aligned} &\min f(x) \\ &sc \\ &Ax \leq b \\ &\\ &x \geq 0 \end{aligned} \tag{3.1}$$

$P = \{x \in \Re^n / Ax \leq b, x \geq 0\}$: est l'esemble réalisable

Condition sur P et f :

1. P non vide et borné.
2. f continue et différentiable.
3. ∇f est continu.
4. $\frac{\partial f(x)}{\partial x_i}$ est continue.

Pour résoudre le problème (3.1), l'algorithme Branch and Bound suit les même étapes que celles présentées dans l'algorithme de base. Sauf que l'ensemble initial P_0 et tous les ensembles des partitions P_k ont une forme plus précise comme : simplexes, cônes polyèdriques ou des pavés.
Présenons une des extension de l'algorithme Branch and Bound dans le cas où les ensembles de partition sont des simplexes.

3.2 Principe de l'algorithme

Calcul du simplexe Δ^1

$$\lambda = \max\{e^T x / x \in P\}, e \in \Re^n. \tag{3.2}$$

$$\Delta^1 = \{x \in \Re^n / e^T x \leq \lambda, x \geq 0\}. \tag{3.3}$$

Dans l'algorithme on subdivise $\Delta^1 \supset P$ au sous ensemble

$$\{\Delta_i,\ i \in S\},\ \cup\Delta_i = \Delta^1,\ S : ensemble\, des\, indices,\ int(\Delta_i) \cap int(\Delta_j) = \emptyset\ si\ i \neq j \tag{3.4}$$

int(.) représente l'ensemble des points intérieurs.

Calcul de la borne inf de $\{f(x)/x \in P \cap \Delta_i\}$

Pour chaque $i \in S$, on calcule $z(\Delta_i)$ comme suit :

$$z(\Delta_i) \leq \min\{f(x)/x \in P \cap \Delta_i\} \tag{3.5}$$

Supposons qu'on a une solution réalisable x^0.
Si $z(\Delta_i) \geq f(x^0)$, il n'existe pas $x \in P \cap \Delta_i$ tel que $f(x) < f(x^0)$.
Donc si $z(\Delta_i) \geq f(x^0)$, pour tout $i \in S$, on voi que x^0 est une solution optimale globale car :

$$\cup\Delta_i = \Delta^1, \; avec \; \Delta^1 \supset P.$$

On subdivise Δ_i au sous simplexes et on le remplace par ces sous simplexes
si $z(\Delta_i) < f(x^0)$ et on calcul la borne inf de $\{f(x)/x \in P \cap \Delta_i\}$.
Le sous problème $\min\{f(x)/x \in P \cap \Delta_i\}$ est aussi un problème de minimisation concave, donc on peut pas le résoudre directement. Par conséquant, on définit une fonction linéaire f_i pour chaque $i \in S$ tel que

$$\forall x \in \Delta_i, \; f_i(x) \leq f(x) \tag{3.6}$$

et on résoud :

$$\min f_i(x)$$

$$sc$$

$$x \in P \cap \Delta_i. \tag{3.7}$$

à la place de résoudre :

$$\min\{f(x)/x \in P \cap \Delta_i\}$$

3.3 Algorithme simplicial Branch and Bound

L'algorithme simplicial Branch and Bound est défini comme suit :
L :=$\{1\}$, k :=1.
Répéter les étapes 1-3 jusqu'a $L = \emptyset$.

Etape 1 : Sélectionner le sous problème

sortir un indice i de L.

Etape 2 : borner les opérations

Calculer la borne inf de $\{f(x)/x \in P \cap \Delta_i\}$.
Mettre à jour x^0 avec la meilleure solution obtenue (x^0 : le titulaire)
Si $z(\Delta_i) \geq f(x^0)$, enlever i de L et retourner à l'étape 1.

Etape 3 : Opération de branchement

Subdiviser le simplexe Δ_i aux deux simplexes Δ^{2k}, Δ^{2k+1} et ajouter leur indices à L
k :=k+1.
La borne inf $z(\Delta_i)$ est mise à ∞ si $P \cap \Delta_i = \emptyset$.
Quand L devient vide dans ce processus, nous voyons que x^0 est une solution optimale globale au problème (3.1).
Cependand, L ne serait pas vide en général et le processus génère une suite infinie de simplexes

$$\{\Delta^{kq}/q = 1, 2, ...\}, \Delta^{k1} \supset \Delta^{k2} \supset ... \; et \; P \cap (\cap_{q=1}^{\infty} \Delta^{kq}) \neq \emptyset \tag{3.8}$$

où Δ^{kq} : le simplexe à l'itération kq.
Pour garantir la fin de l'algorithme, nous devons introduire une tolérence $\epsilon > 0$ pour revenir au critère $z(\Delta_i) \geq f(x^0)$ de l'étape 2 de l'algorithme comme suit :

$$z(\Delta_i) + \epsilon \geq f(x^0), \; z(\Delta_i) + \epsilon |f(x^0)| \geq f(x^0). \tag{3.9}$$

Une règle de subdivision se fait $\cap_{q=1}^{\infty} \Delta_{kq}$ a singleton est appelé exhaustive.
L'exhaustivité est importante pour prouver la convergence de l'algorithme simplicial Branch and Bound.

Régle de la subdivision

Nous subdivisons le simplexe

$$\Delta = conv(\{v_1, v_2, ..., v_{n+1}\}) \tag{3.10}$$

avec $v_1, v_2, ..., v_{n+1}$ des sommets comme suit :
Nous sélectionnons d'abord la plus longue arête de Δ et on la divise.
1-Nous trouvons une paire de sommets comme suit :

$$\|v_p - v_q\| \geq \|v_i - v_j\|, \; \forall (i, j) \neq (p, q). \tag{3.11}$$

2-Nous calculons les subdivisions du point v de l'arête (v_p, v_q) comme :

$$v = (1 - \alpha)v_p + \alpha v_q, \; avec \; \alpha \in [0, \frac{1}{2}]. \tag{3.12}$$

3- Nous déffinissons deux nouveaux simplexes $\Delta^{'}\Delta^{"}$

$$\begin{aligned} \Delta^{'} &= conv(\{v_j/j \neq p\} \cup \{v\}) \\ \Delta^{"} &= conv(\{v_j/j \neq q\} \cup \{v\}) \end{aligned} \tag{3.13}$$

Si la règle de la bissection et (3.9) est adoptée, on peut obtenir une solution $\epsilon - optimale$ globale de (3.1) aprés un nombre fini d'itérations, en utilisant l'une ou l'autre règle de la sélection à l'étape 1 :

Règle de sélection 1 : La première profondeur

Aprés la subdivision du simplexe Δ en $\Delta^{'}\Delta^{"}$, on le met à la tête de la liste pour chaque itération.

règle de la sélection 2 : La meilleure limite

Aprés la subdivision du simplexe Δ en $\Delta^{'}\Delta^{"}$, nous les insérons à la liste comme suit :

$$z(\Delta) \leq z(\Delta^{'})[z(\Delta^{"})].\, \forall \Delta\, devant\, \Delta^{'}[\Delta^{"}]\, dans\, la\, liste. (3.15)$$

$$z(\Delta) \geq z(\Delta^{'})[z(\Delta^{"})].\, \forall \Delta\, au\, fond\, \Delta^{'}[\Delta^{"}]\, dans\, la\, liste. (3.16)$$

Chapitre 4

Mise à jour de l'algorithme simplicial Branch and Bound

4.1 Relaxation linéaire de l'algorithme

A l'étape 2 de l'algorithme, on remplace la fonction objectif $\min\{f(x),\ x \in P \cap \Delta\}$ par son enveloppe convexe g sur Δ et on résoud le problème suivant :

$$Q(\Delta) = \begin{cases} \min g(x) \\ sc \\ x \in P \cap \Delta \quad . \end{cases} \tag{4.1}$$

Avec $w(\Delta)$: valeur optimale

L'enveloppe convexe g est une fonction convexe maximale qui sous-estime f sur Δ c'est à dire une fonction affine qui s'accord avec f au $n+1$ sommets de Δ puisque Δ est donné par les sommets $v_1, v_2, ..., v_{n+1}$ comme :

$$\Delta = \{x \in \Re^n / x = \sum_{i=1}^{n+1} \mu_i v_i,\ \sum_{i=1}^{n+1} \mu_i = 1,\ \mu = (\mu_1, \mu_2, ..., \mu_{n+1})^T \geq 0\} \tag{4.2}$$

Nous pouvons déterminer la valeur de g en tout point $x \in \Delta$. Si x est donné comme combinaison convexe de $v_1, v_2, ..., v_{n+1}$, comme f est continue et concave, nous avons :

$$\forall x \in \Delta \quad g(x) = \sum_{i=1}^{n+1} \mu_i f(v_i).$$

$$\begin{aligned} g(x) &= \mu_1 f(v_1) + ... + \mu_{n+1} f(v_{n+1}) \\ &\leq f(\mu_1 v_1 + ... \mu_{n+1} v_{n+1}) \\ &\leq f(\sum_{i=1}^{n+1} \mu_i v_i) \\ &= f(x) \end{aligned}$$

d'où

$$g(x) \leq f(x) \tag{4.3}$$

Aprés substitution de (4.2) dans le sous problème $Q(\Delta)$, nous avons un programme linéaire équivalent :

$$\begin{gathered} \min f^T \mu \\ sc \\ AV\mu \leq b \\ e^T \mu = 1, \mu \geq 1 \end{gathered} \tag{4.4}$$

où

$$V = [v_1, ..., v_{n+1}], f = [f(v_1), ..., f(v_{n+1})] \tag{4.5}$$

Puisque P est borné, (4.1) a une solution $\mu^{'}$ si seulement si $P \cap \Delta \neq \emptyset$
De l'inégalité (4.3) on peut mettre la borne inf $z(\Delta)$ comme :

$$z(\Delta) = \begin{cases} f^T \mu^{'} & \text{si } P \cap \Delta \neq \emptyset \\ \infty & \text{sinon.} \end{cases} \tag{4.6}$$

Quand $P \cap \Delta \neq \emptyset$ nous avons aussi une solution réalisable $x^{'}$ au sous problème $\min\{f(x), x \in P \cap \Delta\}$.
D'où au problème (3.1) donc on peut mettre à jour x^0 en fonction de $x^{'}$ si nécessaire.

$$et\ soit\ x^{'} = V\mu^{'}. \tag{4.7}$$

4.2 Difficultés de l'algorithme

(a) chaque (4.4) associe avec $Q(\Delta)$ un ensemble different de contraintes.

(b) (4.1) n'hérite pas de la structure du problème (3.1).

Remarque 4.2.1.

* Malgré le nombre vaste de (4.4), on résoud d'abord la convergence.
* Les solutions précédentes sont moins utilisées pour ce problème à cause de (a), d'ailleur même si le problème (3.1) a une structure favorable comme réseau courant nous empèche d'appliquer les algorithmes effectifs à (4.4).

4.3 Algorithme simplicial branch and bound pour la résolution du problème (3.1) avec ϵ

Algorithme SSBB (f, p, ϵ)

début
calculer $\lambda = \max\{e^T x / x \in P\}$

$\Delta^1 = conv(\{0, \lambda e_1, \lambda e_2, ..., \lambda e_n\})$;
$L := \{1\}\,; k := 1; z^0 := +\infty$;
Tant que $L \neq \emptyset$ faire
début
sélectioner un indice $i \in L$;
$L := L \backslash \{i\}$;
$\Delta := \Delta_i$;
Calculer l'enveloppe convexe g de f sur Δ;
Résoudre le problème :
$min\{g(x)/x \in P \cap \Delta\}$;
z^k : la valeur optimale ;
x^k : la solution optimale ;
Si $z^k < z^0 - \epsilon$ alors
début
Si $f(x^k) < z^0$ alors
$z^0 := f(x^k)$;
$x^0 := x^k$;
$(\Delta^{2k}, \Delta^{2k+1}) := SPLIT(\Delta)$;
$L := L \cup \{2k, 2k+1\}$;
fin
k :=k+1 ;
fin
fin.
La procédure SPLIT est définie comme suit :
début
Sélectionner la plus longue arête $(v_p, v_q)\, de\ \Delta$
$v := (1-\alpha)v_p + \alpha v_q$; pour α fixe $\in [0, \frac{1}{2}]$.
$\Delta^{'} = conv(\{v_j/j \neq p\} \cup \{v\})$;
$\Delta^{"} = conv(\{v_j/j \neq q\} \cup \{v\})$;
return$(\Delta^{'}, \Delta^{"})$
fin.

4.4 La convergence de l'algorithme SSBB

Proposition :

On suppose que l'algorithme SSBB est infini. Si chaque suite infinie $\{\Delta^{k_q}\}$, $\Delta^{k_{q+1}} \subset \Delta^{k_q}$ génère successivement par l'algorithme de la bissection des simplexes est exhaustive, alors :

$$\lim_{k\to\infty} w^k = \lim_{k\to\infty} f(x^k) \tag{4.8}$$

Et chaque point d'accumulation x^* de la suite $\{x^k\}$ est une solution optimale.

Preuve :

Il suffit de démontrer que

$$\lim_{q\to\infty}[f(x^*(\Delta^{k_q})) - g(x^*(\Delta^{k_q}))] = 0 \tag{4.9}$$

ce qui implique que

$$\lim_{q\to\infty}(z(\Delta^{k_q}) - w(\Delta^{k_q})) = 0$$

puisque $z(\Delta^{k_q}) \leq f(x^*(\Delta^{k_q})), \text{et } w(\Delta^{k_q}) = g(x^*(\Delta^{k_q}))$
d'aprés l'exhaustivité de $\{\Delta^{k_q}\}$ c-à-d, $\lim_{q\to\infty}\Delta^{k_q} = \{v^{'}\}$, on a

$$\lim_{q\to\infty} x^{k_q} = v^{'},\ \forall\{x^{k_q}\} \subset \Delta,\ q \geq 1,$$

et puisque f est continue, on déduit que $\lim_{q\to\infty} f(x^{k_q}) = f(v^{'})$, il s'ensuit en particulier :

$$\lim_{q\to\infty} f(x^*(\Delta^{k_q})) = f(v^{'})$$

puisque, pour tout $q = 1, 2, ...$ la fonction affine g atteint ses minimiseurs dans Δ^{k_q} sur les sommets de Δ^{k_q} où elle coincide avec f, ce qui laisse a conclure que

$\min\{f(v) : v \in \textit{ au sommets de } \Delta^{k_q}\} \leq g(x^*(\Delta^{k_q})) \leq \max\{f(v) : v \in \textit{ au sommets de } \Delta^{k_q}\},\ q = 1, 2, ...$

En tendant q vers ∞, on a $\lim_{q\to\infty} g(x^*(\Delta^{k_q})) = f(v^{'})$ ce qui donne (4.9)

Chapitre 5

Révision du simplicial algorithme

5.1 Nouvelle relaxation pour la résolution des difficultés

pour résoudre les deux difficultés (a) et (b) déja décrites dans le chapitre 4, nous proposons la relaxation suivante :
Nous faisons tomber la contrainte bornée $x \in \Delta$ de $Q(\Delta)$ et nous remplaçons la fonction objectif g par un plus simple sous-estimateur h de la fonction f.
Pour ce but, premièrement, nous calculons le vecteur gradient $d = \nabla f(\mu)$ de f au point

$$\mu = \sum_{i=1}^{n+1} v_i/(n+1) \; de \; \Delta \tag{5.1}$$

Soit : $\alpha = \min\{f(x) - d^T x / x \in v_1, ..., v_{n+1}\}$ et soit

$$h(x) = d^T x + \alpha \tag{5.2}$$

Par (5.1) et (5.2) et la continuité de f, on obtient le lemme suivant :

Lemme 5.1.1.
Pour chaque $x \in \Delta$, nous avons

$$h(x) \leq f(x) \tag{5.3}$$

Preuve :

Puisque f est concave, nous avons vue de (5.1) que

$$\alpha = \min\{f(x) - d^T x / x \in \Delta\} \tag{5.4}$$

Nous avons donc

$$\forall x \in \Delta, \; f(x) - d^T x \geq \alpha \tag{5.5}$$

c'est à dire

$$d^T x + \alpha \leq f(x)$$

$$h(x) \leq f(x)$$

Par conséquent, nous résolvons le problème suivant :

$$Q^{'}(\Delta) = \begin{cases} \min h(x) \\ sc \\ x \in P \end{cases} \tag{5.6}$$

avec $w^{'}(\Delta)$: la valeur optimale.
On a : P non vide et borné, $Q^{'}(\Delta)$ a une solution $x^{'}$ optimale et nous voyons de lemme (5.1.1) que la valeur optimale $Q^{'}(\Delta)$: est une borne inférieure de $\{f(x)/x \in P \cap \Delta)\}$. C'est à dire que nous pouvons mettre la borne inf $z(\Delta)$ comme

$$z(\Delta) = w^{'}(\Delta) \tag{5.7}$$

Lemme 5.1.2.

$$\min\{h(x)/x \in P\} \leq \min\{f(x)/x \in P \cap \Delta\} \tag{5.8}$$

Preuve :

On a $P \cap \Delta \subset P$, nous avons :

$$\min\{h(x)/x \in P\} \leq \min\{h(x)/x \in P \cap \Delta\} \tag{5.9}$$

Donc nous voyons de lemme (5.1.1) que

$$\min\{h(x)/x \in P \cap \Delta\} \leq \min\{f(x)/x \in P \cap \Delta\} \tag{5.10}$$

de (5.9) et (5.10) nous avons (5.8).
Puisque $x^{'}$ est une solution réalisable pour le problème objectif (3.1), on peut mettre à jour le titulaire x^0 avec $x^{'}$ si nécessaire. Aussi nous voyons qu'on peut obtenir la borne inférieure de $\{f(x)/x \in P \cap \Delta\}$ sans difficultés (a) et (b) si nous résolvons $Q^{'}(\Delta)$.
Mais, rappellons que la fonction objectif g de $Q(\Delta)$ est un enveloppe convexe de f c'est à dire une fonction convexe maximale qui sous-estime f sur Δ. Cela implique que :

$$\forall x \in P \cap \Delta,\ h(x) \leq g(x) \tag{5.11}$$

donc nous avons

$$w^{'}(\Delta) \leq w(\Delta) \tag{5.12}$$

Malheureusement, la borne inférieure obtenue en résolvant $Q^{'}(\Delta)$ peut être inférieure à celle obtenue en résolvant $Q(\Delta)$. Cependant, la solution optimale $x^{'}$ de $Q^{'}(\Delta)$ est un sommet de

P, pendant que celle de $Q(\Delta)$ ne l'est pas. Avec cette propriété nous proposons un chemin pour trouver la meilleure solution réalisable x^* tel que :

$$f(x^*) \leq f(x^{'}) \tag{5.13}$$

Premièrement, nous résolvons le problème suivant par la méthode du simplexe , avec $x^{'}$ solution initiale réalisable.

$$\begin{gathered} \min \nabla f(x')^T x \\ sc \\ x \in P \end{gathered} \tag{5.14}$$

Ce processus peut être décrit comme suit :
Procedure SEARCH($x^{'}$)// Search 0
begin
return $x^* \in argmin\{\nabla f(x')^T x / x \in P\}$.
end
L'autre est que nous résolvons le problème suivant aprés avoir résolu (5.14) par la méthode du simplexe qui utilise $x^{'}$ comme solution initiale réalisable.

$$\begin{gathered} \min f(x) \\ sc \\ x \in S \end{gathered} \tag{5.15}$$

où S est l'ensemble des sommets de P génèré en pivotant la procédure pour résoudre (5.14), ce processus peut être décrit comme suit :
Procedure SEARCH($x^{'}$)//Search 1.
begin
résoudre le problème $\min\{\nabla f(x')^T x / x \in P\}$ avec la méthode du simplexe et soit S l'ensemble des solutions réalisables génèrées par la procédure de pivotage.
return $x^* \in argmin\{f(x)/x \in S\}$.
end
Notre simplicial Branch and Bound pour la résolution du problème (3.1) avec tolérance $\epsilon \geq 0$ peut être décrit comme suit :

Algorithme SBB(f, P, ϵ)

début
calculer
$\lambda := \max\{e^T x / x \in P\}$; $\Delta^1 := conv(\{0, \lambda e_1, ..., \lambda e_n\})$; $L := \{1\}; k := 1; z^0 := +\infty$;
Tantque $L \neq \emptyset$ faire

début
sélectionner un indice $i \in L$
$L := L/\{i\}; \Delta := \Delta_i$ et $v_1, ..., v_{n+1}$ les sommets de Δ;
$\mu = \sum_{i=1}^{n+1} v_i/(n+1)$;
$d := \nabla f(\mu)$;
résoudre le problème $\min\{f(x) - d^T x / x \in \{v_1, ..., v_{n+1}\}\}$ et soit α^k la valeur optimale et v^k la solution optimale.
résoudre le problème $\min\{d^T x + \alpha^k / x \in P\}$ et soit z^k la valeur optimale et x^k la solution optimale.
$\bar{x}^k$:=SEARCH (x^k); et $\bar{z}^k := f(\bar{x}^k)$;
Si $\bar{z}^k < z^0$ alors $z^0 := \bar{z}^k$ et $x^0 := \bar{x}^k$;
Si $\bar{z}^k < z^0 - \epsilon$ alors
début
$(\Delta^{2k}, \Delta^{2k+1}) := SPLIT(\Delta)$;
$L := L \cup \{2k, 2k+1\}$;
fin
k :=k+1 ;
fin.
fin,
La procédure SPLIT (Δ) est définie comme suit :
début
Sélectionner la plus longue arête (v_p, v_q) de (Δ);
$v := (1-\alpha)v_p + \alpha v_q$; pour α fixe dans $[0, \frac{1}{2}]$;
$\Delta^{'} = conv(\{v_j / j \neq p\} \cup \{v\})$;
$\Delta^{"} = conv(\{v_j / j \neq q\} \cup \{v\})$;
return $(\Delta^{'}, \Delta^{"})$;
fin

5.2 Propriété de convergence :

Quand l'algorithme SBB termine, il génère une solution optimale ou $\epsilon - optimale$.

Théorème 5.2.1.

Quand l'algorithme SBB termine aprés un nombre fini d'itérations, il génère une solution réalisable x^0 tel que :

$$\forall x \in P,\ f(x^0) - \epsilon \leq f(x) \tag{5.16}$$

Preuve :

Supposons que l'algorithme termine et soit S l'ensemble des indices du simplexe abondonné. Nous avons :

$$\forall i \in S,\ z^0 - \epsilon \leq w^{'}(\Delta_i) \leq \min\{f(x),\ x \in P \cap (\Delta_i)\} \tag{5.17}$$

$\cup_{i \in S} \Delta_i = \Delta^1 \supset P$ nous avons (5.16)
Si $\epsilon = 0$, alors x^0 est un minimum global.
Si $\epsilon > 0$, alors x^0 est $\epsilon - optimal$ global.
$z^k \leq z^0 - \epsilon|z^0|$ dans l'algorithme, alors (5.16) sera remplacé par :

$$\forall x \in P, f(x^0) - \epsilon|f(x^0)| \leq f(x) \tag{5.18}$$

En suite, nous verons comment que l'algorithme SBB se comporte s'il ne termine pas. Dans ce cas une suite infinie $\{\bar{x}^k / k = 1, 2, ...\}$ est génèrée.

Lemme 5.2.1.

Supposons que l'algorithme SBB génère une suite infinie $\{\bar{x}^k / k = 1, 2, , ...\}$, *il existe une sous-suite* $\{k_1, k_2, ...\}$ *de* $\{1, 2, ...\}$ *tel que*

$$\Delta^{k_1} \supset \Delta^{k_2} \supset ...\ et\ \bar{x}^{k_q} \longrightarrow \bar{x}^{'}, x^{k_q} \longrightarrow x^{'}\ quand\, q \longrightarrow +\infty \tag{5.19}$$

Preuve :

Pour tout $k \neq 1$, il existe $k^{'} < k$ tel que

$$\Delta^{k'} \supset \Delta^{k} \tag{5.20}$$

donc il existe une sous-suite $\{k_1^{'}, k_2^{'}, ...\}$ de $\{1, 2, ...\}$ tel que

$$\Delta^{k_1^{'}} \supset \Delta^{k_2^{'}} \supset ... \tag{5.21}$$

$\bar{x}^{k_1^{'}}, \bar{x}^{k_2^{'}}, ...$ sont génèrées dans $\{k_1^{"}, k_2^{"}\}$ de $\{k_1^{'}, k_2^{'}\}$ tel que

$$\Delta^{k_1^{"}} \supset \Delta^{k_2^{"}} \supset ...et\, \bar{x}^{k^{"}\, q} \rightarrow \bar{x}^{'}\ quand\, q \rightarrow +\infty \tag{5.22}$$

Aussi $\bar{x}^{k_1^{"}}, \bar{x}^{k_2^{"}}, ...$ sont génèrées.
Il existe une sous-suite $\{k_1, k_2, ...\}$ de $\{k_1^{"}, k_2^{"}\}$ tel que (5.19)

Lemme 5.2.2.

Supposons que l'algorithme SBB génère une suite infinie $\{\bar{x}^k / k = 1, 2, ...\}$.
Si $\{k_1, k_2, ...\}$ *est une sous-suite de* $\{1, 2, ...\}$ *tel que* $\Delta^{k_1} \supset \Delta^{k_2} \supset ...$ *et*
$\bar{x}^{k_q} \rightarrow \bar{x}^{'}$ *et* $x^{k_q} \rightarrow x^{'}$ *quand* $q \rightarrow +\infty$ *nous avons*

$$\lim_{q \rightarrow +\infty} (f(\bar{x}^{k_q}) - z^{k_q}) = 0 \tag{5.23}$$

Preuve :

Par la description de l'algorithme, nous avons :

$$z^{k_q} = (d^{k_q})^T x^{k_q} + \alpha^{k_q} = \nabla f(\mu^{k_q})^T x^{k_q} + f(v^{k_q}) - \nabla f(\mu^{kq})^T v^{k_q} \tag{5.24}$$

Depuit la règle de la bisection est adoptée, il existe un $v^{'}$ tel que

$$\cap_{q=1}^{\infty} \Delta^{k_q} = \{v^{'}\} \tag{5.25}$$

Puisque μ^{k_q} et v^{k_q} sont des points de Δ^{k_q}, nous avons

$$\mu^{k_q} \to v^{'} \; et \; v^{k_q} \to v^{'} \; quand \; q \to +\infty \tag{5.26}$$

par la continuité de f et ∇f, nous avons

$$\lim_{q \to +\infty} z^{k_q} = \nabla f(v')^T x^{'} + f(v^{'}) - \nabla f(v')^T v^{'} = \nabla f(v')^T (x^{'} - v^{'}) + f(v^{'}) \tag{5.27}$$

Donc par la concavité de f, nous avons :

$$\lim_{q \to +\infty} z^{k_q} \geq f(x^{'}) \tag{5.28}$$

Puisque f est continue et $f(x^{k_q}) \geq f(\bar{x}^{k_q})$
pour tout q nous avons :

$$\lim_{q \to +\infty} z^{k_q} \geq f(\bar{x}^{'}) \tag{5.29}$$

donc nous avons :

$$\lim_{q \to +\infty} (f(\bar{x}^{k_q}) - z^{k_q}) \leq 0 \tag{5.30}$$

aussi nous avons :

$$\forall q, z^{k_q} < f(\bar{x}^{k_q}) \tag{5.31}$$

Par la continuité de f et de ∇f, nous avons :

$$\lim_{q \to +\infty} (f(\bar{x}^{k_q}) - z^{k_q}) \geq 0 \tag{5.32}$$

de (5.30) et (5.32) nous avons (5.23).
Maintenent, nous montrons que l'algorithme termine aprés un nombre fini d'itérations quand $\epsilon > 0$.

Lemme 5.2.3.

Quand $\epsilon > 0$, l'algorithme SBB termine aprés un nombre fini d'itérations.

Preuve :

Supposons que l'algorithme n'est pas fini et qu'il génère une suite infnie $\{\bar{x}^{k_q}/k = 1, 2, ...\}$.
dans ce cas, il existe une sous suite $\{k_1, k_2, ...\}$ de $\{1, 2, ...\}$ tel que $\Delta^{k_1} \supset \Delta^{k_2} \supset ...$ et $\bar{x}^{kq} \to \bar{x}^{'}$ et $x^{k_q} \to x^{'}$ quand $q \to +\infty$
Nous voyons de lemme (5.2.2) que :
$\lim_{q\to+\infty}(f(\bar{x}^{'}) - z^{k_q}) = 0$
Maintenent, nous supposons que $\epsilon > 0$, alors :
$\forall q, z^{k_q} < f(\bar{x}^{k_q}) - \epsilon$ c'est à dire :

$$f(\bar{x}^{k_q}) - z^{k_q} > \epsilon > 0 \tag{5.33}$$

Par la continuité de f et ∇f, nous avons :

$$\lim_{q\to+\infty}(f(\bar{x}^{k_q}) - z^{k_q}) \geq \epsilon \tag{5.34}$$

contradiction, donc $\epsilon > 0$, l'algorithme termine aprés un nombre fini d'itérations.

Lemme 5.2.4.
supposons que l'algorithme SBB génère une suite infinie $\{\bar{x}^k/k = 1, 2, ...\}$ et z^* est la valeur optimale globale.
Pour chaque itération k, il existe un indice $i \in L^k$ tel que : $w^{'}(\Delta_i) \leq z^*$.

Preuve :

Puisque l'algorithme ne termine pas, nous voyons de lemme (5.2.3) que $\epsilon = 0$.
Supposons qu'il existe un $k^{'}$ tel que :

$$\forall i \in L^{k^{'}}, w^{'}(\Delta_i) > z^* \tag{5.35}$$

. Si la valeur z^0 à l'itération $k^{'}$ est la valeur optimale globale, nous avons :

$$\forall i \in L^{k^{'}}, w^{'}(\Delta_i) > z^* = z^0 \tag{5.36}$$

Par le critère de la recherche inverse, l'algorithme termine aprés un nombre fini d'itérations, c'est une contradiction.

. Si la valeur z^0 à l'itération $k^{'}$ n'est pas la valeur optimale globale, nous avons :

$$\forall x \in P, f(x) > z^* \tag{5.37}$$

c'est une contradiction.

Donc , il existe un indice $i \in L^k$ tel que : $w^{'}(\Delta_i) \leq z^*$ pour chaque itération.
De lemme (5.2.2) et (5.2.4), nous obtenons le théorème suivant :

Théorème 5.2.2.

Supposons que l'algorithme SBB génère une suite infinie $\{\bar{x}^k/k = 1, 2, ...\}$ *et les conditions sur* P *et* f *sont vérifie :*
Si $\{k_1, k_2, ...\}$ *est une suite de* $\{1, 2, ...\}$ *tel que :*
$\Delta^{k_1} \supset \Delta^{k_2} \supset ...$ *et* $\bar{x}^{kq} \to \bar{x}^{'}$ *et* $x^{k_q} \to x^{'}$ *quand* $q \to +\infty$*, alors* $\bar{x}^{'}$ *est une solution optimale globale.*

Preuve :

Supposons que z^* est une solution optimale globale, nous voyons de lemme (5.2.4) que :

$$\forall q, z^{k_q} = \min\{w^{'}(\Delta_i)/i \in L^{k_q}\} \leq z^* \tag{5.38}$$

Depuis $\bar{x}^{k_q}$ est un point de P, nous avons :

$$\forall q, z^* \leq f(\bar{x}^{k_q}) \tag{5.39}$$

Donc nous avons : $\forall q, z^{k_q} \leq z^* \leq f(\bar{x}^{k_q})$. C'est à dire : $0 \leq f(\bar{x}^{k_q}) - z^* \leq f(\bar{x}^{k_q}) - z^{k_q}$.
De la continuité de f et ∇f et de lemme (5.2.2), nous avons :

$$\lim_{q \to +\infty} (f(\bar{x}^{k_q}) - z^*) = 0 \tag{5.40}$$

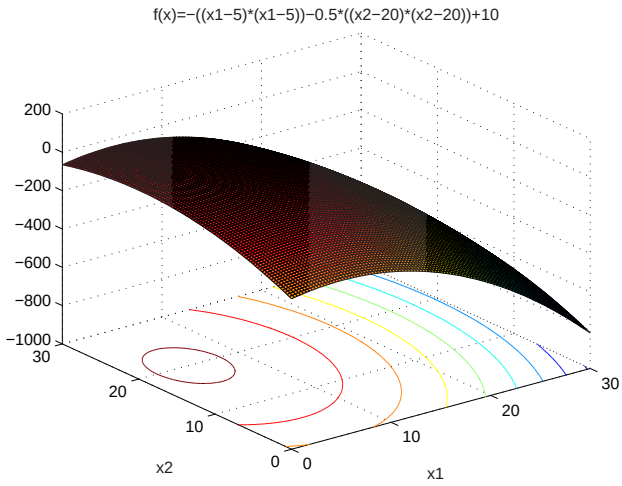
Cela implique que

$$f(\bar{x}^{'}) = z^* \tag{5.41}$$

5.3 Exemple numérique :

Pour bien comprendre l'algorithme SBB, nous proposons l'exemple suivant :

$$\min -(x_1 - 5)^2 - 0.5(x_2 - 20)^2 + 10$$
$$\begin{cases} 0.5x_1 + x_2 \leq 20 \\ x_1 \leq 20 \\ x_1 \geq 0 \\ x_2 \geq 0 \end{cases}$$



Premièrement, nous calculons :
$\lambda = \max\{e^T x / x \in P\}$ *avec* e : *vecteur unitaire*
$= \max\{x_1 + x_2 / 0.5x_1 + x_2 \leq 20, x_1 \leq 20, x_1 \geq 0, x_2 \geq 0\}$

Aprés la résolution de ce problème avec Matlab à l'aide de la commande linprog ou avec la méthode du simplexe, on trouve $\lambda = 30$.
D'où le premier simplexe qui contient l'ensemble des solutions faisables est donné par
$\Delta^1 = conv(0, \lambda e_1, ..., \lambda e_n) = conv([0,0],[30,0],[0,30])$. avec
e_i :vecteur de la base canonique,$i = 1..n$
Aprés, nous calculons :

. $\mu^1 = \sum_{i=1}^{n+1} v_i/(n+1)$
$= ([0,0] + [30,0] + [0,30])/3$
$= [30,30]/3$
$= [10,10]$

. $\nabla f(x) = (-2x_1 + 10, -x_2 + 20)$

. $\nabla f(\mu^1) = (-20 + 10, -10 + 20) = [-10, 10]$

Par conséquent :

. $g_1(x) = \nabla f(\mu^1)x$
$= -10x_1 + 10x_2$

. $\alpha^1 = \min\{f(x) - g_1(x) / x \in sommets\, de\, \Delta^1\}$
$= \min\{-(x_1 - 5)^2 - 0.5(x_2 - 20)^2 + 10 + 10x_1 - 10x_2, x \in \{[0,0],[30,0][0,30]\}$
$= -515$

. $h_1(x) = g_1(x) + \alpha^1 = -10x_1 + 10x_2 - 515$

Pour obtenir la borne inf de f sur $P \cap \Delta^1$, on résoud $\min\{h_1(x)/x \in P\}$ soit par la méthode du simplexe soit avec Matlab à l'aide de la commande linprog.
$\min -10x_1 + 10x_2 - 515$
$$\begin{cases} 0.5x_1 + x_2 \leq 20 \\ x_1 \leq 20 \\ x_1 \geq 0 \\ x_2 \geq 0 \end{cases} .$$

la solution optimale est $x^0 = [20, 0]$ et la valeur optimale est $z(\Delta^1) = -200 - 515 = -715$
$f([20, 0]) = -(20 - 5)^2 - 0.5(-20)^2 + 10 = -415$
Mais la borne inf est $-715 < -415$ donc on subdivise Δ^1 par la règle de la bissection et on continue.
$\Delta^1 = conv(\{[0, 0], [30, 0], [0, 30]\})$
La plus longue arête est $(v_p, v_q) = ([30, 0], [0, 30])$
$v = (1 - \alpha)v_p + \alpha v_q)$, $avec\, \alpha \in [0, 1/2]$
$v = 0.5[30, 0] + 0.5[0, 30] = [15, 15]$
$\Delta^2 = conv(\{v_j/j \neq P\} \cup \{v\}) = conv(\{[0, 0], [15, 15], [0, 30]\})$;
$\Delta^3 = conv(\{v_j/j \neq q\} \cup \{v\}) = conv(\{[0, 0], [30, 0], [15, 15]\})$;
Considérons $\Delta^2 = conv(\{[0, 0], [15, 15], [0, 30]\})$;

. $\mu^2 = (\{[0, 0], [15, 15], [0, 30]\})/3 = [5, 15]$;

. $\nabla f(\mu^2) = [-10 + 10; -15 + 20] = [0, 5]$;

. $g_2(x) = 5x_2$;

. $\alpha^2 = \min\{f(x) - g_2(x)/[x_1, x_2] \in sommets\, de\, \Delta^2\}$
$= \min\{-(x_1 - 5)^2 - 0.5(x_2 - 20)^2 + 10 - 5x_2/[x_1, x_2] \in \{[0, 0], [15, 15], [0, 30]\}\}$
$= -215$

. $h_2(x) = g_2(x) + \alpha^2 = 5x_2 - 215$;

Pour obtenir la borne inf de f sur $P \cap \Delta^2$, on résoud le problème :
$\min\{h_2(x)/x \in P\}$
$\min 5x_2 - 215$
$$\begin{cases} 0.5x_1 + x_2 \leq 20 \\ x_1 \leq 20 \\ x_1 \geq 0 \\ x_2 \geq 0 \end{cases} .$$
La solution optimale est $[0, 0]$, la valeur optimale est $z(\Delta^2) = -215$.
$f([0, 0]) = -215 > f([20, 0])\, donc\, x^0 := [20, 0]$.
$z(\Delta^2) = -215 > f(x^0) = -415$ donc on peut pas subdiviser Δ^2
Considérons le simplexe $\Delta^3 = conv(\{[0, 0], [30, 0], [15, 15]\})$

. $\mu_3 = ([0, 0], [30, 0], [15, 15])/3 = [15, 5]$;

. $\nabla f(\mu_3) = [-30 + 10, -5 + 20] = [-20, 15]$;

. $g_3(x) = -20x_1 + 15x_2$;

. $\alpha^3 = \min\{f(x) - g_3(x)/x \in \textit{sommets de } \Delta^3\}$
$= \min(-(x_1-5)^2 - 0.5(x_2-20)^2 + 10 + 20x_1 - 15x_2/[x_1, x_2] \in \{[0,0],[30,0],[15,15]\})$
$= -215$;

. $h_3(x) = g_3(x) + \alpha^3 = -20x_1 + 15x_2 - 215$;

Pour obtenir la borne inf de f sur $P \cap \Delta^3$, on résoud le problème :
$\min\{h_3(x)/x \in P\}$
$\min -20x_1 + 15x_2 - 215$
$\begin{cases} 0.5x_1 + x_2 \leq 20 \\ x_1 \leq 20 \\ x_1 \geq 0 \\ x_2 \geq 0 \end{cases}$.
On obtient $[20, 0]$ solution optimale et la valeur optimale est $z(\Delta^3) = -615$.
$z(\Delta^3) = -615 < f([20, 0]) = -415$, on continu la subdivision de Δ^3 pourtant on a déja une solution optimale $[20, 0]$.

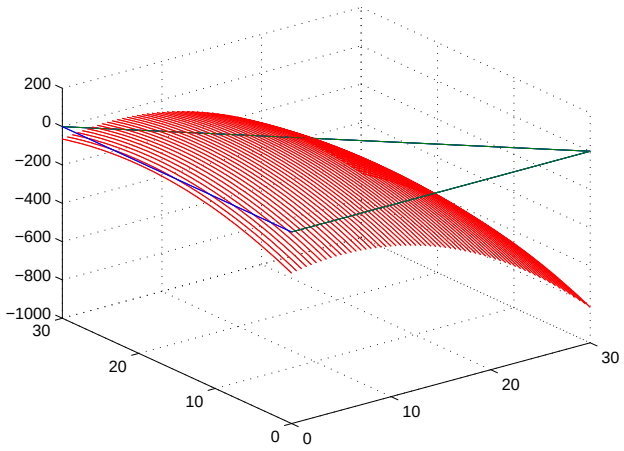


FIG. 5.1 – Le simplexe Δ^1

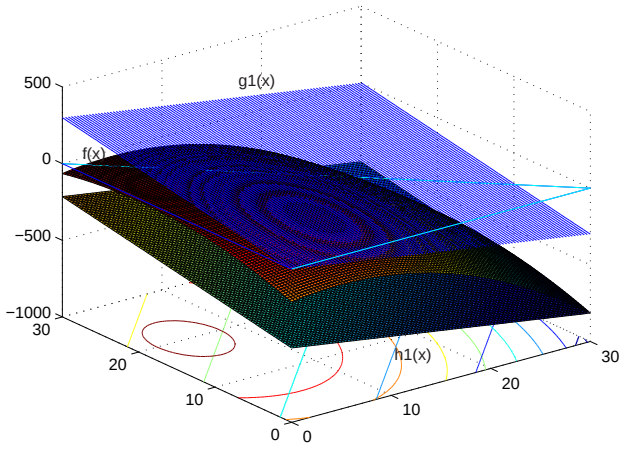


FIG. 5.2 – Les estimateurs de la fonction f dans Δ^1

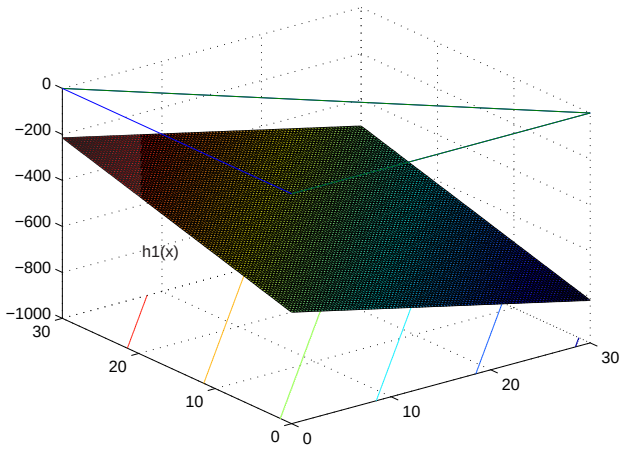


FIG. 5.3 – Le programme linéaire relaxé dans Δ^1

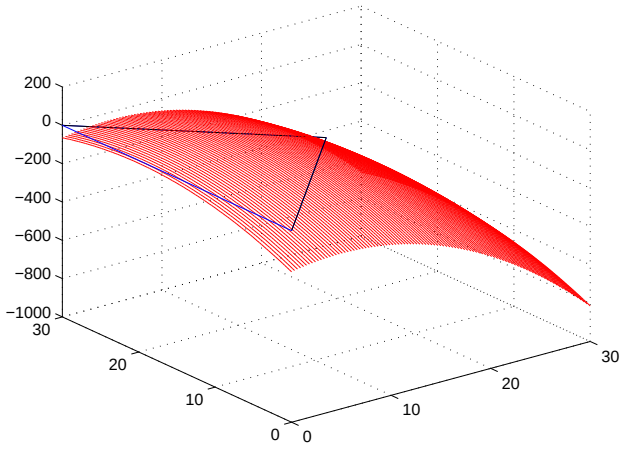


FIG. 5.4 – Le simplexe Δ^2

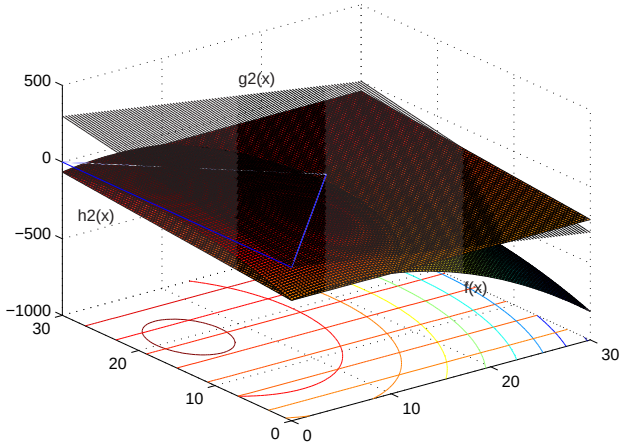


FIG. 5.5 – Les estimateurs de la fonction f dans Δ^2

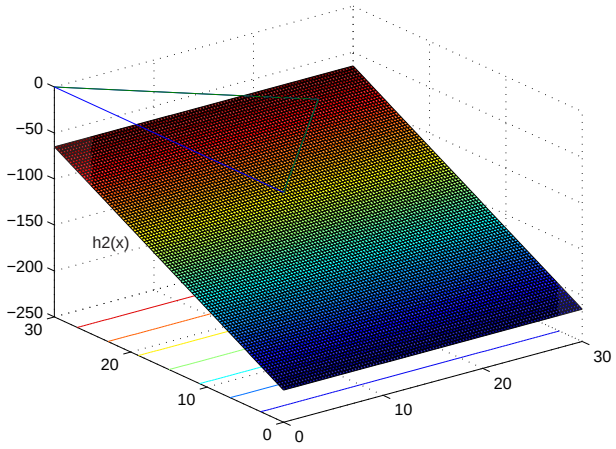


FIG. 5.6 – Le programme linéaire relaxé dans Δ^2

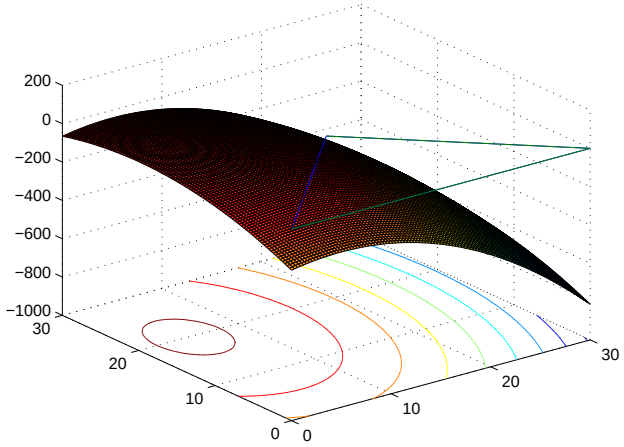


FIG. 5.7 – Le simplexe Δ^3

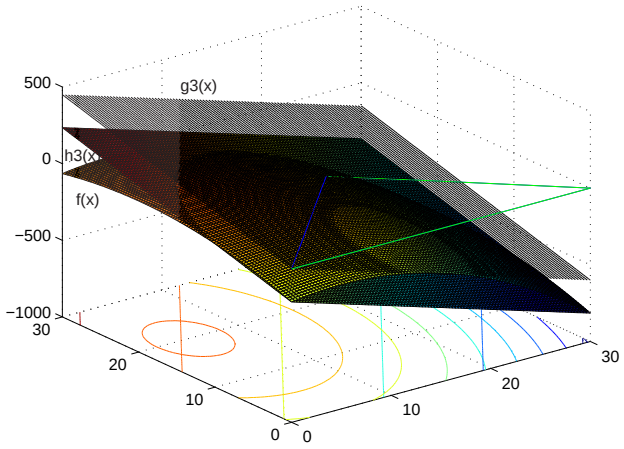


FIG. 5.8 – Les estimateurs de la fonction f dans Δ^3

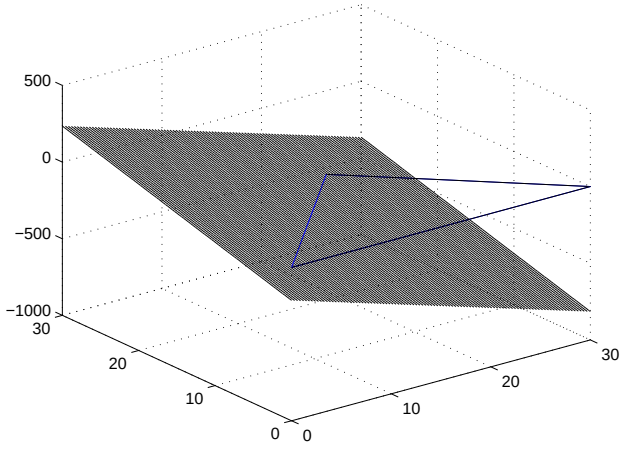


FIG. 5.9 – Le programme linéaire relaxé dans Δ^3

Remarque 5.3.1. Si on fixe pas ϵ, l'algorithme ne termine pas.
A l'itération 3, le sous-estimateur h_3 coincide avec la fonction f dans Δ^3. donc la solution optimale est $[20, 0]$.

Chapitre 6

Implimentation

Algorithme SBB appliqué sur l'exemple numérique

```
sprintf('les paramètres de problème: \n')
type='max'
A=input('A=');
b=input('b=');
[m,n]=size(A);
sprintf('c=')
c=ones(1,n)
varargin='o';
n1=m+1;
sprintf('Calcul de la valeur de lamda: \n')
[x,lamda]=simplex2(type,A,b,c,varargin)
sprintf('Le premier simplexe: \n')
v0=zeros(1,n);
v1=[lamda zeros(1,n-1)];
v2=[0 lamda zeros(1,n-2)];
v=[v0;v1;v2]
disp(sprintf('"Entrer" pour continuer \n'))
pause
sprintf('La valeur de mu1 \n')
j=1;
while j<=n
 for i=1:n
  for k=1:n
   mu1(k)=v(i,j)+v(i+1,j)/n1;
  end
 end
 j=j+1;
end;
```

```
mu1
f=@(x)(-((x(1)-5).^2)-0.5*((x(2)-20).^2)+10);
df=@(x)[-2*x(1)+10,-x(2)+20];
df(mu1);
z1=@(x)((-((x(1)-5).^2)-0.5*((x(2)-20).^2)+10)+10*x(1)-10*x(2));
sprintf('La valeur de alpha1 \n')
alpha1=minimum(m,n,v,z1)
c1=df(mu1);
type='min'
disp(sprintf('"Entrer" pour continuer \n'))
pause
sprintf('La valeur de h1 et la solution optimale\n')
[x1,h]=simplex2(type,A,b,c1,varargin)
h1=h+alpha1
x0=x1
if h1<f(x0)
 a1=1/n;
 i=2;
 while i<=n1
  for j=1:n-1
   vv(i-1)=a1*(v(i,j)+v(i,j+1));
  end
  i=i+1;
 end
 vv;
 v4=vv;
 sprintf('Les deux nouveaux simplexes \n')
 delta2=[v0;v4;v2]
 delta3=[v0;v1;v4]
else
 sprintf('on ne peut pas subdiviser le simplexe et la solution optimale est \n')
 x0
end disp(sprintf('"Entrer" pour continuer \n'))
pause fprintf('le deuxieme simplexe \n')
v=delta2;
sprintf('La valeur de mu2 \n')
mu2=(v0+v4+v2)/3
df(mu2)
c2=df(mu2)
z2=@(x)((-((x(1)-5).^2)-0.5*((x(2)-20).^2)+10)-5*x(2))
delta2=[v0;v4;v2]
sprintf('La valeur de alpha2 \n')
alpha2=minimum(m,n,v,z2)
```

```
sprintf('la valeur de h2 et la solution optimale \n')
[x2,h]=simplex2(type,A,b,c2,varargin)
h2=h+alpha2 if
f(x2)<f(x0)
 x0=x2
end
if h2>f(x0)
 fprintf('on peut pas subdiviser delta2\n')
else
 vv=0.5*v1+0.5*v2
 sprintf('les deux nouveaux simplexes: \n')
 delta4=[v0;v4;v2]
 delta5=[v0;v1;v4]
end
disp(sprintf('"Entrer" pour continuer \n'))
pause
fprintf('le troisieme simplexe: \n')
v=delta3;
sprintf('Calcul de mu3')
mu3=(v0+v1+v4)/3
df(mu3)
z3=@(x)((-((x(1)-5).^2)-0.5*((x(2)-20).^2)+10)+20*x(1)-15*x(2))
sprintf('calcul de alpha3 \n')
alpha3=minimum(m,n,v,z3)
c3=df(mu3);
type='min'
sprintf('Calcul de h3 et la solution optimale \n')
[x3,h]=simplex2(type,A,b,c3,varargin)
h3=h+alpha3
if f(x3)<f(x0)
 x0=x3
end
if h3>=f(x0)
 fprintf('on peut pas subdiviser delta3 \n')
else
 fprintf('on continu la subdivision \n')
end
```

Les fonctions utilisées dans le programme principal

minimum de $f(x) - g_i(x)$

```
function [alpha]=minimum(m,n,v,z)
 alpha=z([v(1,1) v(1,2)]);
 i=1;
 while i<=m
  for j=1:n-1
   if z([v(i,j) v(i,j+1)])< alpha
    alpha=z([v(i,j) v(i,j+1)]);
   end
  end
  i=i+1;
 end
end
```

Algorithme de Simplexe

```
function [x, z] = simplex2(type,A,b,c) clc;
%
%              ************************************************************
%              **     L'algorithme de simplexe pour le problème PL       **
%              **                   max(min) Z = c*x                      **
%              **               Sc: Ax <= b                     **
%              **                            x >= 0                       **
%              ************************************************************
% b: vecteur non negative
% Pour un probleme de maximisation type = 'max', sinon type = 'min'.
% Parametre de sortie:
% A: dernier tableau de la methode de simplexe.
% x: la solution optimale
% Z: lanvaleur de la fonction objective à x.
 if any(b < 0)
  error('Le vecteur b doit etre non negative.')
 end
 if type == 'min'
  tp = -1;
```

```
else
 tp = 1;
 c = -c;
end
[m, n] =size(A);
A = [A eye(m)];
b = b(:);
c = c(:)';
A = [A b];
d = [c zeros(1,m+1)];
A = [A;d];
nargin: renvoie le nombre de variables d'entrée
if nargin == 5
 disp(sprintf('-------------------------------------------------------------'))
 disp(sprintf('             ALGORITHME DU SIMPLEXE                          '))
 disp(sprintf('-------------------------------------------------------------'))
 disp(sprintf('\n\n--->> Le tableau initial \n'))
 A
 disp(sprintf('"Entrer" pour continuer \n\n'))
 pause
end
[mi, col] = Br(A(m+1,1:m+n));
sb = n+1:m+n;

while ~isempty(mi) & mi < 0 & abs(mi) > eps
 t = A(1:m,col);
 if all(t <= 0)
  disp(sprintf('\n Le problème a une fonction objectif sans bornes'));
  x = zeros(n,1);
  if tp == -1
   z =-inf;
  else
   z = inf;
  end
  return;
 end
 row = MRT(A(1:m,m+n+1),A(1:m,col));
 row: l'indice de min(b/min_delta);(on calcul min_teta)
 if ~isempty(row)
  A(row,:)= A(row,:)/A(row,col);
  sb(row) = col;
  for i = 1:m+1
   if i ~= row
```

```
     A(i,:)= A(i,:)-A(i,col)*A(row,:);
    end
   end
  end
  [mi, col] = Br(A(m+1,1:m+n));
 end
 z = tp*A(m+1,m+n+1);
 x =zeros(1,m+n);
 x(sb) = A(1:m,m+n+1);
 x = x(1:n)';
 if nargin == 5
  disp(sprintf('\n--->> Le tableau final '))
  A
  disp('la solution optimale est :\n')
  x
  disp('la valeur optimale est:')
  h=z
  disp(sprintf('"Entrer" pour continuer \n'))
  pause
  clc;
 end
end
```

La fonction MRT

```
function [row, mi] = MRT(a, b)
 % row: l'indice de la ligne pivot
 % mi: la valeur d'un plus petit ratio
 m = length(a);
 c = 1:m;
 a = a(:);
 b = b(:);
 l = c(b > 0);
 [mi,row] = min(a(l)./b(l));
 row = l(row);
end
```

La fonction Br

```
function [m, j] = Br(d)
 % m: premier nombre négative dans le vecteur d.
```

```
 % j: indice de l'entrée m.
 ind = find(d < 0);
 if ~isempty(ind)
  j = ind(1);
  m = d(j);
  for i=2:length(d)
   if m > d(i)
    m = d(i);
    j = i;
   end
  end

 else
  m = [];
  j = [];
 end
end
```

L'exécution

```
ans =

les paramètres de problème:



type =

max

A=[0.5 1;1 0]
b=[20 20]
--------------------------------------------------------------------------------
                                          ALGORITHME DU SIMPLEXE
--------------------------------------------------------------------------------


--->> Le tableau initial
```

```
A =

    0.5000    1.0000    1.0000         0   20.0000
    1.0000         0         0    1.0000   20.0000
   -1.0000   -1.0000         0         0         0

"Entrer" pour continuer



--->> Le tableau final

A =

         0    1.0000    1.0000   -0.5000   10.0000
    1.0000         0         0    1.0000   20.0000
         0         0    1.0000    0.5000   30.0000

la solution optimale est :

x =

    20
    10

la valeur optimale est:

h =

    30

"Entrer" pour continuer

x =

    20
    10


h =

    30
```

```
ans =

Le premier simplexe:


v =

     0     0
    30     0
     0    30

"Entrer" pour continuer



ans =

La valeur de mu1


mu1 =

    10    10


ans =

La valeur de alpha1


alpha1 =

  -515


type =
```

```
min

"Entrer" pour continuer
-------------------------------------------------------------------------------
                                          ALGORITHME DU SIMPLEXE
-------------------------------------------------------------------------------


--->> Le tableau initial


A =

    0.5000    1.0000    1.0000         0   20.0000
    1.0000         0         0    1.0000   20.0000
  -10.0000   10.0000         0         0         0

"Entrer" pour continuer

--->> Le tableau final

A =

         0    1.0000    1.0000   -0.5000   10.0000
    1.0000         0         0    1.0000   20.0000
         0   10.0000         0   10.0000  200.0000

la solution optimale est :

x =

    20
     0

la valeur optimale est:

h =

  -200

"Entrer" pour continuer h1 =

  -715
```

```
x0 =

    20
     0


ans =

Les deux nouveaux simplexes



delta2 =

     0     0
    15    15
     0    30


delta3 =

     0     0
    30     0
    15    15

"Entrer" pour continuer
-------------------------------------------------------------------------------
                                          ALGORITHME DU SIMPLEXE
-------------------------------------------------------------------------------


--->> Le tableau initial


A =

    0.5000    1.0000    1.0000         0   20.0000
    1.0000         0         0    1.0000   20.0000
         0    5.0000         0         0         0

"Entrer" pour continuer
```

```
--->> Le tableau final

A =

    0.5000    1.0000    1.0000         0   20.0000
    1.0000         0         0    1.0000   20.0000
         0    5.0000         0         0         0

la solution optimale est :

x =

     0
     0

la valeur optimale est:

h =

     0

"Entrer" pour continuer

x2 =

     0
     0


h =

     0


h2 =

  -215

on peut pas subdiviser delta2 "Entrer" pour continuer
--------------------------------------------------------------------------------
                                          ALGORITHME DU SIMPLEXE
```

```
---------------------------------------------------------------------------------

--->> Le tableau initial


A =

    0.5000    1.0000    1.0000         0   20.0000
    1.0000         0         0    1.0000   20.0000
  -20.0000   15.0000         0         0         0

"Entrer" pour continuer


--->> Le tableau final

A =

         0    1.0000    1.0000   -0.5000   10.0000
    1.0000         0         0    1.0000   20.0000
         0   15.0000         0   20.0000  400.0000

la solution optimale est :

x =

    20
     0

la valeur optimale est:

h =

  -400

"Entrer" pour continuer


x3 =

    20
     0
```

```
h =

  -400


h3 =

  -615

on continu la subdivision
```

Conclusion

Ce travail a été consacré à l'étude des problèmes de minimisation concave sur un polyèdre et à leur résolution par la mèthode de Branch and Bound où nous avons développé l'algorithme Simplicial Branch and Bound pour génèrer une solution optimale globale.Il est basé sur deux étapes trés importantes :

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

Notre but était de trouver une solution pour les problèmes multi-extrêmaux .C'est à dire, d'eviter les minimums locaux et de trouver une solution optimale globale. Notre simplicial Branch and Bound est un des algorithmes qui résoud ce problème.
Quelques modifications de l'algorithme de base ont été présentées, la première consiste à remplacer la fonction objectif par son enveloppe convexe et nous avons présenté l'algorithme SSBB pour sa résolution et la deuxième par son plus simple sous-estimateur. Notre résultat principal est la présentation de l'algorithme simplicial Branch and Bound.
L'inconvénient de cet algorithme est que malgré que nous trouvons une solution optimale on arrète pas la subdivision donc on est amené à choisir un seuil de précision pour atteindre la solution et c'est le cas de notre étude numérique.

Bibliographie

[1] Christophe Meyer, *A Simplicial finite simplicial covering algorithm for concave minimisation over a polytope*, January(2011)

[2] Mémoire de Licence ,Mlle Hassina Zina, Hameg Sofia *Résolution d'equation non linaires par l'optimisation*,(2010), Tizi-Ouzou.

[3] Mémoire de magister, Mm. Goumeziane Lynda, *Contribution à l'étude des algorithmes de Branch and Bound pour l'optimisation globale non convexe*, (2008), Setif

[4] www.mathworks.com.

[5] Pierre Hansen, Brigitte Jaumard, Christophe Meyer, Hoang Tuy*Best Simplicial and Double-Simplicial Bound for Concave Minimization* www.Simplicial Branch and Bound of concave minimisation,(1996)

[6] Reiner Horst. Panos M.Parbalos and Nguyen V.Thoai, *Introduction to global optimisation*.

[7] Thèse de Doctorat, Christophe Meyer, *ALGORITHMES CONIQUES POUR LA MINIMISATION QUASICONCAVE*, (Août 1996).

[8] Thèse de doctorat Takahito Kuno, *A deterministic Algorithm for Concave Minimization and Its Performance as a Heuristic Tool*,University of Tsukuba, March(2011)

Annexe

Figure de l'exemple d'une fonction avec deux minimums locaux :(sur Matlab)

$[x, y] = meshgrid(0 : .2 : 2, 0 : .2 : 2);$
$z = x^4 + y^4 - (x - y)^2\,;$
$surf(x, y, z)\,;$

La valeur de λ :

$f = [-1; -1];$
$A = [0.5, 1; 1, 0];$
$b = [20; 20];$
$lb = zeros(2, 1);$
$[x, fval, exitflag, output, lambda] = linprog(f, A, b, [], [], lb)$

La valeur de $h_1(x)$:

$f = [-10, 10];$
$A = [0.5, 1; 1, 0];$
$lb = zeros(2, 1);$
$[x, fval, exitflag, output, lambda] = linprog(f, A, b, [], [], lb)$
la valeur de $h_1 = fval - 515.$

Les figures de l'exemple numérique :

1. La fonction objectif
   $[X, Y] = meshgrid(0 : .30 : 30, 0 : .30 : 30);$
   $f = -((X - 5).^2) - 0.5. * ((Y - 20).^2) + 10;$
   $surfc(X, Y, f)$
   $xlabel('x1')$
   $ylabel('x2')$

$title('f(x) = -((x1-5)*(x1-5)) - 0.5*((x2-20)*(x2-20)) + 10')$

2. Le simplexe Δ^1
   $[X, Y] = meshgrid(0:.30:30, 0:.30:30);$
   $f = -((X-5).^2) - 0.5.*((Y-20).^2) + 10;$
   $A = [0\,30\,0];$
   $B = [0\,0\,30];$
   $k = convhull(A, B);$
   grid on
   $plot3(X, Y, f, 'r')$
   hold on
   $plot(A(k), B(k), A, B)$

3. Les estimateurs de la fonction f dans Δ^1
   $[X, Y] = meshgrid(0:.30:30, 0:.30:30);$
   $f = -((X-5).^2) - 0.5.*((Y-20).^2) + 10;$
   $g1 = -10.*X + 10.*Y;$
   $h1 = -10.*X + 10.*Y - 515;$
   $A = [0\,30\,0];$
   $B = [0\,0\,30];$
   $k = convhull(A, B);$
   $surfc(X, Y, f)$
   hold on
   $surfc(X, Y, h1)$
   hold on
   $plot3(X, Y, g1, 'b')$
   hold on
   $plot(A(k), B(k), A, B, 'c')$
   $text(2, 30, 30, 'f(x)')$
   $text(-5, -20, -10, 'h1(x)')$
   $text(10, 20, 600, 'g1(x)')$

4. Programme linèaire relaxé dans Δ^1
   $[X, Y] = meshgrid(0:.30:30, 0:.30:30);$
   $h1 = -10.*X + 10.*Y - 515;$
   $A = [0\,30\,0];$
   $B = [0\,0\,30];$
   $k = convhull(A, B);$
   $surfc(X, Y, h1)$
   hold on
   $plot(A(k), B(k), A, B)$
   $text(10, 30, -800, 'h1(x)')$

5. Le simplexe Δ^2
   $[X, Y] = meshgrid(0 : .30 : 30, 0 : .30 : 30);$
   $f = -((X - 5).^2) - 0.5. * ((Y - 20).^2) + 10;$
   $A = [0\,15\,0];$
   $B = [0\,15\,30];$
   $k = convhull(A, B);$
   grid on
   $plot3(X, Y, f,' r')$
   hold on
   $plot(A(k), B(k), A, B,' k')$

6. Les estimateurs de la fonction f dans Δ^2
   $[X, Y] = meshgrid(0 : .30 : 30, 0 : .30 : 30);$
   $f = -((X - 5).^2) - 0.5. * ((Y - 20).^2) + 10;$
   $g2 = -10. * X + 10. * Y;$
   $h2 = 5. * Y - 215;$
   $A = [0\,15\,0];$
   $B = [0\,15\,30];$
   $k = convhull(A, B);$
   $surfc(X, Y, f)$
   hold on
   $surfc(X, Y, h2)$
   hold on
   $plot3(X, Y, g2,' k')$
   hold on
   $plot(A(k), B(k), A, B,' g')$
   $text(25, 10, -800,' f(x)')$
   $text(0, 28, -300,' h2(x)')$
   $text(10, 20, 600,' g2(x)')$

7. Programme linéaire relaxé dans Δ^2
   $[X, Y] = meshgrid(0 : .30 : 30, 0 : .30 : 30);$
   $h2 = 5. * Y - 215;$
   $A = [0\,15\,0];$
   $B = [0\,15\,30];$
   $k = convhull(A, B);$
   $surfc(X, Y, h2)$
   hold on
   $plot(A(k), B(k), A, B)$
   $text(0, 25, -150,' h2(x)')$
8. Le simplexe Δ^3

$[X, Y] = meshgrid(0 : .30 : 30, 0 : .30 : 30);$
$f = -((X - 5).^2) - 0.5. * ((Y - 20).^2) + 10;$
$A = [0\,30\,15];$
$B = [0\,0\,15];$
$k = convhull(A, B);$
$surfc(X, Y, f)$
hold on
$plot(A(k), B(k), A, B)$

9. Les estimateurs de la fonction f dans Δ^3
$[X, Y] = meshgrid(0 : .30 : 30, 0 : .30 : 30);$
$f = -((X - 5).^2) - 0.5. * ((Y - 20).^2) + 10;$
$h3 = -20. * X + 15. * Y - 215;$
$g3 = -20. * X + 15. * Y;$
$A = [0\,30\,15];$
$B = [0\,0\,15];$
$k = convhull(A, B);$
$surfc(X, Y, f)$
hold on
$surfc(X, Y, h3)$
hold on
$plot3(X, Y, g3,' k')$
hold on
$text(-4, 23, -2,' f(x)')$
$text(18, 38, 10,' g3(x)')$
$text(0.2, 30, 50,' h3(x)')$
hold on
$plot(A(k), B(k), A, B,' g')$

10. Programme linéaire relaxé dans Δ^3
$[X, Y] = meshgrid(0 : .30 : 30, 0 : .30 : 30);$
$h3 = -20. * X + 15. * Y - 215;$
$A = [0\,30\,15]$
$B = [0\,0\,15];$
$k = convhull(A, B);$
grid on $plot3(X, Y, h3,' k')$
hold on
$plot(A(k), B(k), A, B,' k')$
$text(5, 30, -500,' h3(x)')$