الجمهورية الجزائرية الديموقراطية الشعبية
RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
وزارة التعليم العالي والبحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
جامعة عمار ثليجي الأغواط
UNIVERSITÉ AMAR THELIDJI LAGHOUAT

كـلـيـــــةالـعـلـــــوم
FACULTÉ DES SCIENCES
قسم الرياضيات
DÉPARTEMENT DE MATHEMATIQUES

Thèse de Doctorat troisième cycle

Filière : Mathématiques
Spécialité : Analyse Mathématique
Option: Analyse Numérique et EDP

Présentée par : **Telli Mohamed**

Intitulée:

Localisation de l'extremum global des problèmes d'optimisation quadratique avec contraintes

Soutenue publiquement le : 11/04/2021, devant le jury composé de :

Nom et Prénom	***Grade***	***Etabl.***	***Qualité***
Allaoui Salah Eddine	*Professeur*	*UAT Laghouat*	*Président*
Bentobache Mohand	*Professeur*	*UAT Laghouat*	*Directeur de thèse*
Mokhrtari Abdelkader	*Professeur*	*UAT Laghouat*	*Co-directeur de thèse*
Aider Méziane	*Professeur*	*USTHB Alger*	*Examinateur*
Chiter Lakhdar	*Professeur*	*UFA Sétif1*	*Examinateur*
Souid Mohammed Said	*MCA*	*UIK Tiaret*	*Examinateur*
Abita Rahmoune	*MCA*	*UAT Laghouat*	*Examinateur*

Année Universitaire: 2020/2021

Remerciements

Je remercie DIEU, le tout puissant, pour m'avoir accordé santé, courage et patience afin d'accomplir ce travail.

Je tiens à remercier vivement Monsieur le Professeur Mohand Bentobache, mon directeur de thèse, pour ses précieux conseils ainsi que les encouragements qu'il m'a donnés durant la préparation de la thèse. C'est aussi grâce à lui que j'ai pu amélioré considérablement mes compétences en rédaction des textes mathématiques et celles en programmation informatique.

Je tiens ensuite à remercier tout spécialement Monsieur le Professeur Abdelkader Mokhtari, mon co-directeur de thèse, pour ses conseils et son attention constante, sa sympathie et les discussions très intéressantes qu'il a menées pour me suggérer les voies de la recherche.

J'adresse mes remerciements à Prof. Allaoui Salah Eddine, Professeur à l'université de Laghouat, pour l'honneur qu'il m'a fait en acceptant de présider le jury.

Je remercie vivement Prof. Aider Méziane, Professeur à l'USTHB, Prof. Chiter Lakhdar, Professeur à l'université Ferhat-Abbas, Sétif1, Dr. Souid Mohammed Said, MCA à l'université Ibn Khaldoun de Tiaret et Dr. Abita Rahmoune, MCA à l'université de Laghouat pour avoir accepté de juger mon travail.

Merci également à mes amis doctorants du Laboratoire LMPA de l'université de Laghouat pour la convivialité, l'encouragement et l'amitié.

Je remercie plus particulièrement Soumia Bouhadi mon épouse pour sa patience, sa compréhension, son amour et l'équilibre qu'elle a apporté dans ma vie en acceptant de faire de moi son époux. Ces éléments ont été des alliés de poids pendant ces années.

Merci beaucoup à mes frères et ma belle soeur pour leur patience et leur soutien moral.

Finalement, rien de tout cela n'aurait été possible sans mes parents, qui m'ont donné la possibilité de faire des études et qui m'ont toujours encouragé, et que je remercie chaleureusement.

Dédicaces

Je dédie ce modeste travail

♡ ***À mon très cher père***

Tu as toujours été pour moi un exemple du père respectueux, honnête, de la personne méticuleuse, je tiens à honorer l'homme que tu es.
Ma réussite est la tienne. Qu'Allah t'accorde une longue vie et une bonne santé.

♡ ***À ma très chère mère***

Par les inestimables sacrifices que tu as consentis pour moi, tu as tant souhaité que je parvienne à ce but.

♡ ***À ma très chère femme***

Pour la patience et le soutien dont elle a fait preuve pendant toute la durée de cette thèse.

♡ ***À mon très cher encadreur***

Pour l'intelligence sociale, la sensibilité, la rigueur, le sacrifice et l'intimité du travail.
Je te serai reconnaissant toute ma vie. Que Dieu aide vos enfants à réussir dans la vie.

Mohamed TELLI

Table des figures

Liste des tableaux

Table des matières

Introduction

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.

La programmation quadratique concave consiste à minimiser une fonction quadratique concave sur un certain domaine convexe. Il s'agit d'un problème NP-difficile [90], cependant il est connu que le minimum d'une fonction quadratique concave sur un ensemble convexe compact est atteint en un de ses points extrêmes [70], ce qui le rend plus facile à résoudre qu'un problème d'optimisation non-convexe général. La programmation quadratique concave est aujourd'hui une branche particulièrement active de la programmation mathématique, et il y a, à cela, de nombreuses raisons. La première est peut-être le nombre, la variété, et l'importance de ses applications, que ce soit dans les sciences de l'ingénieur, ou dans d'autres domaines des mathématiques appliquées. Sans prétendre être exhaustif, on peut citer : l'optimisation des systèmes électriques [77], l'évaluation des risques-clients (credit scoring) [6, 68], l'évaluation des diagnostiques médicaux [71], etc.

En particulier, si le domaine de la fonction concave à optimiser est polyédral, alors il possède un nombre fini de points extrêmes, ainsi la technique d'énumération pourra être utilisée pour la résolution du problème de programmation quadratique concave si la dimension de celui-ci est petite. Dans le cas contraire, il faudrait appliquer un algorithme itératif pour sa résolution. Plusieurs algorithmes sont développés pour la résolution du problème de minimisation d'une fonction concave sous contraintes linéaires, on cite : les techniques de coupe [95] qui n'assurent pas la convergence par fois [105], les méthodes d'ap-

proximation [31], les méthodes branch-and-bound [29, 88], les techniques qui transforment le problème en un problème linéaire en nombres entiers équivalent, puis le résoudre par des algorithmes de branch-and-bound [103], les techniques de programmation semi-définie [104], etc.

Dans [31], une nouvelle approche pour la recherche du minimum global approché d'un problème d'optimisation d'une fonction quadratique concave sous contraintes linéaires d'inégalité a été proposée. Cet algorithme consiste à commencer d'un minimum local puis de passer à un point extrême amélioré en calculant d'abord des ensembles d'approximation du premier ordre et du second ordre, puis en résolvant plusieurs programmes linéaires. Cependant, l'algorithme proposé se limite à la minimisation d'une fonction quadratique strictement concave sous contraintes linéaires d'inégalité et des variables non-négatives. De plus, les ensembles d'approximation construits, contiennent des points qui doivent être réalisables, ce qui rend coûteux le calcul de ces ensembles.

Notre contribution essentielle dans cette thèse est le développement d'un nouvel algorithme qui généralise l'algorithme proposé dans [31]. En effet, notre algorithme permet de minimiser une fonction quadratique concave quelconque sous contraintes linéaires d'égalité et d'inégalité avec des variables qui peuvent être non-négatives et/ou bornées. Il permet également de résoudre des problèmes de minimisation quadratique concave sous des contraintes de bornes uniquement. Notons que les ensembles d'approximation construits dans notre algorithme contiennent des points qui ne sont pas forcément réalisables, par conséquent les conditions qui doivent être vérifiées par ces points sont plus faibles que celles utilisées dans [31], ce qui nous assure un gain de temps considérable.

Ce travail est constitué d'une introduction, de cinq chapitres et d'une conclusion, le premier chapitre est consacré à des rappels sur quelques notions classiques d'algèbre linéaire, de convexité, ainsi que des conditions d'optimalité locales pour un problème d'optimisation non linéaire avec contraintes linéaires.

Le deuxième chapitre est consacré aux méthodes de résolution des programmes quadratiques convexes. Le troisième chapitre est consacré à la programmation quadratique concave. Nous présentons d'abord quelques approches utilisées pour la résolution des problèmes quadratiques concaves. Par la suite, on présentera une variété d'applications

des problèmes d'optimisation quadratique concave.

Dans le quatrième chapitre, qui est le noyau de cette thèse représentant notre contribution, nous allons décrire et justifier l'algorithme proposé et l'illustrer avec deux exemples numériques. Dans le dernier chapitre, nous allons présenter les résultats expérimentaux des comparaisons numériques de notre algorithme avec l'algorithme branch-and-bound implémenté dans CPLEX12.8 [59], l'algorithme de Rusakov [88] et l'algorithme des ensembles d'approximation développé dans [31]. Enfin, nous clôturons cette thèse par une conclusion générale et quelques perspectives de recherche.

Chapitre 1

Outils de base

1.1 Introduction

En se basant sur les références [10, 37, 47, 49, 60, 75, 86], dans ce chapitre, nous allons rappeler un certain nombre de notions et de résultats relatifs à l'algèbre linéaire (les matrices, les espaces vectoriels de dimension finie, les formes linéaires, les formes bilinéaires et les formes quadratiques), l'analyse convexe et l'optimisation, et dont un usage constant sera fait dans toute la suite de cette thèse.

1.2 Rappels d'algèbre linéaire

1.2.1 Vecteurs et matrices

Soit $V = \{v_1, v_2, \ldots, v_k\}$ un ensemble de k vecteurs de $\mathbb{R}^n$.

• Un vecteur $x \in \mathbb{R}^n$ est une combinaison linéaire des vecteurs de V si

$$x = \sum_{i=1}^{k} \lambda_i v_i, \ \lambda_i \in \mathbb{R}, \ i = 1, 2, \ldots, k.$$

• L'espace vectoriel engendré par les vecteurs de V, noté $Vect(V)$, est l'ensemble des vecteurs de $\mathbb{R}^n$ qui sont des combinaisons linéaires des vecteurs de V.

• Les vecteurs de V sont dits linéairement indépendants si aucun des vecteurs de V ne peut être une combinaison linéaire des autres vecteurs de V.

• Un espace vectoriel E est de dimension m, noté $dim\ E$, s'il existe m vecteurs linéairement indépendants de E, tels que tout vecteur de E peut s'écrire comme une combinaison

linéaire unique de ces vecteurs.

• Une combinaison affine est une combinaison linéaire $\sum_{i=1}^{k} \lambda_i v_i$ où les coefficients λ_i vérifient la relation $\sum_{i=1}^{k} \lambda_i = 1$.

• L'espace affine engendré par les vecteurs de V, noté Aff(V), est l'ensemble des vecteurs de $\mathbb{R}^n$ qui sont des combinaisons affines des éléments de V.

• Les vecteurs de V sont dits affinement indépendants si les vecteurs $v_j - v_1$ $(1 \leq j \leq k)$ sont linéairement indépendants.

• Un espace affine A est de dimension m, noté $dim\ A$, s'il existe $m+1$ points de A, tels que tout élément de A peut s'écrire comme une combinaison affine unique de ces points.

• Soient deux ensembles d'indices :

$$I = \{1, 2, \ldots, i, \ldots, m\}, J = \{1, 2, \ldots, j, \ldots, n\},\ m \leq n.$$

La base canonique de $\mathbb{R}^n$ se compose de vecteurs $\{e_j,\ j \in J\}$, définis par :

$$e_j = \begin{pmatrix} \delta_{1j} \\ \delta_{2j} \\ \vdots \\ \delta_{nj} \end{pmatrix},$$

où δ_{ij} désigne le symbole de Kronecker :

$$\delta_{ij} = \begin{cases} 1, & \text{si } \ i = j; \\ 0, & \text{si } \ i \neq j. \end{cases}$$

Tout vecteur $v \in \mathbb{R}^n$ admet alors une décomposition unique

$$v = \sum_{j \in J} v_j e_j.$$

En notation matricielle, le vecteur $v = \sum\limits_{j \in J} v_j e_j$ sera toujours représenté par le vecteur colonne

$$v = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix},$$

et on désignera par v^T le vecteur ligne suivant :

$$v^T = (v_1, v_2, \ldots, v_n).$$

Le vecteur ligne v^T et le vecteur transposé du vecteur colonne v.

• Une application linéaire

$$\mathcal{A} : \mathbb{R}^n \to \mathbb{R}^m$$

est représentée par la matrice à m lignes et n colonnes

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

où i est l'indice de la ligne et j celui de la colonne. Pour des calculs pratiques, la matrice A se note aussi

$$A = (a_1, a_2, \dots, a_j, \dots, a_n) = \begin{pmatrix} A_1^T \\ A_2^T \\ \vdots \\ A_j^T \\ \vdots \\ A_m^T \end{pmatrix},$$

où

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

est un vecteur-colonne de dimension m, et

$$A_i^T = A(i, J) = (a_{i1}, a_{i2}, \dots, a_{in})$$

est un vecteur-ligne de dimension n. On note $A^T = (a_{ji} = a_{ij}, j \in J, i \in I)$ la transposée de A.

• La matrice A est dite carrée si on a $n = m$, les éléments a_{ii} sont appelés éléments diagonaux, et les éléments $a_{ij}, i \neq j$, sont appelés élément hors-diagonaux, de plus, si $A = A^T$, la matrice est dite symétrique.

• La matrice identité d'ordre n est la matrice

$$I_n = (\delta_{ij}, i, j \in J).$$

• Une matrice A est inversible s'il existe une matrice (unique si elle existe) notée A^{-1} et appelée matrice inverse de la matrice A, telle que $AA^{-1} = A^{-1}A = I_n$. Dans le cas contraire, on dit que la matrice est singulière.

1.2.2 Formes bilinéaires et quadratiques

• Une application f de $\mathbb{R}^m \times \mathbb{R}^n$ dans $\mathbb{R}^k$ est dite bilinéaire si pour tout $y \in \mathbb{R}^n$ l'application $x \mapsto f(x, y)$ est linéaire de $\mathbb{R}^m$ dans $\mathbb{R}^k$ et pour tout $x \in \mathbb{R}^m$, l'application $y \mapsto f(x, y)$ est linéaire de $\mathbb{R}^n$ dans $\mathbb{R}^k$.

• On appelle forme bilinéaire sur $\mathbb{R}^n$ tout application bilinéaire de $\mathbb{R}^n \times \mathbb{R}^n$ dans $\mathbb{R}$.

• Une forme bilinéaire $f : \mathbb{R}^n \times \mathbb{R}^n \longrightarrow \mathbb{R}$ est dite symétrique si elle vérifie :

$$\forall (x, y) \in \mathbb{R}^n \times \mathbb{R}^n,\ f(x, y) = f(y, x).$$

• Soit f est une forme bilinéaire symétrique sur $\mathbb{R}^n$. On appelle forme quadratique associée à f, l'application de $\mathbb{R}^n$ dans $\mathbb{R}$, notée q_f, définie par

$$q_f(x) = f(x, x),\ \text{pour tout } x \in \mathbb{R}^n.$$

$q_f(x)$ est un polynôme homogène de degré 2 en $x_1, x_2, \ldots, x_n$, i.e., une combinaison linéaire d'expressions de la forme x_i^2 ou $x_i x_j, i \neq j$:

$$q_f(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} x_i x_j f(e_i, e_j).$$

Posons $q_{ij} = f(e_i, e_j),\ 1 \leq i \leq n,\ 1 \leq j \leq n$. La matrice $Q = (q_{ij}, 1 \leq i, j \leq n)$ est appelée la matrice de q_f dans la base canonique de $\mathbb{R}^n$. On a alors

$$q_f(x) = x^T Q x. \tag{1.1}$$

En écrivant la matrice Q sous la forme de vecteurs-colonnes :

$$Q = (q_1, q_2, \ldots, q_n),$$

l'expression (1.1) peut se mettre sous la forme suivante :

$$q_f(x) = (x_1, x_2, \ldots, x_j, \ldots, x_n) \begin{pmatrix} q_1^T x \\ q_2^T x \\ \vdots \\ q_j^T x \\ \vdots \\ q_n^T x \end{pmatrix} = \sum_{j=1}^{n} x_j q_j^T x.$$

• Le gradient d'une forme quadratique q_f au point x est donné par :

$$\nabla q_f(x) = \begin{pmatrix} \frac{\partial q_f}{\partial x_1} \\ \frac{\partial q_f}{\partial x_2} \\ \vdots \\ \frac{\partial q_f}{\partial x_n} \end{pmatrix} = 2Qx.$$

• Le hessien d'une forme quadratique q_f au point x est donné par :

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

• q_f est dite définie positive si $q_f(x) > 0, \forall x \in \mathbb{R}^n$ et $x \neq 0$. Elle est dite semi-définie positive ou définie non négative si $q_f(x) \geq 0, \forall x \in \mathbb{R}^n$.

• q_f est dite définie négative si $q_f(x) < 0, \forall x \in \mathbb{R}^n$ et $x \neq 0$. Elle est dite semi-définie négative ou définie non positive si $q_f(x) \leq 0, \forall x \in \mathbb{R}^n$.

• Une matrice symétrique Q est dite matrice définie positive (non négative) et on note $Q > 0$ ($Q \geq 0$) si elle est associée à une forme quadratique définie positive (non négative).

• Les éléments de la diagonale d'une matrice symétrique Q semi-définie positive ne peuvent s'annuler que si les autres éléments de la même ligne et colonne s'annulent aussi. De plus, si pour $x \in \mathbb{R}^n$, $x^T Q x = 0$ alors on aura : $Qx = 0$.

1.3 Rappels d'analyse convexe

1.3.1 Ensembles convexes

A l'école on a appris qu'une figure s'appelle convexe si elle contient, pour n'importe quelle paire de ses points x, y, le segment entier $[x, y]$ liant ces points. C'est exactement la définition d'un ensemble convexe dans le cas multidimensionnel ; il suffit d'exprimer en langage mathématique le sens de la phrase « le segment $[x, y]$ liant les points $x, y \in \mathbb{R}^n$ ».

• Soient x, y deux points dans $\mathbb{R}^n$. L'ensemble

$$[x, y] = \{z = \lambda x + (1 - \lambda)y : 0 \leq \lambda \leq 1\},$$

est appelé segment avec les extrémités x, y.

• Un sous-ensemble C de $\mathbb{R}^n$ est appelé convexe, s'il contient pour toute paire de points x, y, le segment entier $[x, y]$:

$$\forall x, y \in C,\ \forall \lambda \in [0, 1],\ \lambda x + (1 - \lambda)y \in C.$$

Soit $V = \{v_1, v_2, \ldots, v_k\}$ un ensemble de k vecteurs de $\mathbb{R}^n$.

• Un vecteur $x \in \mathbb{R}^n$ est une combinaison convexe des vecteurs de V si

$$x = \sum_{i=1}^{k} \lambda_i v_i,\ \sum_{i=1}^{k} \lambda_i = 1,\ \lambda_i \in \mathbb{R}_+,\ i = 1, 2, \ldots, k.$$

• Un ensemble convexe C est dit de dimention m si l'espace affine engendré par C est de dimension m.

• On définit l'enveloppe convexe des vecteurs de V, notée $Conv(V)$, comme étant l'ensemble de tous les vecteurs de $\mathbb{R}^n$ qui sont des combinaisons convexes des vecteurs de V.

Les exemples les plus simples d'ensemble convexe sont des singletons (points) et l'espace entier $\mathbb{R}^n$. Des exemples beaucoup plus intéressants sont les suivants :

• L'ensemble des solutions d'un système d'inégalités linéaires à n variables réelles x_j et m inégalités :

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \text{ avec } a_{ij}, b_i \in \mathbb{R} \text{ pour } 1 \leq i \leq m$$

est convexe. Ce système s'écrit aussi sous la forme matricielle $Ax \leq b$, où $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m$). Ce type d'ensemble porte le nom de polyèdre. Remarquez que tout polyèdre est

aussi fermé. S'il est de plus borné, on l'appelle polytope.

• Un sous-ensemble non vide C de $\mathbb{R}^n$ s'appelle conique, s'il contient, pour chaque point $x \in C$, le rayon $\mathbf{R}x = \{tx \ : \ t \geq 0\}$ engendré par le point x :

$$x \in C \Rightarrow tx \in C, \ \forall t \geq 0.$$

Un ensemble conique convexe s'appelle cône.

• L'ensemble des solutions d'un système homogène fini de m inégalités linéaires $Ax \leq 0$ (A est une matrice $m \times n$) est un cône ; un cône de ce type s'appelle polyhèdral.

1.3.2 Propriétés algébriques des ensembles convexes

• Les opérations de la somme et la multiplication par des réels préservent la convexité des ensembles :

Si $C_1, C_2, \ldots, C_k$ sont convexes dans R^n et $\lambda_1, \lambda_2, \ldots, \lambda_k$ sont des réels, alors l'ensemble

$$\lambda_1 C_1 + \lambda_2 C_2 + \cdots + \lambda_k C_k = \{\sum_{j=1}^{k} \lambda_j x^j : x^j \in C_j, \ j = 1, 2, \ldots, k\}$$

est convexe.

• L'ensemble $\bigcap_{j=1}^{k} C_j$ est convexe.

Theorème 1.1 (Théorème de Carathéodory [10]). *Soit $S \subset \mathbb{R}^n$. Si $x \in Conv(S)$, alors $x \in Conv(x^1, x^2, \ldots, x^{n+1})$, où $x^j \in S, \ j = 1, 2, \ldots, n+1$.*

1.3.3 Points extrêmes

Géométriquement, un point extrême d'un ensemble convexe C est un point de C qui ne peut pas être une combinaison convexe d'autres points de l'ensemble. La définition exacte d'un point extrême est comme suit :

• Un point $x \in C$ est dit point extrême de C, s'il n'existe aucun segment $[u, v] \subset C$ de longueur positive pour qui x est un point intérieur, c'est-à-dire, si la relation

$$x = \lambda u + (1 - \lambda)v,$$

avec un certain $\lambda \in]0,1[$ et $u, v \in C$ est possible si et seulement si $u = v = x$. Autrement dit x est un point extrême de C si on ne peut pas l'exprimer comme une combinaison convexe de deux autres points différents de C.

• Un point x d'un ensemble convexe C est extrême si et seulement si l'ensemble $C \setminus \{x\}$ est convexe.

• Le nombre de points extrêmes d'un polytope est fini.

• L'enveloppe convexe de $r + 1$ points affinement indépendants dans $\mathbb{R}^n$, est appelée $r-$simplexe. Les $r-$simplexes sont donc exactement les polytopes de dimension r avec $r + 1$ sommets (i.e., points extrêmes).

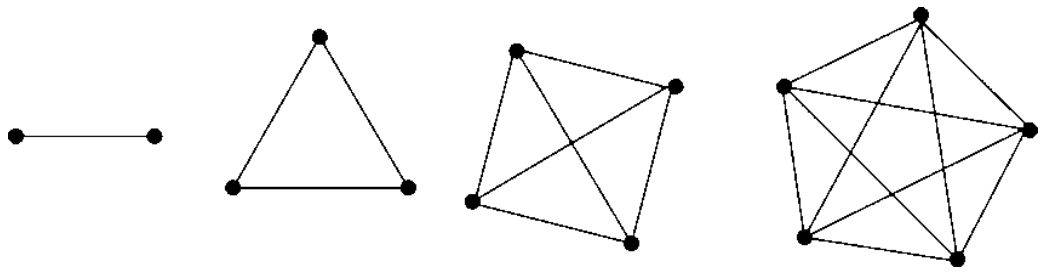

FIGURE 1.1 – Les simplexes.

1.3.4 Fonctions convexes

• Une fonction f définie sur un sous-ensemble convexe C de $\mathbb{R}^n$ et à valeurs dans $\mathbb{R}$ et dite convexe si pour tous les couples $(x, y) \in C^2$ et pour tout $\lambda \in [0, 1]$, on a l'inégalité de convexité suivante :

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y).$$

• Une fonction f est dite concave si $-f$ est convexe.

• Une forme quadratique $q(x) = x^T Qx$, où Q est une matrice symétrique est convexe si et seulement si la matrice Q est semi-définie positive ($Q \geq 0$).

• Soient f est une fonction de $C \subset \mathbb{R}^n$ dans $\mathbb{R}$ et $z \in C$. L'ensemble

$$E_{f(z)}(f) = \{x \in C : f(x) = f(z)\}$$

est appellé ligne de niveau de f au point z.

Soient f une fonction définie de $C \subset \mathbb{R}^n$ dans $\mathbb{R}$ et D son domaine (l'ensemble des $x \in C$

où $f(x) < +\infty$).

• Un vecteur $\eta \in \mathbb{R}^n$ est appelé sous-gradient de f au point $x_0 \in D$ si

$$\forall x \in D,\ \eta^T(x - x_0) \leq f(x) - f(x_0).$$

• L'ensemble de tous les sous-gradients en x_0 est appelé sous-différentiel de f. Il est noté $\partial f(x_0)$.

• La conjuguée de f (non nécessairement convexe) en $y \in \mathbb{R}^n$ est définie par :

$$f^*(y) = \sup_{x \in C} \left(y^T x - f(x)\right).$$

1.4 Optimisation non linéaire

Un problème de minimisation d'une fonction non linéaire f définie de $\mathbb{R}^n$ dans $\mathbb{R}$ sur un ensemble $S \subset \mathbb{R}^n$ se présente sous la forme suivante :

$$\min_{x \in S} f(x). \tag{1.2}$$

Dans ce qui suit, on suppose que S est non vide, convexe et fermé et que f est continue et differentiable sur S.

• Tout vecteur $x \in S$ est appelé une solution réalisable (ou admissible) du problème (1.2).

• Une direction admissible de S en $x^0 \in S$ est un vecteur d, tel que

$$d \neq 0 \text{ et } x^0 + \theta d \in S,\ \forall \theta \in]0, \bar{\theta}] \text{ pour un certain } \bar{\theta} > 0.$$

• Si x^0 est un point intérieur de S, alors toutes les directions de S en x_0 sont admissibles.

• Soit un vecteur $d \in \mathbb{R}^n$ une direction admissible en un point $x \in S$. Le vecteur d est dit une direction de descente si

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

Les minima locaux et globaux de f sur S sont définis de la manière suivante :

• Le vecteur $\tilde{x}$ est un minimum (solution optimale) local du problème (1.2), si

$$\exists \epsilon > 0\ :\ f(\tilde{x}) \leq f(x),\ \forall x \in S,\ \text{avec}\ \|x - \tilde{x}\| \leq \epsilon.$$

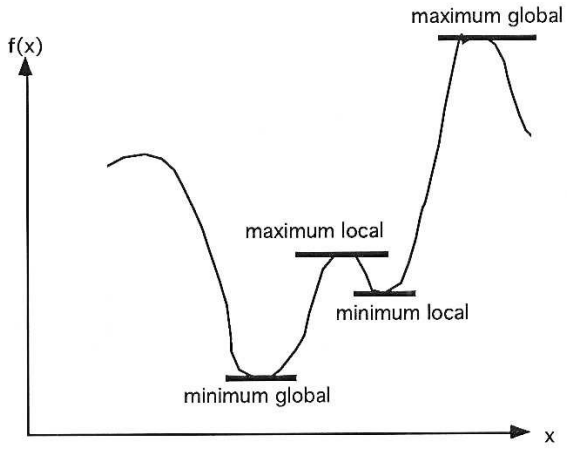

FIGURE 1.2 – Optimum local et global.

• Le vecteur x^* est un minimum global du problème (1.2), si

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

Theorème 1.2. *Soient S un sous-ensemble convexe et compact de $\mathbb{R}^n$, $f : S \to \mathbb{R}$ une fonction concave et continue sur S. Alors le minimum global de f sur S est atteint en un point extrême de S.*

Preuve. Puisque f est continue sur S et que S est un ensemble compact, d'après le théorème de Weierstrass, le minimum global de f sur S existe. Il suffit de montrer que pour tout point x de S, il existe un point extrême x^* de S tel que $f(x) \geq f(x^*)$. En effet, d'après le théorème de Carathéodory, il existe $n+1$ points extrêmes x^i de S tels que :

$$x = \sum_{i=1}^{n+1} \lambda_i x^i,\ \sum_{i=1}^{n+1} \lambda_i = 1,\ \lambda_i \geq 0,\ i = 1, 2, \ldots, n+1.$$

Soit x^* un point extrême satisfaisant $f(x^*) = \min\{f(x^i) :\ i = 1, 2, \ldots, n+1\}$. Puisque f est concave, on obtient

$$f(x) = f(\sum_{i=1}^{n+1} \lambda_i x^i) \geq \sum_{i=1}^{n+1} \lambda_i f(x^i) \geq \sum_{i=1}^{n+1} \lambda_i f(x^*) = f(x^*).$$

□

Remarque 1.1. *Dans un problème de programmation non linéaire où la fonction f est convexe, l'optimum peut être atteint en un point quelconque de S (en un point extrême, point frontière ou encore point intérieur).*

Exemple 1.1. *Considérons le programme quadratique suivant :*

$$\begin{aligned} min \quad & f(x) = (x_1 - 4)^2 + (x_2 - 6)^2, \\ s.c. \quad & -x_1 - x_2 \leq -1, \\ & 2x_1 + 3x_2 \leq 12, \\ & x_1 \geq 0, x_2 \geq 0. \end{aligned}$$

L'ensemble S est un polytope et il est représenté géométriquement sur la figure 1.3

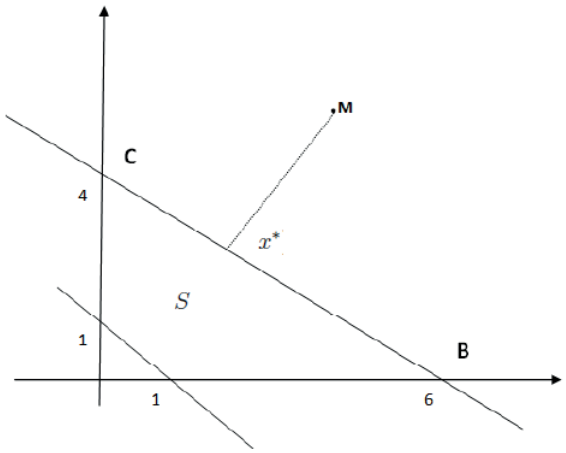

FIGURE 1.3 – Représentation géométrique de l'ensemble S de l'exemple 1.1.

L'équation $f(x) = \alpha =$ constante > 0 représente un cercle de centre $M(4, 6)$ et de rayon $R = \sqrt{\alpha}$. En faisant diminuer le nombre α, la valeur de $f(x)$ diminue. Il s'ensuit alors que le point x^*, qui est l'intersection des droites perpendiculaires BC et Mx^*, réalise le minimum de la fonction f sur l'ensemble S. De plus, le point x^* se trouve sur l'arête BC du polyèdre S. Ici, on trouve

$$x^* = (\frac{24}{13}, \frac{36}{13})^T, \ f(x^*) = \frac{196}{13}.$$

Exemple 1.2. *Considérons la fonction $f(x) = (x_1 - 4)^2 + (x_2 - 1)^2$, et cherchons son minimum sur le même polyèdre précédent S. Cette fois-ci le minimum sera atteint au point intérieur $x^* = (4, 1)^T$, $f(x^*) = 0$.*

Exemple 1.3. *Soit la fonction* $f(x) = (x_1 - 8)^2 + x_2^2$, *et cherchons son minimum sur le même polyèdre précédent* S. *Cette fois-ci le minimum sera atteint au point extrême* $x^* = (6, 0)^T$, $f(x^*) = 4$.

Afin d'anlyser ou résoudre de manière efficace un problème d'optimisation, il est fondamental de pouvoir disposer de conditions d'optimalité. En effet, celles-ci nous servent non seulement à verifier la validité des solutions obtenues, mais souvent l'étude de ces conditions aboutit au développement des algorithmes de résolution eux-mêmes.

Des conditions équivalentes peuvent être obtenues de diverses manières, en procédant à des analyses suivant différentes lignes directrices. L'approche considérée ici pour l'obtention de ces conditions est basée sur les notions de direction de descente et de direction admissible.

1.4.1 Conditions nécessaires d'optimalité

Theorème 1.3. *(Condition nécessaire d'optimalité) Soit* f *une fonction non linéaire, définie de* $\mathbb{R}^n$ *dans* $\mathbb{R}$ *et de classe* $\mathcal{C}^1$. *Si* x^0 *est un minimum (local ou global) du problème (1.2), alors pour toute direction admissible* d *en* x^0, *on a*

$$[\nabla f(x^0)]^T d \geq 0. \tag{1.3}$$

Preuve. Soit d une direction admissible en x^0. Il existe alors $\bar{\theta} > 0$ tel que $x^0 + \theta d \in S$ pour tout $\theta \in [0, \bar{\theta}]$. Considérons la fonction $\varphi(\theta) = f(x^0 + \theta d)$.
Comme x^0 est un minimum local, il existe $\epsilon > 0$ tel que

$$f(x^0) \leq f(x), \forall x \in S \cap \mathcal{B}(x^0, \epsilon)\,^1.$$

Considérons

$$S_{\bar{\theta}} = \left\{x^0 + \theta d, \theta \in [0, \bar{\theta}]\right\}.$$

$\forall x \in S_{\bar{\theta}} \cap S \cap \mathcal{B}(x^0, \epsilon)$, on aura $f(x^0) \leq f(x)$.
Soit $[0, \tilde{\theta}] = \left\{\theta \in [0, \bar{\theta}] \ : \ x^0 + \theta d \in S_{\bar{\theta}} \cap S \cap \mathcal{B}(x^0, \epsilon)\right\}$, et $\varphi(\theta) \geq \varphi(0), \forall \theta \in [0, \tilde{\theta}]$.
D'où $\theta^0 = 0$ est un minimum local de $\varphi(\theta)$ sur $[0, \tilde{\theta}]$.
On a

$$\varphi(\theta) = \varphi(0) + \theta \varphi'_+(0) + o(\alpha)$$

1. $\mathcal{B}(x^0, \epsilon) = \{x \in \mathbb{R}^n : \left\|x - x^0\right\| \leq \epsilon\}$

où

$$\varphi'_+(0) = \lim_{\theta \to 0^+} \frac{\varphi(\theta) - \varphi(0)}{\theta}.$$

Si $\varphi'_+(0)$ était négative, alors pour un nombre θ assez petit, on aurait eu $\varphi(\theta) < \varphi(0)$, ce qui contredirait le fait que φ possède un minimum local en $\theta = 0$. On a donc bien $\varphi'_+(0) \geq 0$.

Par conséquent, comme

$$\varphi'(\theta) = [\nabla f(x^0 + \theta d)]^T d,$$

on obteint

$$\varphi'_+(0) = [\nabla f(x^0)]^T d \geq 0.$$

La relation (1.3) est alors démontrée. □

Remarque 1.2.

- *Si* $\nabla f(x) \neq 0$ *et* $[\nabla f(x)]^T d < 0$*, alors* d *est une direction de descente.*
- *Si* d *est une direction de descente, alors* $[\nabla f(x)]^T d \leq 0$*.*

Dans [21], le théorème 1.3 s'énonce de la façon suivante :

Theorème 1.4. *Soit* f *une fonction définie de* $\mathbb{R}^n$ *à* $\mathbb{R}$*, qui est de classe* $\mathcal{C}^1$ *sur* S *un sous-ensemble non vide, convexe et fermé de* $\mathbb{R}^n$ *et soit* $x^* \in S$*.*

(a) *Si* x^* *est un minimum local de* f *sur* S*, alors*

$$(x - x^*)^T \nabla f(x^*) \geq 0, \forall x \in S. \tag{1.4}$$

(b) *Si* f *est convexe sur* S*, alors la condition de la partie (a) est également suffisante pour que* x^* *minimise* f *sur* S*.*

1.4.2 Conditions d'optimalité pour le cas de contraintes linéaires de type égalités

Considèrons le problème suivant :

$$\min f(x), \tag{1.5}$$

$$g(x) = Ax - b = 0, \tag{1.6}$$

où f est une fonction non linéaire définie de $\mathbb{R}^n$ dans $\mathbb{R}$ et de classe $\mathcal{C}^1$; A est une matrice réelle de dimension $m \times n$, $b \in \mathbb{R}^m$, $x \in \mathbb{R}^n$.
Pour que l'ensemble des solutions réalisables

$$S = \{x \in \mathbb{R}^n : \ g(x) = 0\} = \{x \in \mathbb{R}^n : \ g_i(x) = A_i^T x - b_i = 0, \ i = 1, 2, \ldots, m\},$$

ne soit pas vide ou ne soit pas réduit à un point isolé, on cosidérera que $rangA = m < n$.

Condition nécessaire d'optimalité du premier ordre

Proposition 1.1. *Soit x une solution réalisable du problème (1.5)-(1.6). Un vecteur $d \in \mathbb{R}^n$ est alors une direction admissible en x si et seulement si*

$$Ad = 0. \tag{1.7}$$

De plus, pour un tel vecteur, on a $x(\theta) = x + \theta d \in S, \ \forall \theta \in \mathbb{R}$.

Démonstration. Soit $x \in S$ et d une direction admissible en x, alors $\exists \delta > 0, \ \forall \theta \in]0, \delta], \ x + \theta d \in S$.
On a alors $\forall \theta \in]0, \delta], \ A(x + \theta d) = b \Rightarrow Ax + \theta Ad = b \Rightarrow \theta Ad = 0 \Rightarrow Ad = 0$.
Inversement, soit $d \in \mathbb{R}^n$, tel que $Ad = 0$ et considérons les points $x(\theta) = x + \theta d, \ \theta \in \mathbb{R}$.
On a

$$\forall \theta \in \mathbb{R}^*, \ g(x(\theta)) = Ax(\theta) - b = Ax + \theta Ad - b = 0.$$

Pour $\theta = 0, \ x(\theta)$ se réduit à x qui vérifie $g(x) = 0$. Par conséquent, d est une direction admissible, de plus $\forall \theta \in \mathbb{R}, \ x(\theta) \in S$. □

Pour le problème (1.5)-(1.6), introduisons la fonction $L(x, \lambda)$, avec $x \in \mathbb{R}^n$ et $\lambda = (\lambda_j, \ j = 1, 2, \ldots, m) \in \mathbb{R}^m$, appelée fonction de Lagrange :

$$L(x, \lambda) = f(x) + \lambda^T g(x) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) = f(x) + \lambda^T Ax - \lambda^T b. \tag{1.8}$$

Notons par

$$\nabla L(x, \lambda) = \begin{pmatrix} \nabla_x L(x, \lambda) \\ \nabla_\lambda L(x, \lambda) \end{pmatrix}.$$

Theorème 1.5 (Multiplicateurs de Lagrange)**.** *Si x^0 est un minimum local (ou global) pour le problème (1.5)-(1.6), alors il existe un vecteur unique $\lambda^0 \in \mathbb{R}^m$, appelé vecteur des multiplicateurs de Lagrange, tel que*

$$\nabla L(x^0, \lambda^0) = 0. \tag{1.9}$$

Preuve. Soit x^0 un minimum pour le problème (1.5)-(1.6), alors la fonction $\varphi(\theta) = f(x^0 + \theta d)$ pour $Ad = 0$ admet aussi un minimum au point $\theta = 0$.
On en déduit que

$$\varphi'(0) = [\nabla f(x^0)]^T d = 0. \tag{1.10}$$

L'égalité (1.10) montre que le vecteur d est orthogonal au gradient de f au point x^0. Il est aussi orthogonal à chacun des vecteurs-lignes de la matrice A car $Ad = 0$. Ceci n'est possible que s'il existe un vecteur $\lambda^0 = (\lambda_j, j = 1, \ldots, m) \in \mathbb{R}^m$, tel que

$$\nabla f(x^0) = -\sum_{i=1}^{m} \lambda_i^0 A_i = -A^T \lambda^0. \tag{1.11}$$

En effet, dans le cas contraire, pour tout vecteur $\lambda \in \mathbb{R}^m$, on aura $\nabla f(x^0) \neq -\sum_{i=1}^m \lambda_i^0 A_i$. Donc le vecteur $\nabla f(x^0)$ n'appartient pas au sous-espace vectoriel engendré par les vecteurs-lignes de la matrice A. Il existe alors un vecteur $d^0 \neq 0$ tel que :

$$A_i^T d^0 = 0,\ 1 \leq i \leq m,\ [\nabla f(x^0)]^T d^0 \neq 0.$$

Ce qui contredit le critère d'optimalité (1.10). Donc

$$\nabla f(x^0) + \sum_{i=1}^{m} \lambda_i^0 A_i = 0 \Rightarrow \nabla f(x^0) + \sum_{i=1}^{m} \lambda_i^0 \nabla(g_i)(x^0) = 0 \Rightarrow \nabla_x L(x^0, \lambda^0) = 0. \tag{1.12}$$

Comme x^0 est réalisable,

$$\nabla_\lambda L(x^0, \lambda^0) = \begin{pmatrix} g_1(x^0) \\ g_2(x^0) \\ \vdots \\ g_m(x^0) \end{pmatrix} = g(x^0) = 0.$$

Donc si x^0 est un minimum du problème (1.5)-(1.6), alors le couple (x^0, λ^0) est un point stationnaire de la fonction de Lagrange (1.8). □

Theorème 1.6. *Le vecteur des multiplicateurs de Lagrange λ^0 est unique si et seulement si $rangA = m$.*

Preuve. Soit $rangA = m$ et supposons que pour le minimum x^0, il correspond deux vecteurs λ^0 et $\lambda^{0'}$, tels que

$$\nabla f(x^0) + \sum_{i=1}^{m} \lambda_i^0 A_i = 0,\ \nabla f(x^0) + \sum_{i=1}^{m} \lambda_i^{0'} A_i = 0.$$

On en déduit que

$$\sum_{i=1}^{m} (\lambda_i^0 - \lambda_i^{0'}) A_i = 0.$$

Comme $rangA = m$, alors les vecteurs $A_i, 1 \leq i \leq m$ sont linéairement indépendants. Donc, $\lambda_i = \lambda_i',\ 1 \leq i \leq m$, d'où $\lambda^0 = \lambda^{0'}$.
Inversement, soit λ^0 un vecteur unique correspondant au minimum x^0 et considérons la combinaison linéaire nulle suivante :

$$\sum_{i=1}^{m} \alpha_i A_i = 0.$$

De la relation (1.12)

$$\nabla f(x^0) + \sum_{i=1}^{m} \lambda_i^0 A_i - \sum_{i=1}^{m} \alpha_i A_i = \nabla f(x^0) + \sum_{i=1}^{m} (\lambda_i^0 - \alpha_i) A_i = 0.$$

Donc si $\lambda_i^0 - \alpha_i = \lambda_i'$, on aura $\nabla f(x^0) + \sum_{i=1}^{m} \lambda_i' A_i = 0$. Le vecteur λ' vérifie aussi, comme λ, la relation (1.12). En vertu de l'unicité du vecteur des multiplicateurs de Lagrange, on aura $\lambda^0 = \lambda'$, on déduit alors que $\alpha_i = 0, 1 \leq i \leq m$, et que les vecteurs $A_i, 1 \leq i \leq m$ sont linéairement indépendants. Par conséquent, $rangA = m$. □

Condition nécessaire d'optimalité du second ordre

Considérons le problème (1.5)-(1.6) et supposons de plus que la fonction f est de classe $\mathcal{C}^2$. On a alors la condition nécessaire suivante de second ordre.

Theorème 1.7. *Si x^0 est un minimum du problème (1.5)-(1.6) et λ^0 le vecteur des multiplicateurs de Lagrange correspondant, alors la forme quadratique $\nabla_x^2 L(x^0, \lambda^0)$ est semi-définie positive sur l'ensemble des points vérifiant :*

$$Ad = 0. \tag{1.13}$$

Preuve. Soit d un vecteur de $\mathbb{R}^n$ vérifiant (1.13). On sait déja que le vecteur $x(\theta) = x^0 + \theta d$ est une solution réalisable du problème (1.5)-(1.6) pour tout $\theta \in \mathbb{R}$.
Posons $\varphi(\theta) = f(x(\theta)) = f(x^0 + \theta d)$. Puisque le point x^0 est un minimum pour la fonction f, alors la fonction $\varphi(\theta)$ admet un minimum en $\theta = 0$. D'où

$$\varphi'(0) = d^T \nabla f(x^0) = 0 \text{ et } \varphi''(0) = d^T \nabla^2 f(x^0) d \geq 0.$$

D'autre part, on a

$$L(x, \lambda^0) = f(x) + {\lambda^0}^T g(x) = f(x) + \sum_{i=1}^{m} \lambda_i^0 (A_i^T x - b_i).$$

On en déduit que

$$\nabla_x L(x, \lambda^0) = \nabla f(x) + \sum_{i=1}^{m} \lambda_i^0 A_i, \text{ et } \nabla^2 L(x, \lambda^0) = \nabla^2 f(x).$$

En vertu de la dernière égalité et la dernière inégalité, on obtient finalement

$$d^T \nabla^2 L(x^0, \lambda^0) d \geq 0.$$

□

Condition suffisante d'optimalité du second ordre

Theorème 1.8. *Soit (x^0, λ^0) un couple de vecteurs virifiant la condition nécessaire d'optmalité du premier ordre pour le problème (1.5)-(1.6), i.e., $\nabla_x L(x^0, \lambda^0) = 0$. Pour que x^0 soit un minimum local du problème (1.5)-(1.6), il est alors suffisant que la forme quadratique $d^T \nabla_x^2 L(x^0, \lambda^0) d$ soit définie positive sur l'ensemble $\{d \in \mathbb{R}^n \ : \ Ad = 0\}$.*

Preuve. Soit d un vecteur de $\mathbb{R}^n$ vérifiant $Ad = 0$. Donc $x(\theta) = x^0 + \theta d$ est une solution réalisable pour tout $\theta \in \mathbb{R}$.
Soit $\varphi(\theta) = f(x(\theta)) = f(x^0 + \theta d),\ \theta \in \mathbb{R}$, on a alors

$$\varphi'(\theta) = \nabla f(x^0 + \theta d) \Rightarrow \varphi'(0) = \nabla f(x^0) d,$$
$$\varphi''(\theta) = d^T \nabla^2 f(x^0 + \theta d) d \Rightarrow \varphi''(0) = d^T \nabla^2 f(x^0) d.$$

En outre, de la condition nécessaire du premier ordre (1.11), on déduit que

$$\nabla f(x^0) = -A^T \lambda^0.$$

D'où

$$\varphi'(0) = -A^T\lambda^0 d = 0, \text{ et } \varphi''(0) = d^T\nabla^2 f(x^0)d > 0.$$

La fonction φ admet un minimum local en $\theta = 0$.

Soit maintenant un vecteur $\bar{x}$, solution réalisable du problème (1.5)-(1.6) telle que $\|\bar{x} - x^0\| \leq \epsilon > 0$. Il existe donc un vecteur d tel que $\|d\| = 1, Ad = 0, \bar{x} = x^0 + \epsilon d$.

Il s'ensuit alors que $f(\bar{x}) = f(x^0 + \epsilon d) = \varphi(\epsilon) > \varphi(0) = f(x^0)$. Le point x^0 constitue donc bien un minimum local pour le problème (1.5)-(1.6). □

Cas d'une fonction convexe

Considérons le problème (1.5)-(1.6) où f est une fonction convexe (i.e., $D \geq 0$).

Soit $x^0 \in S$ et λ^0 un m-vecteur tels que le couple (x^0, λ^0) vérifie la condition nécessaire d'optimalité du premier ordre, i.e.,

$$\nabla_x L(x^0, \lambda^0) = \nabla f(x^0) + A^T\lambda^0 = 0. \tag{1.14}$$

Cette dernière relation est aussi suffisante pour l'optimalité du vecteur x^0 dans le problème (1.5)-(1.6).

En effet, puisque S est aussi convexe, on peut écrire

$$f(x) - f(x^0) \geq [\nabla f(x^0)]^T(x - x^0)^T, \ \forall x \in S.$$

D'où

$$\forall x \in S, \ f(x) - f(x^0) \geq [\nabla f(x^0)]^T(x - x^0)^T = -{\lambda^0}^T A(x - x^0).$$

Comme $A(x - x^0) = 0$, on déduit que $f(x) \geq f(x^0)$, $\forall x \in S$. Ce qui prouve que x^0 est un minimum global pour le problème (1.5)-(1.6).

1.4.3 Conditions d'optimalité pour le cas de contraintes linéaires de type inégalités

Avant d'aborder le problème, rappelons un lemme, appelé lemme de Farkas, qui joue un rôle très important en optimisation mathématique.

Lemma 1.1 (Lemme de Farkas). *Soient* $(m+1)$ *vecteurs de* $\mathbb{R}^n, c, A_i, i = 1, 2, \ldots, m, m < n$. *Si pour chaque vecteur* x *vérifiant* $A_i^T x \leq 0, 1 \leq i \leq m$, *on a* $c^T x \leq 0$, *alors il existe des coefficients* $\lambda_i \geq 0, 1 \leq i \leq m$, *tels que* $c = \sum_{i=1}^{m} \lambda_i A_i$.

Preuve. Sans perte de généralité, on peut supposer que les vecteurs $A_i, 1 \leq i \leq m$, sont linéairement indépendants. Montrons alors que le vecteur c appartient au sous-espace engendré par les vecteurs $A_i, 1 \leq i \leq m$, i.e.,

$$c = \sum_{i=1}^{m} \lambda_i A_i. \tag{1.15}$$

Si ce n'était pas le cas, il existerait alors un vecteur $\bar{x}$ orthogonal à tous les vecteurs $A_i, 1 \leq i \leq m$, tel que

$$A_i^T \bar{x} = 0,\ 1 \leq i \leq m,\ c^T \bar{x} = \alpha > 0. \tag{1.16}$$

Soit x^0 un certain vecteur satisfaisant aux conditions du théorème :

$$A_i^T x^0 \leq 0,\ 1 \leq i \leq m,\ c^T x^0 = \beta \leq 0. \tag{1.17}$$

Considérons ensuite le vecteur $x = \bar{x} + \theta x^0$, où

$$\begin{cases} 0 < \theta < \frac{\alpha}{|\beta|}, & \text{si } \beta \neq 0; \\ \theta > 0, & \text{si } \beta = 0. \end{cases}$$

En vertu des relations (1.16) et (1.17), on aura alors

$$\begin{aligned} A_i^T x &= A_i^T \bar{x} + \theta A_i^T x^0 \leq 0, 1 \leq i \leq m, \\ c^T x &= c^T \bar{x} + \theta c^T x^0 = \alpha + \theta\beta > 0. \end{aligned}$$

Cette dernière relation contredit l'hypothèse du lemme, l'égalité (1.15) est donc vraie. Supposons maintenant que dans la relation (1.15), les coefficients λ_i ne sont pas tous positifs ou nuls, par exemple $\lambda_k < 0$. Il existe alors un vecteur $\bar{x}$, tel que

$$A_i^T \bar{x} = 0,\ i \neq k,\ 1 \leq i \leq m,\ A_k^T \bar{x} = \alpha_k < 0.$$

Soit aussi un certain vecteur x^0, satisfaisant aux inégalités $A_i^T x^0 = \beta_i < 0, 1 \leq i \leq m$. On construit le vecteur

$$x = \bar{x} + \theta x^0,\ 0 < \theta < \frac{\alpha_k \lambda_k}{|\sum_{i=1}^{m} \lambda_i \beta_i|}.$$

On aura donc

$$\begin{aligned} A_i^T x &= A_i^T \bar{x} + \theta A_i^T x^0 = \theta\beta_i < 0,\ 1 \leq i \leq m,\ i \neq k; \\ A_k^T x &= A_k^T \bar{x} + \theta A_k^T x^0 = \alpha_k + \theta\beta_k < 0; \end{aligned}$$

mais

$$\begin{aligned} c^T x &= \sum_{i=1}^{m} \lambda_i A_i^T x \\ &= \lambda_k A_k^T x + \sum_{i=1, i\neq k}^{m} \lambda_i A_i^T x \\ &= \lambda_k(\alpha_k + \theta\beta_k) + \sum_{i=1, i\neq k}^{m} \theta\lambda_i\beta_i \\ &= \lambda_k\alpha_k + \theta\sum_{i=1}^{m} \lambda_i\beta_i. \end{aligned}$$

Cette dernière relation contredit l'hypothèse du lemme pour un nombre θ positif assez petit. Notre supposition sur un certain $\lambda_k < 0$ est donc fausse. Le lemme de Farkas est ainsi démontré. □

Remarque 1.3. *Ici, la preuve est faite pour m vecteurs linéairement indépendants, où $m < n$. Ce lemme reste vrai, même pour $m \geq n$. Il suffit alors de prendre les λ_i égaux à zéro pour les vecteurs linéairement dépendants.*

Condition nécessaire d'optimalité du premier ordre

Soit f une fonction non linéaire définie de $\mathbb{R}^n$ dans $\mathbb{R}$ et de classe $\mathcal{C}^1$. Considérons le problème de minimisation de la fonction f sur l'ensemble S défini par les inégalités linéaires suivantes :

$$S = \{x \in \mathbb{R}^n :\ g(x) \leq 0\} = \{x \in \mathbb{R}^n :\ g_i(x) = A_i^T x - b_i \leq 0,\ i \in I\},\ I = \{1, 2, \ldots, m\},$$

où A est une matrice réelle de dimension $m \times n$, $b \in \mathbb{R}^m$. On notera ici que le nombre m peut être plus grand que n.
Le problème de minimisation d'une fonction f avec des contraintes linéaires de type inégalités se résume comme suit :

$$\min f(x), \tag{1.18}$$

$$g(x) = Ax - b \leq 0. \tag{1.19}$$

• Une contrainte $i \in I$ est dite active ou (saturée) au point x si $A_i^T x = b_i$. L'ensemble des indices des contraintes actives au point x, noté $\mathcal{I}_0(x)$, est défini comme suit :

$$\mathcal{I}_0(x) = \{i \in I :\ A_i^T x = b_i\}.$$

Lemma 1.2. *Un vecteur $d \in \mathbb{R}^n$ est une direction admissible au point $x \in S$ pour le problème (1.18)-(1.19) si et seulement si $A_i^T d \leq 0,\ i \in I_0(x)$.*

Preuve. Soit d une direction admissible au point x. Il existe alors un nombre $\bar{\theta} > 0$ assez petit, tel que

$$x(\theta) = x + \theta d \in S, \forall \theta \in]0, \bar{\theta}].$$

On aura donc

$$Ax - b + \theta Ad \leq 0.$$

D'où

$$A_i^T x - b_i + \theta A_i^T d \leq 0,\ i \in I,$$
$$A_i^T x - b_i = 0, \forall i \in \mathcal{I}_0(x).$$

Comme $\theta > 0$, on obtient $A_i^T d \leq 0, \forall i \in \mathcal{I}_0(x)$.
Inversement, soit d un vecteur tel que

$$A_i^T d \leq 0, \forall i \in \mathcal{I}_0(x).$$

Soit le vecteur $x(\theta) = x + \theta d$, où x est une solution réalisable du problème (1.18)-(1.19). On peut alors écrire

$$\begin{aligned} g_i(x(\theta)) &= A_i^T x - b_i + \theta A_i^T d = \theta A_i^T d \leq 0, \forall i \in \mathcal{I}_0(x), \forall \theta \geq 0, \\ g_i(x(\theta)) &= A_i^T x - b_i + \theta A_i^T d = g_i(x) + \theta A_i^T d, \forall i \in I \backslash \mathcal{I}_0(x). \end{aligned}$$

Comme $g_i(x) < 0, \forall i \in I \backslash \mathcal{I}_0(x)$, on peut alors trouver un nombre $\bar{\theta} > 0$ assez petit, tel que

$$g_i(x(\theta)) = g_i(x) + \theta A_i^T d \leq 0, \forall i \in I \backslash \mathcal{I}_0(x), \forall \theta \in [0, \bar{\theta}].$$

Donc

$$g_i(x(\theta)) \leq 0, \forall i \in I \backslash \mathcal{I}_0(x), \forall \theta \in [0, \bar{\theta}].$$

Ce qui revient à dire que

$$x(\theta) = x + \theta d \in S.$$

Le vecteur d considéré est, par conséquent, une direction admissible au point x. □

Theorème 1.9 (Théorème de Karush-Kuhn-Tucker "K.K.T"). *Soit x^0 un minimum du problème (1.18)-(1.19). Il existe alors un $m-$vecteur $\lambda^0 \geq 0$, tel que :*

i) *Pour la fonction de Lagrange $L(x,\lambda) = f(x)+\lambda^T g(x)$, la condition de stationnarité est satisfaite :*

$$\nabla_x L(x^0, \lambda^0) = 0.$$

ii) *La condition de complémentarité est remplie :*

$$\lambda_i^0 g_i(x^0) = 0,\ i \in I.$$

Preuve. Le vecteur x^0 étant un minimum, alors pour toute direction admissible d en x^0, on aura

$$\nabla f(x^0)^T d \geq 0,$$

et ce, en vertu du théorème 1.3. En utilisant le lemme 1.2, on peut écrire

$$A_i^T d \leq 0, \forall i \in \mathcal{I}_0(x^0) \Rightarrow -\nabla f(x^0)^T d \leq 0.$$

En vertu du lemme de Farkas, il existe alors des coefficients $\lambda_i^0 \geq 0,\ i \in \mathcal{I}_0(x^0)$, tels que

$$-\nabla f(x^0) = \sum_{i \in \mathcal{I}_0(x^0)} \lambda_i^0 A_i.$$

Posons $\lambda_i^0 = 0$ pour $i \in I \backslash \mathcal{I}_0(x^0)$. On obtient alors

$$-\nabla f(x^0) = \sum_{i \in I} \lambda_i^0 A_i \Leftrightarrow \nabla_x L(x^0,\ \lambda^0) = 0,\ \lambda_i^0 g_i(x^0) = 0,\ i \in I.$$

Le théorème de K.K.T est ainsi démontré. □

Condition nécessaire d'optimalité du second ordre

Theorème 1.10. *Soit f une fonction de classe $\mathcal{C}^2$. Si x^0 est un minimum du problème (1.18)-(1.19), alors la forme quadratique $d^T \nabla^2 f(x^0) d$ est semi-définie positive sur l'ensemble des points :*

$$A_i^T d = 0,\ i \in \mathcal{I}_0(x^0), \tag{1.20}$$

où $\mathcal{I}_0(x^0)$ est l'ensemble des indices des contraintes actives en x^0.

Preuve. Supposons qu'il existe un vecteur d vérifiant (1.20), tel que $d^T\nabla^2 f(x^0)d < 0$. Considérons le vecteur $x(\theta) = x^0 + \theta d$, pour $\theta > 0$ assez petit, le vecteur $x(\theta)$ est une solution réalisable du problème (1.18)-(1.19).

On peut alors écrire

$$f(x(\theta)) - f(x^0) = \theta[\nabla f(x^0)]^T d + \frac{1}{2}\theta^2 d^T \nabla^2 f(x^0)d + o(\theta^2).$$

En vertu du théorème de K.K.T (théorème 1.9), on a

$$\nabla f(x^0) = -\sum_{i\in\mathcal{I}_0(x^0)} \lambda_i^0 A_i,\ \lambda_i^0 \geq 0,\ i \in \mathcal{I}_0(x^0).$$

D'où

$$[\nabla f(x^0)]^T d = -\sum_{i\in\mathcal{I}_0(x^0)} \lambda_i^0 A_i^T d = 0.$$

Donc

$$f(x(\theta)) - f(x^0) = \theta^2\left(\frac{1}{2}d^T\nabla^2 f(x^0)d + \frac{o(\theta^2)}{\theta^2}\right) < 0,$$

si θ est un nombre positif assez petit.

Cette dernière inégalité contredit le fait que x^0 soit un minimum local de f. Par conséquent, la matrice $\nabla^2 f(x^0)$ est semi-définie positive sur l'ensemble des points vérifiant (1.20). □

Condition suffisante d'optimalité du second ordre

Theorème 1.11. *Soit x^0 une solution réalisable du problème (1.18)-(1.19) vérifiant les conditions nécessaire de K.K.T. Si la matrice $\nabla^2 f(x^0)$ est définie positive sur l'ensemble (1.20), alors x^0 est un minimum du problème (1.18)-(1.19).*

Preuve. La démonstration se fait de la même façon que pour le théorème 1.8. □

Cas de fonction convexe

Considérons le problème (1.18)-(1.19) où f est une fonction convexe.

Soit (x^0, λ^0) un couple de vecteurs vérifiant le théorème de K.K.T :

$$\nabla_x L(x^0, \lambda^0) = 0 \Leftrightarrow \nabla f(x^0) = -\sum_{i=1}^{m} \lambda_i^0 A_i,\ \lambda_i^0 \geq 0,\ 1 \leq i \leq m;$$

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

Alors le vecteur x^0 constitue un minimum global du problème (1.18)-(1.19).

En effet, on peut écrire, du fait que f est convexe et S est convexe,

$$f(x) - f(x^0) \geq [\nabla f(x^0)]^T (x - x^0), \ \forall x \in S.$$

D'où

$$f(x) - f(x^0) \geq -\sum_{i=1}^{m} \lambda_i^0 A_i^T (x - x^0), \ \forall x \in S.$$

Puisque $\lambda_i^0 = 0$, pour $i \in I \backslash \mathcal{I}_0(x^0)$, alors on aura

$$f(x) - f(x^0) \geq -\sum_{i \in \mathcal{I}_0(x^0)} \lambda_i^0 (A_i^T x - A_i^T x^0), \ \forall x \in S.$$

D'où

$$f(x) - f(x^0) \geq -\sum_{i \in \mathcal{I}_0(x^0)} \lambda_i^0 (A_i^T x - b_i), \ \forall x \in S.$$

Par conséquent, x^0 est un minimum global du problème (1.18)-(1.19). Pour une fonction convexe, les conditions de K.K.T sont donc à la fois nécessaires et suffisantes pour l'optimalité de x^0.

Chapitre 2

Méthodes de résolution en programmation quadratique convexe

2.1 Introduction

En se basant sur les références [25, 57, 58, 89], nous présentons dans ce chapitre les différentes approches développées pour la résolution des problèmes de programmation quadratique convexe sous contraintes linéaires, et ce, en donnant leurs principes et leurs algorithmes de base. Il existe plusieurs méthodes pour la résolution des problèmes quadratiques convexes avec contraintes linéaires. Nous citons, par exemple, la méthode du simplexe quadratique de Wolfe [102], la méthode d'activation des contraintes [45], la méthode des points intérieurs [98], etc.

2.2 La méthode du simplexe quadratique de Wolfe

La méthode classique de Wolfe (1959) n'est que la méthode du simplexe légèrement modifiée. Le principe de cette méthode consiste à résoudre le système de Kuhn-Tucker qui se réduit à trouver une solution basique positive ou nulle d'un système linéaire, qui doit vérifier une condition supplémentaire non linéaire appelée condition de complémentarité.

2.2.1 Position du problème

Soit le problème suivant :

$$\min f(x) = \frac{1}{2}x^T Dx + c^T x, \tag{2.1}$$

$$Ax = b, \tag{2.2}$$

$$x \geq 0, \tag{2.3}$$

où D est une matrice réelle semi-définie positive et symétrique d'ordre $n; c, x$ sont des $n-$vecteurs, A est une matrice réelle de dimension $m \times n$, avec $rangA = m$, et b est un $m-$vecteur.

Formulons le théorème de KKT pour le problème (2.1)-(2.3), qui peut s'écrire sous la forme équivalent suivante :

$$\min f(x) = \frac{1}{2}x^T Dx + c^T x, \tag{2.4}$$

$$Ax - b \leq 0, \tag{2.5}$$

$$-Ax + b \leq 0, \tag{2.6}$$

$$-x \leq 0. \tag{2.7}$$

Le point minimum $x^* \geq 0$ est alors caractérisé par les équations et les inégalités suivantes : il existe deux m-vecteurs $\lambda^{*1} \geq 0, \lambda^{*2} \geq 0$, ainsi qu'un n-vecteur $\delta^* \geq 0$ tels que :

$$\begin{cases} \frac{\partial L}{\partial x}(x^*, \lambda^{*1}, \lambda^{*2}, \delta^*) & = 0, \\ \lambda^{*1^T}(Ax^* - b) & = 0, \\ \lambda^{*2^T}(-Ax^* + b) & = 0, \\ \delta^{*^T} x^* & = 0, \\ Ax^* - b & = 0, \end{cases}$$

où

$$\begin{aligned} L(x, \lambda^1, \lambda^2, \delta) &= \frac{1}{2}x^T Dx + c^T x + \lambda^{1^T}(Ax - b) + \lambda^{2^T}(-Ax + b) - \delta^T x, \\ \frac{\partial L}{\partial x}(x, \lambda^1, \lambda^2, \delta) &= Dx + c + A^T\lambda^1 - A^T\lambda^2 - \delta. \end{aligned}$$

En posant $\lambda = \lambda^1 - \lambda^2,\ \lambda \in \mathbb{R}^m$, le point x^* est défini par le système suivant :

$$\begin{cases} Dx^* + A^T\lambda^* - \delta^* & = -c, & (L_1) \\ Ax^* & = b, & (L_2) \\ {\delta^*}^T x^*, & = 0, & (L_3) \\ {\lambda^*}^T(Ax^* - b) & = 0, & (L_4) \\ x^* \geq 0, \lambda \in \mathbb{R}^m, \delta^* \geq 0. & & (L_5) \end{cases}$$

Comme l'équation (L_4) est tout le temps vérifiée, le système se réduit au système :

$$\begin{cases} Dx^* + A^T\lambda^* - \delta^* & = -c, & (L_1) \\ Ax^* & = b, & (L_2) \\ {\delta^*}^T x^*, & = 0, & (L_3) \\ x^* \geq 0, \lambda \in \mathbb{R}^m, \delta^* \geq 0. & & (L_5) \end{cases}$$

Un tel système n'est pas linéaire par rapport au multivecteur $(x^*, \lambda^*, \delta^*)$ à cause de l'équation (L_3). On obtient donc un système linéaire de $(n+m)$ équations à $(2n+m)$ inconnues, constitué des équations (L_1) et (L_2), avec n équations non linéaires $\delta_j x_j = 0, j = 1, 2, \ldots, n$. Donc il suffit d'obtenir une solution basique du système linéaire (L_1)-(L_2) vérifiant les contraintes (L_5), avec x_j basique et δ_j non basique ou vice-versa.
Pour ce faire, il suffit d'appliquer la méthode du simplexe pour trouver une solution réalisable optimale pour le programme linéaire auxiliaire suivant :

$$\begin{gathered} \min z = \tilde{c}^T\tilde{x}, \\ \tilde{A}\tilde{x} + v = \tilde{b}, \\ \tilde{x} = (x, \lambda, \delta, v),\ x \geq 0,\ \lambda \in \mathbb{R}^m,\ \delta \geq 0,\ v \geq 0, \end{gathered}$$

où

$$\tilde{A} = \begin{pmatrix} D & A^T & -I_n \\ A & 0_{\mathbb{R}^{m\times m}} & 0_{\mathbb{R}^{m\times n}} \end{pmatrix},\ \tilde{b} = \begin{pmatrix} -c \\ b \end{pmatrix},\ \tilde{c} = (0_{\mathbb{R}^n}, 0_{\mathbb{R}^m}, 0_{\mathbb{R}^n}, e),\ e = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \in \mathbb{R}^{n+m}.$$

On suppose sans perte de généralité que $\tilde{b} \geq 0$. Alors la solution de base réalisable initiale est $(0_{\mathbb{R}^n}, 0_{\mathbb{R}^m}, 0_{\mathbb{R}^n}, \tilde{b})$.

2.2.2 Algorithme

Le schéma général de l'algorithme de Wolfe se résume de la manière suivante :

Algorithme 2.1 : Algorithme du simplexe quadratique de Wolfe

Entrées : Les matrices A et D, les vecteurs b et c;

Sortie : Une solution optimale x^* du problème (2.1)-(2.3);

Appliquer les conditions de K.K.T au problème;

Déterminer les équations de K.K.T;

Déterminer les paramètres du programme linéaire

- Introduire les variables artificielles v_i ;
- Construire la matrice des contraintes $\tilde{A}$;
- Construire le vecteur du second membre $\tilde{b}$;
- Construire le vecteur des coûts $\tilde{c}$;

Initialisation : poser $k = 0$, Stop $= 0$, soit (x, λ, δ, v) une solution de base réalisable initiale vérifiant $\delta^T x = 0$;

Déterminer l'ensemble des indices J_B et J_N;

tant que *STOP=0* **faire**

  Calculer le vecteur $\pi^T = \tilde{c}_B^T \tilde{A}_B^{-1}$;

  Calculer le vecteur $\Delta_N^T = \pi^T \tilde{A}_N - \tilde{c}_N^T$;

  si $\Delta_N \geq 0$ **alors**

    la solution actuelle est optimale $x^* = x$;

    STOP $= 1$;

  sinon

    Déterminer la variable entrante et celle sortante de la base tout en vérifiant la condition

$$\delta_j x_j = 0, j = 1, 2, \ldots, n;$$

    Mettre à jour $J_B,\ J_N,\ \tilde{A}_B,\ \tilde{A}_N,\ \tilde{c}_B,\ \tilde{c}_N,\ x_B,\ x_N$;

  fin

fin

2.2.3 Exemple numérique

Exemple 2.1. *Soit le problème quadratique convexe suivant :*

$$\begin{array}{ll} \min & f(x) = x_1^2 + x_1x_2 + 6x_2^2 - 2x_1 + 8x_2, \\ s.c. & x_1 + 2x_2 \leq 4, \\ & 2x_1 + x_2 \leq 5, \\ & x_1, x_2 \geq 0. \end{array}$$

En ajoutant deux variables d'écart x_3 et x_4, ce problème peut être transformé en un problème écrit sous forme standard :

$$\begin{array}{ll} \min & f(x) = x_1^2 + x_1x_2 + 6x_2^2 - 2x_1 + 8x_2, \\ s.c. & x_1 + 2x_2 + x_3 = 4, \\ & 2x_1 + x_2 + x_4 = 5, \\ & x_1, x_2, x_3, x_4 \geq 0. \end{array} \tag{2.8}$$

Si $x = (x_1, x_2, x_3, x_4)^T$ est un minimum du problème (2.8), alors $\exists \lambda \in \mathbb{R}^2; \delta_1, \delta_2, \delta_3, \delta_4 \geq 0$ tels que :

$$\begin{aligned} \frac{\partial L}{\partial x_1}(x, \lambda, \delta) &= 0, \\ \frac{\partial L}{\partial x_2}(x, \lambda, \delta) &= 0, \\ \frac{\partial L}{\partial x_3}(x, \lambda, \delta) &= 0, \\ \frac{\partial L}{\partial x_4}(x, \lambda, \delta) &= 0, \\ x_1 + 2x_2 + x_3 &= 4, \\ 2x_1 + x_2 + x_4 &= 5, \\ \delta_j x_j &= 0, \\ x_j \geq 0,\ \delta_j \geq 0\ , j &= 1, 2, 3, 4, \end{aligned}$$

où

$$\begin{aligned} L(x, \lambda, \delta) &= x_1^2 + x_1x_2 + 6x_2^2 - 2x_1 + 8x_2 + \lambda_1(x_1 + 2x_2 + x_3 - 4) \\ &\quad + \lambda_2(2x_1 + x_2 + x_4 - 5) - \delta_1x_1 - \delta_2x_2 - \delta_3x_3 - \delta_4x_4. \end{aligned}$$

Alors on obtient le système suivant :

$$\begin{aligned}
\frac{\partial L}{\partial x_1}(x,\lambda,\delta) &= 2x_1 + x_2 - 2 + \lambda_1 + 2\lambda_2 - \delta_1 = 0,\\
\frac{\partial L}{\partial x_2}(x,\lambda,\delta) &= x_1 + 12x_2 + 8 + 2\lambda_1 + \lambda_2 - \delta_2 = 0,\\
\frac{\partial L}{\partial x_3}(x,\lambda,\delta) &= \lambda_1 - \delta_3 = 0,\\
\frac{\partial L}{\partial x_4}(x,\lambda,\delta) &= \lambda_2 - \delta_4 = 0,\\
x_1 + 2x_2 + x_3 &= 4,\\
2x_1 + x_2 + x_4 &= 5,\\
\delta_j x_j &= 0,\\
x_j \geq 0, \delta_j \geq 0, j &= 1,2,3,4.
\end{aligned}$$

Ce dernier système est équivalent au système suivant :

$$\begin{cases}
2x_1 + x_2 + \delta_3 + 2\delta_4 - \delta_1 = 2, & (L_1)\\
-x_1 - 12x_2 + \delta_2 - 2\delta_3 - \delta_4 = 8, & (L_2)\\
x_1 + 2x_2 + x_3 = 4, & (L_3)\\
2x_1 + x_2 + x_4 = 5, & (L_4)\\
\delta_j x_j = 0, & (L_5)\\
x_j \geq 0, \delta_j \geq 0, j = 1,2,3,4.
\end{cases}$$

Nous avons obtenu un système de 4 équations linéaires avec 8 inconnues et 4 équations non linéaires. Pour trouver une solution telle que $\delta_j x_j = 0$ avec $j = 1,2,3,4$, il suffit d'obtenir une solution basique (x,δ) du système linéaire, vérifiant $x \geq 0,\ \delta \geq 0$, avec x_j basique et δ_j hors base ou vice versa. Pour ce faire, il faut choisir l'indice j_0 entrant dans la base de telle sorte que les vecteurs-colonnes correspondants à x_{j_0} et δ_{j_0} ne se trouvent pas en même temps dans la base.

Appliquons la 1$^{\text{er}}$ phase du simplexe en considérant le problème de programmation linéaire auxiliaire suivant :

$$\begin{aligned}
\max \quad & z = -v_1,\\
s.c. \quad & 2x_1 + x_2 - \delta_1 + \delta_3 + 2\delta_4 + v_1 = 2,\\
& -x_1 - 12x_2 + \delta_2 - 2\delta_3 - \delta_4 = 8,\\
& x_1 + 2x_2 + x_3 = 4,\\
& 2x_1 + x_2 + x_4 = 5,\\
& x_j, \delta_j, v_1 \geq 0, j = 1,2,3,4.
\end{aligned} \tag{2.9}$$

Le vecteur $(x^T, \delta^T, v_1^T) = (0, 0, 4, 5, 0, 8, 0, 0, 2)^T$ est une solution réalisable initiale basique du problème (2.9). Dressons alors les tableaux du simplexe :

$1^{\text{ère}}$ itération :

		x	x_1	x_2	x_3	x_4	δ_1	δ_2	δ_3	δ_4	v_1		
		c	0	0	0	0	0	0	0	0	-1		
c_B	base	b	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	θ	
-1	a_9	2	2	1	0	0	-1	0	1	2	1	1	$\rightarrow j_1 = 9$
0	a_6	8	-1	-12	0	0	0	1	-2	-1	0	∞	
0	a_3	4	1	2	2	0	0	0	0	0	0	4	
0	a_4	5	2	1	0	1	0	0	0	0	0	5/2	
$z = -2$		Δ	-2	-1	0	0	1	0	-1	-2	0		
			$\uparrow j_0 = 1$										

Remarquons que $j_0 = 1$ est l'indice qui entre dans la base et $j_1 = 9$ celui qui sort, et que l'idice $j = 8$ ne peut pas être choisi car nous avons a_4 dans la base ce qui signifie que a_8 doit être hors base.

$2^{\text{ème}}$ itération :

		x	x_1	x_2	x_3	x_4	δ_1	δ_2	δ_3	δ_4	v_1	
		c	0	0	0	0	0	0	0	0	-1	
c_B	base	b	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	θ
0	a_1	1	12	1/2	0	0	-1/2	0	1/2	1	1/2	
0	a_6	10	0	-13	0	0	-1	1	-1	1	1	
0	a_3	3	0	3/2	1	0	1/2	0	-1/2	-1	-1/2	
0	a_4	3	0	0	0	1	1	0	-1	-2	-1	
$z = 0$		Δ	0	0	0	0	1	0	0	0	1	

On a $\Delta \geq 0$, donc la solution basique positive ou nulle du système linéaire est :

$$x^* = (1, 0, 3, 3)^T, \delta^* = (0, 10, 0, 0)^T.$$

Donc le minimum de f est le vecteur $x^* = (1, 0, 3, 3)^T$, avec $f(x^*) = -1$.

2.3 La méthode d'activation des contraintes

La méthode d'activation des contraintes (ASM : Active Set Method) est une méthode calssique dévloppée au début des années soixante-dix pour la résolution des problèmes

linéaires et quadratiques ; elle s'applique pour des problèmes d'optimisation avec contraintes linéaires de types inégalités ou mixtes.

Il existe trois type de méthodes d'activation de contraintes : méthode primale, méthode duale et méthode primale-duale. Dans cette partie, on ne présentera que la méthode primale, cette dernière est itérative et s'exécute en deux phase :

— La première phase consiste à trouver une solution réalisable de départ, qui est obtenue en exécutant la première phase d'un solveur de programmation linéaire.

— La deuxième phase consiste à appliquer la méthode ASM proprement dite. Son principe général est de résoudre à chaque itération un programme quadratqique avec contraintes d'égalité, correspondants aux indices actifs. Par la suite, elle ajuste cet ensemble pour identifier les contarintes active à l'optimum.

2.3.1 Position du problème

Sans perte de généralité, on peut présenter un programme quadratique convexe sous la forme suivante :

$$\min f(x) = \frac{1}{2}x^T Dx + c^T x, \tag{2.10}$$

$$A_i^T x = b_i,\ i \in \mathcal{E}, \tag{2.11}$$

$$A_i^T x \leq b_i,\ i \in \mathcal{I}, \tag{2.12}$$

où D est une matrice réelle définie positive et symétrique d'ordre n ; $I = \mathcal{E} \cup \mathcal{I} = \{1, 2, \ldots, m\}$ est un ensemble fini d'indices des contraintes du problème ; $A_i,\ i \in I, c, x$ sont des $n-$vecteurs et b est un $m-$vecteur.

2.3.2 Critère d'optimalité

En associant aux contraintes du problème (2.10)-(2.12) le m-vecteur multiplicateur λ, la fonction de Lagrange s'écrit :

$$L(x, \lambda) = \frac{1}{2}x^T Dx + c^T x + \sum_{i=1}^{m} \lambda_i (A_i^T x - b_i). \tag{2.13}$$

La condition nécessaire d'optimalité de KKT du premier ordre est la suivante :

$$\begin{aligned} Dx^0 + c + \sum_{i=1}^{m} \lambda_i^0 (A_i^T x^0 - b_i) = 0, & \\ \lambda_i^0 \geq 0, \qquad & \forall i \in \mathcal{I}, \\ \lambda_i^0 (A_i^T x^0 - b_i) = 0, \qquad & \forall i \in \mathcal{I}. \end{aligned}$$

2.3.3 Une itération de la méthode

A une itération k de l'algorithme, notons par x^k la solution courante et W_k l'ensemble des indices des contraintes actives au point x^k. Une itération k de l'algorithme consiste à exécuter les opérations suivantes :

Étape 1. On résout le sous-problème quadratique suivant :

$$\min f(d) = \frac{1}{2} d^T D d + [\nabla f(x^k)]^T d, \tag{2.14}$$

$$A_i^T d = 0, \ i \in W_k. \tag{2.15}$$

Deux cas peuvent se présenter :

— Si $d^k = 0$, alors poser $x^{k+1} = x^k$; aller à l'étape (2).

— Sinon, posons :

$$x^{k+1} = x^k + \theta d^k, \tag{2.16}$$

Le pas θ doit être choisi de telle sorte que la nouvelle solution $\bar{x}$ soit réalisable. Donc toutes les contraintes non saturées doivent vérifier :

$$A_i^T (x^k + \theta d^k) \leq b_i, \ \forall i \in I/W_k.$$

Alors la valeur optimale du pas θ est donnée par :

$$\theta^k = \min\{1, \theta_{i_0}\}, \tag{2.17}$$

où

$$\theta_{i_0} = \min\{\theta_i, \ i \in I/W_k\}, \text{ avec } \theta_i = \begin{cases} \frac{b_i - A_i^T x^k}{A_i^T d^k}, & \text{si } A_i^T d^k > 0, \\ +\infty, & \text{si } A_i^T d^k \leq 0, \end{cases} \quad i \in I \backslash W_k.$$

Si $\theta^k = 1$, alors dans ce cas l'ensemble W_k ne change pas, et on aura :

$$x^{k+1} = x^k + d^k.$$

Sinon, la contrainte i_0 est ajoutée à l'ensemble W_k : $W_k := W_k \cup \{i_0\}$.

Étape 2. Cette étape consiste à vérifier si la nouvelle solution obtenue x^{k+1} est optimale ou non pour le problème (2.10)-(2.12). Ceci se fait en calculant les multiplicateurs de Lagrange λ_i correspondants aux contrainte $i \in W_k$. Rappelons que pour $i \in I \backslash W_k,\ \lambda_i = 0$.

— Si $\lambda_i \geq 0,\ \forall i \in W_k$, alors la condition d'optimalité de KKT est remplie pour le problème (2.10)-(2.12). Donc la solution optimale est trouvée, et on arrête l'algorithme.

— Sinon, il existe une composante du vecteur λ vérifiant $\lambda_{i_1} < 0$, avec $i_1 \in W_k$, alors on peut améliorer la valeur de la fonction objectif en supprimant la contrainte i_1 de l'ensemble W_k. Si plusieurs contraintes ne vérifient pas le critère d'optimalité, alors la contrainte à supprimer doit vérifier la relation suivante :

$$\lambda_{i_1} = \min\{\lambda_i \ :\ \lambda_i < 0,\ i \in W_k\},$$

donc $W_{k+1} := W_k \backslash \{i_1\}$.

On répète le processus de résolution jusqu'à ce que l'optimum soit trouvé.

2.3.4 Algorithme

L'algorithme d'activation des contraintes est donc donné comme suit :

Algorithme 2.2 : ASM

Entrées : Les matrices A et D, les vecteurs b et c, les ensembles d'indices des contraintes $\mathcal{E}$ et $\mathcal{I}$;

Sortie : Une solution optimale x^* du problème (2.10)-(2.12);

1 Initialisation Poser $k = 0$, Stop=0, trouver une solution réalisable de départ x^0;
2 **tant que** $Stop = 0$ **faire**
3 Calculer d^k en résolvant le programme quadratique (2.14)-(2.15);
4 **si** $d^k = 0$ **alors**
5 On pose $x^{k+1} = x^k$;
6 Aller à la ligne 11;
sinon
7 Calculer le pas optimal θ^k avec la relation (2.17);
8 Calculer $x^{k+1} = x^k + \theta^k d^k$;
fin
9 **si** $\theta^k \neq 1$ **alors**
10 On pose : $W_k := W_k \cup \{i_0\}$;
fin
11 Calculer les multiplicateurs de Lagrange $\lambda_i,\ i \in W_k$;
12 **si** $\lambda_i \geq 0,\ \forall i \in W_k$ **alors**
13 $Stop = 1$;
14 $x^* = x^{k+1}$;
sinon
15 Déterminer $\lambda_{i_1} = \min\{\lambda_i : \lambda_i < 0,\ i \in W_k\}$;
16 Poser $W_k := W_k \backslash \{i_1\}$ et $k := k + 1$;
fin
fin

2.3.5 Exemple numérique

Exemple 2.2. *Considérons le programme quadratique convexe suivant :*

$$\begin{array}{ll} \min & f(x) = 2x_1^2 + x_1x_2 + x_2^2 - 12x_1 - 10x_2, \\ s.c. & x_1 + x_2 \leq 4, \\ & x_1, x_2 \geq 0. \end{array} \tag{2.18}$$

On a

$$D = \begin{pmatrix} 4 & 1 \\ 1 & 2 \end{pmatrix}, \quad c = \begin{pmatrix} -12 \\ -10 \end{pmatrix}, \quad A = \begin{pmatrix} 1 & 1 \\ -1 & 0 \\ 0 & -1 \end{pmatrix}, \quad b = \begin{pmatrix} 4 \\ 0 \\ 0 \end{pmatrix}.$$

Itération 1 :

Soit $x^0 = (0,0)^T$ *la solution réalisable de départ pour ce problème. L'ensemble des indices actifs en ce point est* $W_0 = \{2,3\}$. *Alors pour le problème (2.14)-(2.15), on a*

$$A^0 = (A_i,\ i \in W_0) = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}.$$

Il est clair que $d^0 = (0,0)^T$ *est optimal pour le problème (2.14)-(2.15), puisque aucun autre point n'est réalisable pour ce problème. Donc* $x^1 = x^0$.

Le vecteur des multiplicateurs de Lagrange associé est :

$$\lambda^1 = -A^{0^{-1}} \nabla f(x^1) = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} -12 \\ -10 \end{pmatrix} = \begin{pmatrix} -12 \\ -10 \end{pmatrix}.$$

Aucun indice actif n'est optimal et par conséquent la deuxième contrainte sera supprimée de l'ensemble W_0. *Donc, on aura :*

$$W_0 := W_0 / \{2\} = \{3\}.$$

Itération 2 :

La solution du problème (2.14)-(2.15), avec $W_1 = \{3\},\ A^1 = (A_i,\ i \in W_1) = (0,-1)$, *est* $d^1 = (3,0)^T$. *Le pas optimal est* $\theta^1 = 1$, *donc le nouveau point réalisable est*

$$x^2 = x^1 + d^1 = (3,0)^T.$$

Le multiplicateur de Lagrange associé à la contrainte saturée est : $\lambda_3 = -10 < 0$. *Par conséquent, la troisième contrainte est supprimée de l'ensemble des contraintes actives :*

$$W_2 = W_1 / \{3\} = \emptyset.$$

Itération 3 :

La solution du problème (2.14)-(2.15), avec $W_2 = \emptyset$, *est la solution du problème sans contraintes* $\min_{d \in \mathbb{R}^n} f(d) = \frac{1}{2} d^T D d + [\nabla f(x^2)]^T d$ *est*

$$d^2 = -D^{-1} \nabla f(x^2) = -\begin{pmatrix} 2/7 & -1/7 \\ -1/7 & 4/7 \end{pmatrix} \begin{pmatrix} 0 \\ -7 \end{pmatrix} = \begin{pmatrix} -1 \\ 4 \end{pmatrix}.$$

Calcul de θ^2 :

$$A_1^T d^2 = \begin{pmatrix} 1 & 1 \end{pmatrix} \begin{pmatrix} -1 \\ 4 \end{pmatrix} = 3 > 0 \Rightarrow \theta_1 = \frac{b_1 - A_1^T x^2}{A_1^T d^2} = \frac{1}{3};$$

$$A_2^T d^2 = \begin{pmatrix} -1 & 0 \end{pmatrix} \begin{pmatrix} -1 \\ 4 \end{pmatrix} = 1 > 0 \Rightarrow \theta_2 = \frac{b_2 - A_2^T x^2}{A_2^T d^2} = 3;$$

$$A_3^T d^2 = \begin{pmatrix} 0 & -1 \end{pmatrix} \begin{pmatrix} -1 \\ 4 \end{pmatrix} = -4 \leq 0 \Rightarrow \theta_3 = \infty;$$

$$\theta_{i_0} = \min\{\theta_1, \theta_2, \theta_3\} = \theta_1 = \frac{1}{3} \Rightarrow i_0 = 1.$$

Alors

$$\theta^2 = \min\{1, \theta_{i_0}\} = \theta_{i_0} = \frac{1}{3}.$$

La nouvelle solution obtenue, ainsi que l'ensemble W_3 correspondant sont :

$$x^3 = x^2 + \theta^2 d^2 = \begin{pmatrix} 3 \\ 0 \end{pmatrix} + \frac{1}{3} \begin{pmatrix} -1 \\ 4 \end{pmatrix} = \begin{pmatrix} 8/3 \\ 4/3 \end{pmatrix}; \quad W_3 = W_2 \cup \{1\} = \{1\}.$$

Itération 4 :

Résoulution du problème (2.14)-(2.15) avec les paramètres suivants :

$$W_3 = \{1\}, \ A^3 = \begin{pmatrix} 1 & 1 \end{pmatrix}, \ \nabla f(x^3) = \begin{pmatrix} 0 \\ -14/3 \end{pmatrix}.$$

La solution optimale du problème (2.14)-(2.15) nous donne la direction suivante :

$$d^3 = \begin{pmatrix} -7/6 \\ 7/6 \end{pmatrix}.$$

Calcul de θ^3 :

$$A_2^T d^3 = \begin{pmatrix} -1 & 0 \end{pmatrix} \begin{pmatrix} -7/6 \\ 7/6 \end{pmatrix} = \frac{7}{6} > 0 \Rightarrow \theta_2 = \frac{16}{7};$$

$$A_3^T d^3 = \begin{pmatrix} 0 & -1 \end{pmatrix} \begin{pmatrix} -7/6 \\ 7/6 \end{pmatrix} = -\frac{7}{6} \leq 0 \Rightarrow \theta_3 = \infty;$$

$$\theta_{i_0} = \min\{\theta_2, \theta_3\} = \theta_2 = \frac{16}{7} \Rightarrow \theta^0 = \min\{1, \theta_{i_0}\} = 1.$$

Alors on aura :

$$x^4 = x^3 + d^3 = \begin{pmatrix} 8/3 \\ 1 \end{pmatrix} + \begin{pmatrix} -7/6 \\ 7/6 \end{pmatrix} = \begin{pmatrix} 3/2 \\ 5/2 \end{pmatrix}.$$

La valeur du multiplicateur de Lagrange associé à la première contrainte est : $\lambda_1 = \frac{7}{2} \geq 0$. *Donc* x^4 *est la solution optimale pour le problème (2.18).*

2.4 La méthode des points intérieurs primale-duale

En 1984, Karmarkar a mis au point une Méthode de Points Intérieurs "MPI" qui peut résoudre des programmes linéaires en un temps polynomial [61]. Plusieurs chercheurs par la suite ont développé des MPIs pour la résolution des problèmes de programmation quadratique convexe [97].

Les MPIs ont des caractéristiques communes qui se distinguent de celles de la méthode du simplexe. Chaque itération est coûteuse en temps de calcul mais peut faire un progrès significatif vers la solution. Par contre, la méthode du simplexe demande généralemnet un plus grand nombre d'itérations non coûteuses.

La méthode du simplexe passe d'un point extrême à un autre point extrême meilleur jusqu'à ce qu'elle trouve un point optimal, tandis que les MPIs s'approchent de la solution de l'intérieur du domaine réalisable et ne s'approchent de la frontière qu'en limite.

2.4.1 Position du problème

En se basant sur [98], dans cette section, nous présentons la méthode des points intérieurs, dite primale-duale, pour la résolution du problème quadratique convexe suivant :

$$\min f(x) = \frac{1}{2} x^T D x + c^T x, \tag{2.19}$$

$$Ax \geq b, \tag{2.20}$$

$$x \geq 0, \tag{2.21}$$

où D est une matrice réelle semi-définie positive et symétrique d'ordre $n; c, x$ sont des $n-$vecteurs, A est une matrice réelle de dimension $m \times n$ ($rangA = m$), et b est un

$m-$vecteur.

2.4.2 Une itération de la méthode

La méthode des points intérieurs primale-duale consiste à résoudre le problème de programmation non linéaire convexe, dit problème-barrière, suivant :

$$\min \frac{1}{2}x^T Dx + c^T x - \mu \sum_{i=1}^{n} \log(x_i) - \mu \sum_{i=1}^{m} log(w_i), \tag{2.22}$$

$$Ax - w = b, \tag{2.23}$$

où μ est un paramètre de barrière positif. Ici w est un vecteur de variables d'écart utilisé pour transformer les contraintes d'inégalité en contraintes d'égalité.

Le Lagrangien associé à ce problème est donné par :

$$L(x, w, \lambda) = \frac{1}{2}x^T Dx + c^T x - \mu \sum_{i=1}^{n} \log(x_i) - \mu \sum_{i=1}^{m} log(w) + \lambda^T (b - Ax + w).$$

Les conditions d'optimalité du premier ordre pour le problème (2.22)-(2.23) sont

$$\begin{aligned} c + Dx - \mu X^{-1} e - A^T \lambda &= 0, \\ -\mu W^{-1} e + \lambda &= 0, \\ b - Ax + w &= 0, \end{aligned}$$

où

$$e = (1, 1, \ldots, 1)^T \in \mathbb{R}^n,\ X = \begin{pmatrix} x_1 & 0 & \ldots & 0 \\ 0 & x_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & x_n \end{pmatrix},\ W = \begin{pmatrix} w_1 & 0 & \ldots & 0 \\ 0 & w_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & w_m \end{pmatrix}.$$

Ce problème s'écrit aussi comme suit :

$$A^T \lambda + z - Dx = c, \tag{2.24}$$

$$Ax - w = b, \tag{2.25}$$

$$XZe = \mu e, \tag{2.26}$$

$$\Lambda W e = \mu e, \tag{2.27}$$

où

$$z = \mu X^{-1} e,\ \Lambda = \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_m \end{pmatrix}.$$

A une itération donnée, soit $(x, w, \lambda, z) > 0$ le multi-vecteur correspondant au point intérieur courant. Pour calculer les directions de déplacement d_x, d_w, d_λ et d_z, nous remplaçons (x, w, λ, z) par $(x+d_x, w+d_w, \lambda+d_\lambda, z+d_z)$ dans le système (2.24). Nous obtenons alors le système non linéaire suivant :

$$\begin{aligned} A^T d_\lambda + d_z - D d_x &= c - A^T\lambda - z + Dx, \\ A d_x - d_w &= b - Ax + w, \\ Z d_x + X d_z + d_x d_Z e &= \mu e - XZe, \\ W d_\lambda + \Lambda d_w + d_\lambda d_W e &= \mu e - YWe. \end{aligned}$$

Ensuite, nous supprimons les termes non linéaires pour obtenir le système linéaire suivant :

$$\begin{aligned} A^T d_\lambda + d_z - D d_x &= c - A^T\lambda - z + Dx, \\ A d_x - d_w &= b - Ax + w, \\ Z d_x + X d_z &= \mu e - XZe, \\ W d_\lambda + \Lambda d_w &= \mu e - \Lambda We. \end{aligned}$$

Des deux dernières équations, on obtient

$$\begin{aligned} d_z &= X^{-1}(\mu e - XZe - Z d_x), \\ d_w &= \Lambda^{-1}(\mu e - \Lambda We - W d_\lambda). \end{aligned}$$

Nous utilisons ensuite ces expressions pour éliminer d_z et d_w des deux équations restantes dans le système. Après élimination, nous arrivons au système KKT réduit suivant :

$$\begin{aligned} A^T d_\lambda + (X^{-1}Z + D) d_x &= c - A^T\lambda + Dx - \mu X^{-1} e, \\ A d_x + \Lambda^{-1} W d_\lambda &= b - Ax + \mu\Lambda^{-1} e. \end{aligned}$$

Ce système s'écrit sous la forme matricielle suivante :

$$\begin{pmatrix} -(X^{-1}Z + D) & A^T \\ A & \Lambda^{-1}W \end{pmatrix} \begin{pmatrix} d_x \\ d_\lambda \end{pmatrix} = \begin{pmatrix} c - A^T\lambda + Dx - \mu X^{-1} e \\ b - Ax + \mu\Lambda^{-1} e \end{pmatrix}.$$

2.4.3 Algorithme

L'algorithme correspondant est décrit comme suit [98] :

Algorithme 2.3 : MPI

Entrées : Les matrices A et D, les vecteurs b et c;

Sortie : Une solution optimale x^* du problème (2.19)-(2.21);

1 Initialisation : Soit $(x, w, \lambda, z) > 0,\ \delta$ un paramètre entre 0 et 1 et $0 \leq r < 1$ un paramètre très proche de 1;

2 **tant que** *non optimal* **faire**

3 Calculer $\rho = b - Ax + w$;

4 Calculer $\sigma = c - A^T\lambda - z + Dx$;

5 Calculer $\gamma = z^T x + \lambda^T w$;

6 Calculer $\mu = \delta\frac{\gamma}{n+m}$;

7 Résoudre :

$$\begin{pmatrix} -(X^{-1}Z + D) & A^T \\ A & Y^{-1}W \end{pmatrix} \begin{pmatrix} d_x \\ d_\lambda \end{pmatrix} = \begin{pmatrix} c - A^T\lambda + Dx - \mu X^{-1}e \\ b - Ax + \mu\Lambda^{-1}e \end{pmatrix}.$$

8 Calculer $d_z = X^{-1}(\mu e - XZe - Zd_x)$;

9 Calculer $d_w = \lambda^{-1}(\mu e - \Lambda We - Wd_\lambda)$;

10 Calculer le nombre θ_1 comme suit :

$$\theta_1 = r\left(\max_{i,j}\left\{-\frac{d_{x_j}}{x_j}, -\frac{d_{w_i}}{w_i}, -\frac{d_{\lambda_i}}{\lambda_i}, -\frac{d_{z_j}}{z_j}\right\}\right)^{-1};$$

11 Calculer $\theta = \min\{\theta_1, 1\}$;

12 Poser $x = x + \theta d_x,\ w = w + \theta d_w,\ \lambda = \lambda + \theta d_\lambda$ et $z = z + \theta d_z$;

fin

Remarque 2.1. *La notation* $\max_{i,j}$ *est utilisée pour désigner le maximum de tous les ratios de l'ensemble* $\left\{-\frac{d_{x_j}}{x_j}, -\frac{d_{w_i}}{w_i}, -\frac{d_{\lambda_i}}{\lambda_i}, -\frac{d_{z_j}}{z_j}\right\}$.

Critère d'arrêt.

Soient $\epsilon > 0$ une tolérance et $M < \infty$ une tolérance finie très grande. Si $||x||_\infty > M$, alors on arrête l'algorithme, le problème primal est non borné. Si $||y||_\infty > M$, alors on arrête l'algorithme, le problème dual est non borné. Si $||\rho||_1 < \epsilon$, $||\sigma||_1 < \epsilon$ et $\gamma < \epsilon$,

alors on arrête l'algorithme, la solution courante est ϵ-optimale. Pour plus de détails sur la justification du critère d'arrêt et le choix empirique des différents paramètres, voir [98].

Chapitre 3

Méthodes de résolution en programmation quadratique concave

3.1 Introduction

Nous présentons dans ce chapitre les différentes approches développées pour la recherche de solutions locales et globales des problèmes de programmation quadratique concave sous contraintes linéaires, en donnant brièvement leurs principes et leurs algorithmes de base.

Il existe plusieurs méthodes pour la résolution locale et globale de ce type de problème, mais nous nous contentons uniquement de la description des méthodes les plus connues, à savoir deux algorithmes pour la recherche de solutions locales (l'algorithme du gradient conditionnel [43] et l'algorithme DCA : Difference of Convex Functions Algorithm [85]) ainsi que deux autres algorithmes pour la recherche de solutions globales (l'algorithme de Branch and Bound [12] et l'algorithme des ensembles d'approximation [31]). Enfin, nous terminons ce chapitre par la présentation de quelques applications pratiques des problèmes de programmation quadratique concave.

3.2 Recherche de solutions locales

3.2.1 Méthode du gradient conditionnel

Consédérons le problème quadratique concave suivant :

$$\min f(x) = \frac{1}{2}x^T Dx + c^T x, \tag{3.1}$$

$$A_1 x \leq b_1, \tag{3.2}$$

$$A_2 x = b_2, \tag{3.3}$$

$$l \leq x \leq u, \tag{3.4}$$

où D est une matrice réelle semi-définie négative et symétrique d'ordre n ; c, x, l, u sont des vecteurs de $\mathbb{R}^n$, les composantes de l et u peuvent prendre des valeurs finies ou infinies ; A_1 est une matrice réelle de dimension $m_1 \times n$, A_2 est une matrice réelle de dimension $m_2 \times n$, $b_1 \in \mathbb{R}^{m_1}$ et $b_2 \in \mathbb{R}^{m_2}$.
La méthode du gradient conditionnel est développée par Frank et Wolfe en 1956 pour la résolution des programmes quadratiques convexes avec contraintes linéaires [43]. Cette méthode est basée sur les conditions d'optimalité locales du théorème 1.4.

Cet algorithme consiste à commencer d'un point réalisable initial, puis il passe d'un point courant x^k à un nouveau point réalisable x^{k+1}, où $x^{k+1} = x^k + \lambda^*(y^k - x^k)$, avec y^k une solution du programme linéaire $\min_{x \in S}[\nabla f(x^k)]^T x$ et le pas λ^* est calculé de telle sorte à minimiser la fonction $\psi(\lambda) = f(x^k + \lambda(y^k - x^k))$ par rapport à λ sur l'intervalle $[0, 1]$. L'algorithme s'arrête lorsque $\nabla f(x^k)(y^k - x^k) \geq 0$. Cette méthode est particulièrement utile pour la recherche d'un point stationnaire pour les problèmes d'optimisation d'une fonction non linéaire de classe $\mathcal{C}^1$ sur un polytope.

Pour notre programme quadratique concave (3.1)-(3.4), la méthode du gradient conditionnel passe d'un point extrême de l'ensemble réalisable à un nouveau point extrême ayant une meilleure valeur de la fonction objectif. Notons que la valeur du pas λ^* qui minimise la fonction quadratique concave d'une variable réelle $f(x^k + \lambda(y^k - x^k))$, avec $0 \leq \lambda \leq 1$ et y^k est une solution du programme linéaire $\min_{x \in S} x^T \nabla f(x^k)$, est égale à un. En effet, soit $d = y^k - x^k$, alors minimiser $f(x^k + \lambda d)$ équivaut à minimiser l'accroissement :

$$f(x^k + \lambda d) - f(x^k) = \lambda d^T \nabla f(x^k) + \frac{\lambda^2}{2} d^T D d.$$

Définissons la fonction réelle d'une variable réelle $\psi(\lambda) = \lambda d^T \nabla f(x^k) + \frac{\lambda^2}{2} d^T D d$. Puisque $d^T D d \leq 0$ (D est semi-défini négative), la fonction ψ est concave, donc le minimum est atteint en l'une des bornes de l'intervalle $[0, 1]$. Pour un point x^k non stationnaire, on a $d^T \nabla f(x^k) < 0$. Donc

$$\psi(1) = d^T \nabla f(x^k) + \frac{1}{2} d^T D d < 0 = \psi(0).$$

Par conséquent, $\lambda^* = 1$ et le nouveau point $x^{k+1} = x^k + \lambda^*(y^k - x^k) = y^k$, avec $f(x^{k+1}) < f(x^k)$.

L'algorithme du gradient conditionnel pour trouver un minimum local du programme quadratique concave (3.1)-(3.4) est donc décrit comme suit [43, 93] :

Algorithme 3.1 : Gradient conditionnel

Entrées : Les matrices A_1, A_2 et D, les vecteurs b_1, b_2, l, u et c;

Sortie : Une solution optimale locale x^* du problème (3.1)-(3.4);

Initialisation : Poser $k = 0$, $Stop = 0$. Soit x^0 une solution réalisable de départ;

tant que *Stop=0* **faire**

  Calculer $y^k \in \arg\min_{x \in S} x^T \nabla f(x^k)$;

  Poser $d^k = y^k - x^k$;

  si $\nabla f(x^k)^T d^k < 0$ **alors**

    Poser $x^{k+1} = y^k$ et $k = k + 1$;

  sinon

    La solution $x^* = x^k$ est un minimum local;

    $Stop = 1$;

  fin

fin

Remarque 3.1. *Pour une preuve détaillée de la convergence globale de la méthode du gradient conditionnel vers un point stationnaire du problème de minimisation d'une fonction de classe $\mathcal{C}^1$ sur un ensemble compact et convexe non vide, voir [21].*

Exemple numérique

Exemple 3.1. *Considérons le programme quadratique concave suivant :*

$$\begin{array}{ll} \min & f(x) = -x_1^2 - \frac{1}{2}x_1x_2 - \frac{3}{2}x_2^2 - x_1 + 2x_2, \\ s.c. & -6x_1 + 2x_2 \leq 6, \\ & -x_1 - x_2 \leq 2, \\ & 2x_1 - x_2 \leq 2, \\ & 3x_1 + x_2 \leq 3. \end{array}$$

On a $D = \begin{pmatrix} -2 & -1/2 \\ -1/2 & -3 \end{pmatrix}$; $c = \begin{pmatrix} -1 \\ 2 \end{pmatrix}$; $A = \begin{pmatrix} -6 & 2 \\ -1 & -1 \\ 2 & -1 \\ 3 & 1 \end{pmatrix}$; $b = \begin{pmatrix} 6 \\ 2 \\ 2 \\ 3 \end{pmatrix}$.

Posons $k = 0$ *et soit* $x^0 = (-1, -1)^T$ *la solution réalisable de départ pour ce problème, avec* $f(x^0) = -4$.

Itération 1 :

Nous avons $\nabla f(x^0) = Dx^0 + c = (3/2,\ 11/2)^T$.

La solution optimale du PL $\min_{x \in S} x^T \nabla f(x^0)$ *est* $y^0 = (0, -2)^T$, *avec* $f(y^0) = -10$.

On a $d^0 = y^0 - x^0 = (1, -1)^T$ *et* $[\nabla f(x^0)]^T d = -4 < 0$. *Donc* $x^1 = y^0 = (0, -2)^T,\ k = 1$.

Itération 2 :

Nous avons $\nabla f(x^1) = Dx^1 + c = (0,\ 8)^T$.

La solution optimale du PL $\min_{x \in S} x^T \nabla f(x^1)$ *est* $y^1 = (0, -2)^T$, *avec* $f(y^1) = -10$.

On a $d^1 = y^1 - x^1 = (0, 0)^T$ *et* $[\nabla f(x^1)]^T d^1 = 0$. *Donc le minimum local de* f *sur* S *est le vecteur* $x^* = x^1 = (0, -2)^T$, *avec* $f(x^*) = -10$.

3.2.2 Algorithme DCA

L'algorithme de la différence de deux fonctions convexes (DCA) est une méthode itérative d'optimisation locale basée sur l'optimalité locale et la dualité en programmation DC. Cet algorithme a été introduit par Pham Dinh Tao [83, 82] en 1984 et puis intensivement développé par plusieurs chercheurs par la suite [67, 66].

Pour minimiser une fonction $f(x) = g(x) - h(x)$, avec g et h deux fonctions convexes définies de $\mathbb{R}^n$ dans $\mathbb{R}$ sur un domaine convexe $S \subset \mathbb{R}^n$, cet algorithme résous un problème

qui minimise sur $\mathbb{R}^n$ la fonction $\phi(x) - h(x)$, où $\phi(x) = g(x) + \chi_S(x)$, avec

$$\chi_S(x) = \begin{cases} 0, & \text{si } x \in S; \\ +\infty, & \text{Sinon.} \end{cases}$$

L'algorithme commence alors par un point de départ $x^0 \in \mathbb{R}^n$, puis il construit deux suites $\{x^k\}$ et $\{y^k\}$ de la manière suivante :

$$y^k \in \partial h(x^k)$$

et x^{k+1} est une solution du programme convexe suivant :

$$\inf\{\phi(x) - h(x^k) - {y^k}^T(x - x^k),\ x \in \mathbb{R}^n\}.$$

L'algorithme s'arrête lorsqu'un certain critère d'arrêt est vérifié, par exemple lorsque $\left\|x^{k+1} - x^k\right\| \leq \epsilon$ ou $|f(x^{k+1}) - f(x^k)| \leq \epsilon$, où $\epsilon > 0$ est une précision fixée à l'avance. Considérons le problème quadratique suivant :

$$\min f(x) = \frac{1}{2}x^T D x + c^T x, \tag{3.5}$$

$$Ax \leq b, \tag{3.6}$$

où D est une matrice réelle symétrique d'ordre n ; A est une matrice réelle de dimension $m \times n$, $c, x \in \mathbb{R}^n$ et $b \in \mathbb{R}^m$.

Ce problème peut s'écrire comme suit :

$$\inf\{f(x),\ x \in S\},\ \text{où } S = \{x \in \mathbb{R}^n\ :\ Ax \leq b\}.$$

Construisons le problème P_{DC} [85] :

$$\inf\{\phi(x) - h(x),\ x \in \mathbb{R}^n\},$$

où

$$\begin{aligned} \phi(x) &= \frac{1}{2}\rho \left\|x\right\|^2 + c^T x + \chi_S(x), \\ h(x) &= \frac{1}{2}x^T(\rho I_n - D)x, \end{aligned}$$

avec ρ un nombre réel positif ou nul pour lequel la matrice $(\rho I_n - D)$ est définie positive ou bien semi-définie positive. Pour vérifier cette propriété, ρ doit être choisit supérieur ou égal à la plus grande valeur propre de D [85].

L'algorithme DCA appliqué au problème (3.5)-(3.6) consiste à construire deux suites $\{x^k\}$ et $\{y^k\}$ de la manière suivante :

$$y^k \in \partial h(x^k) \Rightarrow y^k = \nabla h(x^k) = (\rho I_n - D)x^k$$

et x^{k+1} est une solution optimale du programme quadratique convexe suivant :

$$\min_{x\in\mathbb{R}^n} \left\{ \phi(x) - h(x^k) - {y^k}^T(x - x^k) \right\}.$$

Comme $-h(x^k) + {y^k}^T x^k$ est une constante qui ne dépend pas de x, alors le problème précédent est équivalent au problème suivant :

$$\min_{x\in\mathbb{R}^n} \left\{ \phi(x) - {y^k}^T x = \frac{1}{2}\rho \|x\|^2 + (c - y^k)^T x + \chi_S(x) \right\}. \tag{3.7}$$

Le problème (3.7) est équivalent au problème d'optimisation quadratique suivant [85, 44] :

$$\min_{x\in S} \left\| x - z^k \right\|^2, \tag{3.8}$$

où

$$z^k = \frac{1}{\rho}(y^k - c) = \frac{1}{\rho}\left((\rho I_n - D)x^k - c\right) = x_k - \frac{1}{\rho}(Dx^k + c).$$

Notons que la solution optimale du problème (3.8) n'est que la projection orthogonale du point z^k sur le polyèdre S.

Finalement, le problème (3.8) est équivalent au problème d'optimisation quadratique convexe suivant :

$$\min_{x\in S} \frac{1}{2}x^T x - {z^k}^T x. \tag{3.9}$$

On peut alors décrire l'algorithme DCA projectif pour trouver un minimum local au problème (3.5)-(3.5) [85, 44] :

Algorithme 3.2 : DCA projectif

Entrées : Les matrices A et D, les vecteurs b et c;

Sortie : Une solution optimale locale x^* du problème (3.5)-(3.6);

Initialisation : soit $\epsilon > 0$ une précision donnée et $x^0 \in \mathbb{R}^n$, poser $k = 0$ et $Stop = 0$;

tant que *Stop=0* **faire**

  $z^k = x^k - \frac{1}{\rho}(Dx^k + c)$;

  si $z^k \in S$ **alors**

    Poser $x^{k+1} = z^k$;

  sinon

    Résoudre le problème (3.9). Notons la solution optimale par p^k;

    Poser $x^{k+1} = p^k$;

  fin

  si $\left\|x^{k+1} - x^k\right\| \leq \epsilon$ **alors**

    $Stop = 1$;

    $x^* = x^{k+1}$;

  sinon

    Poser $k := k + 1$;

  fin

fin

Remarque 3.2. *La décomposition DC n'est pas unique.*

En effet, on peut prendre par exemple la décomposition suivante de f :

$$\begin{aligned} g(x) &= \frac{1}{2}x^T(D + \rho I_n)x + c^T x, \\ h(x) &= \frac{1}{2}\rho x^T x, \end{aligned}$$

où ρ est un nombre réel positif ou nul pour lequel la matrice $(D + \rho I_n)$ est définie positive ou bien semi-définie positive, i.e., $\rho \geq -\lambda_1$, où λ_1 est la plus petite valeur propre de D.

Pour cette décomposition, on arrive à la suite DCA suivante [85] :

$$y^k = \rho x^k \text{ et } x^{k+1} \in arg\min\{\frac{1}{2}x^T(D + \rho I_n)x + (c - \rho x^k)^T x, x \in S\}.$$

Exemple 3.2. *Considérons le programme quadratique convexe suivant :*

$$\begin{aligned} \min \quad & f(x) = -x_1^2 - x_2^2, \\ s.c. \quad & 0 \leq x_1 \leq 4, \\ & 0 \leq x_2 \leq 4. \end{aligned}$$

On a

$$D = \begin{pmatrix} -2 & 0 \\ 0 & -2 \end{pmatrix};\ c = \begin{pmatrix} 0 \\ 0 \end{pmatrix};\ A = I_2;\ b = \begin{pmatrix} 4 \\ 4 \end{pmatrix}.$$

Posons $\rho = 3$, $\epsilon = 0.0001$ *et* $k = 0$.

Soit $x^0 = (12/5, 12/5)^T$ *un point de départ pour ce problème, avec* $f(x^0) = -\frac{288}{25}$.

Itération 1 :

On a

$$z^0 = x^0 - \frac{1}{\rho}(Dx^0 + c) = \begin{pmatrix} 4 \\ 4 \end{pmatrix} \in S \Rightarrow x^1 = z^0.$$

On a $\|x^1 - x^0\| = 2.2627 > \epsilon$. *Donc on pose* $k = 1$.

Itération 2 :

On a

$$z^1 = x^1 - \frac{1}{\rho}(Dx^1 + c) = \begin{pmatrix} 20/3 \\ 20/3 \end{pmatrix} \notin S.$$

La projection du point z^1 *sur* S *est le point* $p^1 = (4, 4)^T$, *donc on pose* $x^2 = p^1$.

On a $\|x^2 - x^1\| = 0$. *Par conséquent, le minimum de* f *sur* S *est le vecteur* $x^* = x^2 = (4, 4)^T$, *avec* $f(x^*) = -32$.

3.3 Recherche de solutions globales

3.3.1 La méthode Branch & Bound

Principe général de la méthode

Un algorithme par séparation et évaluation, plus connu sous son nom anglais Branch and Bound “B&B”, est une méthode générique de résolution de problèmes d’optimisation, et plus particulièrement d’optimisation combinatoire discrète. Dans les méthodes par séparation et évaluation, la séparation permet d’obtenir une méthode générique pour localiser toutes les solutions optimales tandis que l’évaluation évite l’énumération systématique de toutes les solutions.

Séparation

La phase de séparation consiste à diviser le problème en un certain nombre de sous-problèmes qui ont chacun leur ensemble de solutions réalisables de telle sorte que tous ces ensembles forment une partition de l’ensemble de toutes les solutions possibles. Ainsi,

en résolvant tous les sous-problèmes et en prenant la meilleure solution trouvée, on est assuré d'avoir résolu le problème initial.

Ce principe de séparation peut être appliqué de manière récursive à chacun des sous-ensembles de solutions obtenus, et ceci tant qu'il y a des ensembles contenant plusieurs solutions. Les ensembles de solutions (et leurs sous-problèmes associés) construits ont une hiérarchie naturelle en arbre, souvent appelée arbre de recherche ou arbre de décision.

Evaluation

L'évaluation d'un nœud de l'arbre de recherche a pour but de déterminer la meilleure solution de l'ensemble des solutions réalisables associé au nœud en question, ou au contraire, de prouver mathématiquement que cet ensemble ne contient pas de solution intéressante pour la résolution du problème. Lorsqu'un tel nœud est identifié dans l'arbre de recherche, il est donc inutile d'effectuer la séparation de son espace de solutions.

La méthode B&B appliquée à la minimisation d'une fonction concave avec contraintes

En se basant sur [12], nous présentons la méthode de Branch and Bound pour la résolution globale du problème de minimisation concave suivant :

$$(P): \min_{x \in D} f(x),$$

où f est une fonction concave, pas nécessairement séparable, définie de $\mathbb{R}^n$ dans $\mathbb{R}$ et $D = \{x \in \mathbb{R}^n : \ g_i(x) \leq 0, \ i = 1, 2, \ldots, m\}$, où pour chaque $i = 1, 2, \ldots, m$, g_i est une fonction convexe finie sur $\mathbb{R}^n$. Nous supposerons que D est non vide, compact, et il existe un point $p \in \mathbb{R}^n$ tels que $g_i(p) < 0, \ i = 1, 2, \ldots, m$.

En introduisant la fonction convexe $g(x) = \max\{g_i(x), \ i = 1, 2, \ldots, m, \}$ on réécrit le problème (P) sous la forme suivante :

$$\begin{aligned} (P): \quad & \min \quad f(x), \\ & s.c. \quad g(x) \leq 0. \end{aligned}$$

Le problème (P) peut avoir plusieurs solutions optimales locales qui ne sont pas des solutions optimales globales. Il est bien connu qu'une solution optimale globale pour le problème (P) existe, de plus elle est atteinte en un point extrême de D. Soit $\bar{m}$ la valeur optimale de la fonction objectif du problème (P).

Definition 3.1. *Soient S un ensemble compact de $\mathbb{R}^n$ et $S^1, S^2, \ldots, S^q$ des sous-ensembles compacts de S. L'ensemble $\{S^1, S^2, \ldots, S^q\}$ est une partition de S, si*

$$S = \bigcup_{i=1}^{q} S_i,\ S_i \cap S_j = Fr_S(S_i) \cap Fr_S(S_j),\ \forall i, j \in \{1, 2, \ldots, q\},\ i \neq j,$$

où $Fr_S(S_i)$ dénote la frontière relative à S de S_i ($Fr_S(S_i) = (Adh^1(S_i) \backslash Int^2(S_i)) \cap S$).

Definition 3.2. *L'enveloppe convexe d'une fonction f 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 $Conv(T)$;*
2. *$g(x) \leq f(x)$ pour tout $x \in T$;*
3. *Si h est une fonction convexe définie sur $Conv(T)$, et si $h(x) \leq f(x)$ pour tout $x \in T$, alors $h(x) \leq g(x)$ pour tout $x \in T$.*

Pour lancer l'algorithme de Branch and Bound, un polyèdre compact X^0 contenant D est construit et un simplexe S^{01} qui contient X^0 est trouvé. Les étapes suivantes de l'algorithme créent des polyèdres compacts $X^1, X^2, \ldots, X^k, \ldots$, de telle sorte que

$$X^0 \supseteq X^1 \supseteq \ldots \supseteq X^k \supseteq \ldots \supseteq D.$$

À chaque étape k ($k \geq 1$), la technique d'évaluation est utilisée pour calculer une borne inférieure de $\bar{m}$ par le calcul d'une borne inférieure pour la valeur optimale de la fonction objectif du problème relaxé P_k :

$$(P_k) : \quad \min_{x \in X^k} f(x).$$

Pour ce faire, à une étape donnée k ($k \geq 1$), on utilise le polyèdre X^{k-1} et le point $x^{k-1} \in X^{k-1}$ obtenus à l'étape précédente pour créer le nouveau polyèdre X^k dans un premier temps ; puis, étant donnée une partition de S^{01} obtenue à partir de l'étape précédente, la séparation est utilisée pour créer une partition plus raffinée $S^{k,1}, S^{k,2}, \ldots, S^{k,k+1}$ de S^{01}, où $S^{k,j},\ j = 1, 2, \ldots, k+1$, sont des simplexes de dimension n.

Pour tout $j = 1, 2, \ldots, k+1$, l'enveloppe convexe g_{kj} de f sur $S^{k,j}$, comme nous le verrons, est une fonction affine qui est disponible à partir de l'étape précédente ou est

1. L'adhérence ou la fermeture d'un ensemble
2. L'intérieur d'un ensemble

calculée. Le problème P_k est séparé en $k+1$ programmes linéaires sous-estimés $P_{kj}, j = 1, 2, \ldots, k+1$, qui consistent à minimiser g_{kj} sur $X^k \cap S^{k,j}$.

L'algorithme calcule ensuite la valeur minimale parmi les valeurs optimales des fonctions objectif des programmes linéaires $P_{kj}, j = 1, 2, \ldots, k+1$. Cette valeur, notée LB, correspond à une borne inférieure de la valeur optimale de la fonction objectif du problème P_k et par conséquent, c'est un minorant de $\bar{m}$.

Pour toute étape k, $x^k \in X^k \supseteq D$ représente la solution optimale du programme linéaire dont la valeur optimale de la fonction objectif est égale à LB. Notons que x^k peut se trouver ou pas dans D. Si $x^k \in D$, alors l'algorithme calcule la valeur de $f(x^k)$, qui est une borne supérieure pour $\bar{m}$.

À chaque étape, on enregistre la plus petite borne supérieure UB rencontrée ainsi que le point x^c vérifiant $f(x^c) = UB$. Si pour un $\epsilon \geq 0$ choisi, l'inégalité $UB - LB \leq \epsilon$ est satisfaite, alors l'algorithme se termine. Lorsque cela se produit, x^c est une solution $\epsilon-$optimale pour le problème P en ce sens que $x^c \in D$ et $\bar{m} \leq f(x^c) \leq \bar{m} + \epsilon$.

Pour calculer l'enveloppe convexe g_{kj} de f sur $S^{k,j}$, l'algorithme résout les $(n+1)$ équations linéaires suivantes :

$$a^T v^i + \alpha = f(v^i),\ i = 0, 1, \ldots, n, \tag{3.10}$$

où $v^i, i = 0, 1, \ldots, n$, sont les sommets de $S^{k,j}$ et $a \in \mathbb{R}^n$, $\alpha \in \mathbb{R}$ sont des inconnues à calculer. On aura alors

$$g_{kj}(x) = a^T x + \alpha. \tag{3.11}$$

Etant donnée une partition $\{S^{k-1,1}, S^{k-1,2}, \ldots, S^{k-1,k}\}$ de S^{01} de l'étape $k-1$, pour chaque $j = 1, 2, \ldots, k$, $S^{k-1,j}$ est un simplexe de dimension n. Pour créer une partition plus raffinée de S^{01}, l'algorithme utilise une technique dite "procédure de bissection" : Tout d'abord, un simplexe $S^{k-1,\bar{j}}$ contenant x^{k-1} est trouvé. Soient $v^0, v^1, \ldots, v^n$ les points représentant les sommets de ce simplexe.

Ensuite, le point milieu v de l'une des plus longue arrêtes de $S^{k-1,\bar{j}}$ est trouvé. Soient v^d et v^e les points extrêmes de cette arrête. Les deux simplexes $S^{k,1}, S^{k,2}$, avec des sommets

$$v^0, v^1, \ldots, v^{d-1}, v, v^{d+1}, \ldots, v^n \text{ et } v^0, v^1, \ldots, v^{e-1}, v, v^{e+1}, \ldots, v^n$$

respectivement, sont ensuite formés. La paire $\{S^{k,1}, S^{k,2}\}$ est une partition de $S^{k-1,\bar{j}}$ [53]. La partition la plus raffinée de S^{01} est alors obtenue par le remplacement de $S^{k-1,\bar{j}}$, dans la partition $\{S^{k-1,1}, S^{k-1,2}, \ldots, S^{k-1,k}\}$ avec $S^{k,1}$ et $S^{k,2}$.

Nous pouvons maintenant résumer les différentes étapes décrites précédemment dans l'algorithme 3.3 [12].

Remarque 3.3. *Dans [12], la convergence de l'algorithme 3.3 est justifiée et un exemple numérique est résolu d'une façon détaillée.*

Algorithme 3.3 : B&B

1 Initialisation $k = 1$ et $Stop = 0$;
2 Choisir $\epsilon \geq 0$, trouver un point p, tel que $g(p) < 0$. Soient $F = \emptyset, UB = f(p)$ et $x^0 = p$;
3 Trouver le polyèdre compact X^0, tel que $D \subseteq X^0$. Exprimer X^0 comme l'ensemble $\{x \in \mathbb{R}^n \ : \ Ax \leq b\}$, où A est une $(m \times n)$-matrice et $b \in \mathbb{R}^m$;
4 Trouver le simplexe S^{01} de sorte que $X^0 \subseteq S^{01}$, où S^{01} est décrit par les sommets $v_0^0, v_0^1, \ldots, v_0^n \in \mathbb{R}^n$;
5 Trouver l'enveloppe convexe $g_{01} : S^{01} \to \mathbb{R}$ de f sur S^{01}, par la résolution du système d'équations linéaires (3.10) avec $v^i = v_0^i, \ i = 0, 1, \ldots, n$;
6 Trouver le point extrême x^{01}, solution optimale du programme linéaire : $P_{01} : \ \min g_{01}(x), \ \text{s.c. } x \in X^0 \cap S^{01}$;
7 Soit $LB = g_{01}(x^{01}), \ g_0^* = g_{01}$ et $x^0 = x^{01}$;
8 **tant que** *Stop=0* **faire**
  Supposons sans perte de généralité que $x^{k-1} \in S^{k-1,k}$. Supposons également que $v_{k-1}^i, i = 1, 2, \ldots, n$, sont les sommets de $S^{k-1,k}$.
9 **si** $x^{k-1} \notin D$ **alors**
10 aller à la ligne 19;
  sinon
11 **si** $f(x^{k-1}) < UB$ **alors**
12 on pose $UB = f(x^{k-1})$ et $x^c = x^{k-1}$;
  fin
13 **si** $UB - LB \leq \epsilon$ **alors**
14 $Stop = 1$;
15 $x^* = x^c$ est une solution ϵ-optimale pour le problème (P);
16 aller à la ligne 8;
  fin
  fin
17 **si** $x^{k-1} \in D$ **alors**
18 on pose $X^k = X^{k-1}$ et aller à la ligne 28;
  sinon
19 Trouver le point $z^{k-1} \in D$ sur le segment de ligne $[p, x^{k-1}]$, tel que $g(z^{k-1}) = 0$;
20 **si** $f(z^{k-1}) \geq UB$ **alors**
21 aller à la ligne 27;
  sinon
22 On pose $UB = f(z^{k-1})$ et $x^c = z^{k-1}$;
23 **si** $UB - LB \leq \epsilon$ **alors**
24 $Stop = 1$;
25 $x^* = x^c$ est une solution ϵ-optimale pour le problème (P);
26 aller à la ligne 8;
  fin
  fin
  fin
27 On pose $X^k = X^{k-1} \cap \{x \in \mathbb{R}^n \ : \ \left[t^{k-1}\right]^T (x - z^{k-1}) \leq 0\}$, où t^{k-1} est le sous-gradient de g en z^{k-1};
28 Utiliser la bisection pour former la partition $\{S^{k,1}, S^{k,2}\}$ de $S^{k-1,k}$, où $S^{k,1}$ et $S^{k,2}$ sont des simplexes;
29 **pour** $j = 1, 2$, **faire**
30 Trouver l'enveloppe convexe $g_{kj} : S^{k,j} \to \mathbb{R}$ de f sur $S^{k,j}$ par la résolution du système d'équations linéaires (3.10);
31 Trouver la solution optimale x^{kj} pour le problème de PL : $(P_{kj}) \ \min g_{kj}(x), \ \text{s.c. } x \in X^k \cap S^{k,j}$.
  fin
32 **si** $k \neq 1$ **alors**
33 **pour** $j = 1, 2, \ldots, k-1$ **faire**
34 Renommer $S^{k-1,j}$ par $S^{k,j+2}$ et $g_{k-1,j}$ par $g_{k,j+2}$;
  fin
35 **pour** $j = 3, 2, \ldots, k+1$ **faire**
36 **si** $S^{k,j} \notin F$ **alors**
37 Trouver la solution optimale x^{kj} pour le problème de programmation linéaire (P_{kj});
  fin
  fin
  fin
38 **pour** $j = 1, 2, \ldots, k+1$ **faire**
39 **si** $S^{k,j} \notin F \ et \ g_{kj}(x^{kj}) > LB$ **alors**
40 $F = F \cup \{S^{k,j}\}$;
  fin
  fin
41 Calculer $LB = \min\{g_{kj} \ : \ j = 1, 2, \ldots, k+1 \text{ et } S^{k,j} \notin F\}$;
42 Soit $x^k = x^{kj^*}$ et $g_k^* = g_{kj^*}$, où $LB = g_{kj^*}(x^{kj^*})$ et poser $k := k + 1$;
 fin

3.3.2 La méthode des ensembles d'approximation

Dans cette section, nous présentons la méthode des ensembles d'approximation développée dans [31] pour la minimisation d'une fonction objectif quadratique strictement concave sur un domaine délimité par des contraintes linéaires d'inégalité. Cet algorithme construit des ensembles d'approximation du premier et du second ordre de l'ensemble niveau de la fonction objectif au point courant, et ce, pour générer une série de minima locaux qui convergent vers un minimum global approché.

Position du problème

Soit le problème suivant :

$$\min f(x) = \frac{1}{2}x^T Dx + c^T x, \tag{3.12}$$

$$Ax \leq b, \tag{3.13}$$

où D est une matrice réelle définie négative et symétrique d'ordre n ; c, x sont des $n-$vecteurs, A est une matrice réelle de dimension $m \times n$ et b est un $m-$vecteur. On note S l'ensemble des solutions réalisables.

Definition 3.3. *[34] Le sous-ensemble fini de* $E_{f(z)}(f)$

$$R_z = \{y^1, y^2, \ldots, y^r : y^j \in E_{f(z)}(f),\ j = 1, 2, \ldots, r\},$$

où r *est un entier strictement positif, est appelé l'ensemble d'approximation du premier ordre de l'ensemble* $E_{f(z)}(f)$.

Lemma 3.1. *[31] Soient* $z \in S$ *et* $h \in \mathbb{R}^n$ *vérifiant* $h^T \nabla f(z) > 0$. *Supposons qu'il existe un indice* $i \in \{1, 2, \ldots, m\}$, *tel que* $A_i^T h > 0$ *et* $b_i - A_i^T z > 0$, *où* A_i *est la ième ligne de la matrice* A. *Si*

$$-\frac{2h^T(Dz + c)}{h^T Dh} \leq \min_{A_i^T h > 0} \frac{b_i - A_i^T z}{A_i^T h}, \tag{3.14}$$

alors il existe un nombre positif γ, *tel que* $y = z + \gamma h \in S \cap E_{f(z)}(f)$.

Considérons un point réalisable $z \in S$. Dans cette méthode, l'ensemble d'approximation du premier ordre R_z est construit comme suit : D'abord r vecteurs $h^j \in \mathbb{R}^n$ sont choisis de telle sorte à vérifier les conditions :

$${h^j}^T \nabla f(z) > 0,\ \exists i \in \{1, 2, \ldots, m\},\ \text{tel que } A_i^T h^j > 0 \text{ et } b_i - A_i^T z > 0$$

et

$$\gamma_j = -\frac{2{h^j}^T(Dz+c)}{{h^j}^T D h^j} \leq \min_{A_i^T h^j > 0} \frac{b_i - A_i^T z}{A_i^T h^j}.$$

Puis, l'ensemble R_z est formé de la manière suivante :

$$R_z = \{y^1, y^2, \ldots, y^r : \ y^j = z + \gamma_j h^j, \ j = 1, 2, \ldots, r\}.$$

Lemma 3.2. *[31] Soit $z \in S$. S'il existe un point $y^i \in R_z$, tel que $(u^i - y^i)^T \nabla f(y^i) < 0$, où u^i est une solution du programme linéaire $\min_{x \in S} x^T \nabla f(y^i)$, alors $f(u^i) < f(z)$.*

S'il n'existe aucun point y^i de R_z vérifiant la condition $(u^i - y^i)^T \nabla f(y^i) < 0$, alors l'ensemble d'approximation du second ordre est construit grâce au lemme suivant :

Lemma 3.3. *[31] Soient $\hat{x} = -D^{-1}c$ le point maximisant f sur $\mathbb{R}^n$ et $z \in S$ vérifiant $f(z) \neq f(\hat{x})$. Alors les points*

$$\bar{y}^j = \hat{x} + \alpha_j (u^j - \hat{x}) \in E_{f(z)}(f), \ j = 1, 2, \ldots, r, \tag{3.15}$$

où

$$\alpha_j = \left(\frac{2(f(z) - f(\hat{x}))}{(u^j - \hat{x})^T D (u^j - \hat{x})} \right)^{\frac{1}{2}} \tag{3.16}$$

et $u^j, \ j = 1, 2 \ldots, r$ sont des solutions optimales des programmes linéaires (4.11).

Definition 3.4. *L'ensemble*

$$\bar{R}_z = \{\bar{y}^j, \ j = 1, 2 \ldots, r\},$$

où $\bar{y}^j$ est défini par (3.15)-(3.16), est appelé un ensemble d'approximation du second ordre.

Algorithme des ensembles d'approximation "AEA"

L'algorithme des ensembles d'approximation se présente alors de la manière suivante [31] :

Algorithme 3.4 : AEA

Entrées : Les matrices A et D, les vecteurs b et c;

Sortie : Une solution optimale globale approchée x^* du problème (3.12)-(3.13);

1 Initialisation : Poser $k = 0$, $Stop = 0$. Soit x^0 une solution réalisable de départ;
2 **tant que** $Stop=0$ **faire**
3 Trouver un minimal local $z^k \in S$ en utilisant la méthode du gradient conditionnel initialisée par le point x^k;
4 Construire l'ensemble d'approximation du premier ordre A_{z^k};
5 **pour** $y^i \in R_{z^k}$ **faire**
6 Calculer $u^i \in \arg\min_{x \in S} x^T \nabla f(y^i)$;
fin
7 Déterminer l'indice $p \in \{1, 2, \ldots, r\}$, tel que

$$\theta^k = (u^p - y^p)^T \nabla f(y^p) = \min_{i=1,2,\ldots,r} \theta_i = (u^i - y^i)^T \nabla f(y^i);$$

8 **si** $\theta^k < 0$ **alors**
9 Poser $x^{k+1} = u^p$ et $k = k + 1$;
10 Aller à la ligne 3;
sinon
11 Construire l'ensemble d'approximation du second ordre $\bar{R}_{z^k}$;
12 **pour** $\bar{y}^i \in \bar{R}_{z^k}$ **faire**
13 Calculer $v^i \in \arg\min_{x \in S} x^T \nabla f(\bar{y}^i)$;
fin
14 Déterminer l'indice $q \in \{1, 2, \ldots, r\}$, tel que

$$\bar{\theta}^k = (v^q - \bar{y}^q)^T \nabla f(\bar{y}^q) = \min_{i=1,2,\ldots,r} \bar{\theta}_i = (v^i - \bar{y}^i)^T \nabla f(\bar{y}^i);$$

15 **si** $\bar{\theta}^k < 0$ **alors**
16 Poser $x^{k+1} = v^q$ et $k = k + 1$;
sinon
17 La solution $x^* = z^k$ est un minimum global approché;
18 $Stop = 1$;
fin
fin
fin

Exemple numérique

Exemple 3.3. *Considérons le programme quadratique concave suivant :*

$$\begin{array}{ll} \min & f(x) = -x_1^2 - \frac{1}{2}x_1x_2 - \frac{3}{2}x_2^2 - x_1 + 2x_2, \\ s.c. & -6x_1 + 2x_2 \leq 6, \\ & -x_1 - x_2 \leq 2, \\ & 2x_1 - x_2 \leq 2, \\ & 3x_1 + x_2 \leq 3. \end{array}$$

On a $D = \begin{pmatrix} -2 & -1/2 \\ -1/2 & -3 \end{pmatrix}$; $c = \begin{pmatrix} -1 \\ 2 \end{pmatrix}$; $A = \begin{pmatrix} -6 & 2 \\ -1 & -1 \\ 2 & -1 \\ 3 & 1 \end{pmatrix}$; $b = \begin{pmatrix} 6 \\ 2 \\ 2 \\ 3 \end{pmatrix}$.

Le maximum de la fonction f sur $\mathbb{R}^2$ est donné par :

$$\hat{x} = (-16/23, 18/23)^T, \ \hat{f} = f(\hat{x}) = 26/23.$$

Soit $x^0 = (-1,-1)^T$ la solution réalisable de départ pour ce problème, avec $f(x^0) = -4$.

Itération 1 :

En commençant par le point initial x^0, la méthode de gradient conditionnel nous donne le minimum local suivant :

$$z^0 = (0,-2)^T, \ f(z^0) = -10.$$

Posons $r = n = 2$ et construisons l'ensemble d'approximation d'ordre 1, R_{z^0}.

Pour $j = 1, 2$ on pose $h^1 = (1, 1/5)$ et $h^2 = (2, 1/3)$. On aura alors :

$$h^{1T}\nabla f(z^0) = \frac{8}{5} > 0, \ \gamma_1 = -\frac{2h^{1T}(Dz^0 + c)}{h^{1T}Dh^1} = \frac{40}{29} \leq \min_{A_i^T h^1 > 0} \frac{b_i - A_i^T z^0}{A_i^T h^1} = \frac{b_4 - A_4^T z^0}{A_4^T h^1} = \frac{25}{16}.$$

$$h^{2T}\nabla f(z^0) = \frac{8}{3} > 0, \ \gamma_2 = -\frac{2h^{2T}(Dz^0 + c)}{h^{2T}Dh^2} = \frac{16}{27} \leq \min_{A_i^T h^2 > 0} \frac{b_i - A_i^T z^0}{A_i^T h^2} = \frac{b_4 - A_4^T z^0}{A_4^T h^2} = \frac{15}{19}.$$

Calcul des vecteurs $y^j = z^0 + \gamma_j h^j$:

$$y^1 = (40/29, -50/29)^T, \ y^2 = (32/27, -146/81)^T.$$

On a

$$\nabla f(y^1) = (-84/29, 188/29)^T, \ \nabla f(y^2) = (-200/81, 184/27)^T.$$

Les solutions optimales des PL $\min_{x\in S} x^T\nabla f(y^j), j = 1, 2$ *sont*

$$u^1 = (0, -2)^T,\ u^2 = (0, -2).$$

Le nombre θ^0 *est donné par*

$$\theta^0 = (u^p - y^p)^T\nabla f(y^p) = \min_{j=1,2}(u^j - y^j)^T\nabla f(y^j) = \min\{64/29, 128/81\} = 128/81 \Rightarrow p = 2.$$

On a $\theta_0 = \theta_2 > 0$*, donc on construit l'ensemble d'approximation du second ordre* $\bar{R}_{z^0}$:

$$\alpha_1 = \left(\frac{2(f(z^0) - f(\hat{x}))}{(u^1 - \hat{x})^T D(u^1 - \hat{x})}\right)^{1/2} = 1,\ \alpha_2 = \alpha_1 = 1.$$

Donc

$$\bar{y}^1 = \bar{y}^2 = \hat{x} + \alpha_1(u^1 - \hat{x}) = (0, -2)^T,\ \nabla f(\bar{y}^1) = \nabla f(\bar{y}^2) = (0, 8)^T.$$

Les solutions des programmes linéaires $\min\limits_{x\in S} x^T\nabla f(\bar{y}^1)$ *sont*

$$v^1 = v^2 = (0, -2)^T.$$

Le nombre $\bar{\theta}_0$ *est*

$$\bar{\theta}^0 = \min_{j=1,2}\{(v^j - \bar{y}^j)^T\nabla f(\bar{y}^j)\} = 0.$$

Par conséquent, la solution globale approchée et l'optimum global approché sont donnés par :

$$x^* = z^0 = (0, -2)^T \text{ et } f(x^*) = f(z^0) = -10.$$

3.4 Problèmes pratiques modélisés sous forme de programmes quadratiques concaves

La plupart des problèmes pratiques d'optimisation sont généralement très complexes, cependant certains d'entre eux peuvent se modéliser sous forme de problèmes de minimisation concave ou y être réduits. Parmi ces derniers, nous allons présenter quelques exemples [79].

3.4.1 Problèmes à charge fixe

Le problème à charge fixe a été formulé à l'origine par G. B. Dantzig et W. Hirsch en 1954 [51], puis il est étudié par la suite par plusieurs chercheurs [7, 100, 78]. La charge fixe, par exemple un coût d'installation, survient pour un problème de transport dans lequel chaque origine a un coefficient de charge fixe en plus du coefficient de coût habituel. La charge fixe se produit uniquement lorsque la variable apparaît dans la solution avec un niveau positif.

Généralement, le problème à charge fixe se présente sous la forme suivante :

$$\begin{aligned} \min f(x) = \sum_{j=1}^{n} f_j(x_j), \\ Ax = b, \\ x \geq 0, \end{aligned}$$

où $c, x \in \mathbb{R}^n$, A est une matrice réelle de dimension $m \times n$, $b \in \mathbb{R}^m$ et pour $j = 1, 1, \ldots, n$,

$$f_j(x_j) = c_j x_j + k_j \sigma_j, \text{ avec } k_j \in \mathbb{R}, \ \sigma_j = \begin{cases} 0, & \text{si } x_j = 0, \\ 1, & \text{si } x_j > 0. \end{cases}$$

Si certaines ou toutes les charges k_j sont positives, alors la fonction objectif f est concave.

3.4.2 Problèmes à variables binaires

Considérons le problème de programmation $0-1$ suivant :

$$\min z(x) = c^T x, \tag{3.17}$$

$$Ax \leq b, \tag{3.18}$$

$$x \in \{0,1\}^n, \tag{3.19}$$

où c est un n-vecteur, A est une matrice réelle de dimension $m \times n$ et b est un $m-$vecteur. De nombreux algorithmes existent pour résoudre le problème (3.17)-(3.19), nous pouvons citer $B\&B$, les méthodes des coupes, etc. Toutefois, lorsque la dimension du problème est grande, le temps d'exécution des différents algorithmes augmente considérablement. Sous des hypothèses convenables, on peut transformer le problème (3.17)-(3.19) en un problème de programmation quadratique concave.

Pour ce faire, nous procédons à la relaxation du problème discret, et ce, en remplaçant les variables binaires $x \in \{0,1\}^n$ par des variables continues $x \in [0,1]^n$. Ainsi, le problème (3.17)-(3.19) sera équivalent au problème de minimisation concave suivant :

$$\min F_\mu = z(x) + \mu x^T(e - x), \tag{3.20}$$

$$Ax \leq b, \tag{3.21}$$

$$x \in [0,1]^n, \tag{3.22}$$

où e est le vecteur des uns et μ est un paramètre choisit de telle sorte à assurer l'équvalence entre (3.17)-(3.19) et (3.20)-(3.22) et F_μ soit une fonction concave. Cependant, la minimisation de F_μ (qui est non-convexe) est un problème NP-difficile [90] même si elle porte sur des variables continues. À notre connaissance, il n'existe pas de méthodes efficace pour la détermination d'une valeur optimale pour le paramètre μ [4].

3.4.3 Problèmes de production et transport avec coût de production concave

Etant données m usines $U^1, U^2, \ldots, U^m$ fabriquant un certain produit et n entrepôts $E^1, E^2, \ldots, E^n$. On cherche à déterminer le niveau de production y_i de l'usine U^i, $i = 1, 2, \ldots, m$ et la quantité du produit x_{ij} transportée de l'usine U^i à l'entrepôt E^j de façon à satisfaire la demande b_j de chaque entrepôt, tout en minimisant le coût total. Ce coût est la somme du coût de production $g(y)$ des différentes usines et des coûts de transport c_{ij}, supposés linéaires, entre l'usine U^i et l'entrepôt E^j pour $i = 1, 2, \ldots, m$ et $j = 1, 2, \ldots, n$.

La fonction g est supposée concave, ce qui permet de modéliser les économies d'échelle. Contrairement à la plupart des autres modèles semblables de la littérature, g n'est pas supposée séparable, ce qui permet de prendre en compte des situations telles que l'achat groupé de matériaux bruts par les usines auprès d'un même fournisseur. Ce problème

d'optimisation revient à minimiser une fonction concave sur un polytope :

$$\begin{aligned}
\min\ g(y) + \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij},\\
\sum_{j=1}^{n} x_{ij} = y_i,\ i = 1,2,\ldots,m,\\
\sum_{i=1}^{m} x_{ij} = b_j,\ j = 1,2,\ldots,n,\\
y_i \le s_i,\ i = 1,2,\ldots,m,\\
y_i, x_{ij} \ge 0,\ i = 1,2,\ldots,m,\ j = 1,2,\ldots,n,
\end{aligned}$$

où s_i est la capacité de production de l'usine $U^i,\ i = 1,2,\ldots,m$.

3.4.4 Problèmes d'affectation quadratique

Le problème d'affectation quadratique "PAQ" est formulé initialement par Koopmans et Beckmann [65], puis ce modèle est généralisé plus tard par plusieurs chercheurs. On désire affecter n installations à n emplacements différents, de telle sorte à minimiser le coût total d'affectation. Le problème d'affectation quadratique PAQ1 se formule alors de la manière suivante [9] :

$$\begin{aligned}
\min x^T Dx,\\
\sum_{i=1}^{n} x_{ij} = 1,\ j = 1,2,\ldots,n,\\
\sum_{j=1}^{n} x_{ij} = 1,\ i = 1,2,\ldots,n,\\
x_{ij} \in \{0,1\},\ i,j = 1,2,\ldots,n.
\end{aligned}$$

où $x = (x_{ij},\ i,j = 1,2,\ldots,n)$ est le vecteur des variables binaires, avec $x_{ij} = 1$ si l'installation i est placée à l'emplacement j et $x_{ij} = 0$, sinon. La matrice carrée symétrique S est une matrice des coefficients entiers non négatifs s_{ijkl} qui représentent le coût d'affectations simultanées de l'installation i à l'emplacement j et de l'installation k à l'emplacement l. Notons par X_A le domaine admissible du problème PAQ1.

Remarquons que si nous remplaçons la fonction objectif du problème PAQ1 par la fonction $x^T Dx$, où $D = S - \rho I$ (I étant la matrice identité d'ordre n^2 et ρ un scalaire),

alors le problème obtenu sera équivalent au problème original, puisque $x^T Mx = n\rho$ est une constante pour toute solution réalisable du problème original PAQ1. De plus, on peut démontrer que si ρ est choisit te telle sorte qu'il soit supérieur à la plus grande sommes des éléments de chaque ligne de S, alors D devient définie négative et par conséquent $x^T Dx$ devient strictement concave [9]. Enfin, puisque les points extrêmes de

$$X = \{x \in \mathbb{R}^n,\ \sum_{i=1}^{n} x_{ij} = 1,\ j = 1, 2, \ldots, n,\ \sum_{j=1}^{n} x_{ij} = 1,\ i = 1, 2, \ldots, n,\ x \geq 0\}$$

sont en correspondance bijective avec les points du domaine X_A, et comme le minimum d'une fonction strictement concave sur un polytope est atteint en un point extrême, il s'ensuit que le problème PAQ1 peut être résolu d'une façon équivalente :

$$\text{PAQ 2 :}\ \min\{x^T Dx :\ x \in X\}.$$

Notons qu'une solution optimale locale du problème PAQ2 peut ne pas être globale. Aussi nous pouvons démontrer que tout point extreme du problème PAQ2 est un point KKT.

Une variété de problèmes pratiques sont également abordés dans le dernier chapitre de cette thèse. Ces problèmes sont pris de plusieurs références et sont utilisés comme étant des problèmes-test pour comparer les performances de l'algorithme proposé dans cette thèse avec les différents algorithmes existants de la programmation quadratique concave.

Chapitre 4

L'algorithme des approximations linéaires successives pour la minimisation globale d'un PQ concave

4.1 Introduction

Dans ce chapitre, nous allons présenter l'algorithme dit "Algorithme des Approximations Linéaires Successives (AALS)", que nous avons proposé dans [17, 93]. Cet algorithme généralise ceux proposés dans Enkhbat (2003) [34], Chinchuluun et al. (2005)[31] et Bayartugs et al. (2014) [8]. En effet, notre algorithme minimise une fonction quadratique concave générale soumise à des contraintes linéaires qui peuvent être soit des égalités ou des inégalités et les variables sont bornées par des bornes finies ou infinies. Il permet également de résoudre les programmes quadratiques concaves avec des contraintes de bornes sur les variables (box-constrained concave quadratic programs).

AALS commence par un point extrême initial, puis il se déplace du point extrême courant, qui n'est pas forcément une solution locale, à un point ayant une meilleure valeur de la fonction objectif. Le passage d'une solution réalisable de base vers une autre est effectuée par la construction de certains ensembles d'approximation et la résolution d'une suite de programmes linéaires.

Afin de prendre en considération le fait que la matrice de la forme quadratique est

semi-définie, les ensembles d'approximation dans notre algorithme sont construits d'une manière appropriée. De plus, notre algorithme est initialisé par un point extrême correspondant à la solution du problème d'optimisation de la partie linéaire de la fonction objectif sur le domaine réalisable du problème original.

4.2 Position du problème

Le problème de programmation quadratique concave se présente sous la forme générale suivante :

$$\min f(x) = \frac{1}{2}x^T Dx + c^T x, \tag{4.1}$$

$$A_1 x \leq b_1, \tag{4.2}$$

$$A_2 x = b_2, \tag{4.3}$$

$$l \leq x \leq u, \tag{4.4}$$

où D est une matrice réelle semi-définie négative et symétrique d'ordre n ; c, x, l, u sont des vecteurs de $\mathbb{R}^n$, les composantes de l et u peuvent prendre des valeurs finies ou infinies ; A_1 est une matrice réelle de dimension $m_1 \times n$, A_2 est une matrice réelle de dimension $m_2 \times n$, $b_1 \in \mathbb{R}^{m_1}$ et $b_2 \in \mathbb{R}^{m_2}$.

Notons l'ensemble admissible par

$$S = \{x \in \mathbb{R}^n : \ A_1 x \leq b_1, \ A_2 x = b_2 \ \text{ et } l \leq x \leq u\}.$$

Dan ce qui suit, on suppose que S est non vide et borné, ce qui garantira l'existence d'une solution optimale finie pour le problème (4.1)-(4.4). Rappelons les théorèmes suivants :

Theorème 4.1. *(Hiriart-Urruty et Ledyav 1996 [50]) Soit $z \in S$ vérifiant les inégalités $-\infty \leq \inf_S f(x) < f(z)$. Alors, z est une solution du problème (4.1)-(4.4) si et seulement si*

$$(x-y)^T \nabla f(y) \geq 0, \forall y \in E_{f(z)}(f) \cap S \ \textit{et} \ x \in S. \tag{4.5}$$

Theorème 4.2. *(Strekalovsky 1998 [91]) Soit $z \in S$ vérifiant les inégalités $-\infty \leq \inf_{\mathbb{R}^n} f(x) < f(z)$. Alors, z est une solution du problème (4.1)-(4.4) si et seulement si*

$$(x-y)^T \nabla f(y) \geq 0, \forall y \in E_{f(z)}(f) \ \textit{et} \ x \in S. \tag{4.6}$$

4.3 Une itération de l'algorithme

Soit z un point de S. Les conditions d'optimalité globale (4.5) ou (4.6) sont difficiles à vérifier pour le point z car la ligne de niveau $E_{f(z)}(f)$ est un ensemble qui n'est pas fini, c'est pourquoi à chaque itération de notre algorithme, on construit l'ensemble d'approximation du premier ordre noté R_z.

L'algorithme des ensembles d'approximation [31], décrit dans le chapitre 3, construit cet ensemble avec des points $y^j \in S$, puis les conditions d'optimalité globale du théorème 4.1 sont utilisées pour passer à un point réalisable meilleur. Dans ce chapitre, nous construisons l'ensemble R_z avec des points y^j qui ne sont pas nécessairement réalisables, donc afin d'améliorer le point réalisable courant z, on utilise les conditions d'optimalité du théorème 4.2.

Le lemme suivant nous permet de construire l'ensemble d'approximation du premier ordre avec des points qui n'appartiennent pas nécessairement à S.

Lemma 4.1. *Soit $h \in \mathbb{R}^n$, tel que $h^T Dh \neq 0$. Considérons le nombre réel γ calculé comme suit :*

$$\gamma = -\frac{2h^T(Dz+c)}{h^T Dh}. \tag{4.7}$$

Alors le point

$$y_\gamma = z + \gamma h \in E_{f(z)}(f).$$

Preuve. Soit h un vecteur non nul de $\mathbb{R}^n$, tel que $h^T Dh \neq 0$. On a

$$\begin{aligned} f(z+\gamma h) - f(z) &= \frac{1}{2}(z+\gamma h)^T D(z+\gamma h) + c^T(z+\gamma h) - \frac{1}{2}z^T Dz - c^T z \\ &= \gamma z^T Dh + \frac{1}{2}\gamma^2 h^T Dh + \gamma c^T h \\ &= -\frac{2h^T(Dz+c)}{h^T Dh}\left[z^T Dh - \frac{h^T(Dz+c)}{h^T Dh}h^T Dh + c^T h\right] \\ &= -\frac{2h^T(Dz+c)}{h^T Dh}\left[z^T Dh - h^T Dz - h^T c + c^T h\right] \\ &= 0. \end{aligned}$$

Donc $f(z+\gamma h) = f(z)$. Par conséquent, le point $y_\gamma = z + \gamma h \in E_{f(z)}(f)$. □

Dans [35], un algorithme pour la résolution des programmes quadratiques concaves à variables bornées (box-constrained concave quadratic programs) est proposé. L'ensemble

d'approximation du premier ordre est construit avec $n+1$ vecteurs $y^j = z + \gamma_j h^j,\ j = 1, \ldots, n+1$ (n est le nombre de variables), avec $\gamma_j = -\frac{2{h^j}^T(Dz+c)}{{h^j}^T D h^j}$ et h^j sont des vecteurs calculés en utilisant les composantes de z et les bornes inférieures et supérieures des variables.

Dans [34, 31, 8], les auteurs résolvent le problème de minimization d'une fonction quadratique strictement concave soumise à des contraintes linéaires d'inégalité. Dans [34], l'ensemble R_z est construit avec $2n$ vecteurs $y^j = z + \gamma_j h^j,\ j = 1, \ldots, 2n$, où n est le nombre de variables et h^j est calculé en utilisant les vecteurs propres de la matrice de la forme quadratique.

Dans [31], les vecteurs h^j sont générés aléatoirement de telle sorte à vérifier les conditions ${h^j}^T \nabla f(z) > 0$ et $y^j = z + \gamma_j h^j \in S$. Par conséquent, le choix des vecteurs h^j nécessite la résolution de plusieurs inégalités à chaque itération. Donc l'étape de calcul des vecteurs h^j consomme beaucoup de temps d'exécution et d'espace mémoire. Finalement, dans [8], l'ensemble R_z est construit avec m vecteurs $y^j = z + \gamma_j h^j,\ j = 1, \ldots, m$, où m est le nombre de contraintes et ${h^j}^T$ est égal la jème ligne de la matrice des contraintes, qui doit vérifier l'inégalité ${h^j}^T \nabla f(z) > 0$.

Dans ce travail, l'ensemble d'approximation du premier ordre est construit sans supposer aucune condition sur les vecteurs h^j. De plus, la construction prend en considération le fait que la matrice D soit semi-définie. Soit r un entier positif choisit judicieusement. Nous construisons l'ensemble R_z avec la procédure suivante :

Algorithme 4.1 : Construction de l'ensemble d'approximation du premier ordre

1 Poser $R_z = \emptyset$, $J = \emptyset$, $r_1 = 0$ et $j = 1$;
2 **tant que** $j \leq r$ **faire**
3 Choisir un vecteur $h^j \in \mathbb{R}^n$;
4 **si** ${h^j}^T D h^j \neq 0$ **alors**
5 Calculer

$$\gamma_j = -\frac{2{h^j}^T(Dz + c)}{{h^j}^T D h^j}; \tag{4.8}$$

6 Calculer

$$y^j = z + \gamma_j h^j; \tag{4.9}$$

7 Poser $R_z = R_z \cup \{y^j\}$, $J = J \cup \{j\}$, $r_1 = r_1 + 1$ et $j = j + 1$;
  sinon
8 $j = j + 1$;
  fin
fin

Remarque 4.1. *Les vecteurs $h^j,\ j = 1, 2, \ldots, r$ peuvent être choisit d'une façon appropriée ou générés aléatoirement.*

Considérons le problème d'optimisation quadratique suivant :

$$\max_{x \in \mathbb{R}^n} f(x). \tag{4.10}$$

Notons par $\hat{f}$ l'optimum de ce problème. Si $\hat{f} < \infty$, alors on note par $\hat{x}$ l'une de ses solutions optimales. Remarquons que si le problème (4.10) est non borné, alors $\forall x \in \mathbb{R}^n,\ \nabla f(x) \neq 0$. Donc $\nabla f(z) = Dz + c \neq 0$, ainsi pour ${h^j}^T D h^j \neq 0$, on obtient $\gamma_j \neq 0$ et $y^j \neq z$, sinon $y^j = z$. Si $\hat{f} < \infty$, dans ce cas on suppose que le point initial z vérifie la condition $f(z) \neq \hat{f}$.

Remarque 4.2. *Lorsque la matrice D est définie négative (f est strictement concave), on a $\det(D) \neq 0$, donc $\hat{x} = -D^{-1}c$ est la solution optimale unique du problème (4.10) et $\hat{f} = f(\hat{x}) < \infty$. Si la fonction f n'est pas strictement concave, alors au moins l'une des valeurs propres de la matrice D est nulle, donc $\det(D) = 0$ et $rang(D) < n$. Par conséquent, deux cas se présentent :*

— ***Cas 1.*** *Le vecteur $-c$ appartient au sous-espace vectoriel engendré par les vecteurs-colonnes de la matrice D, i.e., le système $\nabla f(x) = Dx + c = 0$ a une infinité de solutions. Dans ce cas, la function f possède un nombre infini de points critiques, qui sont des solutions du problème (4.10) et la valeur de la fonction objectif en ces points est égale à $\hat{f} < \infty$.*

— ***Cas 2.*** *Le vecteur $-c$ n'appartient pas au sous-espace vectoriel engendré par les vecteurs-colonnes de la matrice D, i.e., le système $\nabla f(x) = Dx + c = 0$ n'a pas de solutions. Dans ce cas, on a $\hat{f} = \infty$.*

Maintenant supposons que R_z est déjà calculé. Pour tout $y^j \in R_z$, on résout le programme linéaire suivant [34, 31, 8] :

$$\min_{x \in S} \; x^T \nabla f(y^j). \tag{4.11}$$

Notons par $u^j,\ j \in J$, les points extrêmes correspondant aux solutions du programme linéaire (4.11).

Supposons que $\hat{f} < \infty$. On a alors le lemme suivant :

Lemma 4.2. *Si $(u^j - \hat{x})^T D(u^j - \hat{x}) \neq 0$, alors les points*

$$\bar{y}^j = \hat{x} + \alpha_j(u^j - \hat{x}) \in E_{f(z)}(f), \tag{4.12}$$

où

$$\alpha_j = \left(\frac{2(f(z) - f(\hat{x}))}{(u^j - \hat{x})^T D(u^j - \hat{x})} \right)^{\frac{1}{2}} \tag{4.13}$$

et les vecteurs u^j sont des solutions optimales des programmes linéaires (4.11).

Preuve. Pour $(u^j - \hat{x})^T D(u^j - \hat{x}) \neq 0$, on a

$$\begin{aligned} f(\bar{y}^j) = f(\hat{x} + \alpha_j(u^j - \hat{x})) &= c^T(\hat{x} + \alpha_j(u^j - \hat{x})) + \frac{1}{2}(\hat{x} + \alpha_i(u^j - \hat{x}))^T D(\hat{x} + \alpha_j(u^j - \hat{x})) \\ &= \alpha_j(u^j - \hat{x})^T(D\hat{x} + c) + \frac{1}{2}\alpha_j^2(u^j - \hat{x})^T D(u^j - \hat{x}) + f(\hat{x}) \\ &= \frac{1}{2}\alpha_j^2(u^j - \hat{x})^T D(u^j - \hat{x}) + f(\hat{x}) \\ &= f(z) - f(\hat{x}) + f(\hat{x}) \\ &= f(z). \end{aligned}$$

Par conséquent, $\bar{y}^j \in E_{f(z)}(f)$. □

Le lemme précédent nous permet de construire l'ensemble d'approximation du second ordre.

Definition 4.1. *Soit r_2 un entier positif tel que $r_2 = r_1 - s$, avec s égal au nombre de vecteurs u^j qui vérifient $(u^j - \hat{x})^T D(u^j - \hat{x}) = 0$. Alors, l'ensemble*

$$\bar{R}_z = \{\bar{y}^j, j = 1, 2 \ldots, r_2\},$$

où $\bar{y}^j$ est défini par (4.12)-(4.13), est appelé un ensemble d'approximation du second ordre.

Dans [34, 31], la focntion quadratique f est supposée être strictement concave, i.e., le dénominateur des nombres α_j est supposé être strictement négatif. Dans notre algorithme, on travaille avec une matrice semi-définie négative D, donc le dénominateur peut s'annuler. Afin de prendre en considération ce cas, l'ensemble $\bar{R}_z$ est construit uniquement avec des points $\bar{y}^j$ vérifiant $(u^j - \hat{x})^T D(u^j - \hat{x}) \neq 0$. Si $\hat{f} = \infty$, on pose $\bar{R}_z = \emptyset$. Sinon, on construit l'ensemble d'approximation du second ordre avec la pocédure suivante :

Algorithme 4.2 : Construction de l'ensemble d'approximation du second ordre

```
1 Poser R̄_z = ∅, J̄ = ∅ et r_2 = 0;
2 tant que j ∈ J faire
3 |   Calculer d_j = (u^j − x̂)^T D(u^j − x̂);
4 |   si d_j ≠ 0 alors
5 |   |   Calculer α_j avec (4.13) et ȳ^j = x̂ + α_j(u^j − x̂);
6 |   |   Poser R̄_z = R̄_z ∪ {ȳ^j}, J̄ = J̄ ∪ {j} et r_2 = r_2 + 1;
  |   fin
  fin
```

En remplaçant l'ensemble $E_{f(z)}(f)$ par l'ensemble d'approximation $R_z \cup \bar{R}_z$ dans la condition d'optimalité globale (4.6), on peut donner la définition suivante pour une solution globale approchée :

Definition 4.2. *[93] Une solution réalisable z est appelée une solution globale approchée si*

$$(x - y)^T \nabla f(y) \geq 0, \forall y \in R_z \cup \bar{R}_z \text{ et } x \in S. \tag{4.14}$$

Remarque 4.3. *Notez que le concept de solution globale approchée utilisé dans cette thèse est différent du concept de solution globale connu en optimisation globale. Cependant,*

lorsqu'un bon ensemble d'approximation est construit, la solution globale approchée peut être une solution ε-globale.

Afin d'améliorer la solution réalisable courante z, nous calculons les nombres réels $\theta_j,\ j \in J$ et θ_p comme suit :

$$\theta_j = (u^j - y^j)^T \nabla f(y^j) \text{ et } \theta_p = \min\{\theta_j,\ j \in J\}. \tag{4.15}$$

Remarquons que si $\theta_p = (u^p - y^p)\nabla f(y^p) < 0$, alors $f(u^p) < f(z)$. En effet, comme $y^p \in E_{f(z)}(f)$ et f est concave, on a

$$f(u^p) - f(z) = f(u^p) - f(y^p) \leq (u^p - y^p)\nabla f(y^p) < 0 \Rightarrow f(u^p) < f(z).$$

Dans ce cas, on recommence une nouvelle iteration avec $z = u^p$. Sinon, deux cas peuvent se présenter :

Cas 1. Si $\hat{f} = \infty$, alors on arrête l' algorithme avec z une solution optimale globale approchée.

Cas 2. Si $\hat{f} < \infty$, alors pour tout $\bar{y}^j \in \bar{R}_z$, on détermine le vecteur v^j et les nombres $\bar{\theta}_j$, $\bar{\theta}_q$ comme suit :

$$v^{j^T} \nabla f(\bar{y}^j) = \min_{x \in S} x^T \nabla f(\bar{y}^j), \tag{4.16}$$

$$\bar{\theta}_j = (v^j - y^j)^T \nabla f(\bar{y}^j) \text{ et } \bar{\theta}_q = \min\{\bar{\theta}_j,\ j \in \bar{J}\}. \tag{4.17}$$

Si $\bar{\theta}_q < 0$, alors $f(v^q) < f(z)$; donc on recommence une nouvelle itération avec $z = v^q$. Sinon, on arrête l'algorithme avec $x^* = v^q$ une solution optimale globale approchée pour le problème (4.1)-(4.4).

4.4 Algorithme

Soit z^0 la solution du programme linéaire $\min_{x \in S} c^T x$; $\hat{f}$ le maximum de la function f sur $\mathbb{R}^n$ et $\hat{x}$ une solution optimale du problème (4.10) si elle existe ; $\nabla f(z^0)$ le gradient de f au point z^0. Supposons que $f(z^0) \neq \hat{f}$. Les différentes étapes de l'algorithme des approximations linéaires successives pour la résolution du programme quadratique concave

(4.1)-(4.4) sont résumées dans l'algorithme suivant :

Algorithme 4.3 : AALS

Entrées : Les matrices A_1, A_2 et D, les vecteurs b_1, b_2, l, u et c;

Sortie : Une solution globale approchée z^* du problème (4.1)-(4.4);

1 Poser $r = n$, $k = 0$ et $Stop = 0$;
2 **tant que** $Stop=0$ **faire**
3 Construire l'ensemble R_{z^k} avec l'algorithme 4.1;
4 **tant que** $y^j \in R_{z^k}$ **faire**
5 calculer $u^j \in \arg\min_{x \in S} x^T \nabla f(y^j)$;
6 Déterminer l'indice p et le nombre θ_p avec (4.15);
7 **si** $\theta_p < 0$ **alors**
8 poser $k := k + 1,\ z^k = u^p$ et aller à la ligne 3;
sinon
9 **si** $\hat{f} < \infty$ **alors**
10 construire l'ensemble $\bar{R}_{z^k}$ avec l'algorithme 4.2;
sinon
11 le point z^k est une solution optimale globale apprchée pour le problème (4.1)-(4.4), $Stop = 1$;
12 aller à la ligne 2;
fin
fin
13 **tant que** $\bar{y}^j \in \bar{R}_{z^k}$ **faire**
14 calculer $v^j \in \arg\min_{x \in S} x^T \nabla f(\bar{y}^j)$;
15 Déterminer l'indice q et le nombre $\bar{\theta}_q$ avec (4.17);
16 **si** $\bar{\theta}_q < 0$ **alors**
17 poser $k := k + 1,\ z^k = v^q$ et aller à la ligne 3;
sinon
18 le point $z^* = z^k$ est une solution globale approchée pour le problème (4.1)-(4.4), $Stop = 1$;
fin
fin
fin
fin

Remarque 4.4.

AALS peut être initialisé par une solution réalisable arbitraire z^0, telle que $f(z^0) \neq \hat{f}$. Cependant, les résultats numériques obtenus dans ce travail montrent qu'il est judicieux de choisir z^0 égal à la solution optimale du programme linéaire $\min_{x \in S} c^T x$. Lorsque $c = 0$, on génère aléatoirement un vecteur $\tilde{c}$, puis on initialise notre algorithme avec la solution du PL $\min_{x \in S} \tilde{c}^T x$.

Notons que différents choix des vecteurs h^j peuvent aboutir à de différentes solutions globales approchées. C'est pourquoi, nous effectuons plusieurs exécutions de AALS, et ce, dans le but d'atteindre des solutions globales approchées d'une meilleure qualité. Notons par NE l'entier positif qui désigne le nombre d'exécutions de AALS et e_j le $j$ème vecteur de la matrice identité d'ordre n. Afin de trouver une solution globale approchée améliorée, on propose l'algorithme appelé "Algorithme des Approximations Linéaires Successives Stochastique (AALSS)" :

Algorithme 4.4 : AALSS

Entrées : Les matrices A_1, A_2 et D, les vecteurs b_1, b_2, l, u et c, le nombre d'exécutions NE;

Sortie : Une solution globale approchée x^* du problème (3.1)-(3.4);

Poser $k = 1$;

tant que $k \leq NE + 1$ **faire**

  si $k = 1$ **alors**

    appliquer AALS (Algorithme 4.3) avec $h^j = e_j,\ j = 1, 2, \ldots, n$;

    Soit x^1 la solution globale approchée et $f^1 = f(x^1)$;

  sinon

    appliquer AALS avec des composantes des vecteurs h^j générées aléatoirement avec la distribution uniforme dans l'intervalle $[-1, 1]$;

    Soit x^k la solution globale approchée obtenue et $f^k = f(x^k)$;

  fin

fin

Calculer la solution globale approchée améliorée $\bar{x}$, telle que $f(\bar{x}) = \min\{f^k,\ k = 1, 2, \ldots, NE + 1\}$;

Appliquez la méthode du gradient conditionnel pour atteindre un point stationnaire x^* en commençant par le point $\bar{x}$.

4.5 Convergence de l'algorithme

Theorème 4.3. *Chaque exécution i de AALSS (Algorithme 4.4 décrit ci-dessus) trouve un minimum global approché $\bar{x}$ en un nombre fini d'itérations. De plus, la linge 9 de l'algorithme 4.4 s'arrête avec un point stationnaire x^* qui vérifie $f(x^*) \leq \min\{f(x^i) : i = 1, 2, \ldots, NE + 1\}$.*

Preuve. Dans la $k^{\text{ième}}$ itération de la $i^{\text{ème}}$ exécution, on note z^{ik} le point extrême courant. Alors deux cas peuvent se présenter :

Cas 1 il existe un indice $p \in J$, tel que $\theta_p = (u^p - y^p)^T \nabla f(y^p) < 0$. Puisque f est concave et $y^p \in E_{f(z^{ik})}(f)$, on a

$$f(u^p) - f(z^{ik}) = f(u^p) - f(y^p) \leq (u^p - y^p)^T \nabla f(y^p) < 0 \Rightarrow f(z^{ik+1}) < f(z^{ik}), z^{ik+1} = u^p.$$

Cas 2 $\theta_p \geq 0, \hat{f} < \infty$ et il existe $q \in \bar{J}$, tel que $\bar{\theta}_q = (v_q - \bar{y}^q)^T \nabla f(\bar{y}^q) < 0$. Puisque f est concave et $y^p \in E_{f(z^{ik})}(f)$, on a

$$f(v^q) - f(z^{ik}) = f(v^q) - f(\bar{y}^q) \leq (v^q - \bar{y}^q)^T \nabla f(\bar{y}^q) < 0 \Rightarrow f(z^{ik+1}) < f(z^{ik}), z^{ik+1} = v^q.$$

Dans tous les cas, on recommence une nouvelle itération avec un nouveau point extrême z^{ik+1}, qui améliore strictement la valeur de la fonction objectif, i.e., $f(z^{ik+1}) < f(z^{ik})$. Puisque le nombre de points extrêmes de l'ensemble polyédrique compact S est fini, la $i^{\text{ème}}$ exécution de l'algorithme AALSS s'arrête en un nombre fini d'itérations avec un point x^i qui vérifie : $\theta_p \geq 0$. Par conséquent,

$$\forall y^j \in R_{x^i}, \forall x \in S, (x - y^j)^T \nabla f(y^j) \geq (u^j - y^j)^T \nabla f(y^j) = \theta_j \geq \theta_p \geq 0.$$

De plus, si $\hat{f} = \infty$, alors $\bar{R}_z = \emptyset$. Donc

$$\forall y \in R_{x^i} \cup \bar{R}_{x^i}, \forall x \in S, (x - y)^T \nabla f(y) \geq 0.$$

Par conséquent, x^i est un minimum global approché.
Si $\hat{f} < \infty$, alors x^i vérifie également la condition $\bar{\theta}_q \geq 0$. Donc

$$\forall \bar{y}^j \in R_{x^i}, \forall x \in S, (x - \bar{y}^j)^T \nabla f(\bar{y}^j) \geq (v^j - \bar{y}^j)^T \nabla f(\bar{y}^j) = \bar{\theta}_j \geq \bar{\theta}_q \geq 0.$$

D'où

$$\forall y \in R_{x^i} \cup \bar{R}_{x^i}, \forall x \in S, (x-y)^T \nabla f(y) \geq 0.$$

Par conséquent, x^i est un minimum global approché.

Enfin, on applique la méthode du gradient conditionnel en commençant par le point extrême initial $\bar{x}$ qui vérifie $f(\bar{x}) = \min\{f(x^i) : i = 1, 2, \ldots, NE+1\}$. Si $\bar{x}$ est un point stationnaire, alors la méthode du gradient conditionnel se termine en une itération avec $x^* = \bar{x}$ et $f(x^*) = f(\bar{x})$. Cependant, si $\bar{x}$ n'est pas un point stationnaire, alors la méthode du gradient conditionnel construit une suite de points extrêmes $\{x^k\}$, telle que $f(x^{k+1}) < f(x^k)$, qui converge vers un point stationnaire x^*, avec $f(x^*) < f(\bar{x})$. □

Remarque 4.5. *Dans [34, 31, 35, 8], un algorithme local est appliqué à chaque itération pour obtenir un minimum local. Notre algorithme commence par un point extrême initial qui est une solution du programme linéaire* $\min_{x \in S} c^T x$ *et il se déplace, à chaque itération, d'un point extrême à un autre meilleur, qui n'est pas nécessairement un minimum local. Cependant, pour atteindre un point stationnaire, nous appliquons la méthode du gradient conditionnel une fois que toutes les exécutions sont effectuées. Cela économise beaucoup de temps de calcul, car appliquer un algorithme local à chaque itération et à chaque exécution peut affecter les performances de l'algorithme global.*

4.6 Exemples numériques

Exemple 4.1. *Considérons le programme quadratique concave suivant :*

$$\begin{aligned}
&min && f(x) = 2x_1 - x_3 - 2x_1^2 - 2x_3^2 + 2x_1x_3,\\
&s.c. && x_1 + 3x_2 + x_3 \leq 2,\\
& && 3x_1 - x_2 + x_3 \leq 0,\\
& && 2x_1 - x_2 - x_3 = 0,\\
& && 0 \leq x_1 \leq +\infty,\ 0 \leq x_2 \leq 1,\ -\infty \leq x_3 \leq 1.
\end{aligned}$$

On a

$$A_1 = \begin{pmatrix} 1 & 3 & 1 \\ 3 & -1 & 1 \end{pmatrix},\ b_1 = \begin{pmatrix} 2 \\ 0 \end{pmatrix},\ A_2 = \begin{pmatrix} 2 & -1 & -1 \end{pmatrix},\ b_2 = 0,$$

$$l = \begin{pmatrix} 0 \\ 0 \\ -\infty \end{pmatrix},\ u = \begin{pmatrix} +\infty \\ 1 \\ 1 \end{pmatrix},\ c = \begin{pmatrix} 2 \\ 0 \\ -1 \end{pmatrix},\ D = \begin{pmatrix} -4 & 0 & 2 \\ 0 & 0 & 0 \\ 2 & 0 & -4 \end{pmatrix}.$$

Premièrement, on calcule le maximum de f sur $\mathbb{R}^3$. Comme les valeurs propres de la matrice D sont respectivement -6, -2 et 0, donc D est semi-définie négative et $\det(D) = 0$. *De plus, nous avons $rang(D) = rang(D| - c) = 2$, donc le vecteur $-c$ appartient au sous-espace vectoriel engendré par les colonnes de D. Par conséquent, le système linéaire $\nabla f(x) = Dx + c = 0$ possède une infinité de solutions, qui sont de la forme $(\frac{1}{2}, \beta, 0),\ \beta \in \mathbb{R}$. Nous pouvons prendre par exemple le point $\hat{x} = (\frac{1}{2}, 0, 0)$. La fonction f a une infinité de points critiques avec une valeur de la fonction objectif égale à $\hat{f} = f(\hat{x}) = \frac{1}{2}$.*

La solution du programme linéaire $\min_{x \in S} c^T x$ *est*

$$z^0 = (0, 0, 0)^T,\ f(z^0) = 0.$$

Appliquons l'algorithme 4.4 avec NE = 0 (nous effectuons une exécution de AALSS avec $h^j = e_j$).

Posons $r = n = 3$ et $k = 0$.

Itération 1 :

Le gradient de la fonction f au point initial z^0 est

$$\nabla f(z^0) = Dz^0 + c = (2, 0, -1)^T.$$

Construisons l'ensemble d'approximation du premier ordre R_{z^0} :

Pour $j = 1, 2, 3$, on pose $h^j = e_j \in \mathbb{R}^3$. On a

$$h^{1T} D h^1 = -4,\ h^{2T} D h^2 = 0,\ h^{3T} D h^3 = -4 \Rightarrow R_{z^0} = \{y^1, y^3\},$$

où

$$\gamma_1 = -\frac{2h^{1T} \nabla f(z^0)}{h^{1T} D h^1} = 1,\ \gamma_3 = -\frac{2h^{3T} \nabla f(z^0)}{h^{3T} D h^3} = -1/2,$$

$$y^1 = z^0 + \gamma_1 h^1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix},\ y^3 = z^0 + \gamma_3 h^3 \begin{pmatrix} 0 \\ 0 \\ -1/2 \end{pmatrix}.$$

On a

$$\nabla f(y^1) = (-2, 0, 1)^T,\ \nabla f(y^3) = (1, 0, 1)^T.$$

Les solutions optimales des PL $\min_{x\in S} x^T\nabla f(y^j)$, $j=1,3$ *sont*

$$u^1 = u^3 = (0,1,-1)^T.$$

Le nombre θ_p *est donné par*

$$\theta_p = \min\{\theta_1,\theta_3\} = \min\{(u^j-y^j)^T\nabla f(y^j),\ j=1,3\} = \min\{1,-\frac{1}{2}\} = -\frac{1}{2}.$$

On a $\theta_p = \theta_3 < 0$, *donc le nouveau point extrême est*

$$z^1 = u^3 = (0,1,-1)^T,\ f(z^1) = -1 < f(z^0) = 0.$$

Itération 2 :

Le gradient de la fonction f *au point* z^1 *est*

$$\nabla f(z^1) = Dz^1 + c = (0,0,3)^T.$$

Posons $h^j = e_j,\ j=1,2,3$ *et construisons les éléments de l'ensemble* R_{z^1} *:*

$$\gamma_1 = 0,\ \gamma_3 = \frac{3}{2},\ y^1 = \begin{pmatrix} 0 \\ 1 \\ -1 \end{pmatrix},\ y^3 = \begin{pmatrix} 0 \\ 1 \\ 1/2 \end{pmatrix},\ R_{z^1} = \{y^1, y^3\}.$$

On a

$$\nabla f(y^1) = (0,0,3)^T,\ \nabla f(y^3) = (3,0,-3)^T.$$

Les solutions optimales des programmes linéaires $\min_{x\in S} x^T\nabla f(y^j),\ j=1,3$ *sont*

$$u^1 = (0,1,-1)^T,\ u^3 = (0,0,0)^T.$$

Le nombre θ_p *est calculé comme suit :*

$$\theta_p = \min\{\theta_1,\theta_3\} = \min\{0,\frac{3}{2}\} = 0.$$

On a $\theta_p = \theta_1 = 0$ *et* $\hat{f} < \infty$, *donc on construit l'ensemble d'approximation du second ordre* $\bar{R}_{z^1}$ *:*

$$(u^1-\hat{x})^T D(u^1-\hat{x}) = -3,\ (u^3-\hat{x})^T D(u^3-\hat{x}) = -1,$$

$$\alpha_1 = \left(\frac{2(f(z^1)-f(\hat{x}))}{(u^1-\hat{x})^T D(u^1-\hat{x})}\right)^{\frac{1}{2}} = 1,\ \alpha_3 = \left(\frac{2(f(z^1)-f(\hat{x}))}{(u^3-\hat{x})^T D(u^3-\hat{x})}\right)^{\frac{1}{2}} = \sqrt{3}.$$

Donc

$$\bar{y}^1 = \begin{pmatrix} 0 \\ 1 \\ -1 \end{pmatrix},\ \bar{y}^3 = \begin{pmatrix} \frac{1-\sqrt{3}}{2} \\ 0 \\ 0 \end{pmatrix},\ \bar{R}_{z^1} = \{\bar{y}^1, \bar{y}^3\}.$$

Nous avons

$$\nabla f(\bar{y}^1) = (0,0,3)^T,\ \nabla f(\bar{y}^3) = (2\sqrt{3}, 0, -\sqrt{3})^T.$$

Les solutions des programmes linéaires $\min_{x\in S} x^T \nabla f(\bar{y}^j),\ j = 1,3$ *sont*

$$v^1 = (0,1,-1)^T,\ v^3 = (0,0,0)^T.$$

Le nombre $\bar{\theta}_q$ *est*

$$\bar{\theta}_q = \min\{\bar{\theta}_1, \bar{\theta}_3\} = \min\{(v^j - \bar{y}^j)^T \nabla f(\bar{y}^j),\ j = 1,3\} = \min\{0, 3-\sqrt{3}\} = 0.$$

Par conséquent, la solution globale approchée et l'optimum global approché sont donnés par

$$x^* = z^1 = (0,1,-1)^T \text{ et } f(x^*) = f(z^1) = -1.$$

La première itération de la méthode du gradient conditionnel n'a pas amélioré le point x^**, il est donc un point stationnaire.*

Remarque 4.6. *Pour cet exemple, nous avons trouvé le même minimum global que celui trouvé par la méthode d'énumération des points extrêmes et celle du solveur CPLEX12.8. Notons que les algorithmes proposés dans [34, 31, 8] pour la résolution des programmes quadratiques concaves ne peuvent pas être appliquées pour la résolution du problème de cet exemple car la fonction* f *n'est pas strictement concave, i.e.,la matrice* D *est semi-définie négative.*

Exemple 4.2. *Considérons le programme quadratique concave à variable bornées (Chinchuluun et al. 2005) :*

$$\begin{aligned} \min \quad & f(x) = -x_1^2 - x_2^2 - x_3^2 - (x_3 - x_4)^2, \\ s.c. \quad & -23/10 \le x_i \le 27/10,\ j = 1,2,3,4. \end{aligned}$$

On a

$$l = \begin{pmatrix} -23/10 \\ -23/10 \\ -23/10 \\ -23/10 \end{pmatrix},\ u = \begin{pmatrix} 27/10 \\ 27/10 \\ 27/10 \\ 27/10 \end{pmatrix},\ c = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix},\ D = \begin{pmatrix} -2 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 \\ 0 & 0 & -4 & 2 \\ 0 & 0 & 2 & -2 \end{pmatrix}.$$

Dans cet exemple, la matrice D est définie négative (f est strictement concave), donc le maximum de la fonction f sur $\mathbb{R}^4$ est donné par

$$\hat{x} = -D^{-1}c = (0,0,0,0)^T,\ \hat{f} = f(\hat{x}) = 0.$$

La solution optimale du programme linéaire $\min_{x\in S} c^T x$ est

$$z^0 = (-23/10, -23/10, -23/10, -23/10)^T,\ f(z^0) = -\frac{1587}{100}.$$

Appliquons AALSS avec $NE = 0$.

Posons $r = n = 4$ et $k = 0$.

Itération 1 :

Le gradient de la fonction f au point z^0 est

$$\nabla f(z^0) = Dz^0 + c = (23/5, 23/5, 23/5, 0)^T.$$

Construction de l'ensemble d'approximation du premier ordre R_{z^0} :

Pour $j = 1,2,3,4$, on pose $h^j = e_j \in \mathbb{R}^4$. On a

$$h^{1^T} Dh^1 = -2,\ h^{2^T} Dh^2 = -2,\ h^{3^T} Dh^3 = -4,\ h^{4^T} Dh^4 = -2 \Rightarrow R_{z^0} = \{y^1, y^2, y^3, y^4\}.$$

Les vecteurs $\gamma = (\gamma_j,\ j = 1,2,3,4)$, y^j, $j = 1,2,3,4$ sont

$$\gamma = (23/5, 23/5, 23/10, 0)^T,$$

$$y^1 = (23/10, -23/10, -23/10, -23/10)^T,\ y^2 = (-23/10, 23/10, -23/10, -23/10)^T,$$

$$y^3 = (-23/10, -23/10, 0, -23/10)^T,\ y^4 = (-23/10, -23/10, -23/10, -23/10).$$

Les vecteurs gradients $\nabla f(y^j)$ sont

$$\nabla f(y^1) = (-23/5, 23/5, 23/5, 0)^T,\ \nabla f(y^2) = (23/5, -23/5, 23/5, 0)^T,$$

$$\nabla f(y^3) = (23/5, 23/5, -23/5, 23/5)^T,\ \nabla f(y^4) = (23/5, 23/5, 23/5, 0).$$

Les solutions optimales des programmes linéaires $\min_{x\in S} x^T \nabla f(y^j)$, $j = 1,2,3,4$ sont

$$u^1 = (27/10, -23/10, -23/10, -23/10)^T,\ u^2 = (-23/10, 27/10, -23/10, -23/10)^T,$$

$$u^3 = (-23/10, -23/10, 27/10, -23/10)^T,\ u^4 = (-23/10, -23/10, -23/10, -23/10)^T.$$

Le nombre θ_p est calculé comme suit :

$$\theta_p = \min\{\theta_j, j = 1, 2, 3, 4\} = \min\{-\frac{46}{25}, -\frac{46}{25}, -\frac{621}{50}, 0\} = -\frac{621}{50}.$$

On a $\theta_p = \theta_3 < 0$, donc le nouveau point extrême est

$$z^1 = u^3 = (-23/10, -23/10, 27/10, -23/10)^T,\ f(z^1) = -\frac{4287}{100} < f(z^0) = -\frac{1587}{100}.$$

À la quatrième itération, la solution courante est

$$z^3 = (27/10, 27/10, 27/10, -23/10)^T,\ f(z^3) = -\frac{4687}{100}.$$

Les vecteurs $u^j,\ j = 1, 2, 3, 4$ sont

$$u^1 = (-23/10, 27/10, 27/10, -23/10)^T,\ u^2 = (27/10, -23/10, 27/10, -23/10)^T,$$

$$u^3 = u^4 = (27/10, 27/10, -23/10, 27/10)^T.$$

On a $\theta_p = \frac{54}{25} > 0$ et $\hat{f} < \infty$. Donc

$$v^1 = (-23/10, 27/10, 27/10, -23/10)^T,\ v^2 = (27/10, -23/10, 27/10, -23/10)^T,$$

$$v^3 = v^4 = (27/10, 27/10, -23/10, 27/10)^T \text{ et } \bar{\theta}_q = \frac{2040}{1009} > 0.$$

Par conséquent, la solution globale approchée et le minimum global approché sont donnés par :

$$x^* = z^3 = (27/10, 27/10, 27/10, -23/10)^T \text{ and } f(x^*) = -46.87.$$

La première itération de l'algorithme du gradient conditionel n'a pas amélioré le point x^*, il est donc un point stationnaire.

Remarque 4.7. *Pour cet exemple, nous avons trouvé le même minimum global que celui obtenu par la méthode d'énumération des sommets et celui obtenu par l'algorithme de branch-and-bound implémenté dans CPLEX12.8 [59]. Cependant, le minimum global approché trouvé par l'algorithme proposé dans [31] est -39.58.*

Chapitre 5

Résultats expérimentaux

5.1 Introduction

L'évolution des outils informatiques a profondément influencé les méthodes de travail des ingénieurs et chercheurs sans oublier l'enseignement. Le traitement numérique des données, leur visualisation, ainsi que les techniques de modélisation et de simulation se sont notamment généralisés. Afin de comparer notre algorithme avec les algorithmes existants pour la résolution des problèmes de programmation quadratique concave : l'Algorithme des Ensembles d'Approximation (AEA) [31], l'algorithme de branch-and-bound de Rusakov (ABBR) [88], l'algorithme de branch-and-bound de CPLEX12.8 (CPLEX) [59], nous avons implémenté notre algorithme (AALSS) avec le langage de programmation Matlab2018b [72].

5.2 Les problèmes-test

Dans l'étude expérimentale que nous avons effectuée, nous avons comparé les différents algorithmes sur 156 problèmes-test divisés en plusieurs classes :

A. Douze programmes quadratiques concaves avec des variables bornées (box-constrained concave quadratic programs) [31] :

• Six problèmes-test de dimension $n = 10, 50, 100, 200, 500, 1000$, où n est le nombre de variables du problème :

Problème 1

$$\begin{aligned} \min \quad & f(x) = -\sum_{i=1}^{n}(n-1-0.1i)x_i^2, \\ s.c. \quad & -1-i \leq x_i \leq 1+5i,\ i=1,2,\ldots,n. \end{aligned}$$

• Six problèmes-test de dimension $n = 10, 50, 100, 200, 500, 1000$, où n est le nombre de variables du problème :

Problème 2

$$\begin{aligned} \min \quad & f(x) = -\left\|x\right\|^2, \\ s.c. \quad & -(n-i+1) \leq x_i \leq n+0.5i,\ i=1,2,\ldots,n. \end{aligned}$$

Les caractéristiques de ces problèmes-test sont résumées dans la table 5.1, où f^* désigne l'optimum global connu du problème-test.

TABLE 5.1 – Caractéristiques des problèmes-test de norme [31]

Nom	Problème	n	f^*	$f(AEA)$
norm-qp1	1	10	-83712	-83712
norm-qp2	1	50	-49103210	-49103210
norm-qp3	1	100	-778330545	-778330545
norm-qp4	1	200	-12393657590	-12393657590
norm-qp5	1	500	-482716617725	-482716617725
norm-qp6	1	1000	-7715908147950	-7715908147950
norm-qp7	2	10	-1646,25	-1636
norm-qp8	2	50	-199481,25	-199431
norm-qp9	2	100	-1589587,5	-1578020,25
norm-qp10	2	200	-12691675	-11460835
norm-qp11	2	500	-198072937,5	-178801974
norm-qp12	2	1000	-1583958375	-1429690604

Notons que pour l'algorithme AEA, de nombreux détails d'implémentation, qui permettent le développement de son code, ne sont pas donnés dans [31], comme le nombre d'éléments de l'ensemble d'approximation du premier ordre (r), la manière de choisir le point initial et la manière de calculer les vecteurs h^j pour vèrifier les inègalités ${h^j}^T \nabla f(z) > 0$ et $z + \gamma_j h^j \in S$. Par conséquent, nous n'avons rapporté que les valeurs

globales approchées obtenues par AEA [31] dans la dernière colonne du tableau 5.1 ; puis nous avons calculé l'erreur Δf et nous l'avons rapportée dans la table 5.6.

B. Les problèmes-test dits "SPIN-GLASS" [84] : ces modèles permettent de formuler un problème issu de la physique statistique. Le modèle mathématique discret de ce problème est donné par [23, 62] :

$$\min\left\{\sum_{i=1}^{n} g_{ij}x_ix_j : x_i \in \{-1,1\}\right\}, \tag{5.1}$$

où $x = (x_1, x_2, \ldots, x_n)^T$ représente le vecteur des états des spin, $G = (g_{ij}, i, j = 1, 2, \ldots, n)$ est la matrice des interactions. Pour résoudre le problème (5.1), il est écrit d'abord sous la forme

$$\min\{x^T(G - 4I_n)x + 4 : x \in \{-1,1\}^n\}. \tag{5.2}$$

Ce problème est équivalent au programme quadratique concave suivant :

Problème 3

$$\min \qquad f(x) = x^T(G - 4I_n)x + 4, \tag{5.3}$$

$$s.c. \qquad -1 \leq x_i \leq 1,\ i = 1, 2, \ldots, n. \tag{5.4}$$

On considère cinq problèmes-test de dimension $n = 20, 40, 60, 80, 100$, où n est le nombre de variables du problème 3. Les caractéristiques de ces problèmes-test sont résumées dans la table 5.2.

C. Les problèmes-test dits "Difficult Class of Test Quadratic Problems (DCTQP)" [81] : ces problèmes sont écrits sous la forme :

Problème 4

$$\min \qquad f(x) = -n(n-1)\sum_{i=1}^{n} x_i - \sum_{i=1}^{n/2} x_i + n\sum_{i<j} x_ix_j + n\sum_{i>j} x_ix_j, \tag{5.5}$$

$$s.c. \qquad x_i \in \{0,1\},\ i = 1, 2, \ldots, n. \tag{5.6}$$

Dans [81], les auteurs ont considéré ces problèmes discrets comme étant des problèmes difficiles, vu leurs nombre exponentiel de minimum locaux discrets. Pour n choisi, la solution globale du problème 4 est le point $x = (1, \ldots, 1, 0, \ldots, 0)^T$ avec exactement $n/2$ uns, suivis de $n/2$ zéros. Pour résoudre ces problèmes, ils sont écrits sous la forme (4.1)-(4.4) où D, c, l et u sont définis comme suit :

$$D = -2n^2I_n + 2nee^T;\ l_i = 0,\ u_i = 1,\ i = 1, 2, \ldots, n;$$

$$c_i = -1,\ i = 1, \ldots, n/2,\ c_i = 0,\ i = (n/2) + 1, \ldots, n.$$

Nous pouvons facilement voir que la plus grande valeur propre de D est nulle. Ainsi D est une matrice semi-définie négative.

Nous avons considéré six problèmes-test de dimension $n = 4, 8, 12, 16, 20, 24$, où n est le nombre de variables du problème 4. Les caractéristiques de ces problèmes-test sont résumées dans la table 5.2.

TABLE 5.2 – Caractéristiques des problèmes-test SPIN-GLASS [84] et DCTQP [81]

Nom	Problème	n	f^*	Nom	Problème	n	f^*
SpinGlass-qp1	3	20	-120	dctqp1	4	4	-18
SpinGlass-qp2	3	40	-240	dctqp2	4	8	-132
SpinGlass-qp3	3	60	-360	dctqp3	4	12	-438
SpinGlass-qp4	3	80	-480	dctqp4	04	16	-1032
SpinGlass-qp5	3	100	-600	dctqp5	4	20	-2010
				dctqp6	4	24	-3468

D. Quatres problèmes-test pris de l'article de Rusakov [88] :$n = 5, 10, 15, 20$, où n représente le nombre de variables du problème suivant :

Problème 5

$$\begin{aligned} \min \quad & f(x) = \sum_{i=1}^{n} -2x_i - \frac{i}{n}x_i^2, \\ s.c. \quad & \sum_{i=1}^{n} ix_i = n^2, \\ & 0 \leq x_i \leq i,\ i = 1, 2, \ldots, n. \end{aligned}$$

Les caractéristiques de ces problèmes-test sont résumées dans la table 5.3.

E. Neuf problèmes d'optimisation quadratique concave tirés du livre [42] :

Problème 6

$$\begin{aligned} \min \quad & f(x) = c^T x - 0.5x^T Dx, \\ s.c. \quad & 20x_1 + 12x_2 + 11x_3 + 7x_4 + 4x_5 \leq 40, \\ & 0 \leq x_j \leq 1,\ j = \overline{1, 5}, \end{aligned}$$

TABLE 5.3 – Caractéristiques des problèmes-test de Rusakov [88]

Nom	Problème	m	n	f^*
Rusakov-qp1	5	1	5	-35
Rusakov-qp2	5	1	10	-120
Rusakov-qp3	5	1	15	-255
Rusakov-qp4	5	1	20	-440

avec

$$c = (42, 44, 45, 47, 47.5)^T,\ D = 100I_5.$$

Problème 7

$$\begin{aligned} \min_{x,y} \quad & f(x,y) = c^T x - 0.5x^T Dx + dy, \\ s.c. \quad & 6x_1 + 3x_2 + 3x_3 + 2x_4 + x_5 \leq 6.5, \\ & 10x_1 + 10x_3 + y \leq 20, \\ & 0 \leq x_j \leq 1,\ j = \overline{1,5}, \\ & 0 \leq y, \end{aligned}$$

avec

$$c = (-10.5, -7.5, -3.5, -2.5, -1.5)^T,\ d = -10,\ D = 100I_5.$$

Problème 8

$$
\begin{aligned}
\min_{x,y} \quad & f(x,y) = c^T x - 0.5 x^T D x + dy, \\
s.c. \quad & 2x_1 + 2x_2 + y_6 + y_7 \leq 10, \\
& 2x_1 + 2x_3 + y_6 + y_8 \leq 10, \\
& 2x_2 + 2x_3 + y_7 + y_8 \leq 10, \\
& -8x_1 + y_6 \leq 0, \\
& -8x_2 + y_7 \leq 0, \\
& -8x_3 + y_8 \leq 0, \\
& -2x_4 - y_1 + y_6 \leq 0, \\
& -2y_2 - y_3 + y_7 \leq 0, \\
& -2y_4 - y_5 + y_8 \leq 0, \\
& 0 \leq x_j \leq 1, \ j = \overline{1,4}, \\
& 0 \leq y_j \leq 1, \ j = 1,2,3,4,5,9, \\
& 0 \leq y_j \leq 3, \ j = 6,7,8,
\end{aligned}
$$

avec

$$c = (-5,-5,-5,-5)^T, \ d_j = -10, \ j = \overline{1,9}, \ D = 100 I_4.$$

Problème 9

$$
\begin{aligned}
\min_{x,y} \quad & f(x,y) = 6.5x - 0.5x^2 - y_1 - 2y_2 - 3y_3 - 2y_4 - y_5, \\
s.c. \quad & Az \leq b, \ z = (x,y)^T, \\
& 0 \leq x \leq 1, \\
& y_j \leq 1, \ j = 3,4, \\
& y_5 \leq 2,
\end{aligned}
$$

avec

$$A = \begin{pmatrix} 1 & 2 & 8 & 1 & 3 & 5 \\ -8 & -4 & -2 & 2 & 4 & -1 \\ 2 & 0.5 & 0.2 & -3 & -1 & -4 \\ 0.2 & 2 & 0.1 & -4 & 2 & 2 \\ -0.1 & -0.5 & 2 & 5 & -5 & 3 \end{pmatrix}, \ b = \begin{pmatrix} 16 \\ -1 \\ 24 \\ 12 \\ 3 \end{pmatrix}.$$

Problème 10

$$\begin{array}{ll} \min\limits_{x,y} & f(x,y) = c^T x - 0.5 x^T D x + d^T y, \\ s.c. & Az \leq b,\ z = (x,y)^T, \\ & 0 \leq z \leq 1, \end{array}$$

avec

$$A = \begin{pmatrix} -2 & -6 & -1 & 0 & -3 & -3 & -2 & -6 & -2 & -2 \\ 6 & -5 & 8 & -3 & 0 & 1 & 3 & 8 & 9 & -3 \\ -5 & 6 & 5 & 3 & 8 & -8 & 9 & 2 & 0 & -9 \\ 9 & 5 & 0 & -9 & 1 & -8 & 3 & -9 & -9 & -3 \\ -8 & 7 & -4 & -5 & -9 & 1 & -7 & -1 & 3 & -2 \\ -7 & -5 & -2 & 0 & -6 & -6 & -7 & -6 & 7 & 7 \\ 1 & -3 & -3 & -4 & -1 & 0 & -4 & 1 & 6 & 0 \\ 1 & -2 & 6 & 9 & 0 & -7 & 9 & -9 & -6 & 4 \\ -4 & 6 & 7 & 2 & 2 & 0 & 6 & 6 & -7 & 4 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{pmatrix}, \ b = \begin{pmatrix} -4 \\ 22 \\ -6 \\ -23 \\ -12 \\ -3 \\ 1 \\ 12 \\ 15 \\ 9 \\ -1 \end{pmatrix},$$

$$c = (-20, -80, -20, -50, -60, 90, 0)^T,\ d = (10, 10, 10, 10)^T,\ D = 10 I_7.$$

Problème 11

$$\begin{array}{ll} \min & f(x) = c^T x - 0.5 x^T D x, \\ s.c. & Ax \leq b,\ 0 \leq x \leq 1,\ x \in \mathbb{R}^{10}, \end{array}$$

avec

$$A = \begin{pmatrix} -2 & -6 & -1 & 0 & -3 & -3 & -2 & -6 & -2 & -2 \\ 6 & -5 & 8 & -3 & 0 & 1 & 3 & 8 & 9 & -3 \\ -5 & 6 & 5 & 3 & 8 & -8 & 9 & 2 & 0 & -9 \\ 9 & 5 & 0 & -9 & 1 & -8 & 3 & -9 & -9 & -3 \\ -8 & 7 & -4 & -5 & -9 & 1 & -7 & -1 & 3 & -2 \end{pmatrix}, \ b = \begin{pmatrix} -4 \\ 22 \\ -6 \\ -23 \\ -12 \end{pmatrix},$$

$$c = (48, 42, 48, 45, 44, 41, 47, 42, 45, 46)^T,\ D = 100 I_{10}.$$

Considérons le programme quadratique concave suivant :

$$\begin{array}{ll} \min & f(x) = -0.5 \sum_{i=1}^{20} \lambda_i (x_i - \alpha_i)^2, \\ s.c. & Ax \leq b,\ x \geq 0,\ x \in \mathbb{R}^{20}, \end{array} \tag{5.7}$$

où

$$b = (-5, 2, -1, -3, 5, 4, -1, 0, 9, 40)^T,$$

$A = (A_1, A_2)$, avec

$$A_1 = \begin{pmatrix} -9 & 3 & 5 & 0 & 0 & 1 & 7 & -7 & -4 & -6 \\ 3 & 5 & 0 & 0 & 1 & 7 & -7 & -4 & -6 & -3 \\ 5 & 0 & 0 & 1 & 7 & -7 & -4 & -6 & -3 & 7 \\ 0 & 0 & 1 & 7 & -7 & -4 & -6 & -3 & 7 & 0 \\ 0 & 1 & 7 & -7 & -4 & -6 & -3 & 7 & 0 & -5 \\ 1 & 7 & -7 & -4 & -6 & -3 & 7 & 0 & -5 & 1 \\ 7 & -7 & -4 & -6 & -3 & 7 & 0 & -5 & 1 & 1 \\ -7 & -4 & -6 & -3 & 7 & 0 & -5 & 1 & 1 & 0 \\ -4 & -6 & -3 & 7 & 0 & -5 & 1 & 1 & 0 & 2 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix},$$

$$A_2 = \begin{pmatrix} -3 & 7 & 0 & -5 & 1 & 1 & 0 & 2 & -1 & -1 \\ 7 & 0 & -5 & 1 & 1 & 0 & 2 & -1 & -1 & -9 \\ 0 & -5 & 1 & 1 & 0 & 2 & -1 & -1 & -9 & 3 \\ -5 & 1 & 1 & 0 & 2 & -1 & -1 & -9 & 3 & 5 \\ 1 & 1 & 0 & 2 & -1 & -1 & -9 & 3 & 5 & 0 \\ 1 & 0 & 2 & -1 & -1 & -9 & 3 & 5 & 0 & 0 \\ 0 & 2 & -1 & -1 & -9 & 3 & 5 & 0 & 0 & 1 \\ 2 & -1 & -1 & -9 & 3 & 5 & 0 & 0 & 1 & 7 \\ -1 & -1 & -9 & 3 & 5 & 0 & 0 & 1 & 7 & -7 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Ce problème est équivalent au problème suivant :

Problème 12

$$\begin{array}{ll} \min & f(x) = c^T x - 0.5 x^T D x, \\ s.c. & Ax \leq b,\ x \geq 0,\ x \in \mathbb{R}^{20}, \end{array}$$

où $c = (\lambda_i \alpha_i,\ i = 1, \ldots, 20)$, $D = (d_{ij},\ i, j = 1, \ldots, 20)$, avec

$$d_{ij} = \begin{cases} -\lambda_i, & \text{si } i = j\,; \\ 0, & \text{sinon.} \end{cases}$$

Si l'on note l'optimum global du problème (5.7), donné dans [42], par $\bar{f}$ et celui du problème 12 par f^*, alors on a

$$f^* = \bar{f} + \frac{1}{2}\sum_{i=1}^{20} \lambda_i \alpha_i^2.$$

Les caractéristiques de ces problèmes-test sont résumées dans la table 5.4.

TABLE 5.4 – Caractéristiques des problèmes-test de Floudas [42]

Nom	Problème	m	n	λ_i	α_i	$\bar{f}$	$0.5\sum_{i=1}^{20} \lambda_i \alpha_i^2$	f^*
Floudas-qp1	5	1	5	-	-	-17	-	-17
Floudas-qp2	6	2	6	-	-	-361.5	-	-36105
Floudas-qp3	7	9	13	-	-	-195	-	-195
Floudas-qp4	8	6	5	-	-	-15.49	-	-15049
Floudas-qp5	9	11	10	-	-	-268.01	-	-268.01
Floudas-qp6	10	5	10	-	-	-39	-	-39
Floudas-qp7	11	10	20	1	2	-394.7506	40	-354.7506
Floudas-qp8	11	10	20	1	-5	-884.7506	250	-634.7506
Floudas-qp9	11	10	20	1	8	-754.7506	640	-114.7506

F. Vingt problèmes-test correspondant au programmes quadratiques concaves de la librairie Globallib [46] qui représente une collection de problèmes d'optimisation globale. Les caractéristiques de ces problèmes-test sont résumées dans la table 5.5.

TABLE 5.5 – Caractéristiques des problèmes-test Globallib [46]

No.	Nom	m	n	f^*	No.	Nom	m	n	f^*
1	ex2-1-5	11	10	-268,015	11	st-qpc-m1	5	5	-473,778
2	ex2-1-7	10	20	-3730,410	12	st-qpc-m3b	10	10	0
3	ex2-1-8	10	24	15639	13	st-qpc-m3c	10	10	0
4	st-fp7a	10	20	-354,75	14	st-qpc-m4	10	10	0
5	st-fp7b	10	20	-634,75	15	st-rv1	5	10	-59,944
6	st-fp7c	10	20	-8695,012	16	st-rv2	10	20	-64,481
7	st-fp7d	10	20	-114,75	17	st-rv3	20	20	-35,761
8	st-fp7e	10	20	-3730,41	18	st-rv7	20	30	-138,187
9	st-m1	11	20	-461356,939	19	st-rv8	20	40	-132,662
10	st-m2	21	30	-856648,819	20	st-rv9	20	50	-120,153

G. Cinquante problèmes générés aléatoirement en utilisant la procédure de génération décrite comme suit [94] : Soit $S = \{x \in \mathbb{R}^n : a_i^T x \leq b_i,\ i = 1, \ldots, m\}$, où $a_i \in \mathbb{R}^n$, $b_i \in \mathbb{R}$. Soit v un point frontière de S, et $I(v) = \{i : a_i^T v = b_i\}$. On définit $l(x) = a^T x$, où $a = \sum_{i \in I(v)} \lambda_i a_i,\ 0 \leq \lambda_i \leq 1,\ i \in I(v),\ \sum_{i \in I(v)} \lambda_i = 1$. Soit w la solution du problème linéaire suivant $\min_{x \in S} a^T x$. Le problème $\min_{x \in S} -(ax)^2$ a une solution globale $u \in \{v, w\}$.

H. Cinquante problèmes-test générés avec l'algorithme présenté dans [92] qui sont de la forme : $\min\ f(x) = c^T x + 0.5 x^T Q x,$ s.c. $Ax \leq b,\ x \geq 0$, où $A \in \mathbb{R}^{(n+1)\times n}$, $b \in \mathbb{R}^{n+1}$, $c, x \in \mathbb{R}^n$, $Q \in \mathbb{R}^{n \times n}$ est symérique semi-définie négative.

5.3 Comparaison numérique

Nous avons comparé AALSS, avec le paramètre $NE = 10$, aux algorithmes suivants :

AEA : l'algorithme basé sur les ensembles d'approximation proposé dans [31].

ABBR : l'algorithme de branch-and-bound développé par Rusakov dans [88].

CPLEX : l'algorithme de branch-and-bound implémenté dans CPLEX12.8 (la fonction "cplexqp" avec les paramètres "optimalitytarget" et "timelimit" réglés à 3 et 10800 s, respectivement) [59].

Nous avons exécuté les différents solveurs sur un PC Intel Core i7-4790, CPU @3.60 Ghz, 8GO RAM, qui fonctionne sous le système d'exploitation Windows 10. Pour résoudre les programmes linéaires intermédiaires, l'algorithme du simplexe [33] ou les algorithmes présentés dans [15, 22, 39] peuvent être utilisés. Cependant, nous avons utilisé l'algorithme des points intérieurs (la fonction "cplexlp" avec le paramètre lpmethod=4) et la fonction "cplexqp" pour trouver le maximum de f sur $\mathbb{R}^n$.

La signification des paramètres mentionnés dans les différentes tables est comme suit : m et n désignent le nombre de contraintes ainsi que le nombre de variables du problème-test ; f^* représente l'optimum global connu du problème test ; f_0 désigne l'optimum du programme linéaire $\min_{x \in S} c^T x$; f désigne l'optimum global approché obtenu par l'algorithme exécuté ; $\triangle f$ désigne la valeur absolue de la différence entre l'optimum global connu et l'optimum global approché obtenu par l'algorithme exécuté, i.e., $\triangle f = |f^* - f|$; "IT" désigne le nombre d'itérations de l'algorithme exécuté ; NE^* désigne le numéro de l'exécution au cours de laquelle AALSS a trouvé l'optimum global f^*, i.e.,

$f^{NE} = \min\{f^i : i = 1, 2, \ldots, NE+1\}$, où f^i représente l'optimum global approché trouvé dans la i$^{\text{ème}}$ exécution. "CPU" représente le temps d'exécution de l'algorithme considéré. Notons que pour AALSS, "CPU" et "IT" représentent respectivement la somme des temps d'exécution et la somme des nombres d'itérations des $NE + 1$ exécutions effectuées de l'algorithme 4.3. Lorsque le temps CPU de AALSS est meilleur que celui de CPLEX, nous le mettons en évidence par le caractère foncé.

Nous avons résolu les 156 problèmes-test listés dans la section 5.2 avec les solveurs mentionnés ci-dessus. Les résultats numériques obtenus sont présentés dans de différents tables et ils sont représentés graphiquement dans plusieurs figures, et ce, selon le tableau suivant :

Classe du problème-test	Nom	Tableau des résultas	Figure
A	Norme	5.6	5.1
B	SPIN-GLASS	5.7	5.2
C	DCTQP	5.8	5.2
D	Rusakov	5.9	5.3
E	Floudas	5.10	5.4
F	Globallib	5.11	5.4
G	Thoai	5.12	5.5
H	Rosen	5.13	5.5

A partir des différentes tables et graphes, nous pouvons remarquer ce qui suit :

— Pour les problèmes-test à variables bornées (tables 5.6 et 5.7), CPLEX est plus rapide que AALSS, en particulier pour les problèmes-test de norme avec une dimension supérieure à 200 variables (voir la table 5.6). Cependant, pour les problèmes test "DCTQP" présentés dans la table 5.8, nous remarquons que notre algorithme est plus précis et plus rapide que CPLEX, en particulier pour les problèmes-test de dimension $n \geq 20$. Par exemple, le problème-test "dctqp5" de dimension 20 est résolu en 0,250 s avec AALSS et en 86,609 s avec CPLEX et le problème-test "dctqp6" avec 24 variables est résolu en 0,211 s avec AALSS, tandis que CPLEX a été interrompu après 3 heures. La performance de notre algorithme pour ce genre de problèmes vient du fait que la solution du programme linéaire $\min_{x \in S} c^T x$ coïncide avec la solution optimale globale du problème ($f^0 = f^*$).

— De plus, nous remarquons que AALSS a réussi à trouver l'optimum global connu pour tous les problèmes à variables bornées, et ce, lors de la première exécution (les vecteurs h^j sont égaux aux vecteurs unitaires e_j), après aucune amélioration n'a été constatée au cours des dix autres exécutions ($NE^* = 1$). Enfin, nous constatons que l'algorithme AEA a trouvé des valeurs avec $\triangle f$ variant entre $1.0E+01$ et $1.5E+08$ pour les problèmes-test "norm-qp7" ,..., "norm-qp12". Cela montre que les techniques utilisées dans notre algorithme ont considérablement amélioré la précision de l'algorithme proposé dans [31]. à savoir :

1. La technique d'initialisation qui consiste à partir d'un point extrême correspondant à une solution optimale du programme linéaire $\min_{x \in S} c^T x$;
2. La technique de choix du nombre d'éléments de l'ensemble d'approximation du premier ordre. Ce nombre est posé égal au nombre de variables du problème considéré ;
3. La technique de choix des vecteurs h^j, qui consiste à les générer de manière à obtenir des vecteurs $y^j \in E_{f(z)}(f)$, qui n'appartiennent pas nécessairement à S, tout en prenant en compte le fait que la matrice D est semi-définie ;
4. La technique stochastique qui consiste à exécuter l'algorithme 11 fois, où dans la première exécution, les vecteurs h^j sont pris égaux aux vecteurs de la matrice identité. Cette technique stochastique a permis de trouver de meilleures approximations pour l'ensemble niveau de la fonction objectif au point courant.

— Pour les problèmes-test de Rusakov (voir les tables 5.3 et 5.9), nous remarquons que AALSS a réussi à trouver les valeurs globales exactes connues pour les problèmes-test "Rusakov-qp1" et "Rusakov-qp3" ($NE^* = 3$), "Rusakov-qp2" ($NE^* = 2$) et "Rusakov-qp4" ($NE^* = 5$). En termes de temps de calcul, nous remarquons que CPLEX est plus rapide que AALSS et ABBR. De plus, AALSS est plus rapide que l'algorithme de branch-and-bound de Rusakov (voir le graphe de la figure 5.3).

— Pour les problèmes-test de Floudas présentés dans les tables 5.4 et 5.10, nous remarquons que tous les problèmes-test sont résolus avec une bonne précision et un temps de calcul court par AALSS et CPLEX. De plus, toutes les valeurs globales des problèmes-test de Floudas ont été retrouvées lors de la première exécution de

AALSS, à l'exception du problème "Floudas-qp6", l'optimum global connu était trouvé lors de la quatrième exécution ($NE^* = 4$). En termes de temps de calcul, on note que AALSS est plus rapide que CPLEX sur les problèmes 1, 4 et 5, mais il est plus lent que CPLEX sur les six autres problèmes-test (voir le graphe de la figure 5.4).

— Pour les problèmes-test de la bibliothèque Globallib (voir les tables 5.5 et 5.11), nous notons que CPLEX et AALSS ont réussi à trouver l'optimum global connu des problèmes-test. En termes de temps de calcul, on note que CPLEX est plus rapide que AALSS, sauf pour les trois problèmes st_m1, st_rv1 et st_rv2, AALSS est plus rapide que CPLEX (voir le graphe de la figure 5.4).

— Pour les problèmes-test de Thoai générés aléatoirement (voir la table 5.12), CPLEX est plus rapide que AALSS. Cependant, pour les problèmes-test de Rosen indiqués dans la table 5.13, notre algorithme est plus précis et plus rapide que CPLEX, en particulier dans la résolution des problèmes de grandes dimensions. En effet, AALSS a trouvé des valeurs globales approchées avec une erreur entre $8,6E-10$ et $2,7E-08$, alors que CPLEX a trouvé des valeurs approchées avec une erreur entre $3,7E-08$ et $2,6E-06$. De plus, le temps de calcul de AALSS varie entre $0,288$ s et $2,227$ s, tandis que le temps CPU de CPLEX varie entre $0,403$ s et $363,250$ s pour les problèmes-test avec $n \leq 30$. Pour les problèmes-test avec une dimension supérieure à 30 variables, CPLEX a atteint le temps limite de 3 heures et AALSS les a résolu en moins de $3s$.

— Parmi les 156 problèmes-test, AALSS est plus rapide que CPLEX dans 61 problèmes (5 problèmes de DCTQP, 3 problèmes de Globallib, 3 problèmes de Floudas et les 50 problèmes-test de Rosen), soit 39,10% des problèmes-test. De plus, l'erreur moyenne de AALSS est meilleur que celle de CPLEX pour cinq classes de problèmes-test (norme, DCTQP, Rusakov, Thoai et Rosen) et l'erreur moyenne des deux solveurs est égale pour les trois autres classes (SPIN-GLASS, Floudas et Globallib).

— Le temps de calcul de la méthode du gradient conditionnel est très petit (inférieur à $0,08s$) pour tous les problèmes-test. De plus, son nombre d'itérations est égal

à un pour tous les problèmes-test, sauf pour les problèmes de Thoai, il est égal à 2. Cela signifie que le les solutions globales approchées obtenues par les deux premières étapes de AALSS sont des points stationnaires pour tous les problèmes-test, à l'exception des problèmes de Thoai, qui ont un vecteur c égal au vecteur nul, la méthode du gradient conditionnel a amélioré le point non stationnaire obtenu en deux itérations pour atteindre un point stationnaire.

— Parmi les 156 problèmes test, 147 problèmes (94,23%) ont été résolus jusqu'à l'optimalité $\varepsilon-$globale lors de la première exécution ($NE^* = 1$). Cela signifie que l'application de l'algorithme 4.4 avec h^j choisi comme vecteurs de la matrice identité est efficace et précis, en particulier pour les problèmes à variables bornées, mais il est moins précis que AALSS (avec $NE = 10$) pour les problèmes-test Floudas, Rusakov et Globallib.

— Bien que la convergence de AALSS vers une solution ε-globale ne soit pas garantie (par exemple, si nous appliquons AALSS avec $NE = 5$ ou avec $r = [n/2]$, nous n'obtiendrons pas la solution globale pour certains problèmes), on peut remarquer que la solution globale approchée trouvée par notre méthode (AALSS avec $NE = 10$ et $r = n$) est aussi ε-globale pour tous les problèmes-test et c'est une solution globale exacte ($\triangle f = 0$) pour les problèmes-test : norme, SPIN-GLASS, DCTQP, Rusakov et certains des problèmes-test de Floudas et Globallib.

TABLE 5.6 – Résultats numériques de AALSS, CPLEX et AEA pour les problèmes-test de norme

	AALSS				CPLEX		AEA
Problème	IT	CPU	Δf	NE^*	CPU	Δf	Δf
norm-qp1	64	0.461	0	1	0.063	0	0
norm-qp2	143	3.903	0	1	0.016	0	0
norm-qp3	241	15.218	0	1	0.016	0	0
norm-qp4	417	50.609	0	1	0.015	0	0
norm-qp5	1388	516.709	0	1	0.000	6.1E-05	0
norm-qp6	1485	2371.026	0	1	0.015	0	0
norm-qp7	58	0.429	0	1	0.016	0	1.0E+01
norm-qp8	159	4.297	0	1	0.016	0	5.0E+01
norm-qp9	253	15.361	0	1	0.015	0	1.2E+04
norm-qp10	398	45.278	0	1	0.000	0	1.2E+06
norm-qp11	785	239.065	0	1	0.015	0	1.9E+07
norm-qp12	1287	1830.461	0	1	0.000	0	1.5E+08
Moy	556.5	424.401	**0**	1	0.016	5.1E-06	1.5E+07

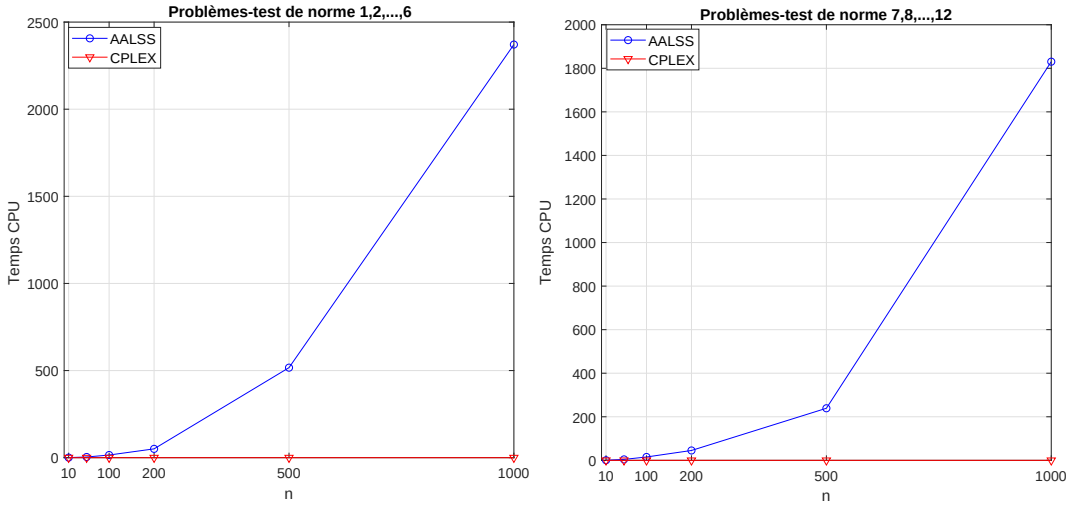

FIGURE 5.1 – Comparaison de AALSS et CPLEX pour les problèmes-test de norme

TABLE 5.7 – Résultats numériques de AALSS et CPLEX pour les problèmes-test SPIN-GLASS

			AALSS					CPLEX	
Problème	n	f^*	f_0	IT	CPU	Δf	NE^*	CPU	Δf
SpinGlass-qp1	20	-120	-40	54	0.778	0	1	0.219	0
SpinGlass-qp2	40	-240	-80	68	1.679	0	1	0.016	0
SpinGlass-qp3	60	-360	-120	94	3.314	0	1	0.015	0
SpinGlass-qp4	80	-480	-160	116	5.354	0	1	0.016	0
SpinGlass-qp5	100	-600	-200	106	6.651	0	1	0.015	0
Moy				87.6	3.555	0	1	0.056	0

TABLE 5.8 – Résultats numériques de AALSS et CPLEX pour les problèmes-test DCTQP

			AALSS					CPLEX	
Problème	n	f^*	f_0	IT	CPU	Δf	NE^*	CPU	Δf
dctqp1	4	-18	-18	11	0.174	0	1	0.031	0.0E+00
dctqp2	8	-132	-132	11	**0.152**	0	1	0.219	0.0E+00
dctqp3	12	-438	-438	11	**0.178**	0	1	0.203	2.4E-09
dctqp4	16	-1032	-1032	11	**0.196**	0	1	3.594	2.2E-08
dctqp5	20	-2010	-2010	11	**0.250**	0	1	86.609	1.6E-07
dctqp6	24	-3468	-3468	11	**0.211**	0	1	¿10800	9.3E-09
Moy				11	**0.194**	**0**		18.131	3.2E-08

TABLE 5.9 – Résultats numériques de AALSS, CPLEX et ABBR pour les problèmes-test de Rusakov

	AALSS				CPLEX		ABBR	
Problème	IT	CPU	Δf	NE^*	CPU	Δf	CPU	Δf
Rusakov-qp1	28	0.233	0	3	0.140	0.0E+00	0.412	0
Rusakov-qp2	38	0.390	0	2	0.187	1.6E-13	0.849	0
Rusakov-qp3	39	0.549	0	3	0.031	0.0E+00	1.184	0
Rusakov-qp4	37	0.673	0	5	0.188	0.0E+00	1.313	0
Moy	35.50	0.461	**0**	3.25	0.137	3.9E-14	0.940	0

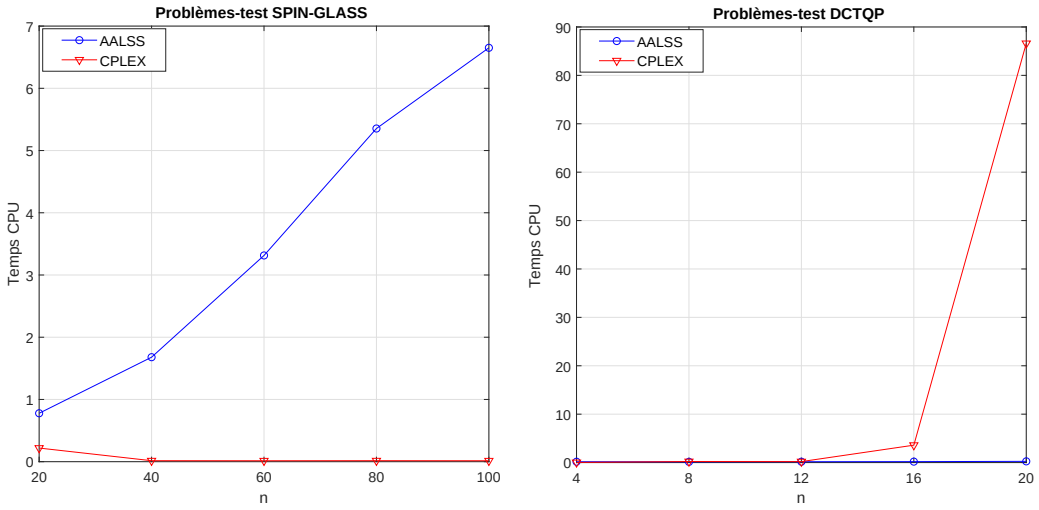

FIGURE 5.2 – Comparaison de AALSS et CPLEX pour les problèmes-test SPIN-GLASS et DCTQP

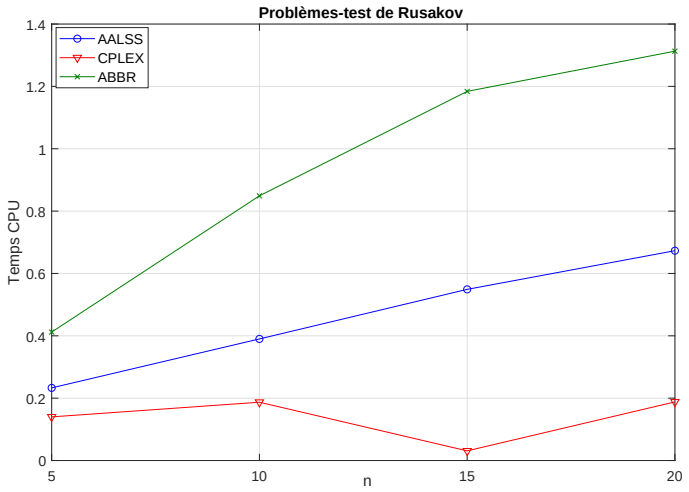

FIGURE 5.3 – Comparaison de AALSS, CPLEX et ABBR pour les problèmes-test de Rusakov

TABLE 5.10 – Résultats numériques de AALSS et CPLEX pour les problèmes-test de Floudas

		AALSS					CPLEX	
Problème	f^*	f_0	IT	CPU	Δf	NE^*	CPU	Δf
Floudas-qp1	-17.00	0.00	28	**0.212**	0.0E+00	1	0.719	0.0E+00
Floudas-qp2	-361.50	-361.50	11	0.160	0.0E+00	1	0.016	0.0E+00
Floudas-qp3	-195.00	-23.47	25	0.236	0.0E+00	1	0.031	0.0E+00
Floudas-qp4	-15.49	-15.49	11	**0.140**	8.6E-05	1	0.453	8.6E-05
Floudas-qp5	-268.01	-268.01	11	**0.174**	1.5E-06	1	0.203	1.5E-06
Floudas-qp6	-39.00	6.73	38	0.360	0.0E+00	4	0.203	0.0E+00
Floudas-qp7	-354.75	1.48	36	0.609	8.4E-05	1	0.172	8.4E-05
Floudas-qp8	-634.75	-360.55	43	0.657	3.2E-05	1	0.188	3.2E-05
Floudas-qp9	-114.75	6.55	16	0.364	6.8E-04	1	0.172	6.8E-04
Moy			24.33	0.324	9.9E-05	1.33	0.240	9.9E-05

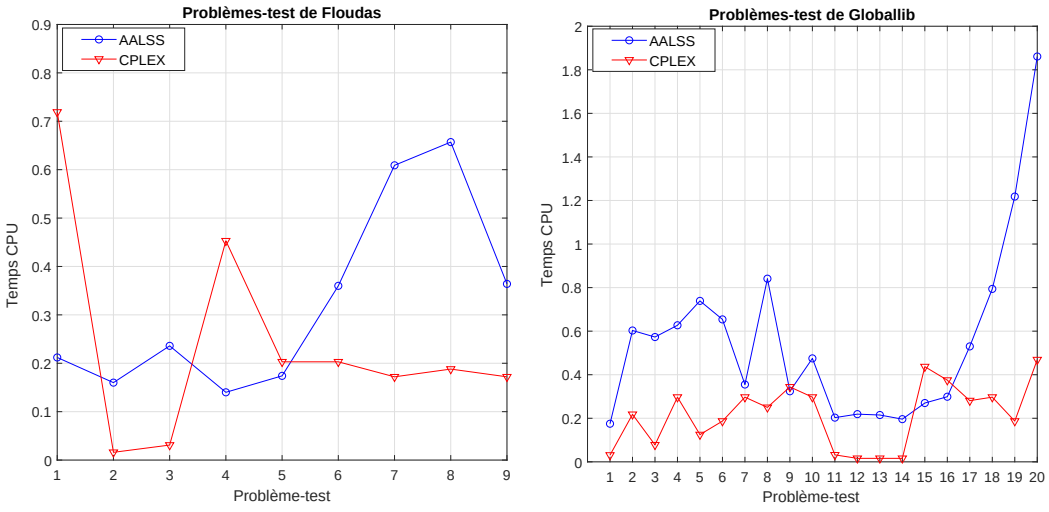

FIGURE 5.4 – Comparaison de AALSS et CPLEX pour les problèmes-test de Floudas et Globallib

TABLE 5.11 – Résultats numériques de AALSS et CPLEX pour les problèmes-test de Globallib

	AALSS					CPLEX	
Problème	f_0	IT	CPU	Δf	NE^*	CPU	Δf
ex2_1_5	-268.01	11	0.175	4.7E-10	1	0.031	4.7E-10
ex2_1_7	12.75	36	0.603	1.7E-09	2	0.218	1.7E-09
ex2_1_8	18423.00	25	0.573	0.0E+00	1	0.078	0.0E+00
st_fp7a	1.48	36	0.627	6.1E-04	1	0.297	6.1E-04
st_fp7b	-360.55	47	0.739	6.1E-04	1	0.125	6.1E-04
st_fp7c	-3.16	43	0.654	3.2E-04	11	0.187	3.2E-04
st_fp7d	6.55	16	0.355	6.1E-04	1	0.297	6.1E-04
st_fp7e	12.75	52	0.841	1.3E-04	2	0.250	1.3E-04
st_m1	-461356.94	11	**0.324**	0.0E+00	1	0.344	3.5E-10
st_m2	-856648.82	11	0.475	0.0E+00	1	0.297	8.1E-10
st_qpc-m1	55.32	22	0.203	0.0E+00	1	0.032	4.5E-09
st_qpc-m3b	0.00	11	0.219	0.0E+00	1	0.016	0.0E+00
st_qpc-m3c	0.00	11	0.215	0.0E+00	1	0.016	0.0E+00
st_qpc-m4	0.00	11	0.196	0.0E+00	1	0.016	0.0E+00
st_rv1	-59.90	20	**0.270**	0.0E+00	1	0.437	1.4E-14
st_rv2	-64.48	11	**0.299**	0.0E+00	1	0.375	1.4E-14
st_rv3	-35.38	27	0.530	0.0E+00	1	0.281	0.0E+00
st_rv7	-131.70	32	0.794	0.0E+00	1	0.297	7.9E-10
st_rv8	-128.80	34	1.218	0.0E+00	1	0.187	0.0E+00
st_rv9	-118.03	39	1.861	0.0E+00	3	0.469	0.0E+00
Moy		25.30	0.559	1.1E-04	1.70	0.213	1.1E-04

TABLE 5.12 – Résultats numériques de AALSS et CPLEX pour les problèmes-test de Thoai

			AALSS				CPLEX	
Problème	m	n	IT	CPU	Δf	NE^*	CPU	Δf
Thoai-qp1	10	15	11	0.198	1.7E-15	1	0.011	6.0E-15
Thoai-qp2	20	30	11	0.270	1.1E-14	1	0.013	1.6E-14
Thoai-qp3	30	45	11	0.369	2.6E-14	1	0.015	1.4E-14
Thoai-qp4	40	60	11	0.487	2.8E-14	1	0.016	2.2E-13
Thoai-qp5	50	75	11	0.652	8.5E-14	1	0.024	2.6E-13
Moy			11	0.395	**3.0E-14**	1	0.016	1.0E-13

TABLE 5.13 – Résultats numériques de AALSS et CPLEX pour les problèmes-test de Rosen

			AALSS				CPLEX	
Problème	m	n	IT	CPU	Δf	NE^*	CPU	Δf
Rosen-qp1	11	10	22.00	**0.288**	9.2E-10	1	0.403	3.7E-08
Rosen-qp2	21	20	26.50	**0.571**	8.6E-10	1	9.438	7.1E-07
Rosen-qp3	31	30	25.90	**0.916**	2.3E-09	1	363.250	5.2E-07
Rosen-qp4	41	40	36.70	**1.887**	2.7E-08	1	¿10800	6.3E-07
Rosen-qp5	51	50	25.50	**2.227**	1.6E-08	1	¿10800	2.6E-06
Moy			27.32	**1.178**	**9.4E-09**	1.00	124.364	9.0E-07

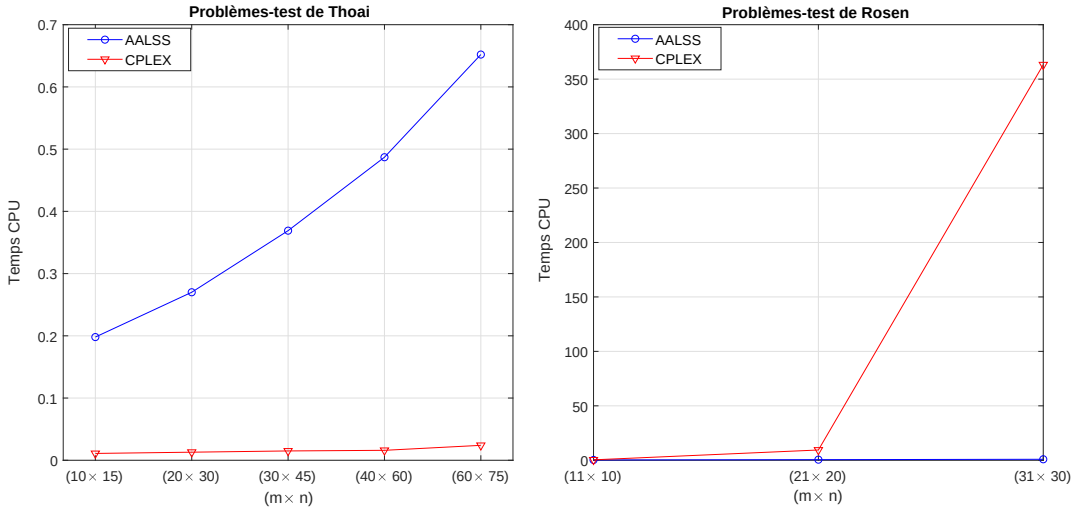

FIGURE 5.5 – Comparaison de AALSS et CPLEX pour les problèmes-test de Thoai et Rosen

Conclusion

Dans ce travail, en se basant sur les algorithmes proposés dans [34, 31, 8, 17], nous avons proposé un nouvel algorithme appelé algorithme des approximations linéaires successives pour la résolution des problèmes de programmation quadratique concave qui se présentent sous la forme générale (la matrice de la forme quadratique est semi-définie négative, les contraintes du problème sont du type égalité et/ou inégalité, les variables sont bornées par des bornes qui peuvent être finies ou infinies).

Dans le but de réduire le temps d'execution et d'améliorer la précision des algorithmes développés dans [34, 31, 8], nous avons proposé plusieurs techniques, à savoir

1. la technique d'initialisation qui consiste à partir du point extrême correspondant à une solution optimale du programme linéaire minimisant la partie linéaire de la fonction objectif $\min_{x \in S} c^T x$;
2. la technique qui consiste à fixer le nombre d'éléments de l'ensemble d'approximation du premier ordre égal au nombre de variables du problème considéré ;
3. la technique de choix des vecteurs h^j, qui consiste à les générer de telle sorte à obtenir des vecteurs y^j de la ligne niveau de la fonction objectif au point courant, qui ne sont pas forcément réalisables, toute en prenant en considération le fait que la matrice D est semi-définie négative ;
4. la technique stochastique qui consiste à exécuter l'algorithme plusieurs fois, où dans le première exécution, les vecteurs h^j sont pris égaux aux vecteurs de la matrice identité. Cette technique stochastique a permis de trouver de meilleures approximations pour l'ensemble niveau de la fonction objectif au point courant.

Afin de tester l'efficacité de notre approche, nous avons développé une implémentation avec le langage de programmation MATLAB ; puis nous avons comparé les performances

de notre algorithme avec celles de trois algorithmes existants : l'algorithme de branch-and-bound implémenté dans « CPLEX12.8 », l'algorithme de branch-and-bound de Rusakov implémenté dans le solveur « CONCAVE » et l'algorithme des ensembles d'approximation, et ce, sur plusieurs classes de problèmes de programmation quadratique concave.

Les résultats numériques obtenus montrent bien que notre algorithme est plus rapide que l'algorithme de Rusakov ; il est plus précis que l'algorithme des ensembles d'approximation et il est compétitif avec l'algorithme de branch-and-bound de CPLEX [59] que ça soit en terme de précision ou bien en terme de temps d'exécution, particulièrement pour la résolution des problèmes DCTQP et les problèmes-test de ROSEN. De plus, la supériorité de notre algorithme par rapport à CPLEX, pour ce type de problèmes, augmente avec l'augmentation de la dimension du problème.

Néanmoins, il reste à accomplir d'autres travaux dans le futur, à savoir :

— la généralisation de AALS pour la résolution des problèmes de programmation quadratique concave en nombres entiers.

— la généralisation de AALS pour la résolution des problèmes de programmation quadratique non-convexe.

— l'application de l'algorithme local développée dans [18] pour passer d'un point extrême non local à un point extrême vérifiant les conditions d'optimalité locale.

Bibliographie

[1]

[2] M. Al Kharboutly. *Résolution d'un problème quadratique non convexe avec contraintes mixtes par les techniques de l'optimisation DC.* Thèse de doctorat, Université Le Havre Normandie, Normandie, France, 2018.

[3] F. Al Khayyal, R. Horst, and P.M. Pardalos. Global optimization of concave functions subject to quadratic constraints : an application in nonlinear bilevel programming. *Annals of Operations Research*, 34 :125–147, 1992.

[4] M. Aourid and B. Kaminska. Minimization of the 0-1 linear programming problem under linear constraints by using neural networks : synthesis and analysis. *IEEE Transactions on Circuits and Systems I : Fundamental Theory and Applications*, 43 :421, 1996.

[5] D. Avis and K. Fukuda. Reverse search for enumeration. *Discrete applied mathematics*, 65 :21–46, 1996.

[6] B. Baesens, T. Van Gestel, S. Viaene, M. Stepanova, J. Suykens, and J. Vanthienen. Benchmarking state-of-the-art classification algorithms for credit scoring. *Journal of the operational research society*, 54 :627–635, 2003.

[7] M. Balinski. Fixed-cost transportation problems. *Naval Research Logistics Quarterly*, 8 :41–54, 1961.

[8] T. Bayartugs, C. Battuvshin, and R. Enkhbat. Quadratic optimization over a polyhedral set. 9 :621–629, 2014.

[9] M. Bazaraa and H. Sherali. On the use of exact and heuristic cutting plane methods for the quadratic assignment problem. *Journal of the Operational Research Society*, 33 :991–1003, 1982.

[10] M. Bazaraa, H. Sherali, and C. Shetty. *Nonlinear programming : theory and algorithms*. John Wiley & Sons, 2013.

[11] H.P. Benson. Concave minimization : theory, applications and algorithms. *Handbook of global optimization*, 43–148, 1995.

[12] H.P. Benson and R. Horst. A branch and bound-outer approximation algorithm for concave minimization over a convex set. *Computers & Mathematics with Applications*, 21 :67–76, 1991.

[13] H.P. Benson and S. Sayin. A finite concave minimization algorithm using branch and bound and neighbor generation. *Journal of Global Optimization*, 5 :1–14, 1994.

[14] M. Bentobache. *Sur les méthodes mathématiques de la programmation linéaire et quadratique*. Thèse de doctorat, Université Abderrahmane Mira de Béjaïa, Béjaïa, Algérie, 2013.

[15] M. Bentobache and M.O. Bibi. A two-phase support method for solving linear programs : Numerical experiments. *Mathematical Problems in Engineering*, 2012 :01–28, 2012.

[16] M. Bentobache, M. Telli, and A. Mokhtari. A sequential linear programming algorithm for continuous and mixed-integer nonconvex quadratic programming. *In Le Thi H. et al. (eds) Optimization of Complex Systems : Theory, Models, Algorithms and Applications, Advances in Intelligent Systems and Computing*, 991 :26–36, 2020.

[17] M. Bentobache, M. Telli, and A. Mokhtari. A global minimization algorithm for concave quadratic programming. *In proceedings of EURO 2018, 29th European Conference on Operational Research*, 329, July 08-11, 2018.

[18] M. Bentobache, M. Telli, and A. Mokhtari. A simplex algorithm with the smallest index rule for concave quadratic programming. *In proceedings of INFOCOMP2018 : The Eighth International Conference on Advanced Communications and Computation*, 88–93, July 22-26, 2018.

[19] M. Bergounioux. *Optimisation et contrôle des systèmes linéaires*. Dunod, 2001.

[20] L. Berkovitz. *Convexity and optimization in* $\mathbb{R}^n$. John Wiley & Sons, 2003.

[21] D.P Bertsekas, W. Hager, and O. Mangasarian. *Nonlinear Programming*. Athena Scientific Belmont, Massachusets, USA, 1999.

[22] M. Ouamer Bibi and M. Bentobache. A hybrid direction algorithm for solving linear programs. *International Journal of Computer Mathematics*, 92 :201–216, 2015.

[23] L. Bieche, J. Uhry, R. Maynard, and R. Rammal. On the ground states of the frustration model of a spin glass by a matching method of graph theory. *Journal of Physics A : Mathematical and General*, 13 :2553–2576, 1980.

[24] M. Bierlaire. *Introduction à l'optimisation différentiable*. Presses Polytechniques et Universitaires Romandes, Romandie, Suisse, 2006.

[25] B. Brahmi. *Méthodes primale et duale pour la résolution des problèmes de programmation quadratique convexe*. Mémoire de Magister, Université Abderrahmane Mira de Béjaïa, Béjaïa, Algérie, 2006.

[26] B. Brahmi. *Méthodes primales et duales pour la programmation quadratique : extension et applications*. Thèse de doctorat, Université Abderrahmane Mira de Béjaïa, Béjaïa, Algérie, 2012.

[27] R.E. Burkard, E. Cela, P.M. Pardalos, and L.S. Pitsoulis. The quadratic assignment problem. *Handbook of combinatorial optimization*, pages 1713–1809, 1998.

[28] R.H. Byrd, G. Liu, and J. Nocedal. On the local behavior of an interior point method for nonlinear programming. *Technical Report OTC 98/02, Optimization Technology Center, Northwestern University, 1997*.

[29] J. Chen and S. Burer. Globally solving nonconvex quadratic programming problems via completely positive programming. *Mathematical Programming Computation*, 4 :33–52, 2012.

[30] P. Chen, P. Hansen, and B. Jaumard. On-line and off-line vertex enumeration by adjacency lists. *Operations Research Letters*, 10 :403–409, 1991.

[31] A. Chinchuluun, P.M. Pardalos, and R. Enkhbat. Global minimization algorithms for concave quadratic programming problems. *Optimization*, 54 :627–639, 2005.

[32] P. Ciarlet. *Introduction à l'analyse numérique matricielle et à l'optimisation*. Dunod, 2007.

[33] G.B. Dantzig. *Linear Programming and Extensions.* Princeton University Press, Princeton, New Jersey, USA, 1963.

[34] R. Enkhbat. On some theory, methods and algorithms for concave programming. *Optimization and Optimal Control, Series on Computers and Operations Research*, 1 :79–102, 2003.

[35] R. Enkhbat and Y. Bazarsad. General quadratic programming and its applications in response surface analysis. *Optimization and Optimal Control, Springer Optimization and Its Applications, Springer, New York*, 39 :121–137, 2010.

[36] R. Gabasov et F.M. Kirillova. Méthodes de programmation linéaire, volumes 1, 2 et 3. *Edition de l'Université de Minsk (en russe)*, 1977, 1978 et 1980.

[37] F. Demengel et G. Demengel. *Convexité dans les espaces fonctionnels : théorie et illustration par les exemples.* Ellipses Editions, Paris, France, 2004.

[38] D. Werra T. Lieblin et J. Hêche. *Recherche opérationnelle pour ingénieurs.* Lausanne : Presses polytechniques et universitaires romandes, Suisse, 2003.

[39] M. Bentobache et M.O. Bibi. *Méthodes numériques de la programmation linéaire et quadratique : théorie et algorithmes.* Presses Académiques Francophones, Allemagne, 2016.

[40] J. Falk and K. Hoffman. A successive underestimation method for concave minimization problems. *Mathematics of operations research*, 1 :251–259, 1976.

[41] A.V. Fiacco and G.P. Mc Cormick. *Nonlinear programming : sequential unconstrained minimization techniques.* SIAM, 1990.

[42] C. Floudas, P.M. Pardalos, C. Adjiman, W. Esposito, Z.H. Gümüs, S.T. Harding, J.L. Klepeis, Clifford A. Meyer, and C. Schweiger. *Handbook of test problems in local and global optimization.* Springer, 2013.

[43] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3 :95–110, 1956.

[44] B. Gasmi. *Contribution Ã l'étude des méthodes de résolution des problèmes d'optimisation quadratique.* Mémoire de Magister, Université de Batna 2, Batna, Algérie, 2007.

[45] P.E. Gill and W. Murray. *Numerical methods for constrained optimization*. Academic Press, Inc., (London) LTD, 1974.

[46] Globallib. Gamsworld global optimization library, Consulté le 15 janvier 2019.

[47] J. Hiriart-Urruty. *Les Mathématiques du Mieux Faire-Volume 1-Premiers Pas en Optimisation*. Ellipses Editions, Paris, France, 2008.

[48] J. Hiriart-Urruty. *Bases, outils et principes pour l'analyse variationnelle*. Springer, 2012.

[49] J. Hiriart-Urruty. *Mathematical Tapas : Volume 2 (from Undergraduate to Graduate, L3-M1)*. Springer, 2016.

[50] J. Hiriart-Urruty and Y. Ledyaev. A note on characterization of the global maxima of a (tangentially) convex function over a convex set. *Journal of Convex Analysis*, 3 :55–62, 1996.

[51] W. Hirsch and G.B. Dantzig. *Notes on Linear Programming, Part XIX : The Fixed Charge Problem*. Rand Corporation, California, USA, 1954.

[52] R.A. Horn and C.R. Johnson. *Matrix analysis*. Cambridge university press, 2012.

[53] R. Horst. An algorithm for nonconvex progrmnming problems. *Mathematical Programming*, 10 :312–321, 1976.

[54] R. Horst. A general class of branch-and-bound methods in global optimization with some new approaches for concave minimization. *Journal of Optimization Theory and Applications*, 51 :271–291, 1986.

[55] R. Horst, P.M. Pardalos, and N. Thoai. *Introduction to global optimization*. Springer, 2000.

[56] R. Horst and H. Tuy. *Global optimization : Deterministic approaches*. Springer, 2013.

[57] P. Hungerländer. Algorithms for convex quadratic programming. *arXiv preprint arXiv :1409.5222*, 2014.

[58] N. Ikheneche. *Méthode de support pour la minimisation d'une fonctionnelle quadratique convexe*. Mémoire de Magister, Université Abderrahmane Mira de Béjaïa, Béjaïa, Algérie, 2003.

[59] IBM ILOG. Cplex optimizer 12.8, 2017.

[60] A. Iouditski. Optimization, analyse convexe, théorie de programmation non-linéaire. *Cours en ligne de l'Université de Joseph Fourier, Grenoble, France*, 2007.

[61] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4 :373–395, 1984.

[62] S. Kauffman et al. *The origins of order : self-organization and selection in evolution.* Oxford University Press, USA, 1993.

[63] Z. Kebbiche. *Etude et extensions d'algorithmes de points interieurs pour la programmation non lineaire.* Thèse de doctorat, Université Ferhat Abbas-Setif, Setif, Algérie, 2007.

[64] A. Khurana and S. Arora. Fixed charge bi-criterion indefinite quadratic transportation problem with enhanced flow. *Investigación Operacional*, 32 :133–145, 2014.

[65] T. Koopmans and M. Beckmann. Assignment problems and the location of economic activities. *Econometrica : journal of the Econometric Society*, 25 :53–76, 1957.

[66] H. Le Thi. An efficient algorithm for globally minimizing a quadratic function under convex quadratic constraints. *Mathematical programming*, 87 :401–426, 2000.

[67] H. An Le Thi and T. Pham Dinh. A continuous approch for globally solving linearly constrained quadratic. *Optimization*, 50 :93–120, 2001.

[68] S. Lessmann, B. Baesens, H. Seow, and L. Thomas. Benchmarking state-of-the-art classification algorithms for credit scoring : An update of research. *European Journal of Operational Research*, 247 :124–136, 2015.

[69] E. Levitin and B. Polyak. Constrained minimization methods. *USSR Computational mathematics and mathematical physics*, 6 :1–50, 1966.

[70] D. Luenberger. *Linear and nonlinear programming.* Springer, 2008.

[71] O. Mangasarian, W. Street, and W. Wolberg. Breast cancer diagnosis and prognosis via linear programming. *Operations Research*, 43 :570–577, 1995.

[72] MATLAB. Version 9.4. 0 (r2018a). *The MathWorks Inc., Natick, MA, USA*, 2018.

[73] C. Meyer. *Algorithmes coniques pour la minimisation quasiconcave.* École polytechnique de Montréal, Québec, Canada, 1996.

[74] M. Minoux. Multiflots de coût minimal avec fonctions de coût concaves. *Annales des télécommunications*, 31 :77–92, 1976.

[75] M. Minoux. *Programmation mathématique : théorie et algorithmes*. Tec & Doc - Lavoisier, 2007.

[76] M. Moeini. *La programmation DC et DCA pour l'optimisation de portefeuille*. Thèse de doctorat, Université Paul Verlaine-Metz, Metz, France, 2008.

[77] J. Momoh, L. Dias, S. Guo, and R. Adapa. Economic operation and planning of multi-area interconnected power systems. *IEEE transactions on power systems*, 10 :1044–1053, 1995.

[78] M. Monteiro, D. Fontes, and F. Fontes. *Solving concave network flow problems*. Faculty of Economics, University of Porto, Porto, Portugal, 2012.

[79] A. Ouazib and B. Aizel. Minimisation d'une forme quadratique concave soumise Ã des contraintes linéaires de type inégalités. *Mémoire de Master, Université Abderrahmane Mira de Béjaïa, Béjaïa, Algérie*, 2014.

[80] P. Pardalos and L. Pitsoulis. The quadratic assignment problem. *Handbook of combinatorial optimization*, 4 :241–337, 1998.

[81] P.M. Pardalos and G.P. Rodgers. Computational aspects of a branch and bound algorithm for quadratic zero-one programming. *Computing*, 45 :131–144, 1990.

[82] T. Pham Dinh. Algorithmes de calcul du maximum des formes quadratiques sur la boule unité de la norme du maximum. *Numerische Mathematik*, 45 :377–401, 1984.

[83] T. Pham Dinh et al. Duality in dc (difference of convex functions) optimization. subgradient methods. pages 277–293, 1988.

[84] T. Pham Dinh et al. A branch and bound method via dc optimization algorithms and ellipsoidal technique for box constrained nonconvex quadratic problems. *Journal of Global Optimization*, 13 :171–206, 1998.

[85] T. Pham Dinh, H. Le Thi, and F. Akoa. Combining dca (dc algorithms) and interior point techniques for large-scale nonconvex quadratic programming. *Optimization Methods and Software*, 23 :609–629, 2008.

[86] S. Radjef. *Optimisation cours et exercices*. Cours en ligne d'Université des Sciences et de la Technologie d'Oran Mohamed-Boudiaf, Oran, Algérie, 2020.

[87] M. Raghavachari. On connections between zero-one integer programming and concave programming under linear constraints. *Operations Research*, 17 :680–684, 1969.

[88] A. Rusakov. Concave programming under simplest linear constraints. *Computational mathematics and mathematical physics*, 43 :908–917, 2003.

[89] F. Saci. Sur la programmation quadratique convexe. *Mémoire de Master, Université Abderrahmane Mira de Béjaïa, Béjaïa, Algérie*, 2014.

[90] S. Sahni. Computationally related problems. *SIAM Journal on computing*, 3 :262–279, 1974.

[91] A. Strekalovsky. Global optimality conditions for nonconvex optimization. *Journal of Global Optimization*, 12 :415–434, 1998.

[92] Y. Sung and J. Rosen. Global minimum test problem construction. *Mathematical Programming*, 24 :353–355, 1982.

[93] M. Telli, M. Bentobache, and A. Mokhtari. A successive linear approximation algorithm for the global minimization of a concave quadratic program. *Computational and Applied Mathematics*, 39 :1–28, 2020.

[94] N. Thoai. On the construction of test problems for concave minimization algorithms. *Journal of Global Optimization*, 5 :399–402, 1994.

[95] H. Tuy. Concave programming under linear constraints. *Soviet Math.*, 5 :1437–1440, 1964.

[96] H. Tuy, S. Ghannadan, A. Migdalas, and P. Väarbrand. Strongly polynomial algorithm for a production-transportation problem with concave production cost. *Optimization*, 27 :205–227, 1993.

[97] R.J. Vanderbei. Loqo user's manual-version 4.05. *Princeton University*, 2000.

[98] R.J. Vanderbei. *Linear programming : foundations and extensions*, volume 285. Springer, 2020.

[99] A. Wächter and L.T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106 :25–57, 2006.

[100] W. Walker. *Contributions to the methodology of decomposition and fixed charge problems*. Cornell University, Ithaca, New York, USA, 1968.

[101] W. Walker. A heuristic adjacent extreme point algorithm for the fixed charge problem. *Management Science*, 22 :587–596, 1976.

[102] P. Wolfe. The simplex method for quadratic programming. *Econometrica : Journal of the Econometric Society*, 27 :382–398, 1959.

[103] W. Xia, J. Vera, and L. Zuluaga. Globally solving non-convex quadratic programs via linear integer programming techniques. *arXiv preprint arXiv :1511.02423*, 2015.

[104] M. Zamani. A new dual for quadratic programming and its applications. *arXiv preprint arXiv :1808.01266*, 2018.

[105] P. Zwart. Nonlinear programming : counterexamples to two global optimization algorithms. *Operations Research*, 21 :1260–1266, 1973.

ملخص

في هذا العمل، قمنا بتطوير خوارزمية جديدة لحل مشكل البرمجة التربيعية المقعرة تحت قيود خطية. تبدأ الخوارزمية المقترحة بحل تقريبي أولي، ثم تنتقل إلى حل جديد ذات قيمة أفضل لدالة الهدف. يتم الانتقال من حل إلى حل أخر عن طريق بناء مجموعات تقريبية معينة وحل سلسلة من البرامج الخطية. من أجل مقارنة الخوارزمية المقترحة بالخوارزميات الأخرى، قمنا ببرمجة طريقتنا بالماتلاب، ثم قارنا مختلف الخوارزميات على عدة مجموعات من المشاكل التربيعية المقعرة. نتائج المقارنة المتحصل عليها تبين بوضوح دقة وفعالية خوارزميتنا.

الكلمات المفتاحية: البرمجة التربيعية المقعرة، البرمجة الخطية، المجموعة التقريبية، خط المستوى.

Résumé

Dans cette thèse, nous proposons un nouvel algorithme pour la recherche d'un minimum global d'une fonction quadratique concave sous contraintes linéaires d'égalité et d'inégalité avec des variables non-négatives et/ou bornées. L'algorithme proposé commence par un sommet initial quelconque, puis il passe à un nouveau sommet ayant une meilleure valeur de la fonction objectif. Le passage d'un sommet à autre sommet s'effectue par la construction de certains ensembles d'approximation et la résolution d'une série de programmes linéaires. Afin de comparer notre algorithme avec les approches existantes, nous avons implémenté notre méthode avec MATLAB; puis nous avons comparé les différents algorithmes sur plusieurs collections de problèmes-test. Les résultats numériques obtenus montrent bien la précision et l'efficacité de notre algorithme.

Mots clés : Programmation quadratique concave, Optimisation globale, Programmation linéaire, Ensemble d'approximation, Ligne de niveau.

Abstract

In this thesis, we propose a new algorithm for finding an approximate global minimum of a concave quadratic function subject to linear equality and inequality constraints, where the variables are bounded with finite or infinite bounds. The proposed algorithm starts with an initial extreme point, then it moves from the current extreme point to a new one with a better objective function value. The passage from one basic feasible solution to a new one is done by the construction of certain approximation sets and solving a sequence of linear programming problems. In order to compare our algorithm with the existing approaches, we have developed an implementation with MATLAB and conducted numerical experiments on numerous collections of test problems. The obtained numerical results show the accuracy and the efficiency of our approach.

Keywords: Concave quadratic programming, Global optimization, Linear programming, Approximation sets, Level curve.