République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université M'hamed Bouguerra Boumerdes

Faculté des Sciences
Département des Mathématiques

Pour l'obtention du diplôme de master en Recherche Opérationnelle

Option : Recherche Opérationnelle, Modélisation et Aide à la Décision

(ROMAD)

Mémoire présenté le 26/06/2016

Par

M^{elle} Bradai Nadia

M^{elle} Rebai Sara

Programmation quadratique bi-objectif en varibles mixtes

devant le jury composé de :

Président	M^{me} S. Zouaoui	M.A.A	U.M.B.B.
Promoteur	M^{r} M. Bezoui	M.A.A	U.M.B.B.
Examinateur	M^{me} N. Larbi	M.A.A	U..M.B.B.

Année Universitaire 2015 – 2016

Table des matières

Table des figures

Liste des Algorithmes

Remerciement

$\mathcal{N}$oun tenons à exprimer notre profonde gratitude à monsieur M.BEZOUI, notre promoteur de mémoire, pour la confiance qu'il nous a faite en acceptant de diriger nos recherches, et pour ses précieux conseils et orientations, ainsi que pour l'intérêt particulier qu'il a accordé à ce travail. Nous le remercions pour sa grande contribution à l'aboutissement de ce travail.

$\mathcal{N}$ous adressons nos remerciements à madame S.Zouaoui pour l'honneur qu'elle nous a fait en acceptant de présider le jury de ce mémoire.

$\mathcal{N}$ous remercions également madame N.Laarbi pour avoir accepté de juger ce travail.

$\mathcal{N}$ous remercions nos trés chers parents, qui ont toujours été là pour nous,
"Vous avez tout sacrifié pour vos enfants n'épargnant ni santé ni efforts. Vous nous avez donné un magnifique modèle de labeur et de persévérance. Nous sommes redevable d'une éducation dont nous sommes fiers".

$\mathcal{N}$ous remercions nos frères et soeurs pour leur encouragement, anisi que toute la famille.

Sans oublier de remercier aussi tous nos collègues, nos amies et tous ceux qui ont contribué de prés ou de loin à la réalisation de ce travail.

À les plus beaux créatures que Dieu a créé sur terre „,

À ces sources de tendresse, de patience et de générosité„,

À mes parents !

À mes frères Ahmed,Lyes,Zoubir et Adel

À mes sœurs Assia,Fatiha et Khadidja

ainsi que leurs enfants Mohamed,Aridje,Ihabe et Ala'a

À mon binôme Rebai Sara et sa famille

À tou(te)s mes ami(e)s et collègues

À tous les étudiants de la promotion 2015/2016

Option :Recherche Opérationnelle

A tous ceux qui, par un mot, m'ont donné la force de continuer .....

Je dedie ce travail

.

Nadia

À les plus beaux créatures que Dieu a créé sur terre „,

À ces sources de tendresse, de patience et de générosité„,

À mes parents !

À mes frères Islam, Houssam et le petit coucou Nassim

À celle qui je la considère toujours ma soeur 'Rym'

ainsi qu'à mes chères amies Hassina et Hanaa

À mes deux grands mères que dieux les gardes, et toutes mes tantes et mes cousines

À mon chère oncle Djamel

À mon fiançé Mohamed et sa famille

À mon binôme Bradai Nadia et sa famille

À tou(te)s mes ami(e)s et collègues

À tous les étudiants de la promotion 2015/2016

Option :Recherche Opérationnelle

A tous ceux qui, par un mot, m'ont donné la force de continuer .....

Je dedie ce travail

.

Sara

Introduction

L'optimisation et plus particulièrement la programmation mathématique, vise à résoudre des problèmes où l'on cherche à déterminer parmi un grand nombre de solutions candidates, celle qui donne le meilleur résultat de la fonction objectif. Plus précisèment, on cherche à trouver une solution satisfaisant un ensemble de contraintes, et qui minimise ou maximise une fonction donnée. L'application de la programmation mathématique est en expansion croissante et trouve beaucoup d'applications dans plusieurs domaines pratiques.

L'optimisation quadratique est l'une des théories de la programmation mathématique la mieux adaptée à la formulation des problèmes pratiques. Cette branche est trés importante d'un point de vue pratique que théorique. Du point de vue applications ; plusieurs problèmes en économie, en mathématiques, dans les sciences de l'ingénieur, la recherche opérationnelle et le contrôle optimal, sont modélisés comme des problèmes d'optimisation quadratique. Du point de vue théorique, elle est une transition naturelle entre la programmation linéaire et non linéaire. Les algorithmes développés pour le cas de l'optimisation non linéaire reposent essentiellement sur l'approche quadratique, tels que les méthodes de programmation quadratique séquentielle (PQS).

Plusieurs méthodes ont été développées pour la résolution de ce type de problèmes, parmi lesquelles on peut citer :

✓ La méthode d'activation des contraintes (Active-set method, ASM).

✓ Les méthodes de points intérieurs.

✓ Méthodes adaptées de support.

✓ Méthode du simplexe quadratique de Wolf(1959).

✓ Méthode directe de support.

L'objet de ce memoire est d'implémenter une nouvelle méthode de résolution de problème de programmation quadratique bi-objectif à variables mixtes et d'appliquer cette méthode sur le problème de portefeuille, et ce en s'inspirant des méthodes adaptées de support pour la résolution des problèmes de programmation linéaire et quadratique, conçues par R. Gabassov et al.[29]. L'avantage de cette méthode réside dans le fait qu'elle manipule les contraintes de problème telles qu'elles se présentent sans chercher à les modifier. De plus, elle permet d'avoir une solution optimale ϵ prés, où ϵ est une précision choisie à l'avance.

Ce travail s'articule autour de cinque chapitres :

Le premier chapitre est consacré à la programmation multiojectif, conceptes de bases et méthodes de résolution, le deuxième chapitre contient des définitions, théorèmes sur la programmation quadratique et ses méthodes de résolution et pour le troisième chapitre est sur la programmation multiobjectif en nombre entier.

Dans le quatrième chapitre 'Méthode adaptée pour la programmation quadratique biobjectif en variables mixtes', nous allons faire une étude explicite de chaque étape de la méthode, tout en donnant aussi l'algorithme de la méthode.

Le dernier chapitre est consacré à l'implémentation de notre méthode adaptée et application sur les problèmes d'optimisation des portefeuilles, nous allons, alors, définir quelques notions du langage Matlab, expliquer le fonctionnement de notre programme et terminer par appliquer cette méthode sur les problèmes d'optimisation des portefeuilles, tout en discutant les résultats obtenus.

Enfin, ce mémoire s'achève par une conclusion générale et quelques perspectives.

Chapitre 1

Programmation Multiobjectif

"Je ne suis pas le produit de mes circonstances. je suis le produit de mes décisions".

(Stephen Covey)

1.1 Introduction

L'optimisation multiobjectif à vu le jour au 19ème siécle avec les travaux en économie d'Edgeworth, reprise ensuite par Pareto en 1896[73].
Dans ce chapitre, nous présenterons les concepts de base de l'optimisation multiobjectif, les définitions principales, et surtout les méthodes de résolutions concernant ce type de problèmes.

1.2 Concepts de Base

1.2.1 Formulation mathématique du problème multiobjectif

Un probleme multiobjectif linéaire (programmation multi−objectif) consiste à optimiser simultanement K fonctions objectifs, souvent contradictoires, notées $Z_i(i = 1....K)$ avec $K \geq 2$.

$$\begin{cases} "optimiser" Z_1 = C^1 X \\ "optimiser" Z_2 = C^2 X \\ \vdots \\ \vdots \\ "optimiser" Z_K = C^K X \\ S.C \\ X \in S \end{cases}$$

Où :

$X = (x_1, x_2, ..., x_n)$ = Vecteur représentant les variables de décision.

$S = \{x \in \mathbb{R}^n \setminus Ax \leq b; x \geq 0\}$ = Ensemble des solutions réalisables associe à des contraintes d'égalité, d'inégalité et des bornes explicites (espace de décision).

$A \in R^{n \times m}; b \in \mathbb{R}; m, n \in N$ et $C^i \in \mathbb{R}^{i \times n}$ pour $i = 1, ..., K$

$Z = (Z_1(x), Z_2(x), \cdot, Z_i(x))$ = Vecteur des critères à optimiser.

Remarque 1

Le symbole "." signifie qu'il n'est, généralement, pas possible de trouver dans S une action qui optimise simultanément les K critères. Ainsi, de cette façon, un problème multicritère est correctement formulé par rapport à la réalité concernée par le problème de décision.

Il faut donc déterminer une action $x^* \in S$, tel que (par rapport aux actions de S) le vecteur $Z(x^*) = (Z_1(x^*), ..., Z_i(x^*))$ soit bon, acceptable selon les préférences du decideur [1], voire optimal, à condition de se doter d'un cadre décisionnel donnant une signification à cette notion. L'action x^* est appelée souvent(solution de meilleur compromis).

1.2.2 Espace des décisions

L'espace $\mathbb{R}^n$ dans lequel se situe l'ensemble des actions S ($S \subset \mathbb{R}$) est appelé "espace des decisions".

1.2.3 Espace des critères

L'éspace $\mathbb{R}^K$ dans lequel se situe Z_s, tel que Z_s est l'image de S dans $\mathbb{R}^K$ par l'application linéaire associée à la matrice $C = (c_1, c_2, \cdots, c_K)$:

1. Un décideur :est un individu (ou un groupe d'individus) qui est face à une situation de décision.

$Z_s = C(S) : x = (x_1, x_2, \cdots, x_n)S \rightarrow C_x = (Z_1, Z_2, \cdots, Z_K)$, Z_S est appelé "éspace des critères ".

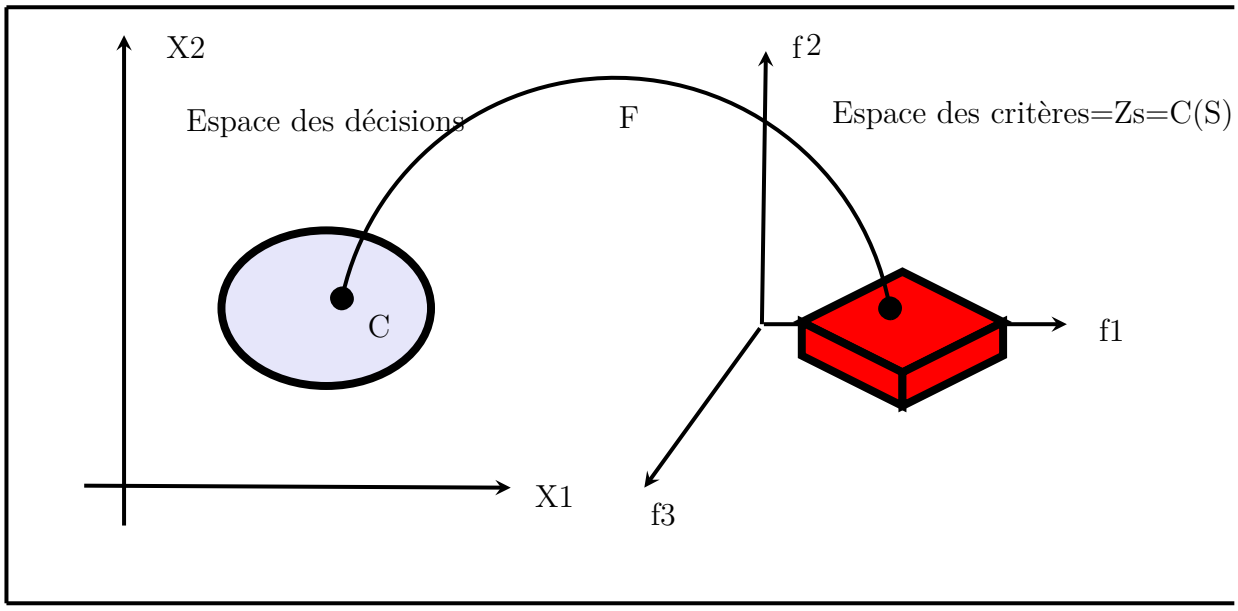

FIGURE 1.1 – Espace des décisions et espace des critères

1.2.4 Dominance

Soient deux vecteurs critères Z, $Z' \in Z_s$. On dit que Z **domine** Z' si et seulement si[15] :

- $Z \leq Z'$ (cas de minimisation)
- $Z \neq Z'$ (c'est-a-dire : $Z_i \leq Z'_i$ pour tout $i = 1, ..., K$, et $Z_i < Z'_i$ pour au moins un indice i).

1.2.5 Dominance forte

Soient deux vecteurs critères Z,$Z' \in Z_s$. On dit que Z **domine fortement** Z' si et seulement si [15] :

- $Z < Z'$(c'est-à-dire : $Z < Z'$ pour tout $i = 1, ..., K$).
- Si Z **domine fortement** Z', alors Z est meilleur que Z' pour tous les critères.

1.2.6 Efficacité

Définition 1 (Solution efficace)

Une solution $x^* \in S$ est une solution **efficace** s'il n'existe pas de $x \in S$ tel que $Z(x)$ **domine** $Z(x^*)$.

Remarque 2

Les solutions efficaces sont aussi connues sous le nom de solutions "Pareto optimales, admissibles et de compromis".

A partir d'un point efficace, il est impossible d'augmenter la valeur d'un critère sans diminuer la valeur d'au moins un autre critère.
L'ensemble des solutions efficaces est appelé "le front Pareto" ou "la surface de compromis".
La notion de "Pareto localement optimale" qualifie une solution qui n'est dominée par aucune solution de son voisinage.
Le voisinage d'une solution x est composé des solutions pouvant être atteintes depuis x à l'aide d'une transformation élémentaire, alors appelée opérateur de voisinage

1.2.7 Point idéal

Les coordonnées du point idéal correspondent aux valeurs obtenues en optimisant chaque fonction objectif séparément. Le vecteur est défini par :[15]

$$\overline{Z} = (min_{x \in s} Z_1(x), ...min_{x \in s} Z_r(x)) \in \mathbb{R}^r$$

1.2.8 Point anti-idéal

Les coordonnées du point anti-idéal correspondent aux pires valeurs de chaque fonction objectif. Le vecteur est défini par :

$$\underline{Z} = (max_{x \in s} Z_1(x), ...max_{x \in s} Z_r(x)) \in \mathbb{R}^r$$

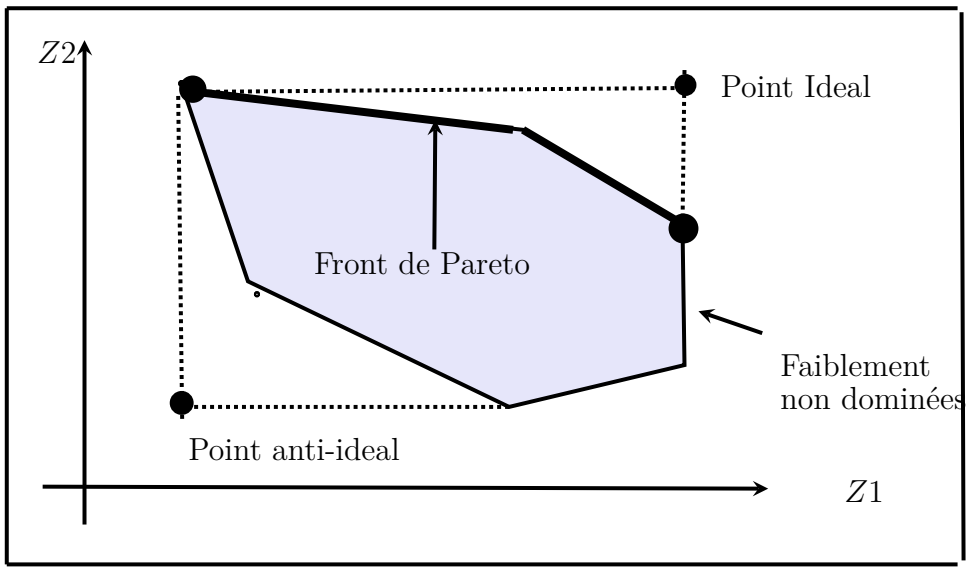

FIGURE 1.2 – Point idéal, point anti-idéal et front de Pareto.

1.2.9 Point Nadir

Les coordonnées du point Nadir correspondent aux pires valeurs de chaque objectif des points du front Pareto. Le vecteur est défini par :

$$n_i = (max_{j=1,..,r} Z_{ij}, i = 1, .., r)$$

1.3 Difficulté d'un problème multiobjectif

La difficulté principale d'un problème multiobjectif est qu'il n'a pas une seule solution optimale, mais potentiellement plusieurs, en fonction des objectifs que l'on suppose plus ou moins importants. Ces solutions sont appelées **solutions de Pareto** et l'ensemble de solutions que l'on obtient à la fin de la recherche est **la surface de compromis**. Il est alors assez difficile d'établir une définition de ce que doit être l'optimum.

C'est aprés avoir trouvé les solutions du problème multiobjectif que d'autres difficultés surviennent : il faut sélectionner une solution dans cet ensemble. La solution qui sera choisie par l'utilisateur va refléter les compromis opérés par le décideur vis-à-vis des différentes fonctions objectif.

Le décideur étant "humain", il va faire des choix et l'un des buts de l'optimisation multiobjectif va être de modéliser les choix du décideur ou plutôt ses préférences. Pour modéliser ces choix, on pourra s'appuyer sur deux grandes théories :

– La théorie de l'utilité multi-attribut.[1]

– La théorie de l'aide à la décision.[71]

Ces deux théories ont des approches différentes.

La première approche va chercher à modéliser les préférences du décideur, en postulant qu'implicitement chaque décideur cherche à maximiser une fonction appelée fonction d'utilité.

La seconde approche va chercher à reproduire le processus de sélection du ou des décideurs. Pour cela, on s'appuiera sur des outils permettant d'opérer des sélections de solutions parmi un ensemble de solutions.

L'autre difficulté à laquelle on sera confronté avant d'effectuer une optimisation sera la sélection de la méthode d'optimisation. Le décideur peut intervenir dans différentes phases de l'optimisation.[31](voir la figure (1.3))

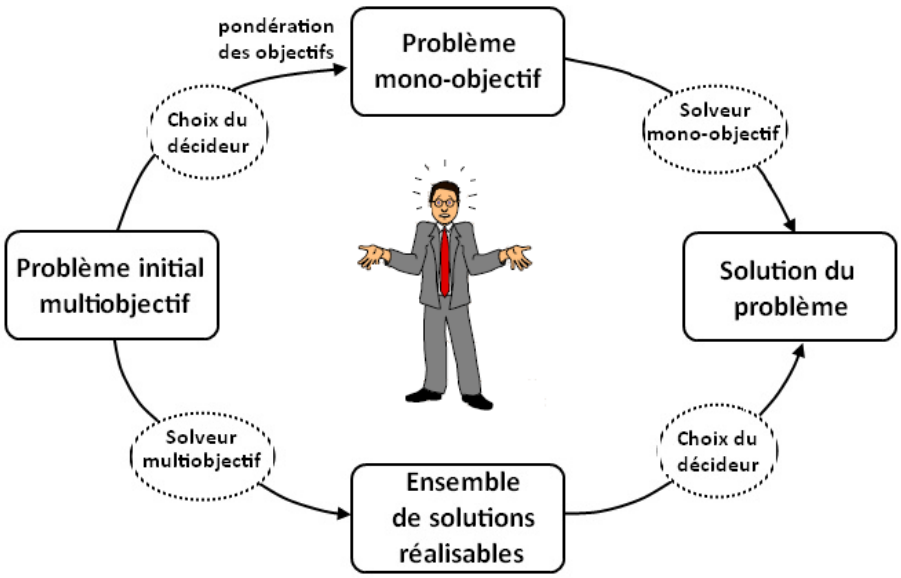

FIGURE 1.3 – Différents types de résolution d'un problème multiobjectif, en fonction de l'intervention du décideur.

1.4 Classification des méthodes de résolution des problèmes multiobjectif

Dans les différentes publications, nous rencontrons deux classifications différentes des approches de résolution de problème multiobjectifs. Le premier classement adopte un point de vue décideur, les approches sont classées en fonction de l'usage que l'on désire en faire. Le deuxième classement adopte un point de vue concepteur, les approches sont triées de leur façon de traiter les fonctions objectifs. Ainsi avant de se lancer dans la résolution d'un problème multiobjectif,

il faut se poser la question du type d'approche de résolution à utiliser.[6]

- **Classification "point de vue décideur"**

  Cette classification est essentiellement utilisée en recherche opérationnelle. Les décisions étant considérées comme un compromis entre les objectifs et les choix spécifiques du décideur (contraintes de coût, de temps, etc.), un décideur choisit une méthode en fonction de l'aide qu'elle va lui apporter.[6] [7]

1.4.1 Approches d'optimisation à priori :

Dans ces méthodes, le compromis que l'on désire effectuer entre les différentes fonctions objectif a été déterminé avant l'éxécution de la méthode d'optimisation. Pour obtenir la solution de notre problème, il suffira de faire une et une seule recherche et, en fin d'optimisation, la solution obtenue reflétera le compromis que l'on désirait effectuer entre les fonctions objectif avant de lancer la recherche.

Cette méthode est intéressante dans le sens où il suffit d'une seule recherche pour trouver la solution. Cependant, il ne faut pas oublier le travail de modélisation du compromis qui a été effectué avant d'obtenir ce résultat. Il ne faut pas, non plus, oublier que le décideur est "humain" et qu'il sera susceptible de constater que la solution obtenue en fin d'optimisation ne le satisfait finalement pas et qu'il souhaite maintenant, aprés avoir examiné cette solution, une solution qui opère un autre compromis entre les fonctions objectif.[15]

1.4.2 Approches d'optimisation progressive (intéractive) :

Dans ces méthodes, on cherchera à questionner le décideur au cours de l'optimisation afin que celui-ci puisse réorienter la recherche vers des zones susceptibles de contenir des solutions qui satisfassent les compromis qu'il souhaite opèrer entre les fonctions objectif.
Ces méthodes, bien qu'appliquant une technique originale pour modéliser les préférences du décideur, présentent l'inconvénient de monopoliser l'attention du décideur tout au long de l'optimisation.

Cet inconvénient n'est pas majeur lorsque l'on a affaire à des problèmes d'optimisation pour lesquels la durée d'une évaluation de la fonction objectif n'est pas importante. Pour les autres problèmes, l'utilisation d'une telle méthode peut être délicate. Il est en effet difficile de monopoliser l'attention du décideur pendant une période qui peut s'étendre sur plusieurs heures en lui posant, de plus, des questions toutes les dix minutes, voire même toutes les heures.[15]

1.4.3 Approches à posteriori :

Dans ces méthodes, on va chercher un ensemble de solutions bien réparties dans l'espace de solutions. Le but sera ensuite de proposer ces solutions au décideur pour qu'il puisse sélectionner la solution qui le satisfait le plus en jugeant les différentes solutions proposées.

Ici, il n'est plus nécessaire de modéliser les préférences du décideur. On se contente de produire un ensemble de solutions que l'on transmettra au décideur. Il y a donc un gain de temps non négligeable vis-à-vis de la phase de modélisation des préférences de la famille des méthodes à priori.
L'inconvénient qu'il nous faut souligner est que, maintenant, il faut générer un ensemble de solutions bien réparties. Cette tâche est non seulement difficile, mais, en plus, peut requérir un temps d'exécution prohibitif.[15]

► **Classification "point de vue concepteur"**

Ce classement adopte un point de vue plus théorique articulé autour des notions d'agrégation et d'optimum de Pareto. Ces notions sont développées dans les sections suivantes car nous adoptons cette classification pour présenter les différentes méthodes.[6] [7]

1.4.4 Approches heuristiques

Les approches heuristiques peuvent êtres divisées en deux classes : d'une part les algorithmes spécifiques à un problème donnée qui utilisent des connaissances du domaine et d'autre part des algorithmes généraux (méta-heuristique) applicables à une grande variété de PMO. Les approches heuristiques ne garantissent pas de trouver de manière exacte tout l'ensemble des solutions Pareto, mais une approximation, aussi bonne que possible, de cet ensemble. Les approches heuristiques peuvent être classées en deux catégories :

- **Approches Pareto**, ces méthodes sont fondées sur la notion de dominance au sens de Pareto qui privilégie une recherche satisfaisant au mieux tous les objectifs. Une seule résolution permet d'approximer l'ensemble de la frontière Pareto.
  Ces approches appartiennent également aux approches de type à "posteriori".[15]
- **Approches non-Pareto**, ne traitent pas le problème comme un véritable problème multiobjectif. Elles cherchent à ramener le problème initial à un ou plusieurs problèmes mono-objectifs.
  Les approches non Pareto sont classées en deux catégories : les approches scalaires, qui transforment le problème multiobjectif en problème mono-objectif et les approches non scalaires, qui gardent l'approche multiobjectif, mais en traitant séparément chacun des objectifs.

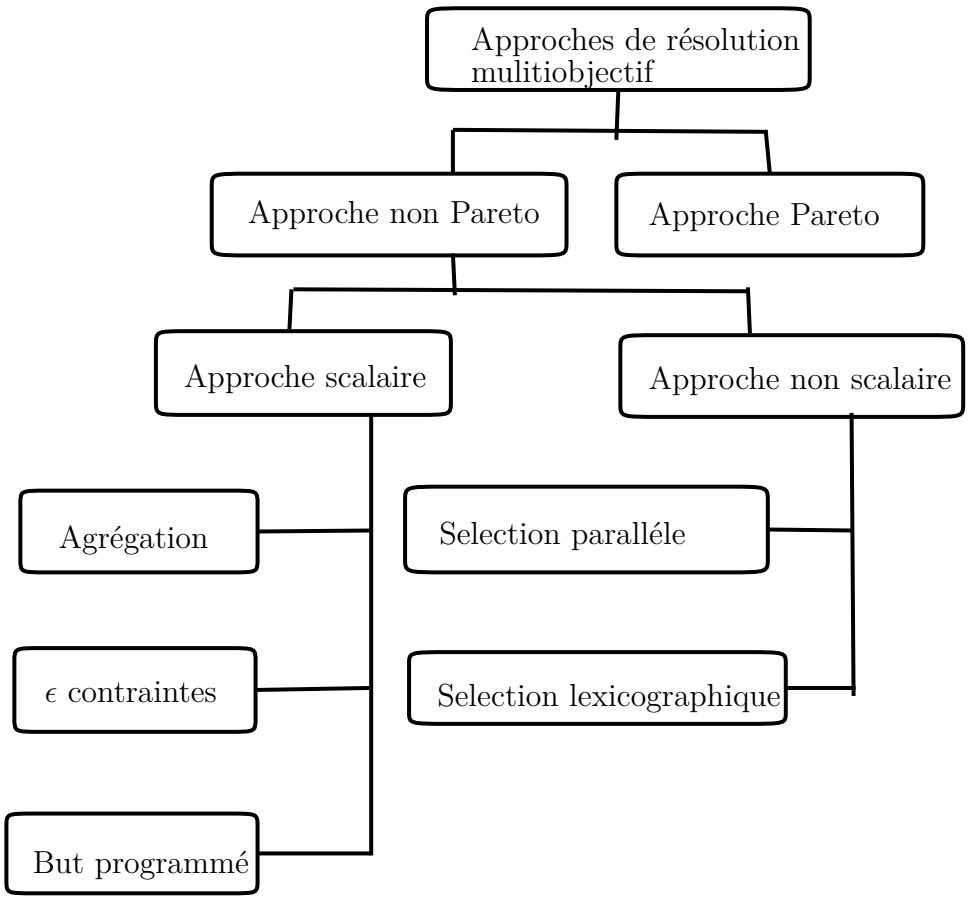

FIGURE 1.4 – Approches de résolution multiobjectif

1.5 Méthodes de résolution des problèmes multiobjectifs

Il existe un nombre important de méthodes, peuvent aussi être rangées en cinq groupes :

★ Méthodes scalaires,

★ Méthodes interactives,

★ Méthodes floues,

★ Méthodes exploitant une métaheuristique,

★ Méthodes d'aide à la décision.

1.5.1 Les méthodes scalaires

A l'origine, les problèmes multiobjectifs étaient transformés en problèmes mono-objectifs. Plusieurs approches différentes ont été mises au point pour transformer les problèmes multiobjectifs en problèmes mono-objectifs, au moyen d'une fonction de scalarisation (fonctions scalarisantes) $f : \mathbb{R}^n \rightarrow \mathbb{R}$. Typiquement f est une fonction des fonctions objectifs du problème, scalaire ou paramètres de préférence.[23]

La méthode de pondération des fonctions objectifs

C'est l'une des premières méthodes employées de la résolution d'un problème d'optimisation multiobjectif, est la plus évidente. D'ailleurs, on appelle aussi cette méthode l'approche naîve de l'optimisation multiobjectif [14]. Le but, ici, est de revenir à un problème d'optimisation monobjectif, dont il existe de nombreuses méthodes de résolution. La manière la plus simple de procéder consiste à prendre chacune des fonctions objectif, à leur appliquer un coefficient de pondération et à faire la somme des fonctions objectif pondérées. On obtient alors une nouvelle fonction objectif [15].

Le problème

$$(P) : \begin{cases} "min" f_i(x), & i = \overline{1, K},\ K \geq 2 \\ avec\ g(x) \leq 0, & x \in \mathbb{R}^n \end{cases}$$

se transforme de la manière suivante

$$(P_{cor.}) : \begin{cases} "min" f_{eq}(x, w) = \sum_{i=1}^{k} w_i f_i(x), & K \geq 2 \\ g(x) \leq 0, & i = \overline{1, K} \\ & x \in \mathbb{R}^n \end{cases}$$

où les poids $w_i \in [0, 1]$ et $\sum_{i=1}^{k} w_i = 1$

Diffèrents poids fournissent diffèrentes solutions supportées. Et la même solution peut être générée en utilisant des poids diffèrents[69].

Critique

Cette méthode est simple à mettre en œuvre et elle est d'une grande efficacité. Mais les difficultés essentielles de cette approche sont[51] :

1. Comment le décideur détermine-t-il les poids de chaque critère ?
2. Comment exprimer l'interaction entre les différents critères ?

L'utilisation de ces modéles impose que les objectifs soient commensurables [2]. Il est donc trés difficile d'utiliser cette méthode lorsque l'ensemble des critères est composé à la fois de critères qualitatifs et quantitatifs[51].

2. Exprimés dans la même unité

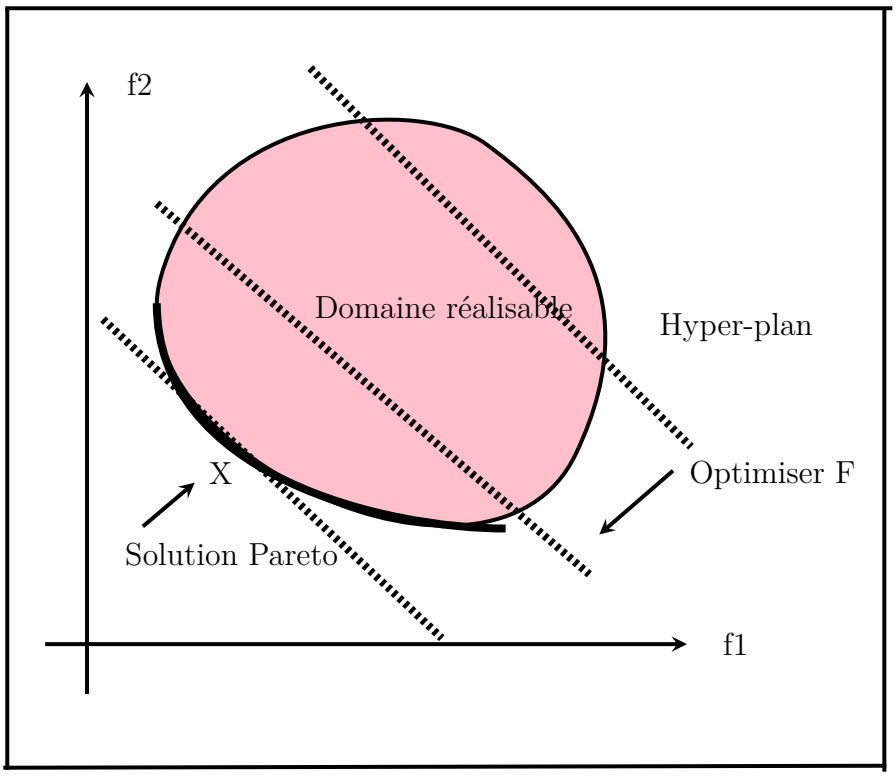

FIGURE 1.5 – Méthode de pondération des fonctions objectifs

La figure(1.5.1) illustre le fonctionnement de la méthode de pondération. Fixer un vecteur poids revient à trouver un hyper-plan dans l'espace objectif (une droite pour un problème biobjectif) avec une orientation fixée. La solution Pareto optimale est le point où l'hyperplan possède une tangente commune avec l'espace réalisable (point x dans la figure (1.5.1)). Donc, pour une agrégation donnée, il n'existe en général qu'une seule valeur optimale pour le problème. Ces approches résolvent le problème en utilisant différentes valeurs pour les poids qui fournissent différentes solutions supportées. Mais, dans le cas d'une frontière Pareto concave, les solutions non supportées sont alors négligées.

But programmé

Dans les approches de ce type, le décideur doit définir des buts T_i ou références qu'il désire atteindre pour chaque objectif f_i. Ces valeurs sont introduites dans la formulation du problème, le transformant en un problème mono-objectif. La nouvelle fonction objectif est modifiée de façon à minimiser les écarts entre les résultats et les buts à atteindre.

$$"min" \sum_{i=1}^{K} |f_i(s) - T_i| \; avec \; s \in \Omega$$

Différentes approches sont envisageables, comme celles du min-max [14], ou du but à atteindre. Ces approches, bien que travaillant par agrégation des objectifs, permettent de générer des

solutions non-supportées.[2]

La méthode de compromis (approche par "ϵ-contrainte)

C'est une autre façon de transformer un problème d'optimisation multiobjectif en un problème mono-objectif est de convertir $K-1$ des K objets du problème en contraintes et d'optimiser séparément l'objectif restant[3]. La démarche est la suivante :

- On choisit un objectif à optimiser prioritairement ;
- On choisit un vecteur de contraintes initial ;
- On transforme le problème en conservant l'objectif prioritaire et en transformant les autres objectifs en contraintes d'inégalité.

On appelle aussi cette méthode la méthode de la "ϵ-contrainte".[52]

On part du problème P ;

On suppose que la fonction objectif prioritaire est la fonction de rang 1 ;

On choisit un vecteur de contraintes $\epsilon_i \geq 0$, i $\in$ 2,...,k ;

On transforme le problème P de la manière suivante[11] :

$$(P_{cor}) \begin{cases} Minimiser f_1(x) \\ f_j(x) \leq \epsilon_j, \text{ pour } j \neq 1 \\ g(x) \leq 0 \\ x \in \mathbb{R}^n \end{cases}$$

En générale le problème peut être reformulé de la manière suivante :

$$(P_{cor}) \begin{cases} Minimiser f_i(x) \\ Avec \quad f_1(x) \leq \epsilon_1 \\ \vdots \\ f_{i-1}(x) \leq \epsilon_{i-1} \\ f_{i+1}(x) \leq \epsilon_{i+1} \\ \vdots \\ f_k(x) \leq \epsilon_k \\ et \quad g(x) \leq 0 \\ x \in \mathbb{R}^n \end{cases}$$

L'approche par ϵ-contrainte doit être appliquée plusieurs fois en faisant varier le vecteur ϵ pour trouver un ensemble de points pareto optimaux. cette approche a l'avantage par rapport aux autres de ne pas être trompée par les problèmes non convexes.[2]

Pour des problèmes simples, le choix des valeurs de ϵ_i peut donner une borne répartition des solutions sur la surface de compromis. En revanche, dans la plupart des cas concrets, le

choix(arbitraire) des valeurs de ϵ_i ne permet pas d'obtenir une bonne répartition des solutions sur la surface de compromis. Alors on ne peut pas avoir assez de points et rencontrer de grandes difficultés pour extrapoler l'allure de la surface de compromis.[11]

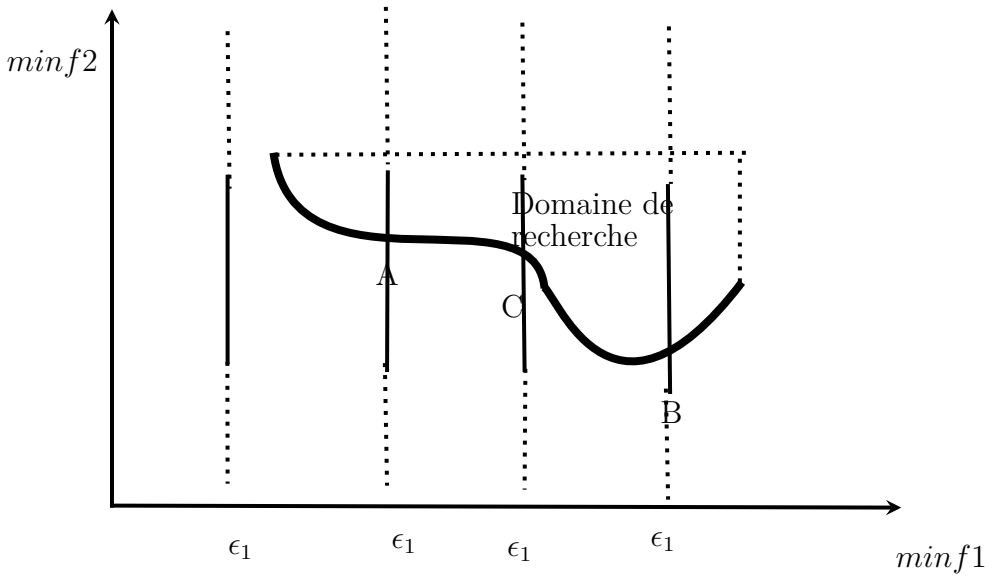

FIGURE 1.6 – Illustration de l'approche de $\epsilon-$ contraintes

La méthode de Keeney-Raiffa

Cette méthode utilise le produit des fonctions objectifs pour se ramener à un problème d'optimisation monobjectif. L'approche utilisée ici est semblable à celle utilisée dans la méthode de pondération des fonctions objectives. La fonction objectif ainsi obtenir s'appelle la fonction d'utilité de Keeney-Raiffa.[15]

La méthodes hybrides

La méthode hybride la plus connue est la méthode de Corley. Cette méthode utilise la méthode de pondération des fonctions objectif et la méthode du compromis. On part du problème P. On le transforme de la manière suivante :[15]

$$(P_{cor.}) : \begin{cases} "Minimiser" & \sum_{i=1}^{K} w_i \cdot f_i(x); \\ telque, & f_j(x) \leq \varepsilon_j,\ j \in 1, ..., K \\ et & g(x) \leq 0 \\ & x \in \mathbb{R}^n \end{cases}$$

1.5.2 Les approches non scalaires (non Pareto)

Ces approches ne transforment pas le problème multiobjectif en un problème mono-objectif, mais utilisent des opérateurs qui traitent séparément les différents objectifs, elles n'utilisent pas non plus la notion de dominance Pareto : sélection parallèle, sélection lexicographique.

Sélection parallèle

Cette approche a été la première proposant un algorithme génétique pour la résolution de problèmes multiobjectifs [68]. L'algorithme proposé, VEGA (Vector Evaluated Genetic Algorithm), sélectionne les individus selon chaque objectif de manière indépendante. L'idée est simple : Pour k objectifs et une population de n individus, une sélection de n/k meilleurs individus est effectuée pour chaque objectif. Ainsi k sous-populations vont être créées et ensuite mélangées afin d'obtenir une nouvelle population de taille n. le processus se termine par l'application des opérateurs génétiques (croisement et mutation).

Sélection lexicographique

Cette approche, proposée par Fourman [Fourman, 1985], elles classent les objectifs en fonction d'un ordre d'importance proposé par le décideur. Ensuite l'optimum est obtenu en optimisant tout d'abord la fonction objectif la plus importante puis la deuxième en intégrant les valeurs obtenues comme contraintes pour la résolution sur des objectifs moins prioritaire et ainsi de suite. La solution obtenue à l'étape k sera la solution du problème. Le risque essentiel de cette méthode est la grande importance attribuée aux objectifs classés en premier. La meilleure solution trouvée pour l'objectif le plus important va faire converger l'algorithme vers une zone restreinte de l'espace d'état et enfermer les points dans une niche.[14] [7]

1.5.3 Méthodes interactives

Les méthodes interactives permettent de chercher une et une seule solution. Elles forment la famille des méthodes progressives et permettent à l'utilisateur de déterminer ses préférences vis-à-vis d'un compromis entre objectifs au cours de l'optimisation.[64]

Ces méthodes sont à comparer aux méthodes a préférences a priori et a préférences a posterioré ;[15]

- Méthode du Simplexe,[56] [54]
- Méthode de compromis par substitution,[41][15]
- Méthode de Fandel,[21]

- Méthode STEP,[21]
- Méthode de Jahn,[21]
- Méthode de Geoffrion.[21]

1.5.4 Méthodes floues ou brouillées

- Méthode de Sakawa,[15][66][67]
- Méthode de reardon.[60][61]

1.5.5 Méthodes exploitant une métaheuristique

Méthodes souvent inspirées de mécanismes d'optimisation rencontrés dans la nature. Elles sont utilisées pour les problèmes où on ne connait pas d'algorithmes de résolution en temps polynomial et pour lesquels on espère trouver une solution approchée de l'optimum global. Elles cherchent à produire une solution de meilleure qualité possible dictée par des heuristiques avec un temps de calcul raisonnable en examinant seulement une partie de l'espace de recherche.
Dans ce cas l'optimalité de la solution n'est pas garanti ni l'écart avec la valeur optimal. Parmi ces heuristiques, on trouve les métaheuristiques qui fournissent des schémas de résolution généraux permettant de les appliquer potentiellement à tous les problèmes.
Plusieurs classifications des métaheuristiques ont été proposées, la plupart distinguent globalement deux catégories : celles se basant sur une solution unique et celles se basant sur une population de solution.

La recherche locale

La recherche locale (RL) est une méthode classique en recherche opérationnelle qui consiste à explorer à une itération donnée le voisinage de la solution courante. L'exploration se fait en modifiant un ensemble de composantes de la solution courante pour se déplacer vers une nouvelle solution. Le processus est répété itérativement jusqu'à ce qu'un critère d'arrêt choisi soit rencontré. Ce type d'approche nécessite la définition de plusieurs concepts : celui de voisinage associé à une solution, et celui de mouvement effectué entre deux itérations. Le voisinage d'une solution est défini en fonction du problème à résoudre. Il représente, par définition, l'ensemble des solutions accessibles depuis la solution courante x.[3][2]

Le recuit simulé (simulated annealing)

Le recuit simulé (Simulated Annealing) s'inspire du processus du recuit physique. Le processus du recuit simulé répète une procédure itérative qui cherche des solutions de coût plus

faible tout en acceptant de manière contrôlée des solutions qui dégradent la fonction de coût.[49]

Avantages et inconvénients de récuit simulé :

Avantages

- La méthode du recuit simulé a l'avantage d'être souple vis-à-vis des évolutions du problème et facile à implémenter.
- Contrairement aux méthodes de descente, sa évite le piège des optima locaux.
- Excellents résultats pour un nombre de problèmes complexes.

Inconvénients

- Nombreux tests nécessaires pour trouver les bons paramètres .
- Définir les voisinages permettant un calcul efficace de ΔE.

Algorithme 1 Récuit simulé

---

1. Engendrer une configuration initiale S_0 de S :s$\leftarrow$ S_0
2. Initialiser la température T en fonction du schéma de refroidissement
3. Répéter
4. Engendrer un voisin aléatoire S' de S
5. Calculer $\Delta E = f(S') - f(S)$
6. Si $\Delta E \leq 0$alors $S \leftarrow S'$
7. Sinon accepter S' comme la nouvelle solution avec la probabilité $P(E,T) = \exp^{\frac{-\Delta E}{T}}$
8. Fin Si
9. Mettre T à jour en fonction de schéma de refroidissement (réduire la température)
10. Jusqu'à la condition d'arrêt
11. Retourner la meilleure configuration trouvée

La recherche Tabou (Tabu Search)

La recherche tabou (TS) est une méthode de recherche locale combinée avec un ensemble de techniques permettant d'éviter d'être piégé dans un minimum local ou la répétition d'un cycle. La recherche tabou est introduite principalement par Glover [34][35][36]. Cette méthode a montré une grande efficacité pour la résolution des problèmes d'optimisation difficiles. En effet, à partir d'une solution initiale s dans un ensemble de solutions local S, des sous-ensembles de solution $N(s)$ appartenant au voisinage S sont générés. Par l'intermédiaire de la fonction d'évaluation nous retenons la solution qui améliore la valeur de f, choisie parmi l'ensemble de solutions voisines $N(s)$.

L'algorithme accepte parfois des solutions qui n'améliorent pas toujours la solution courante. Nous mettons en oeuvre une liste tabou (tabu list) T de longueur k contenant les k dernières solutions visitées, ce qui ne donne pas la possibilité a une solution déjà trouvée d'être acceptée et stockée dans la liste tabou. Alors le choix de la prochaine solution est effectué sur un ensemble des solutions voisines en dehors des éléments de cette liste tabou. Quand le nombre k est atteint, chaque nouvelle solution sélectionnée remplace la plus ancienne dans la liste. La construction de la liste tabou est basée sur le principe FIFO, c'est-à-dire le premier entré est le premier sorti. Comme critère d'arrêt on peut par exemple fixer un nombre maximum d'itérations sans amélioration de $s*$, ou bien fixer un temps limite après lequel la recherche doit s'arrêter.
[5][42]

Algorithme 2 La recherche tabou

1. Initialisation :

   s_0 une solution initiale

   $s \leftarrow s_0$, $s^* \leftarrow s_0$, $C^* \leftarrow f(s_0)$

   $T = \emptyset$

2. Générer un sous-ensemble de solution au voisinage de s

   $s' \in N(s)$ telque $\forall x \in N(s), f(x) \geq f(s')$ et $s' \notin T$

   Si $f(s') < C^*$ alors $s^* \leftarrow s'$ et $C^* \leftarrow f(s')$.

   Mise à jour de T

3. Si la condition d'arrêt n'est pas satisfaite retour à l'étape 2

Avantages et inconvénients :

La recherche tabou est une méthode de recherche locale, et la structure de son algorithme de base est proche de celle du recuit simulé, avec l'avantage d'avoir un paramétrage simplifié : le paramétrage consistera d'abord à trouver une valeur indicative t d'itérations pendant lesquelles les mouvements sont interdits. Il faudra également choisir une stratégie de mémorisation. En revanche, la méthode tabou exige une gestion de la mémoire de plus en plus lourde en mettant des stratégies de mémorisation complexe. L'efficacité de la méthode tabou offre son utilisation dans plusieurs problèmes d'optimisation combinatoire classiques tels que le problème de voyageur de commerce, le problème d'ordonnancement, le problème de tournéees de véhicules, etc.

Les algorithmes génétiques

Les algorithmes génétiques (AG) sont des algorithmes d'optimisation stochastique fondés sur les mécanismes de la sélection naturelle et de la génétique. Ils ont été adaptés à l'optimisation par John Holland [45], également les travaux de David Goldberg ont largement contribué à les enrichir [37]

Le vocabulaire utilisé est le même que celui de la théorie de l'évolution et de la génétique, on emploie le terme individu (solution potentielle), population (ensemble de solutions), génotype (une représentation de la solution), gène (une partie du génotype), parent, enfant, reproduction, croisement, mutation, génération, etc.

Leur fonctionnement est extrêmement simple, on part d'une population de solutions potentielles (chromosomes) initiales, arbitrairement choisies. On évalue leur performance (Fitness) relative. Sur la base de ces performances on crée une nouvelle population de solutions potentielles en utilisant des opérateurs évolutionnaires simples : la sélection, le croisement et la mutation. Quelques individus se reproduisent, d'autres disparaissent et seuls les individus les mieux adaptés sont supposés survivre. On recommence ce cycle jusqu'à ce qu'on trouve une solution satisfaisante. En effet, l'héritage génétique à travers les générations permet à la population d'être adaptée et donc répondre au critère d'optimisation, Un algorithme génétique recherche le ou les extrema d'une fonction définie sur un espace de données. Son mise en oeuvre nécessite :

- **Le codage des données**

  La première étape est de définir et coder convenablement le problème. Cette étape associe à chaque point de l'espace de recherche une structure de données spécifique, appelée génotype ou ensemble de chromosomes, qui caractérisera chaque individu de la population. Le codage de chaque individu en séquence est essentielle dans l'élaboration d'un algorithme génétique dont dépend notamment l'implémentation des opérateurs de transformations. Ainsi, cette phase détermine la structure de données qui sera utilisée pour coder le génotype des individus de la population. Le codage doit donc être adapté au problème traité. Plusieurs types de codages sont utilisés dans la littérature, les premiers résultats théoriques sur les algorithmes génétiques ont opté pour un codage par une séquence binaire de longueur fixe à travers la notion de schéma [37]. L'efficacité de l'algorithme génétique dépend donc du choix convenable du type de codage.

- **Génération de la population initiale**

  La génération de la population initiale, c'est-à-dire le choix des dispositifs de départ que nous allons faire évoluer, ce choix de la population initiale d'individus conditionne fortement la rapidité de l'algorithme. Néanmoins, une initialisation aléatoire est plus simple à réaliser : les valeurs des gènes sont tirées au hasard selon une distribution uniforme.

Toutefois, il peut être utile de guider la génération initiale vers des sous domaines intéressants de l'espace de recherche. Par exemple lors d'une recherche d'optima dans un problème d'optimisation sous contraintes, il est préférable de produire des éléments satisfaisant les contraintes. La population initiale doit être suffisamment diversifiée et de taille assez importante pour que la recherche puisse parcourir l'espace d'état dans un temps limité.

- **Fonction d'adaptation (Fitness)**

  L'évaluation de la Fitness est généralement l'étape dans laquelle on mesure la performance de chaque individu. Pour pouvoir juger la qualité d'un individu et ainsi le comparer aux autres, il faut établir une mesure commune d'évaluation. Aucune règle n'existe pour définir cette fonction, son calcul peut ainsi être quelconque, que ce soit une simple équation ou une fonction affine. La manière la plus simple est de poser la fonction d'adaptation comme la formalisation du critère d'optimisation.

- **Sélection**

  La sélection permet d'identifier statistiquement les meilleurs individus d'une population et d'éliminer les mauvais, pendant le passage d'une génération à une autre, ce processus est basé sur la performance de l'individu. L'opérateur de sélection doit être conçu pour donner également une chance aux mauvais éléments, car ces éléments peuvent, par croisement ou mutation, engendrer une descendance pertinente par rapport au critère d'optimisation. Il existe différentes techniques de sélection, on propose quelques unes.

  Sélection uniforme : On ne s'interesse pas à la valeur d'adaptation fitness et la sélection s'effectue d'une manière aléatoire et uniforme telle que chaque individu i a la même probabilité Prob(i) = $1/T_{pop}$ comme tous les autres individus (T_{pop} est la taille de la population).

  Sélection par tournoi : Deux individus sont choisis au hasard, on compare leurs fonctions d'adaptation et le mieux adapté est sélectionné.

  Élitisme : Cette méthode de sélection permet de favoriser les meilleurs individus de la population. Ce sont donc les individus les plus prometteurs qui vont participer à l'amélioration de notre population. On peut constater que cette méthode induisait une convergence prématurée de l'algorithme.

- **Croisement**

  L'opérateur de croisement favorise l'exploration de l'espace de recherche et enrichit la diversité de la population en manipulant la structure des chromosomes, le croisement fait avec deux parents et génère deux enfants, en espérant qu'un des deux enfants au moins héritera de bons gènes des deux parents et sera mieux adapté qu'eux. Il existe plusieurs méthodes de croisement par exemple le croisement en un point, ou en multiples points

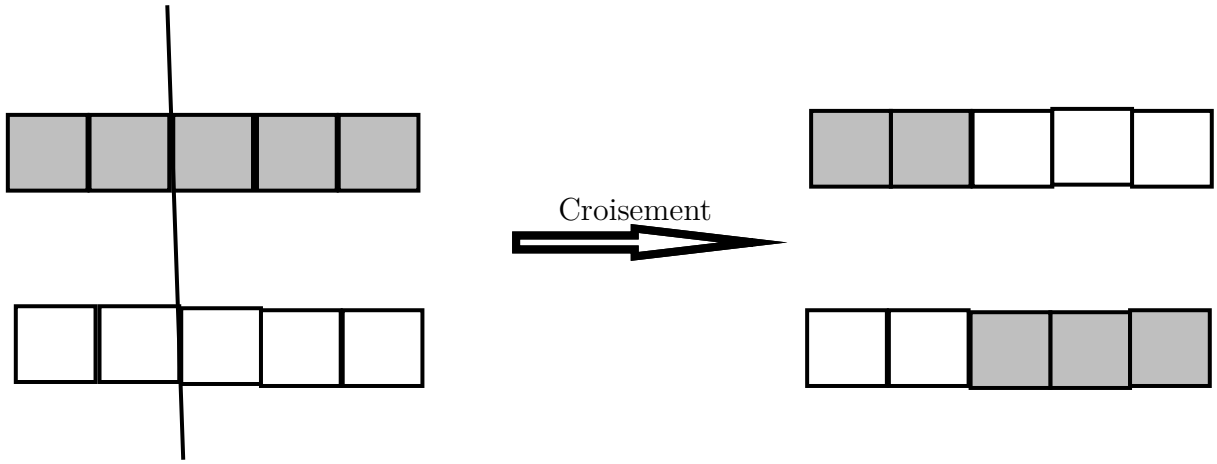

FIGURE 1.7 – Croisement en un point.

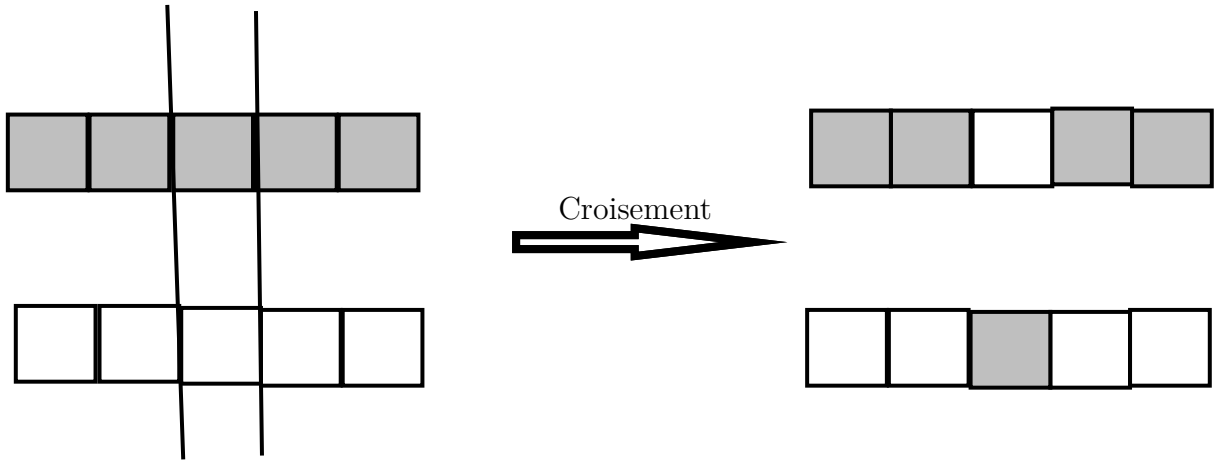

FIGURE 1.8 – Croisement en deux points.

– **Mutation**

L'opérateur de mutation est un processus où un changement mineur du code génétique appliqué à un individu pour introduire de la diversité et ainsi d'éviter de tomber dans des optimums locaux. Cet opérateur est appliqué avec une probabilité P_m généralement inférieure à celle du croisement P_c

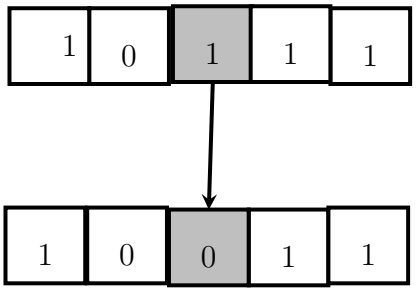

FIGURE 1.9 – Représentation schématique d'une mutation (codage binaire)

.

L'efficacité des algorithmes génétiques dépend fortement du réglage des différents paramètres caractérisant ces algorithmes, et qui sont parfois difficiles à déterminer. Des paramètres comme la taille de la population, le nombre maximal des générations, la probabilité de mutation p_m, et la probabilité de croisement p_c.

Les deux premièrs paramètres dépendent directement de la nature du problème et de sa complexité, et leurs choix doit représenter un compromis entre la qualité des solutions et le temps d'exécution.

La probabilité de croisement p_c est liée à la forme de la fonction d'évaluation. Son choix est en général heuristique. Plus sa valeur est élevée, plus la population subit des changements importants.

La probabilité de mutation p_m est généralement faible puisqu'un taux élevé risque de conduire vers un optimum local. En revanche, une probabilité faible permet d'assurer une bonne exploration de l'espace de recherche sans perturber la convergence.

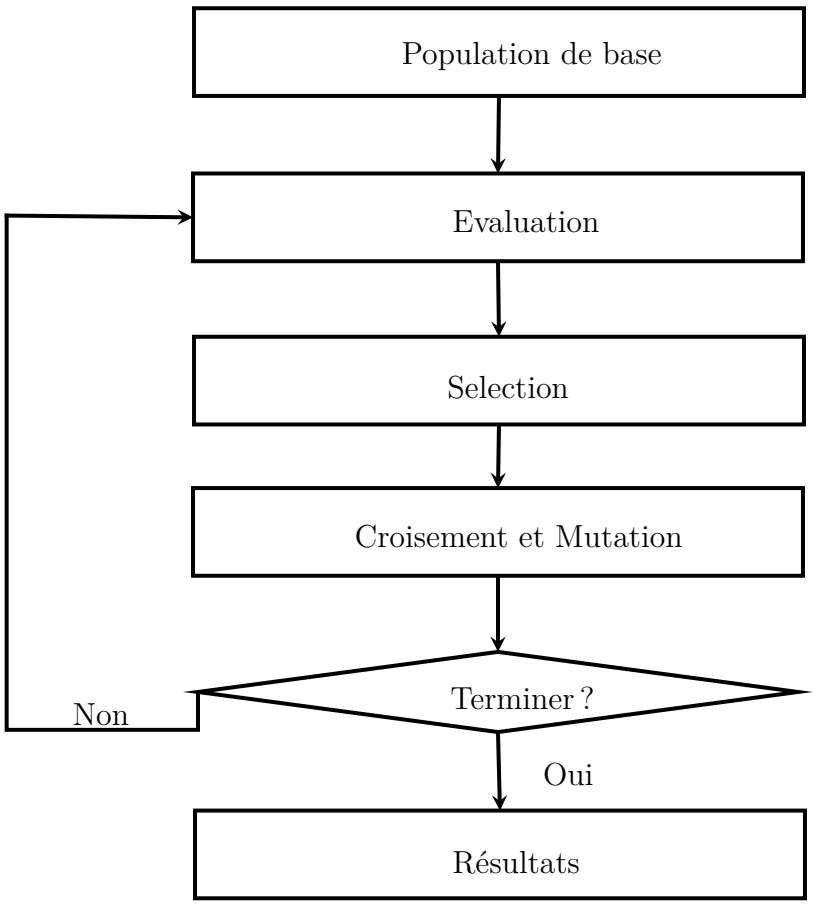

FIGURE 1.10 – Fonctionnement d'un algorithme génétique.

1.5.6 Méthodes d'aide à la décision

Chacune de ces méthodes traite une problématique bien précise. Les éléments importants qui différencient ces méthodes sont la définition de la relation de classement des actions (chaque méthode exploite une relation de préférence différente) et la méthode d'exploitation des résultats.[15][72][23][70

- La méthode ELECTRE I
- La méthode ELECTRE IS
- La méthode ELECTRE II
- La méthode ELECTRE III
- La méthode ELECTRE IV
- La méthode ELECTRE TRI
- La méthode PROMETHEE I
- La méthode PROMETHEE II

1.6 Conclusion

A travers ce chapitre, nous avons présenté, d'une part les définitions nécessaires à la compréhension des problèmes d'optimisation multi-objectifs, et d'autre part, les différentes problèmatiques liées aux spécificités du multi-objectif, comme l'intervention du décideur dans le processus de décision et les choix des méthodes d'optimisation à utiliser pour insister sur l'étendue du spectre des recherches dans ce domaine.
Dans le prochain chapitre nous présenterons la programmation quadratique.

Chapitre 2

Programmation Quadratique

"De la même façon que personne n'est capable d'expliquer pourquoi les étoiles sont belles, c'est difficile d'exprimer la beauté des mathématiques".

(Yoko Ogawa)

2.1 Introduction

En optimisation mathématique, un problème d'optimisation quadratique est un problème d'optimisation dans lequel on minimise (ou maximise) une fonction quadratique.
Dans ce chapitre, nous parlerons d'abord sur la programmation linéaire, la programmation non linéaire et sur la programmation quadratique(défintions, formules, méthodes de résolutions).

2.2 Programmation linéaire

La programmation linéaire est une technique fondamentale de la recherche operationnelle qui permettre la résolution des nombreux des problèmes, elle consiste à modilisé les problémes sous forme d'une fonction objectif linéaire à n inconnus qu'on cherche à l'optimiser soumise à des contraintes aussi linéaire formuler comme des égalités ou des inégalités non strictes ou les deux en même temps afin de les résoudre.

Définition 2 (Programme linéaire[10])

Un programme linéaire est tout problème d'optimisation dans lequel :

- Le domaine D des solutions réalisables est défini par un ensemble des égalités ou des inégalités linéaires ou les deux à la fois appelées "contraintes". Notons que les inéquations linéaires strictes sont interdites car elle sont dépourvues de signification physique ;
- La fonction Z, dite "fonction objectif ou économique" est linéaire ;
- Toute les variables de décisions sont astreintes à être non négatifs.

2.2.1 Formes d'un programme linéaire

Un programme linéaire peut être écrit sous une :

Forme générale : [63][10]

Tout programme linéaire s'écrit sous la forme générale suivante :

$$\begin{cases} Max \ ou \ Min(Z) = \sum_{j=1}^{n} c_j x_j. \\ \forall i = 1, 2, \cdots, m : \sum_{j=1}^{n} a_{ij} x_j \leq, =, \geq b_i. \\ \forall i = 1, 2, \cdots, n : x_j \geq 0. \end{cases}$$

Où :

Z = la fonction objectif (ou économique) qu'on cherche à l'optimiser ;

x_j = variable de décision ;

a_{ij} = valeur donner selon le probleme posé ;

b_i = le second membre des contraintes ;

c_j = les coûts des variables dans la fonction objectif.

Définition 3 (Solution réalisable)

Une solution réalisable est toute affectation des variables qui respecte les contraintes du problème ;

$$D = \{x \in \mathbb{R}^n \setminus \sum_{j=1}^{n} a_{ij} x_j \leq, =, \geq b_i; x_j \geq 0.\}$$

Définition 4 (Point extrême)

$x^0 \in D$, est un point extrême s'il ne peut pas être exprimé à l'aide d'une combinaison linéaire de deux autres points de D [10] :

$$\nexists x, y \in D \ \ tq : x^0 = \lambda x + (1-\lambda)y; \lambda \in [0,1]$$

Forme canonique :[10]

Un probléme de maximisation(resp. minimisation) doit être écrit sous la forme canonique comme suite :

$$\begin{cases} Max \ \ (resp.Min) \ \ Z = C^T x. \\ s.c \ \ Ax \leq (resp. \geq) b. \\ x \geq 0. \end{cases}$$

Où :

$x = (x_1, x_2, x_3, \cdots, x_n)^t$ vecteur des variables de décision ;

$b = (b_1, b_2, b_3, \cdots, b_n)^t$ vecteur de second membre de taille m ;

$C = (c_1, c_2, c_3, \cdots, c_n)$ vecteur coût de taille n ;

$A =$ la matrice des contraintes de taille $m \times n$.

Forme standard [10] :

On dit qu'un programme linéaire est écrit sous forme standard si toutes les contraintes(sauf les contraintes de non-négativité) sont des égalités d'où le PL devient :

$$\begin{cases} Max \ \ (resp.Min) \ \ Z = C^T x. \\ s.c \ \ Ax = b. \\ x \geq 0. \end{cases}$$

2.2.2 La notion de dualité

La dualité est un concepte fondamentale de la programmation linéaire et conduit à un résultat de grande portée théorique et pratique.

A tout probléme de programmation linéaire on peut associer un autre probléme dual qui est de même nature que lui.

En fait, il faut considérer que deux problèmes linéaires duaux ne constituent pas deux problèmes distincts, mais deux aspects du même problème.

Le dual d'un PL sous forme canonique :[63] Considérons le programme linéaire(P) écrit sous la forme canonique :

$$\begin{cases} Max\ \ Z = Cx. \\ s.c\ \ Ax \leq b. \\ x \geq 0. \end{cases}$$

Son dual est le programme linéaire suivant :

$$\begin{cases} Min\ \ W = b^t y. \\ yA^t \geq c. \\ y \geq 0. \end{cases}$$

fonction objectif Max	fonction objectif Min
le vecteut coùt	le vecteut de second membre
A matrice des contraintes	A^t matrice des contraintes
contrainte $i =$	variable y_i quelconque
contrainte $i \leq$	variable $y_i \geq$
contrainte $i \geq$	variable $y_i \leq$
variable $x_j \geq$	contrainte $j \geq$

- Si l'un des problèmes (PL) ou (D) posséde une solution optimale finie, il en est de même pour l'autre et leurs valeurs optimales sont égales.
- Si l'un des problèmes (PL) ou (D) est non borné, alors le domaine réalisable de l'autre est vide.

Note : le dual du dual est le problème de départ appelé primale. [10]

2.2.3 Méthodes de résolution d'un programme linéaire

Parmi les méthodes que les mathématiciens ont eu à utiliser on peut citer :

1. La méthode d'énumération.
2. La méthode géométrique ou graphique
3. La méthode du simplexe.

1- Méthode d'énumération

C'est la méthode la plus ancienne, elle est basée sur le fait que la solution optimale est un point extrême du polyèdre défini par les contraintes. Elle se résume selon les étapes suivantes :

1. Déterminer tous les points extrêmes.
2. Evaluer la fonction objectif au niveau de chacun de ces points.
3. Comparer les valeurs obtenues.

Cependant cette méthode est très fastidieuse si nous avons à faire à des problèmes où les points extrêmes sont nombreux ou bien les composantes des vecteurs sont très nombreuses.[50]

2- Méthode géométrique ou graphique

L'ensemble des points qui satisfont à un système d'inéquations à deux inconnues est un polygone convexe ; s'il y a plus de deux variables, c'est un polyèdre convexe. Dans la suite cet ensemble de points sera appelé domaine réalisable ou domaine des solutions. La méthode graphique est basée sur le fait que si le problème admet au moins une solution optimale, alors celle - ci est l'un des points extrêmes du domaine réalisable.
Dans le cas de deux variables, la fonction objectif est de la forme :

$$Z = F(x_1, x_2) = \lambda_1 x_1 + \lambda_2 x_2$$

Pour $\lambda_1 \neq 0, \lambda_2 \neq 0$ et quel que soit f_1 fixé, la fonction objectif est une équation linéaire à deux variables, $\lambda_1 x_1 + \lambda_2 x_2 = f_1$ et sa représentation graphique est une droite passante par les points$(0, \frac{f_1}{\lambda_2})$ et $(\frac{f_1}{\lambda_1}, 0)$ En faisant varier f, on obtient une famille de droites qui sont appelées droites d'appui parallèles à la droite d'équation $\lambda_1 x_1 + \lambda_2 x_2 = f_1$. Les droites les plus proches de l'origine des coordonnées rendent minimale la fonction $F(x_1, x_2)$ tandis que les plus éloignées la rendent maximale. La solution au problème est soit un point ou bien un segment de droite. Dans le cas où la solution est un point, ce dernier est l'intersection de la droite d'appui la plus extrême parmi toutes les droites d'appui avec le polygone des solutions, selon qu'il s'agit d'une minimisation ou d'une maximisation. En notant $A \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}$ ce point, la valeur de la fonction objectif est alors : $f(A) = a_1\lambda_1 + a_2\lambda_2$, la solution est un segment, si l'intersection définie précédemment était un segment. En notant $B \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$ et $C \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$ les points extrêmes de ce segment, les autres points solutions sont une combinaison convexe linéaire de ces deux points. Cette méthode malgré sa simplicité n'est applicable que pour des problèmes de dimensions 2 ou 3 au maximum et il est parfois impossible de donner la valeur exacte de la solution.[50]

Méthode du simplexe

L'algorithme de simplexe est une des méthodes les plus utilisés depuis de nombreuses années dans le cadre de la recherche operationnelle pour la résolution des problèmes de programmation linéaire, cette méthode a été conçue en 1947 par le mathématicien George B.Dantzig[17]. Elle a pris plus de quarante ans de développement par des travaux de recherche qui ont aboutis sur plusieurs améliorations en terme de performance et de stabilité.
Il existe deux versions principales du simplexe :

– La première méthode dite standard où l'algorithme est appliqué sur une version non modifiée du problème. Cette méthode est aussi appelée "full table"(tableau complet), elle est basé sure une évolution de la matrice compléte modélisant le problème .
– La seconde méthode est la méthode révisé. Cette dernière utilise une décomposition du problème de base en sous-matrices. Cette modélisation du problème à l'aide de quelques astuces, diminue la quantité de calculs nécessaires afin de faire évoluer la matrice vers une solution optimale.

Le principe de la méthode du simplexe

Dans tous qui suit on considére un problème de maximisation. Pour obtenir la solution réalisable d'un (PL) il faut le rendre premièrement sous forme standard, puis, si on a un programme linéaire standard constitué de n variables et m contraintes alors une solution de base est obtenue en annulant $(n-m)$ variables et en résolvant les m contraintes pour déterminer les valeurs des autres m variables, généralement une telle solution peut être retrouvée en annulant toutes les variables de décision.
Apartir de cette solution, la méthode de simplexe va générer successivement des solutions réalisables de base pour notre systeme d'équations en s'assurant que la valeur de la fonction objectif est en train d'augmenter jusqu'à localiser la solution optimale du problème qui est un point extrême de l'espace des solutions réalisables donc une solution réalisable de base. Ainsi, on peut décrire la méthode de simplexe comme étant une procédure itérative qui passe d'une solution réalisable à une autre jusqu'à atteindre la solution optimale.[63][10]

L'algorithme du simplexe[10]

Pour pouvoir appliquer l'algorithme du simplexe il est nécessaire de connaitre une base réalisable. On suppose donc que l'on dispose d'une telle base de départ. A l'itération k. Soit B la base courante.

Algorithme 3 Algorithme de simplexe

Début

étape 1 : Rechercher une solution optimale à partir d'une solution réalisable de base :$x_B = A_B^{-1}b$

étape 2 : Examiner les $(z_j - c_j)$ pour toutes les colonnes a_j qui ne sont pas dans la base.

si $(z_j - c_j) \geq 0$ **alors**

passer à l'étape 6

sinon

passser à l'étape 3.

finsi

étape 3 :

si il existe un $(z_j - c_j)$ négatif pour lequel il n'y a pas d'élément positif dans b **alors**

arrêter la recherche puisque la solution optimale est infinie.

sinon

choisir le vecteur(et donc la variable) associée à la valeur la plus négative des $(z_j - c_j)$ pour entrer dans la base : $(z_k - c_k) = \min_{j \in J}(z_j - c_j)$ avec $(z_j - c_j) < 0$

finsi

étape 4 : Déterminer le vecteur(et donc la variable) qui sort de la base à l'aide du critère :

$x_{Br}/y_{rk} = min_i(x_{Bi}/y_{ik}, y_{ik}) > 0$

étape 5 : Etablir la nouvelle base, calculer la nouvelle solution réalisable de base et la nouvelle valeur de la fonction objectif. retourner à l'étape 2.

étape 6 : la solution réalisable de base courante est la solution optimale. La solution optimale n'est pas unique s'il existe un $(z_j - c_j)$ nul pour un vecteur a_j qui ne se trouve pas dans la base.

2.3 Programmation non linéaire

2.3.1 Formes d'un programme non linéaire

Nous parlons de modèle de programmation non linéaire lorsque l'on doit maximiser ou minimiser une fonction sous contraintes et que cette fonction objectif a au moins une contrainte est non linéaire.

Nous distinguons trois types de problèmes non linéaire :

- **Problème sans contraintes** : $\min_{x \in \mathbb{R}^n} f(x)$
- **Problème avec contraintes de type égalité** :

  $\min_{x \in S} f(x)$ avec S de la forme : $S = \{x \in \mathbb{R}^n \text{tq} : g_i(x) = 0\}$ pour $i \in \{1...n\}$ avec $g_i : \mathbb{R}^n \to \mathbb{R}$

- **Problème avec contraintes de type inégalité** :

  $\min_{x \in S} f(x)$ avec S de la forme :$S = \{x \in \mathbb{R}^n \text{ tq} : h_i(x) \leq 0\}$ pour $i \in \{1...n\}$ avec $h_i : \mathbb{R}^n \to \mathbb{R}$

2.3.2 Existence d'une solution

Théorème 2.1 *(Théorème de Weierstrass)*

Soit f une fonction continue sur $S \subset R^n$ avec S fermé borné. Alors :

$\exists\ x_0 \in S \quad tel\ que \quad f(x_0) = \min_{x \in S} f(x)$

$\exists\ x_1 \in S \quad tel\ que \quad f(x_1) = \max_{x \in S} f(x)$

cela signifie que f est bornée sur S, $f(x_0) \leq f(x) \leq f(x_1)$ et ses bornes (le minimum et le maximum) sont atteint dans ce cadre là.

On veut résoudre $\min_{x \in S} f(x)$ ou $\max_{x \in S} f(x)$ ie on cherche v valeur optimale et x_0 tel que $f(x_0) = v$.

Définition 5

Soit $x_0 \in S$:

1. On dit que x_0 est un minimum local si : $\exists v \in V(x_0) \quad telque$
   $\forall x \in V : f(x) \geq f(x_0)$
2. On dit que x_0 est un minimum local strict si : $\exists v \in V(x_0) \quad telque$
   $\forall x \in V : f(x) > f(x_0)$
3. On dit que x_0 est un minimum global si : $\forall x \in S, f(x) \geq f(x_0)$
4. On dit que x_0 est un minimum global strict si : $\forall x \in S, f(x) > f(x_0)$

On définit de la même façon un maximum local, maximum local strict, maximum global, maximum global strict, en renversant les inégalités .

2.3.3 Condition du premier ordre

Dérivabilité-notion de gradient

Définition 6

Si $f : \mathbb{R}^n \to \mathbb{R}$ est différentiable alors on définit le gradient de f en x par : $\begin{pmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \\ \vdots \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{pmatrix}$

On le note $\nabla f(x)$.

Recherche des points stationnaires

La recherche de minimum local ou global consiste à chercher d'abord les points stationnaires.

Définition 7

x_0 est un point stationnaire si et seulement si $\nabla f(x_0) = 0$.

Proposition 2.1 (Condition nécessaire d'optimalité)
Supposons que S est ouvert et que f est différentiable sur S. Si x_0 est un minimum ou maximum local alors $\nabla f(x_0) = 0$

Corollaire 2.1 Les minima et les maxima locaux sont des points stationnaires et par conséquent, le minimum global et le maximum global aussi.
Un point stationnaire qui n'est ni un maximum ni un minimum est un point singulier (point selle ....).

Point selle : Un point selle ou col pour f : $\mathbb{R}^2 \to \mathbb{R}$ est un point $x_0 = (x_0^1, x_0^2)$ tel que Δ_1,Δ_2 deux droites passant par x_0 telles que f(Δ_1) présente un maximum en x_0 et f(Δ_2) présente un minimum en x_0.

2.3.4 Conditions du second ordre

La matrice Hessienne

Définition 8

Si $f : \mathbb{R}_n \to \mathbb{R}$ est de classe C^2 alors on définit la matrice Hessienne de f en x par :

$$Hf(x) = \begin{pmatrix} \frac{\partial^2 f(x)}{\partial^2 x_1} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \cdots & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \cdots & \cdots & \cdots & \frac{\partial^2 f(x)}{\partial^2 x_n} \end{pmatrix}$$

Proposition 2.2 $Hf(x)$ est une matrice symétrique car :
$\frac{\partial^2 f(x)}{\partial x_i \partial x_j} = \frac{\partial^2 f(x)}{\partial x_j \partial x_i}$ dés que f est deux fois dérivable (Schwarz).

Remarque 3

1. Pour connaitre est ce que la fonction quadratique est définie ou semi-définie positive on applique le critère de sylvester sur sa matrice hessienne.
2. La programmation quadratique n'est qu'un cas particulier de la programmation non linéaire donc tous les définitions de cette dernière sont valable pour la première.

2.4 Programmation quadratiques

L'optimisation quadratique est l'une des théories de la programmation mathématique la plus utilisée pour modéliser des problèmes pratiques. Cette branche est trés importante d'un point de vue pratique que théorique, de nombreux domaines d'application ont été touchés par cette théorie, notamment l'économie, les sciences de l'ingénieur,la physique, etc...

2.4.1 Représentation d'une forme quadratique

Définition 9

Une forme quadratique de dimension n est une fonction réelle de n variables $x_1, x_2, ..., x_n$ ayant la forme suivante :

$$F(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_i x_j \qquad (1)$$

2.4.2 Représentation matricielle d'une forme quadratique

Soient $x = (x_1, x_2, ..., x_n)$ *et* $A = (a_{ij}, 1 \leq i \leq n, 1 \leq j \leq n)$. Pour $n = 3$ et d'aprés la formule (1) on a :

$$F(x) = \sum_{i=1}^{3}\sum_{j=1}^{3} a_{ij}x_i x_j$$

$$F(x) = a_{11}x_1^2 + a_{12}x_1x_2 + a_{13}x_1x_3 + a_{21}x_2x_1 + a_{22}x_2^2 + a_{23}x_2x_3 + a_{31}x_3x_1 + a_{32}x_3x_2 + a_{33}x_3^2$$

$$= x_1(a_{11}x_1 + a_{12}x_2 + a_{13}x_3) + x_2(a_{21}x_1 + a_{22}x_2 + a_{23}x_3) + x_3(a_{31}x_1 + a_{32}x_2 + a_{33}x_3) =$$

$$\begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} \begin{pmatrix} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 \end{pmatrix} = \begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

$$= x^t Ax$$

la forme (1) peut encore s'écrire comme suit :

$$F(x) = a_{11}x_1^2 + a_{22}x_2^2 + a_{33}x_3^2 + x_1x_2(a_{12} + a_{21}) + x_1x_3(a_{13} + a_{31}) + x_3x_2(a_{32} + a_{23})$$

En posant $d_{ij} = d_{ji} = (a_{ij} + a_{ji}) \setminus 2, \forall i, j \in \{1, ..., n\}$ on remarque que pour $i \neq j$ le coefficient devant le produit $\{x_i x_j\}$ est égale à $(a_{ij} + a_{ji})$ on aura alors :

$$F(x) = x^t Ax = x^t Dx$$

où $D = \{d_{ij}, 1 \leq i, j \leq n\}$ est une matrice symétrique .

2.4.3 Gradient d'une forme quadratique

Définition 10

Soit $F : \mathbb{R}^n \to \mathbb{R}$ une fonction réelle continément différentiable. Son gradient au point x est défini par :

$$\nabla F(x) = \begin{pmatrix} \frac{\partial F}{\partial x_1} \\ \frac{\partial F}{\partial x_2} \\ \vdots \\ \frac{\partial F}{\partial x_n} \end{pmatrix}. \tag{2.1}$$

Soit une forme quadratique et D sa matrice symétrique associée :

$$F(x) = x^T Dx. \tag{2.2}$$

En écrivant la matrice D sous forme de vecteurs colonnes

$$D = (d_1, d_2, \ldots, d_n),$$

l'expression (2.2) peut se mettre sous la forme suivante :

$$F(x) = (x_1, x_2, \ldots, x_j, \ldots, x_n) \begin{pmatrix} d_1^T x \\ d_2^T x \\ \vdots \\ d_j^T x \\ \vdots \\ d_n^T x \end{pmatrix} = \sum_{j=1}^{n} x_j d_j^T x.$$

La dérivée partielle de F par rapport à chaque variable x_j est donnée par :

$$\begin{aligned} \frac{\partial F}{\partial x_j} &= x_1 d_{1j} + \cdots + x_{j-1} d_{(j-1)(j)} + d_j^T x + x_j d_{jj} + \cdots + x_n d_{nj} \\ &= x_1 d_{1j} + \cdots + x_{j-1} d_{(j-1)(j)} + x_j d_{jj} + \cdots + x_n d_{nj} + d_j^T x \\ &= 2 d_j^T x. \end{aligned}$$

Par conséquent, le gradient de $F(x)$ est :

$$\nabla F(x) = 2Dx. \tag{2.3}$$

Définition 11

Soit une fonction réelle de classe $\mathcal{C}^2, F : \mathbb{R}^n \to \mathbb{R}$. Le Hessien de la fonction F est défini par :

$$\begin{aligned} \nabla^2 F(x) &= \left(\nabla \frac{\partial F}{\partial x_1}, \nabla \frac{\partial F}{\partial x_2}, \ldots, \nabla \frac{\partial F}{\partial x_j}, \ldots, \nabla \frac{\partial F}{\partial x_n} \right) \\ &= \begin{pmatrix} \frac{\partial^2 F}{\partial^2 x_1} & \frac{\partial^2 F}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 F}{\partial x_1 \partial x_n} \\ \frac{\partial^2 F}{\partial x_2 \partial x_1} & \frac{\partial^2 F}{\partial^2 x_2} & \cdots & \frac{\partial^2 F}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 F}{\partial x_n \partial x_1} & \frac{\partial^2 F}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 F}{\partial^2 x_n} \end{pmatrix}. \end{aligned} \tag{2.4}$$

Définition 12

Soit $F : \mathbb{R}^n \to \mathbb{R}$ une fonction de classe $\mathcal{C}^1$. La dérivée directionnelle de F dans la direction d au point x est :

$$\begin{aligned} \frac{\partial F}{\partial d} &= \lim_{t \to 0^+} \frac{F(x + td) - F(x)}{t} \\ &= \frac{\partial F(x + td)}{\partial x_1}|_{t=0}\ d_1 + \cdots + \frac{\partial F(x + td)}{\partial x_n}|_{t=0}\ d_n \\ &= \nabla F(x)^T d. \end{aligned}$$

2.5 La convexité

La convexité joue un rôle central dans la théorie classique de l'optimisation. Elle est un outil indispensable pour la recherche des conditions à la fois nécessaires et suffisantes d'optimalité.[15, 51]

2.5.1 Les ensembles convexes

Définition 13

un ensemble $K \subset \mathbb{R}^n$ est dit convexe si pour tout couple$(x, y) \in K^2$ et $\lambda \in [0, 1]$, on a [9] :

$$\lambda x + (1 - \lambda)y \in K$$

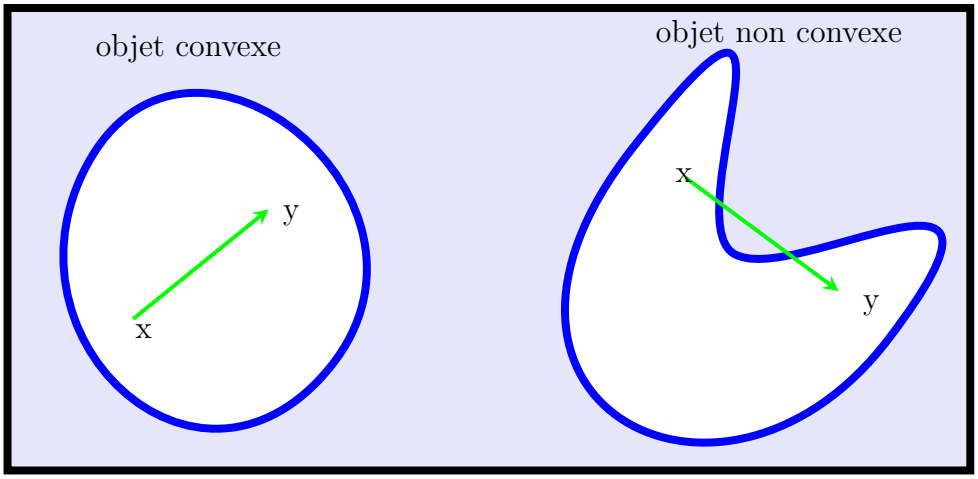

FIGURE 2.1 – Ensemble convexe et non convexe

Géométriquement : cette notion s'interpréte comme suit : "pour tout segment reliant deux points quelconques x et y de K, le segment $[x, y]$ doit être aussi dans K".[9]

2.5.2 Propriétés des ensembles convexes

Propriété 2.1 Soit une famille $\{C_i\}_{i=1,\dots,k}$ d'ensembles convexes, alors on a :

- $C = \cap_{i=1}^{k} C_i$ est un ensemble convexe.
- $C = \prod_{i=1}^{k} C_i$ est un ensemble convexe.

Propriété 2.2 Soient C_1 et C_2 sont deux ensembles convexes de $\mathbb{R}^n$, alors l'ensemble $K = C_1 \cap C_2$ est convexe.

Propriété 2.3 Si C est convexe, et $\lambda \in \mathbb{R}$, alors l'ensemble $K = \{x | x = \lambda x_1, x_1 \in C\}$ est convexe.

2.5.3 Fonctions convexes

Définition 14

Une fonction réelle F définie sur un ensemble convexe C de $\mathbb{R}^n$, est dite convexe, si pour tous les points $x, y \in C$, et pour tout nombre réel positif ou nul λ tel que $0 \leq \lambda \leq 1$, l'inégalité suivante est vérifiée [9] :

$$F(\lambda x + (1-\lambda)y) \leq \lambda F(x) + (1-\lambda)F(y). \tag{2.5}$$

Définition 15

Une fonction convexe $F(x), x \in C$, est dite *strictement convexe* si l'inégalité (2.5) est stricte pour tous les points $x_1, x_2 \in C$, avec $x_1 \neq x_2$ et $\lambda \in [0, 1]$

Définition 16

Soit une forme quadratique $F(x) = x^T Dx$ et x un point quelconque non nul de $\mathbb{R}^n$. Alors :

- F est dite définie positive si $F(x) > 0$.
- F est dite semi-définie positive ou définie non négative si $F(x) \geq 0$.
- F est dite définie négative si $F(x) < 0$.
- F est dite semi-définie négative si $F(x) \leq 0$.
- F est dite non définie si $F(x)$ est positive pour certaines valeurs de x et négative pour d'autres.[9]

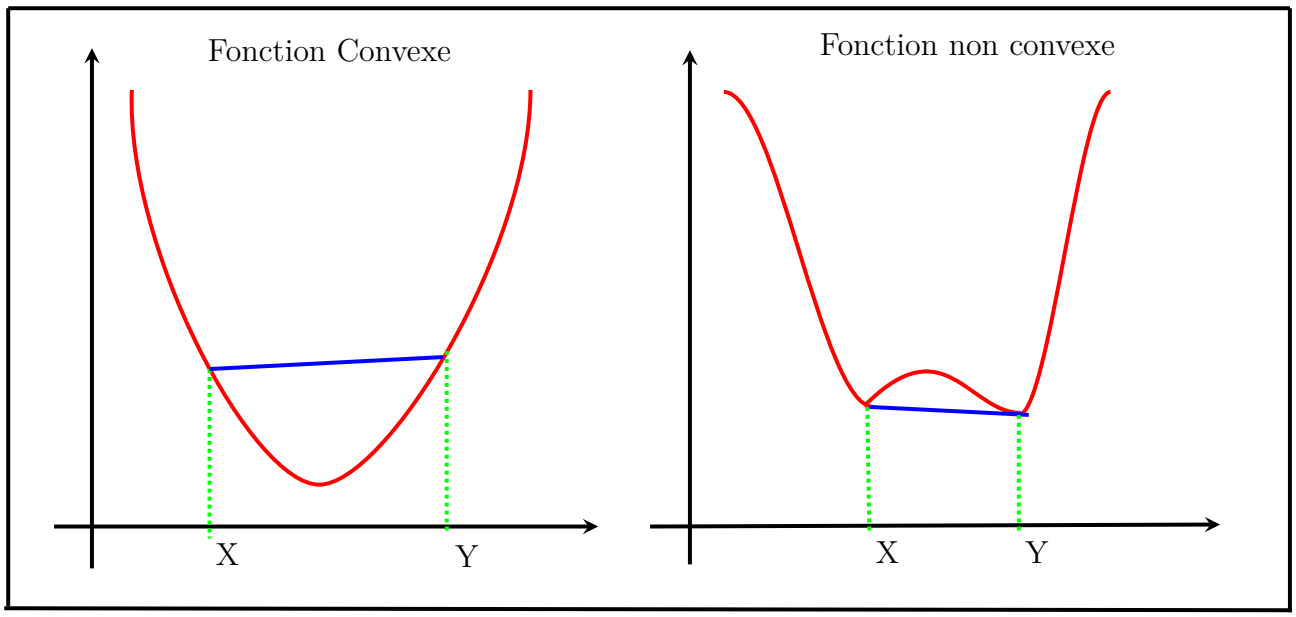

FIGURE 2.2 – Fonction convexe

2.5.4 Propriétés des fonctions convexes

Propriété 2.4 [62] Soit F une fonction réelle définie sur un ensemble convexe $C \subset \mathbb{R}^n$. Alors F est convexe si et seulement si son épigraphe : $epi(F) = \{(x, r) \in \mathbb{R}^n \times \mathbb{R} : x \in C, F(x) \leq r\}$ est un ensemble convexe.

Théorème 2.2 *[53] Si F est continument différentiable, les conditions (a) et (b) ci-dessous sont équivalentes ; de plus, les conditions (a), (b) et (c) ci-dessous sont équivalentes si F est deux fois continument différentiable :*

- ***(a)*** *F est convexe ;*
- ***(b)*** $\forall x \in C,\ \forall y \in C : F(y) - F(x) \geq [\nabla F(x)]^T (y - x)$;
- ***(c)*** *$\forall x \in C$, le Hessien $\nabla^2 F(x)$ est une matrice semi-définie positive.*

Propriété 2.5 [62, 44] Soit F une fonction réelle définie sur un ensemble convexe $C \subset \mathbb{R}^n$, alors F est convexe si et seulement si

$$F\left(\sum_{i=1}^{p} \lambda_i s_i\right) \leq \sum_{i=1}^{p} \lambda_i F(s_i), \tag{2.6}$$

où $s_i \in C, i = 1 \cdots p, \lambda_i \geq 0, \sum_{i=1}^{p} \lambda_i = 1$.

Propriété 2.6 [62, 44] Soit une fonction réelle de classe $\mathcal{C}^1$, définie sur un ensemble convexe $C \in \mathbb{R}^n$. Alors f est convexe si et seulement si

$$f(y) - f(x) \geq (y - x)^T \nabla f(x), \forall x, y \in C. \tag{2.7}$$

Propriété 2.7 [62, 44] Soit une fonction réelle de classe $\mathcal{C}^2$, définie sur un ensemble convexe $C \in \mathbb{R}^n$. Alors f est convexe si et seulement si

$$(y-x)^T H(x)(y-x) \geq 0, \forall x, y \in C. \tag{2.8}$$

Propriété 2.8 [62, 44] Si $C \subset \mathbb{R}^n$ est un ouvert convexe, alors f est convexe si et seulement si

$$H(x) \geq 0, \forall x \in C.$$

Alors d'aprés ce théoréme, on déduit facilement qu'une fonction quadratique $F(x) = x^T Dx + c^T x$ est convexe si et seulement si sa matrice associée D est semi-définie positive.

2.5.5 Critère de Sylvester pour les formes quadratiques définies et semi-définies

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

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

Le mineur de la matrice D, formé des lignes $i_1, i_2, \ldots, i_p$ et les colonnes $j_1, j_2, \ldots, j_p$, sera noté comme suit :

$$D\begin{pmatrix} i_1, i_2, \ldots, i_p \\ j_1, j_2, \ldots, j_p \end{pmatrix} = \begin{vmatrix} d_{i_1 j_1} & d_{i_1 j_2} & \cdots & d_{i_1 j_p} \\ d_{i_2 j_1} & d_{i_2 j_2} & \cdots & d_{i_2 j_p} \\ \vdots & \vdots & \ddots & \vdots \\ d_{i_p j_1} & d_{i_p j_2} & \cdots & d_{i_p j_p} \end{vmatrix}.$$

Ce mineur est dit principal si $i_1 = j_1, i_2 = j_2, \ldots, i_p = j_p$, c'est-à-dire s'il est formé de lignes et de colonnes portant les mêmes numéros. Les mineurs suivants

$$D_1 = d_{11}, \quad D_2 = \begin{vmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{vmatrix}, \ldots, \quad D_n = \begin{vmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & \cdots & d_{nn} \end{vmatrix},$$

sont appelés mineurs principaux successifs. Alors, le critère de Sylvester se formule comme suit :

Théorème 2.3 ***(critère de Sylvester)***[59]

– *Pour qu'une matrice symétrique D soit définie positive ($D > 0$), il est nécessaire et suffisant que tous ses mineurs principaux successifs soient positifs :*

$$D_1 > 0, \;\; D_2 > 0, \ldots, \;\; D_n > 0, \tag{2.9}$$

– *Pour que la matrice D soit semi-définie positive ($D \geq 0$), il est nécessaire et suffisant que tous ses mineurs principaux soient non négatifs :*

$$D\begin{pmatrix} i_1, i_2, \ldots, i_p \\ i_1, i_2, \ldots, i_p \end{pmatrix} \geq 0, \quad 1 \leq i_1 < i_2 < \cdots < i_p \leq n, \;\; p = 1, 2, \cdots, n. \tag{2.10}$$

2.6 Méthodes de résolution des problémes quadratiques

Plusieurs méthodes ont été développées pour la résolution de ce type de problèmes, parmi lesquelles on peut citer :

2.6.1 Méthode d'activation des contraintes

C'est une méthode classique, développée au début des années soixante-dix pour la résolution des problèmes de programmation linéaire et quadratique. Elle s'applique pour des problèmes d'optimisation avec des contraintes linéaires de type inégalités ou mixtes (égalités et inégalités). La première méthode est mise au point par Fletcher en 1971[24] ; par la suite d'autres auteurs ont fait des raffinements numériques à cette dernière tels que Gill et Murray en 1978 [33] ainsi que Gould en 1991 [40]. Goldfarb et Idnani en 1983 ont développé la méthode duale [38] pour le cas des programmes quadratiques strictement convexes, tandis que Boland [13] l'a généralisée en 1997 pour le cas convexe. Le principe général de la méthode consiste à écarter temporairement un certain nombre de contraintes d'inégalités et de résoudre à chaque itération un problème avec uniquement des contraintes d'égalités, correspondant aux contraintes actives. Par la suite, l'ensemble des indices actifs est ajusté en ajoutant ou/et en supprimant une contrainte à la fois jusqu'à l'obtention de la solution optimale.

2.6.2 Méthodes des points intérieurs

Les méthodes des points intérieurs sont apparues dans les années cinquante, notamment dans le livre de Fiacco et McCormick[22]. C'est dans ce livre que le terme de points intérieurs sera introduit. Les itérés générés par ces méthodes sont strictement réalisables : ils restent à l'intérieur du domaine réalisable, d'où le nom donné à ces méthodes.

Les méthodes de points intérieurs forment une classe d'algorithmes qui permettent de résoudre des problèmes d'optimisation mathématique. Elles ont l'intérét d'être polynomiales lorsqu'on les applique aux problèmes d'optimisation linéaire, quadratique convexe, semi-définie positive ; et plus généralement aux problèmes d'optimisation convexe, pourvu que l'on dispose d'une barrière auto-concordante représentant l'ensemble admissible, calculable en temps polynomial (ce n'est pas toujours le cas, car certains problèmes d'optimisation convexe sont $NP-$difficiles).

Les méthodes de points intérieurs se répartissent en plusieurs familles :

- Les méthodes " affine scaling " (optimisation sur des ellipsoïdes)
- Les méthodes de réduction du potentiel (notion de barriére , chemin central, relaxation).

L'intérêt pour ces méthodes principalement développées depuis les années 60 [22], a connu un renouveau pour les problèmes non linéaires et a ouvert un nouveau domaine pour les problèmes linéaires. La distinction entre la programmation linéaire et non linéaire n'était plus aussi nette. Den Hertog (1994) a classé les méthodes de points intérieurs en quatre catégories :

- Méthodes de chemin central.
- Méthodes affines et mise à l'échelle.
- Méthodes projectives avec potentiel.
- Méthodes affines avec potentiel.

2.6.3 Méthode du simplexe quadratique de Wolf(1959)

C'est une modification de la méthode du simplexe [16][18][75].

Son principe consiste à résoudre le systéme d'optimalité de Karush-Kuhn-Tucker, en ajoutant la condition de complémentarité.

L'algorithme de la méthode nécessite une solution réalisable de départ, Elle est obtenue en utilisant la première phase du simplexe.

Algorithme 4 Algorithme de Wolf

Début

1− Introduire les données, D, A, b, c ;

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

– Déterminer les équations de K.K.T ;

2− Déterminer les paramètres du programme linéaire ;

– Introduire les variables artificielles υ_i ;

– Construire la matrice des contraintes A ;

– Construire le vecteur du second membre b ;

– Construire le vecteur des coûts c ;

3− Initialiser le vecteur solution $(x, \lambda, \delta, \upsilon)$;

– Déterminer l'ensemble des indices J_B et J_N ;

– Extraire les éléments de base x_B, c_B, A_B ;

4 Calculer le vecteur des potentiels $u^T = c_B^T A_B^{-1}$;

– Calculer le vecteur des estimations $E_N^T = u^T A_N - c_N^T$;

– **si** $E_N \geq 0$ **alors**

la solution actuelle est **optimale**, aller à **fin**, ;

– **sinon**

Aller à l'étape **5** ;

– **finsi**

5− Déterminer la variable qui entre dans la base tout en vérifiant la condition $\delta_j x_j = 0, j = \overline{1,n}$;

– Déterminer la variable qui sort de la base ;

– Mettre à jour A_B, x_B, c_B, J_B, J_N et aller à l'étape **4**.

2.6.4 Méthode directe de support

Cette méthode est basée sur le principe des méthodes adaptées développées par les professeurs R. Gabassov et F.M. Kirillova [25][29]. Ces méthodes sont intermédiaires entre les méthodes d'activation des contraintes et celles de points intérieurs.

Algorithme 5 Méthode directe pour la programmation quadratique convexe à variables hybrides

Début

1−Soit un plan de support initial$\{x, J_P\}$ du problème (P) ;

2−Calculer le vecteur des estimations $E_N^T = E^T(J_N) = g_N^T - u^T A_N$

si $\beta(x, J_B) \leq \epsilon$ **alors**

x est $\epsilon-$optimal ;

Aller à fin.

finsi

3−**Changement de plan**

Choisir l'indice j_0 tel que $|E_{j_0}| = max(|E_j|, j \in J_{NNO})$ où J_{NNO}est l'ensemble des indices non optimaux ;

Calculer la direction d'amélioration ℓ ;

Calculer le pas $\theta^0 = min(1, \theta_{j1}, \theta_{j_S}, \theta_F)$.

Calculer $\overline{x} = x + \theta^0 \ell$,

Calculer l'estimation de suboptimalité $\beta(\overline{x}, J_B) = (1 - \theta^0)(\beta - \theta^0 \ell^T D \ell)$

si $\beta(\overline{x}, J_B) \leq \epsilon$ **alors**

Aller à fin.

sinon

Aller à l'étape'4'.

finsi

4−**Changement de support**

si $\theta^0 = 1$ **alors**

$\overline{x} = x + \ell; \beta(\overline{x}, J_B) = 0;$

$\overline{x}$ est optimal

aller à fin.

sinon

si $\theta^0 = \theta_{j_1}$ **alors**

si $j_0 \in J_S$ **alors**

$\overline{J_B} = (J_B \setminus j_1) \cup j_0, \overline{J_S} = (J_S \setminus j_0).$

sinon

si $j_0 \in J_{NN}$ **alors**

$\overline{J_B} = (J_B \setminus j_1) \cup j_0$ et $\overline{J_S} = J_S.$

finsi

sinon

si $\theta^0 = \theta_{j_s}$ **alors**

$\overline{J_B} = J_B, \overline{J_S} = J_S \setminus j_s.$

sinon

si $\theta^0 = \theta_F$ **alors**

$\overline{J_B} = J_B, \overline{J_S} = J_S \cup j_*$

finsi

finsi

finsi

si $\beta(\overline{x}, \overline{J_B}) > \epsilon$ **alors**

aller à l'étape "2",en partant de $\{x, J_P\}$;

finsi

finsi

finsi

2.7 Conclusion

La programmation quadratique est un cas particulier de la programmation non linéaire, dans ce chapitre, nous avons présenter la formule mathématique et les méthodes de résolution de la programmation linéaire, la programmation non linéaire et la programmation quadratique. Dans le chapitre suivant nous allons parler sur la programmation en nombres entiers.

Chapitre 3

PROGRAMMATION EN NOMBRES ENTIERS

"Lorsque notre objectif à atteindre est très clair alors, rien ne peut nous arrêter.".

(Zenman)

3.1 Introduction :

La programmation en nombres entiers concerne les programmes d'optimisation sous contraintes pour lesquels les variables doivent prendre des valeurs entières. Elle est un domaine très riche de la programmation mathématique. Les recherches dans ce domaine sont nombreuses et les commencements de vraie étaient avec **Gomory** en 1958[39].

Dans beaucoup de problèmes d'optimisation une solution à valeurs entières est exigée. Par exemple dans le problème de la production, on cherche le nombre optimal(entier)de pièces à fabriquer.

3.2 Forme générale d'un programme linéaire en nombres entiers "PLNE"

La forme générale d'un programme linéaire en nombres entiers [32]se présente comme suit :

$$\begin{cases} \max ou \min(Z) = \Sigma_{j=1}^{n} c_j x_j. \\ \forall i = \overline{1,m} : \Sigma_{j=1}^{n} a_{ij} x_j \leq, = ou \geq b_i. \\ \forall j = \overline{1,n} : x_j \geq 0. \\ x_j \quad entier, j = \overline{1,k} \quad (k \leq n) \end{cases}$$

Remarque 4

Dans le cas d'un programme linéaire avec des variables binaires,on aura,à la suite des contraintes fonctionnelles.

$x_j = 0$ *ou* $1, j = \overline{1,n}$

Relaxation

soit la forme générale du PLNE [76]suivante :

$$(PLNE)\begin{cases} \max(\min) Z = C^T X \\ s.c \\ AX(\geq, \leq, =)B \\ \forall x_j \in X, \qquad x_j \geq 0 \qquad (j = \overline{1,n}) \\ x_j \in Z^+ (j = \overline{1,k}) \qquad avec \qquad k \leq n. \end{cases}$$

Notez bien que si on remplace les derniers restrictions par $x_j \geq 0$ pour $j = \overline{1,k}$ avec $(k \leq n)$ on aura un programme linéaire qui sera une relaxation du problème original .

Ce fait est très important, ça implique qu'on peut résoudre la relaxation par des méthodes qu'on a déja considérés dans les chapitres précédents.

Définition 17

La relaxation linéaire (ou continue)R_p de P est obtenue par relachement (enlevement) des contraintes d'intégralité.

Définition 18

Soit $F(P)$ la région des solutions possibles de P :

on a $F(P) \subseteq F(R_p)$.

3.3 Méthodes de résolution

On distingue deux classes de méthodes de résolution des problèmes d'optimisation linéaire en nombre entiers :

Les méthodes exactes : Ces méthodes sont capables de trouver une solution optimale d'un problème donné dans un intervalle de temps bien déterminé mais exponentiel sur les problèmes

NP-complets(notamment les PLNE), parmi ces méthodes on trouve la méthode de séparation et d'évaluation (Branch and Bound), la méthode des coupes (cutting planes), la méthode qui combine les deux dernières (Branch-and-cut), la programmation dynamique.

3.3.1 Méthode de séparation et d'évaluation(Branch and bound)

La téchnique de Branch and Bound [65] est une méthode algorithmique classique pour résoudre un problème d'optimisation linéaire en nombres entiers. Il s'agit de rechercher une solution optimale dans un ensemble linéaire de solutions possibles, la méthode repose d'abord sur la séparation (branch)de l'ensemble des solutions en sous-ensembles plus petits. L'exploration de ces solutions utilise ensuite une évaluation optimiste pour majorer (bound) les sous-ensembles, ce qui permet de ne plus considérer que ceux susceptibles de contenir une solution potentiellement meilleure que la solution courante.

Le principe de "Branch and Bound"

La dénomination séparation et évaluation (Branch and Bound[65]) recouvre deux idées :

La séparation (Branch) consiste à séparer un ensemble de solutions en sous-ensembles.

L'évaluation (Bound) consiste à évaluer les solutions d'un sous-ensemble de façon optimiste, c'est-à-dire en majorant la valeur de la meilleure solution de ce sous-ensemble.

Résumons les diverses étapes de cette méthode de résolution par séparations et évaluations successives comme suit :

Pour résoudre un problème linéaire en nombres entiers (PLNE[76]), nous donnons ici la démarche pour un problème donné de maximisation ou minimisation :

Séparation :

- Choisir une variable non entière pour générer deux sous problèmes, soit X_f cette variable devant être entière dans la solution optimale initiale.
- Représentons par X_e la partie entière de X_f.
- Pour éliminer la solution non entière X_f, on crée deux branches et deux sous problèmes. On obtient une branche avec $X_f \leq X_e$ et l'autre avec $X_f \geq X_e+1$.
- Les deux sous problèmes sont obtenus en ajoutant au programme linéaire (PLNE) précédent (celui ou provient la séparation), la containte $X_f \leq X_e$ pour un des sous problèmes (SP_1) et la contrainte $X_f \geq X_e+1$ pour l'autre sous problèmes (SP_2).

  Toutes les solutions entières réalisables sont maintenant contenues dans l'un ou l'autre des sous-problèmes.

Evaluation optimiste :

Après la résolution des deux sous problèmes on obtient les trois cas suivants :

1^{er} **cas :** La solution du sous problème (SP_1) est entière et la solution du sous problème (SP_2) est entière. Nous choisissons alors la solution du sous problème dont la fonction objectif est la

plus élevée (cas de la maximisation).

$2^{ème}$ **cas :** Solution du sous problème (SP_1) entière et la solution de sous problème (SP_2) fractionnaire pour les variables devant être entières.

Si la valeur de la fonction objectif du sous problème (SP_2) est inférieure (cas de la maximisation) ou supérieure (cas de minimisation) à la solution de sous problème (SP_1), nous arrêtons le processus. la solution de sous problème (SP_1) est optimal.

Dans le cas contraire, (supérieur dans le cas de la maximisation et inférieur dans le cas de la minimisation), nous divisons le sous problème (SP_2) en deux nouveaux sous problèmes avec $X_f^{'} \leq X_e^{'}$ et $X_f^{'} \geq X_e^{'}$+1, $X_f^{'}$ étant la variable fractionnaire devant être entière du sous problème (SP_2).

3^{eme} **cas :** Solution du sous problème (SP_1) fractionnaire et solution du sous problème (SP_2) entière. même raisonnement en renversant les rôles entre les sous problème (SP_1) et (SP_2) .

critères d'arrêt :

Nous continuons le processus jusqu'au moment où :

- Le sous problème considéré n'a pas de solution .
- Le sous problème considéré a une solution optimale entière.
- La valeur de la solution optimale d'un sous problème qui contient une ou plusieurs variables de base (devant être entières) fractionnaires est inférieure à la valeur de la solution optimale entière d'un autre sous problème dans le cas de la maximisation et supérieure dans le cas de la minimisation.

Algorithme 6 Algorithme générale de Branche and Bound

début

$K \leftarrow 0$;

Résoudre le problème relaxé (P_r^K) ;

Soit x^K la solution trouvé ; Soit $Z^K = c^T * x$ la valeur de la fonction objectif en ce point ;

si $x^0 \in Z^n$ **alors**

STOP ;

tantque Il existe une branche non exploré **faire**

Choisir la stratégie d'exploration ;

Procédure de séparation : ;

Choisir une composante $x_i \in x^0 Z$;

il est d'usage de choisir la composante de x^0 contenant la plus grande partie fractionnaire ;

$K \leftarrow K + 1$; Ajouter la contrainte $\lfloor x_i \rfloor$ au problème P^K ;

$K \leftarrow K + 1$; Ajouter la contrainte $\lceil x_i \rceil$ au problème $P^K v$

fin tantque

finsi

Exemple d'application "Branch and Bound"

soit le problème initial suivant :

$$(P)\begin{cases} \max(Z) = 2x_1 + 3x_2 \\ s.c \\ 2x_1 + x_2 \leq 10 \\ x_1 + 4x_2 \leq 20 \\ x_i \in Z^+; (i = \overline{1,2}) \end{cases}$$

et soit (Pr) le problème relaxé(continu) :

$$(Pr)\begin{cases} \max(Z) = 2x_1 + 3x_2 \\ s.c \\ 2x_1 + x_2 \leq 10 \\ x_1 + 4x_2 \leq 20 \\ x_i \geq 0; (i = \overline{1,2}) \end{cases}$$

Solution de base

	x_1	x_2	x_3	x_4	$\overline{B}$
x_3	2	1	1	0	10
x_4	1	4	0	1	20
-Z	2	3	0	0	0

En résolvant le programme linéaire avec la méthode du simplexe, on avait obtenu le tableau optimal suivant :

Tableau optimal au (Pr)

	x_1	x_2	x_3	x_4	$\overline{B}$
x_1	1	0	4/7	-1/7	20/7
x_2	0	1	-1/7	2/7	30/7
-Z	0	0	-5/7	-4/7	-130/7

La solution optimal est : x_1^*=20/7=2.85,x_2^*=30/7,Z^*=130/7

cette solution n'est pas entière donc on applique Branch and Bound avec $2 \leq x_1^* = 2.85 \leq 3$

La séparation

En intégrant la contrainte $x_1 \leq 2$ à la solution optimale précédente nous obtenons le sous problème 1 (SP_1)

$$(SP_1)\begin{cases} \max(Z) = 2x_1 + 3x_2 \\ s.c \\ 2x_1 + x_2 \leq 10 \\ x_1 + 4x_2 \leq 20 \\ x_1 \leq 2 \\ x_i \geq 0; (i = \overline{1,2}) \end{cases}$$

En intégrant la contrainte $x_1 \geq 3$ à la solution optimale précédente nous obtenons le sous problème 2 (SP_2)

$$(SP_2)\begin{cases} \max(Z) = 2x_1 + 3x_2 \\ s.c \\ 2x_1 + x_2 \leq 10 \\ x_1 + 4x_2 \leq 20 \\ x_1 \geq 3 \\ x_i \geq 0; (i = \overline{1,2}) \end{cases}$$

L'évaluation :

• Résolvons le sous problème (SP_1).

On doit ajouter la contrainte $x_1 \leq 2$ au (Pr),sous forme d'équation ,on a $x_1 + x_5$=2

On obtient alors le tableau suivant en ajoutant cette équation (contrainte) au tableau précédent.

	x_1	x_2	x_3	x_4	x_5	$\overline{B}$
x_1	1	0	4/7	-1/7	0	20/7
x_2	0	1	-1/7	2/7	0	30/7
x_5	1	0	0	0	1	2
-Z	0	0	-5/7	-4/7	0	-130/7

On doit toute fois modifier ce tableau puisque x_1 est une variable dans la base et il doit lui correspondre un vecteur unitaire dans le tableau.
Multipliant la $1^{ère}$ ligne par (-1) et additionnant à la $3^{ème}$ ligne, on obtient le tableau suivant :

	x_1	x_2	x_3	x_4	x_5	$\overline{B}$
x_1	1	0	4/7	-1/7	0	20/7
x_2	0	1	-1/7	2/7	0	30/7
x_5	0	0	-4/7	1/7	1	-6/7
-Z	0	0	-5/7	-4/7	0	-130/7

On a (x_{B_3}=-6/7 < 0) alors on applique l'algorithme dual du simplexe,on trouve :
variable sortant :x_5 (x_{B_3}=-6/7 < 0)
variable entrante :x_3 [min $-Z_j/\mu_{3j}, \mu_{3j} < 0$]=[(-5/7)/(-4/7)]=5/4
le pivot est -4/7(μ_{33}) = $-4/7$
On obtient le tableau optimal suivant effectuant l'opération de pivotement.

	x_1	x_2	x_3	x_4	x_5	$\overline{B}$
x_1	1	0	0	0	1	2
x_2	0	1	0	1/4	-1/4	63/14
x_3	0	0	1	-1/4	-7/4	3/2
-Z	0	0	0	-3/4	-5/4	-245/14

La solution optimale au sous problème (SP_1) est :
$x_1 = 2$, x_2=63/14=4+1/2, Z_1=17+1/2

•On doit maintenant procéder de la même façon pour déterminer la solution optimale du sous problème (SP_2).
Il faut ajouter la contrainte $x_1 \geq 3 \Leftrightarrow$ -$x_1 \leq -3$ et sous forme d'équation on a : -x_1+x_5=-3.
Ajoutons cette dernière équation au tableau optimal du (Pr), on obtient :

	x_1	x_2	x_3	x_4	x_5	$\overline{B}$
x_1	1	0	4/7	-1/7	0	20/7
x_2	0	1	-1/7	2/7	0	30/7
x_5	-1	0	0	0	1	-3
-Z	0	0	-5/7	-4/7	0	-130/7

Additionnant la $1^{ère}$ ligne à la $3^{ème}$ (pour obtenir le vecteur unité sous x_1), on obtient le tableau suivant :

	x_1	x_2	x_3	x_4	x_5	$\overline{B}$
x_1	1	0	4/7	-1/7	0	20/7
x_2	0	1	-1/7	2/7	0	30/7
x_5	0	0	4/7	-1/7	1	-1/7
-Z	0	0	-5/7	-4/7	0	-130/7

Appliquant l'algorithme dual du simplexe , on obtient :
variable sortante : x_5
variable entrante : x_4
pivot=-1/7
Le tableau optimal est alors le suivant :

	x_1	x_2	x_3	x_4	x_5	$\overline{B}$
x_1	1	0	0	0	-1	3
x_2	0	1	1	0	2	4
x_4	0	0	-4	1	-7	1
-Z	0	0	-3	0	-4	-18

La solution optimale au sous problème (SP_2) est :
x_1=3, x_2=4, Z_2= 18
• La solution optimale entière Z_2 est la solution recherchée car Z_2=18 > Z_1=17+1/2 ;
le sous problème (SP_1) ne sera pas subdivisé en deux nouveaux sous problèmes.

3.3.2 Coupes de Gomory

Soit (P) un programme linéaire en nombres entiers écrit sous forme canonique par rapport à une base J.

$$(P)\begin{cases} \max(Z) = C^T X \\ AX = b \\ x_j \in N \qquad j = \overline{1,n} \end{cases}$$

on sait que si les trois conditions suivantes :

$$C \leq 0 \tag{3.1}$$

$$b \geq 0 \tag{3.2}$$

$$b \qquad entier \tag{3.3}$$

sont satisfaites alors $\overline{x}$, solution de base relative à la base J est solution optimale de (P). si (3.1) et (3.2) sont vérifiés $\overline{x}$ est solution optimale du programme linéaire (P') obtenu à partir de (P) en relâchant les contraintes d'intégrité sur les variables.

Définition 19

Etant donné le programme linéaire en nombres entiers (P), on dit que l'inéquation ax$\leq \alpha$ est valide si elle est satisfaite par toute solution réalisable de (P), une coupe est une inéquation valide qui n'est pas satisfaite pour tout point de D' domaine des solutions réalisables de la relaxation (P') de (P).

Définition 20

Si α est un scalaire quelconque, on désigne par :

- $\lfloor \alpha \rfloor$ le plus grand entier inferieur à α
- $\lceil \alpha \rceil$ le plus petit entier superieur à α
- $\langle \alpha \rangle$ =α- $\lfloor \alpha \rfloor$

  $\langle \alpha \rangle$ est appelée la partie fractionnaire de α et $\lfloor \alpha \rfloor$ sa partie entière inferieure

Théorème 3.1 *Pour toute valeur du scalaire α,l'inéquation :*

$$\sum_{j \notin J} (\lfloor \alpha \rfloor A_i^j - \lfloor \alpha A_i^j \rfloor) x_j \geq \lfloor \alpha \rfloor b_i - \lfloor \alpha b_i \rfloor \tag{3.4}$$

est valide, pour certaines valeurs de α (et en particulier α=1), c'est une coupe.

Démonstration 3.1 Multiplions chaque terme de la i^{eme} contrainte de (P) par le scalaire α. il vient :

$$\alpha x_r + \sum_{j \notin J} \alpha A_i^j x_j = \alpha b_i \quad (r = col(i))$$

et décomposons chaque coéfficient en la somme de sa partie entière et de sa partie fractionnaire :

$$\lfloor\alpha\rfloor x_r + \sum_{j\notin J}\lfloor\alpha A_i^j\rfloor x_j + \langle\alpha\rangle x_r + \sum_{j\notin J}\langle\alpha A_i^j\rangle x_j = \lfloor\alpha b_i\rfloor + \langle\alpha b_i\rangle$$

compte tenu de ce que,par définition : $0 \leq \langle\alpha\rangle < 1$ et $x_j \geq 0$ ceci implique :

$$\lfloor\alpha\rfloor x_r + \sum_{j\notin J}\lfloor\alpha A_i^j\rfloor x_j \leq \lfloor\alpha b_i\rfloor + \langle\alpha b_i\rangle$$

Or le membre de gauche est entier et $\langle\alpha b_i\rangle < 1$, par conséquent cette dernière inéquation implique :

$$\lfloor\alpha\rfloor x_r + \sum_{j\notin J}\lfloor\alpha A_i^j\rfloor x_j \leq \lfloor\alpha b_i\rfloor$$

et en soustrayant cette dernière inéquation de la i^{eme} contrainte de (P) multipliée au préalable par $\lfloor\alpha\rfloor$, on obtient (3.4) qui est donc bien une inéquation valide. Supposons les inéquations (3.1) et (3.2) soient vérifiées mais que, pour certain indice i, b_i ne soit pas entier(i.e.$\langle b_i\rangle > 0$). posons α=1 dans (3.4), on a :

$$\sum_{j\notin J}\langle A_i^j\rangle x_j \geq \langle b_i\rangle \tag{3.5}$$

qui est une coupe puisque (3.5) n'est pas satisfait pour la solution réalisable de base de (P').

Principe de méthodes de coupe[65, 39]

- Introduire de nouvelles contraintes linéaires au problème pour réduire le domaine réalisable du problème relaxé sans pour autant éliminer de points du domaine réalisable du problème en nombres entiers.
- La procédure consiste à résoudre une suite de problèmes relaxés jusqu'a ce qu'une solution optimale en nombre entier soit obtenue.
- Un problème de la suite est obtenu du précédent en lui ajoutant une contrainte linéaire (coupe) supplémentaire.

et voici l'algorithme qu'on suit :

Algorithme 7 Coupes de Gomory

début

(0) A et b étant entiers, résoudre le problème par une méthode de programmation linéaire en oubliant les contraintes d'intégrité. Aller à **(1)**

(1)

si la solution optimale est entière **alors**

Terminer, et cette solution est aussi la solution optimale du problème posé.

sinon

Aller à **(2)**

finsi

(2) Déterminer $\langle b_r \rangle = \max \langle b_i \rangle$, ajouter au système d'équations obtenu à la dernière itération la coupe :

$$-\sum_{j=1}^{p} \langle a_{rj} \rangle x_j + x_{p+1} = -\langle b_r \rangle$$

où p est le nombre de variables avant d'ajouter cette coupe.Aller à **(3)**

(3) Appliquer l'algorithme dual du simplexe à partir du tableau optimal précédent et retourner à **(1)**

3.3.3 Branch and Price

Principe du branch and price :[4]

Le branch and price est une méthode de résolution du problème linéaire en nombres entiers. Les algorithmes de branch and bound avec génération de colonnes sont appelés algorithmes de branch and price. Plus précisément dans un branch and price on a une exploration arborescente de la même façon que dans le branch and bound mais au lieu d'avoir une simple résolution de (PL) à chaque nœud on a à résoudre un (PL) avec génération de colonnes. Les colonnes générées peuvent être valides dans tout l'arbre ou seulement dans la branche courante. Les deux principales difficultés dans la mise en œuvre d'un branch and price sont :

- Formaliser le sous-problème et être capable de le résoudre rapidement (même difficulté que dans le cadre de la génération de colonnes)
- Trouver une règle de branchement adaptée qui ne perturbe pas le sous-problème (difficulté propre au branch and price).

Les difficultés liées au branchement dans un branch and price :

En général les branchements classiques (typiquement le branchement binaire sur[31] une variable) fonctionnent assez mal dans le cadre d'une résolution par branch and price . En effet dans un branch and price on est susceptible de générer de nouvelles variables à chaque nœud et par conséquent de faire augmenter rapidement le nombre de variables. Le risque est donc de parvenir difficilement à obtenir des solutions entières (qui correspondent aux feuilles de

l'arbre) puisqu'on a régulièrement de nouvelles variables sur lesquelles brancher. De plus les contraintes de branchement viennent grossir l'ensemble des contraintes présentent initialement dans le PLNE. Elles sont biensûr associées à de nouvelles variables duales qui doivent être prises en compte dans le sous-problème (au moment où on cherche les variables améliorantes). Il n'est souvent pas facile de prendre en compte ces contraintes.

3.3.4 Programmation dynamique

La programmation dynamique [8]est une technique mathématique et algorithmique à implémenter qui a pour objet d'aider à prendre des décisions séquentielles indépendantes les unes des autres et calculer éventuellement des stratégies optimales. Il n'y a pas de formalisme mathématique standard. C'est une approche de résolution où les équations doivent être spécifiées selon le problème à résoudre.

Formalisme générale du problème

On s'intéresse à un système dynamique à temps discret, avec une fonction de coût additive dans le temps, l'état du système étant représenté par :

$$x_{k+1} = f_k(x_k, u_k, w_k), k = \overline{1, N}$$

Où :

- k numérote les périodes (dont le nombre est fixé à N).
- x_k décrit l'état du système au début de la période k.
- u_k est la décision devant être prise à la période k.
- w_k est la perturbation aléatoire de la période k.
- f_k est la fonction de transfert (transition) entre les périodes k et $k+1$.

La fonction de coût associée à la période k s'écrit : $g_k(x_k, u_k, w_k), k = \overline{1, N}$
On rajoute en plus un coût pour l'état final du système : $g_{N+1}(x_{N+1})$
Le coût total est la somme des coûts (c'est une variable aléatoire).
La fonction-objectif du problème d'optimisation s'écrit alors :
$E[g_{N+1}(x_{N+1}) + \sum g_k(x_k, u_k, w_k)]$,(espérance par rapport aux (w_k)),pour laquelle on souhaite déterminer le minimum pris par rapport aux décisions (u_k).

Les méthodes approchées : Elles cherchent à produire une solution de meilleure qualité possible dictée par des heuristiques avec un temps de calcul raisonnable en examinant seulement une partie de l'espace de recherche.

3.3.5 Heuristiques et métaheuristiques

une heuristique[74] est un algorithme approché qui permet d'identifier en temps polynomial au moins une solution réalisable rapide, pas obligatoirement optimale. L'usage d'une heuristique est efficace pour calculer une solution approchée d'un problème et ainsi accélérer le processus de résolution exacte. Généralement une heuristique est conçue pour un problème particulier, en s'appuyant sur sa structure propre sans offrir aucune garantit quant à la qualité de la solution calculée.

Face aux difficultés rencontrées par les heuristiques pour avoir une solution réalisable de bonne qualité pour des problèmes d'optimisation difficiles, les métaheuristiques ont fait leur apparition. Ces algorithmes sont plus complets et complexes qu'une simple heuristique, et permettent généralement d'obtenir une solution de très bonne qualité pour des problèmes issus des domaines de la recherche opérationnelle ou de l'ingénierie dont on ne connait pas de méthodes efficaces pour les traiter ou bien quand la résolution du problème nécessite un temps élevé ou une grande mémoire de stockage.

Le rapport entre le temps d'exécution et la qualité de la solution trouvée d'une métaheuristique reste alors dans la majorité des cas très intéressant par rapport aux différents types d'approches de résolution.

La plupart des métaheuristiques utilisent des processus aléatoires et itératifs comme moyens de rassembler de l'information, d'explorer l'espace de recherche et de faire face à des problèmes comme l'explosion combinatoire. Une métaheuristique [19]peut être adaptée pour différents types de problèmes, tandis qu'une heuristique est utilisée à un problème donné. Plusieurs d'entre elles sont souvent inspirées par des systèmes naturels dans de nombreux domaines tels que : la biologie (algorithmes évolutionnaires et génétiques) la physique (recuit simulé), et aussi l'éthologie (algorithmes de colonies de fourmis).

Un des enjeux de la conception des métaheuristiques est donc de faciliter le choix d'une méthode et le réglage des paramètres pour les adapter à un problème donné.

Les métaheuristiques peuvent être classées de nombreuses façons. On peut distinguer celles qui travaillent avec une population de solutions de celles qui ne manipulent qu'une seule solution à la fois. Les méthodes qui tentent itérativement d'améliorer une solution sont appelées méthodes de recherche locale ou méthodes de trajectoire. Ces méthodes construisent une trajectoire dans l'espace des solutions en tentant de se diriger vers des solutions optimales. Les exemples les plus connus de ces méthodes sont : La recherche Tabou et le Recuit Simulé. Les algorithmes génétiques, l'optimisation par essaim de particules et Les algorithmes de colonies de fourmis présentent les exemples les plus connus des méthodes qui travaillent avec une population.

Heuristiques :[74]

1. **Heuristiques de type glouton :** Les heuristiques de type glouton sont généralement simples à programmer, et requièrent un faible temps d'exécution pour générer une solution réalisable du problème. En partant d'une solution nulle (resp. avec toutes les variables fixées à 1), le principe de ces heuristiques est d'ajouter (resp. retirer) des éléments dans la solution courante tant que (resp. jusqu'à ce que) la solution reste (resp. devienne) réalisable dans le cas des heuristiques gloutonnes dites primales (resp. duales).
2. **Heuristiques basées sur des relaxations :** Différentes heuristiques basées sur des relaxations permettent d'obtenir de bonnes solutions pour les problèmes en nombres entiers. Nous commençons ici par présenter des heuristiques simples consistant à arrondir une solution optimale de la relaxation en continu. Nous abordons ensuite des heuristiques plus perfectionnées basées sur la notion de pivot.

   - Relaxation en continu :

     Une heuristique simple permettant de générer rapidement une solution (réalisable ou non) du problème consiste à arrondir une solution optimale de la relaxation en continu. L'idée derrière cette approche est qu'une solution optimale de la relaxation en continu peut être proche d'une solution optimale du problème. Même si cela est vrai dans certains cas, il n'y a cependant pas de garantie d'obtenir une solution réellement proche de l'optimum dans tous les cas.

   - Heuristique pivot et complément :

     L'heuristique pivot et complément (PC) a été proposée pour la résolution des problèmes en variables $0-1$. Elle permet généralement l'obtention de solutions entières réalisables du problème. Le principe de l'heuristique repose sur le fait que le problème(PLNE) est équivalent au programme linéaire suivant :

$$\begin{cases} \max ou \min(Z) = \Sigma_{j=1}^{n} c_j x_j. \\ s.c \\ Ax + s = b \\ 0 \leq x_j \leq 1 \;\; j \in N \\ s_i \geq 0 \;\; i \in M \\ s \;\text{ variable en base.} \end{cases}$$

Métaheuristiques :[43]

De nombreuses métaheuristiques ont été et sont encore proposées pour résoudre le problème(PLNE), le principe et l'algorithme de ces méthodes est présenté dans le chapitre (1).

– Recuit simulé,

- Recherche tabou,
- Recherche local,
- Les algorithmes génétiques,
- Les algorithmes évolutionnaires.

3.4 Conclusion

De nombreux problèmes de recherche opérationnelle et d'algorithmique peuvent être traduits en problème d'PLNE, dans ce chapitre, nous avons déterminé la forme générale d'un PLNE et les méthodes de résolution relative à ce type de problème.
Le prochain chapitre est consacré à la présentation de notre nouvelle méthode adaptée pour la résoltion des problèmes quadratiques biobjectifs en variables mixtes.

Chapitre 4

Méthode adaptée pour la programmation quadratique biobjectif à variables mixtes

"La découverte, c'est de voir ce que tout le monde voit et de penser ce que personne ne pense.".
(Gyorgi Szent)

4.1 Introduction

La méthode directe de support que nous avons présentée dans le chapitre(2) est une méthode d'amélioration basée sur la métrique du simplexe, mais les plans de support admettent encore d'autres métriques pour les directions d'amélioration. Le principe de notre méthode est pratiquement le même que celui de la méthode directe de support, c'est-à-dire, qu'au lieu d'utiliser la métrique du simplexe en changeant un seul indice non basique j_0, on utilisera plutôt une autre métrique dite adaptée qui consiste à considérer tous les indices non optimaux en fonction desquels on contruit une direction d'amélioration de la fonction objectif, et le pas le long de cette direction.

4.2 Position du problème

Le probléme(P) de la programmation quadratique bi-objectif en variables mixtes se présente sous la forme canonique suivante :

$$F_1(x) = \frac{1}{2}x^T Dx + c^T x \to \min, \tag{4.1}$$

$$F_2(x) = C^T x \to \max, \tag{4.2}$$

$$Ax = b, \tag{4.3}$$

$$d^- \leq x \leq d^+, \tag{4.4}$$

$$Y = \begin{cases} 1 \text{ si } x > 0 \\ 0 \quad \text{sinon} \end{cases} \tag{4.5}$$

où $D^T = D \geq 0, c \in \mathbb{R}^n, b \in \mathbb{R}^m, d^-, d^+ \in \mathbb{R}^n, rangA = m < n$

Notations :

$I = \{1, 2, ..., m\}$:l'ensemble d'indices des lignes de A,

$J = \{1, 2, ..., n\}$:l'ensemble d'indices des colonnes de$A$,

J_B :ensemble d'indices des variables basiques, tel que $|J_B| = m$,

$J_N = J \backslash J_B$: ensemble d'indices des variables hors-base,

$x = (\frac{x_B}{x_N}), x_B = x(J_B) = (x_j, j \in J_B), x_N = x(J_N, j \in J_N)$,

$c = (\frac{c_B}{c_N}), c_B = c(J_B), c_N = c(J_N)$,

$A = A(I, J) = (a_{ij}, 1 \leq i \leq m, 1 \leq j \leq n), A = (a_j, j \in J)$, $a_j = \begin{pmatrix} a_{1j} \\ \vdots \\ \vdots \\ a_{mj} \end{pmatrix}$

$A = (A_B|A_N), A_B = A(I, J_B), A_N + (I, J_N)$.

4.3 Définitions

Définition 21 (Solution réalisable)

Un vecteur vérifiant les contraintes (4.3)-(4.4) est appelé plan ou solution réalisable du probléme (P).[9][47]

Définition 22 (Solution optimale)

Un plan (x^0) *est dit optimal si :*

$$F(x^0) = \frac{1}{2}x^{0^T}Dx^0 + c^T x^0 = \min(\frac{1}{2}x^T Dx + c^T x)$$

où x *est pris parmi tous les plans du probléme (4.2)-(4.3).*[9][47]

Définition 23 (solution suboptimale)

Un plan (x^ϵ) *est appelé* ϵ*-optimal ou suboptimal si*

$$F(x^\epsilon) - F(x^0) \leq \epsilon,$$

où ϵ *est un nombre positif ou nul, donné à l'avance et* (x^0) *est une solution optimale du problème.*[9][47]

Définition 24 (Support)

L'ensemble $J_B \subset J, |J_B| = m$, est appelé support des contraintes du problème (P) si[9][47] :

$$detA_B \neq 0$$

Définition 25 (Plan de support)

Le couple $\{x, J_B\}$, formé du plan x et du support J_B est appelé plan de support des contraintes.[9][47]

Définition 26 (Plan de support non dégénéré)

Le plan de support est dit non dégénéré, si [9][47]

$$d_j^- < x_j < d_j^+, j \in J_B.$$

4.4 Formule d'accroissement de la fonction objectif

Soit $\{x, J_B\}$ un plan de support des contraintes du problème (P) et considérons un autre plan quelconque $\overline{x} = x + \Delta x$. L'accroissement de la fonction objectif s'écrit comme suit [9][47] :

$$F(\overline{x}) - F(x) = g^T(x)\Delta x + \frac{1}{2}(\Delta x)^T D(\Delta x) \tag{4.6}$$

Où $g(x) = Dx + c$ est le gradient de la fonction objectif F au point x, avec $g(x) = g(J) = (g_j, j \in J)$. Il peut être fractionné de la maniére suivante : $g = (\frac{g_B}{g_N}); g_B = g(J_B); g_N = g(J_N)$.
Par ailleur on a :

$$\begin{cases} Ax = b \\ A\overline{x} = b \end{cases}$$

Donc $\overline{x} = A(\Delta x + x) = A\Delta x + Ax$, ce qui implique que :$A\Delta x = 0$.
En posant : $\Delta x = (\frac{\Delta x_B}{\Delta x_N}); \Delta x_B = \Delta x(J_B); \Delta x_N = \Delta x(J_N)$.
on a : $A\Delta x = 0 \Leftrightarrow A_B\Delta x_B + A_N\Delta x_N = 0$, alors :

$$\Delta x_B = -A_B^{-1}A_N\Delta x_N. \tag{4.7}$$

Introduisons maintenant le vecteur des potentiels u et le vecteur des estimations E tels que :

$$u^T = g_B^T A_B^{-1}, E^T = E^T(J) = g^T - u^T A = (E_B^T, E_N^T), \tag{4.8}$$

où $E_B^T = 0, et E_N^T = g_N^T - u^T A_N$.
La formule(4.6) devient alors :

$$F(\overline{x}) - F(x) = g_B^T\Delta x_B + g_N^T\Delta x_N + \frac{1}{2}(\Delta x_B, \Delta x_N)^T D(\Delta x_B, \Delta x_N) \tag{4.9}$$

D'après l'égalité (4.7), on aura :

$$\begin{aligned} F(\overline{x}) - F(x) &= g_B^T(-A_B^{-1}A_N\Delta x_N) + g_N^T\Delta x_N + \frac{1}{2}(-A_B^{-1}A_N\Delta x_N, \Delta x_N)D(-A_B^{-1}A_N\Delta x_N, \Delta x_N) \\ &= [-g_B^TA_B^{-1}A_N + g_N^T]\Delta x_N + \frac{1}{2}\Delta^T x_N \\ &\qquad \begin{pmatrix} -A_B^{-1}A_N \\ I_N \end{pmatrix}^T D \begin{pmatrix} -A_B^{-1}A_N \\ I_N \end{pmatrix} \Delta x_N \end{aligned}$$

où $I_N = I(J_N, J_N)$ est une matrice d'identité d'ordre$(n-m)$.

Soit maintenant :

$$Z = Z(J, J_N) = \begin{pmatrix} -A_B^{-1}A_N \\ I_N \end{pmatrix}, M = M(J_N, J_N) = Z^T DZ. \tag{4.10}$$

Finalement (4.6) aura la forme suivante :

$$F(\overline{x}) - F(x) = E_N^T\Delta x_N + \frac{1}{2}(\Delta x_N)^T M(\Delta x_N) \tag{4.11}$$

4.5 Critère d'optimalité

Théorème 4.1 ***"Critère d'optimalité"***

Soit $\{x, J_B\}$ un plan de support des contraintes du problème (P), alors : Pour que x soit optimal, il suffit que le vecteur des estimations E vérifie les conditions suivantes :

$$\begin{cases} E_j \geq 0, & si \quad x_j = d_j^- \\ E_j \leq 0, & si \quad x_j = d_j^+ \\ E_j = 0, & si \quad d_j^- < x_j < d_j^+, \quad j \in J_N\ ; \end{cases} \tag{4.12}$$

Si de plus, le plan de support des contraintes est non dégénéré, alors les conditions (4.12) sont aussi nécessaires pour l'optimalité de x. [9][47]

Preuve [9]

Condition suffisante : Soit $\{x, J_B\}$ un plan de support des contraintes vérifiant les relations (4.12) . Pour un autre plan $\overline{x}$ du problème (P), la formule d'accroissement (4.11)devient :

$$F(\overline{x}) - F(x) = E_N^T\Delta x_N + \frac{1}{2}(\Delta x_N)^T M(\Delta x_N) \geq E_N^T\Delta x_N$$

car la matrice M est semi-définie positive. D'où

$$F(\overline{x}) - F(x) \geq \sum_{j\in J_N, E_j>0} E_j(\overline{c}_j - x_j) + \sum_{j\in J_N, E_j<0} E_j(\overline{c}_j - x_j)$$

Comme $d_j^- < \overline{x}_j < d_j^+$, on a :$\overline{x}_j - d_j^- \geq 0$ et $\overline{x}_j - d_j^+ \leq 0$. En vertu des relations d'optimalité (4.12), on aura :

$$F(\overline{x}) - F(x) \geq \sum_{j \in J_N, E_j > 0} E_j(\overline{c}_j - x_j) + \sum_{j \in J_N, E_j < 0} E_j(\overline{c}_j - x_j) \geq 0$$

d'où :

$$F(\overline{x}) \geq F(x)$$

Par conséquent, x est optimal.

Condition nécessaire : Raisonnons par l'absurde. Soit $\{x, J_B\}$ un plan de support optimal non dégénéré du probleme (P). Supposons que les relations (4.12) ne sont pas vérifiées, c'est à dire qu'il existe au moins un indice $j_0 \in J_N$ tel que :

$$E_{j_0} < 0, x_{j_0} < d_{j_0}^+$$

ou :

$$E_{j_0} > 0, x_{j_0} > d_{j_0}^-$$

On construit un plan $\overline{x} = x + \theta\ell$, où θ est un nombre réel positif et $\ell = \ell(J)$ un vecteur de direction d'amélioration à construire comme suit :

$$\begin{cases} \ell_{j_0} = -signE_{j_0}, \\ \ell_j = 0, j \neq j_0, j \in J_N. \end{cases}$$

D'autre part, on doit avoir :

$$A\overline{x} = Ax + \theta A\ell = b \Leftrightarrow A\ell = A_B\ell_B + A_N\ell_N = 0,$$

d'où :

$$\ell_B = -A_B^{-1}A_N\ell_N = A_B^{-1}a_j signE_{j_0}.$$

On a donc :

$$\begin{cases} \ell_{j_0} = -signE_{j_0}, \\ \ell_j = 0, j \neq j_0, j \in J_N, \\ \ell_B = A_B^{-1}a_{j_0} signE_{j_0}. \end{cases}$$

En outre, le vecteur $\overline{x}$ doit vérifier l'inégalité $d^- \leq \overline{x} \leq d^+ \Leftrightarrow d^- \leq x + \theta\ell \leq d^+$. Soit, en écrivant composante par composante

$$\begin{cases} d_j^- - x_j \leq \theta\ell_j \leq d_j^+ - x_j, j \in J_B \\ d_{j_0}^- - x_{j_0} \leq \theta\ell_{j_0} \leq d_{j_0}^+ - x_{j_0} \end{cases}$$

Comme le plan de support est non dégénéré, on peut toujours trouver un nombre positif θ assez petit tel que le vecteur $\overline{x}$ sera un plan pour le problème (P). D'aprés la formule d'accroissement (4.11), on aura alors :

$$\begin{aligned} F(\overline{x}) - F(x) &= E_N^T \Delta x_N + \frac{1}{2}(\Delta x_N)^T M(\Delta x_N) \\ &= \theta E_N^T \ell + \frac{1}{2}\theta^2 \ell_N^T M \ell_N \\ &= \theta(-E_{j_0} sign E_{j_0} + \frac{1}{2}\theta \ell_N^T M \ell_N) \\ &= \theta(-|E_{j_0}| + \frac{1}{2}\theta \ell_N^T M \ell_N) \end{aligned}$$

Pour θ et ℓ choisis on aura $-|E_{j_0}| + \frac{1}{2}\theta \ell_N^T M \ell_N < 0$ ce qui implique que : $F(\overline{x}) - F(x) < 0$. Par consequent, x n'est pas optimal, ce qui est contradictoire avec l'hypothese de départ. Donc les relations(4.12) sont forcément vérifiées si le plan de support optimal est non-dégénéré.

4.6 Critère de suboptimalité

Définition 27 (Estimation de suboptimalite)

Le nombre $\beta(x, J_B)$ dit estimation de suboptimalite est une valeur qui estime l'écart entre la valeur optimale $F(x^0)$ et une autre valeur $F(x)$ d'un plan de support des contraintes quelconque $\{x, J_B\}$. [9][47]

Théorème 4.2 *: Soit $\{x, J_B\}$ un plan de support des contraintes du problème (P) et β un nombre positif ou nul arbitraire. Si le nombre :*

$$\beta(x, J_B) = \sum_{j \in J_N, E_j > 0} E_j(x_j - d_j^-) + \sum_{j \in J_N, E_j < 0} E_j(x_j - d_j^+) \leq \epsilon, \tag{4.13}$$

alors $\{x, J_B\}$ est ϵ-optimal. [9]

Preuve : Soit x^* une solution optimale du problème (P). En remplacant x par x^* dans la formule d'accroissement (4.6) et en minorant l'éxpression, on aura [9] :

$$\begin{aligned} F(x^*) - F(x) &= E_N^T \Delta x_N + \frac{1}{2}\Delta x_N^T M \Delta x_N \\ &= \sum_{j \in J_N} E_j(x_j^* - x_j) + \frac{1}{2}\Delta x_N^T M \Delta x_N \\ &\geq \sum_{j \in J_N} E_J(x_j^* - x_j). \end{aligned}$$

D'où :

$$F(x) - F(x^*) \leq \sum_{j \in J_N} E_j(x_j - x_j^*) = \sum_{j \in J_N, E_j > 0} E_j(x_j - x_j^*) + \sum_{j \in J_N, E_j < 0} E_j(x_j - x_j^*)$$

Nous avons $d^- \leq x_j^* \leq d^+$, $j \in J_N$, alors on aura :

$$\begin{cases} E_j(x_j - x_j^*) \leq E_j(x_j - d_j^-) & Si \;\; E_j > 0, \\ E_j(x_j - x_j^*) \leq E_j(x_j - d_j^+) & Si \;\; E_j < 0. \end{cases}$$

D'où :

$$F(x) - F(x^*) = \sum_{j \in J_N, E_j > 0} E_j(x_j - d_j^-) + \sum_{j \in J_N, E_j < 0} E_j(x_j - d_j^+) \leq \epsilon.$$

Par conséquent, x est ϵ-optimal.

- Si $\beta(x) = \epsilon = 0$, alors x est optimal.
- Si $\beta(x) > \epsilon$, il faut améliorer le plan x.

4.7 Construction de l'algorithme

Avant de présenter la méthode de résolution, donnons quelques définitions essentielles.[9]

Définition 28 (Support de la fonction objectif)

On appelle support de la fonction objectif (4.1) l'ensemble des indices $J_S \subset J_N$ tel que $detM_S = detM(J_S, J_S) \neq 0$, où M est la matrice (4.10) . On pose $J_{NN} = J_N \setminus J_S$.

Définition 29 (Support d'un problème)

L'ensemble $J_p = \{J_B, J_S\}$, formé du support des contraintes J_B et de celui de la fonction objectif J_S, est appelé support du problème (P).

Définition 30 (Plan de support)

On appelle plan de support du problème (P) la paire $\{x, J_p\}$ formée du plan x et du support J_p ; il est dit accordé si $E(J_S) = 0$.

4.7.1 Construction d'une direction d'amélioration adaptée

Notation :

ℓ_s : direction correspondant aux indices appartenant à J_S,

ℓ_{NN} : direction correspondant aux indices appartenant à J_{NN},

ℓ_B : direction correspondant aux indices appartenant à J_B,

ℓ_N : direction correspondant aux indices appartenant à J_N.

Dans cet algorithme on choisira la métrique suivante pour les composantes non basiques de la direction admissible ℓ.

$$d_j^- - x_j \leq \ell_j \leq d_j^+ - x_j, j \in J_{NN} = J_N \setminus J_S. \tag{4.14}$$

Cette métrique dépend du plan x, et de ce fait, elle est dite adaptée. Afin de calculer les composantes de la direction admissible d'amélioration ℓ, considérons l'accroissement

$$\triangle F = F(x+\ell) - F(x) = \sum_{j \in J_N, E_j \geq 0} E_j \ell_j + \sum_{j \in J_N, E_j \leq 0} E_j \ell_j + \frac{1}{2} \ell_N^T M \ell_N.$$

En tenant compte de la métrique (4.14), la partie linéaire de ΔF atteint son minimum pour les valeurs des composantes de ℓ_{NN} suivantes :

$$\ell_j = \begin{cases} d_j^- - x_j, & Si \;\; E_j > 0, \\ d_j^+ - x_j, & Si \;\; E_j < 0, \\ 0, & Si \;\; E_j = 0, j \in J_{NN}. \end{cases} \tag{4.15}$$

On peut maintenant déduire les composantes de ℓ_S à partir de

$$E_j(x+\ell) = 0, j \in J_S$$

Par conséquent,

$$M(J_S, J_S)\ell_S + M(J_S, J_N)\ell_{NN} = 0$$

Ainsi on peut déduire :

$$\ell_S = -M_S^{-1} M(J_S, J_{NN})\ell_{NN}. \tag{4.16}$$

Quand aux composantes $\ell(J_B)$, elles sont déduites à partir de $A\ell = 0$, d'où :

$$\ell_B = -A_B^{-1}(A_S \ell_S + A_{NN}\ell_{NN}). \tag{4.17}$$

4.7.2 Changement de plan :

Construisons le nouveau plan $\overline{x}$ sous la forme suivante :

$$\overline{x} = x + \theta^0 \ell,$$

où ℓ est une direction d'amelioration définie par les formules (4.15)- (4.17)et le nombre θ est le pas le long de cette direction, avec $\theta^0 = min(1, \theta_{j1}, \theta_{j_S}, \theta_F)$. Les nombres $1, \theta_{j1}$et θ_{j_S}se calculent de telle sorte que les contraintes directes sur le vecteur $\overline{x}$ soient vérifiées :

$$d_j^- - x_j \leq \theta^0 \ell \leq d_j^+ - x_j, j \in J_{NN}. \tag{4.18}$$

$$d_j^- - x_j \leq \theta^0 \ell \leq d_j^+ - x_j, j \in J_B. \tag{4.19}$$

$$d_j^- - x_j \leq \theta^0 \ell \leq d_j^+ - x_j, j \in J_S. \tag{4.20}$$

Donc

$$\theta_{j_1} = \min_{j \in J_B} \theta_j, \theta_{j_s} = \min_{j \in J_S} \theta_j$$

Où

$$\theta_j = \begin{cases} \frac{d_j^+ - x_j}{\ell_j}, & Si \ \ \ell_j > 0, \\ \frac{d_j^- - x_j}{\ell_j}, & Si \ \ \ell_j < 0, \\ \infty, & Sinon. \end{cases}$$

Le nombre $\theta_0 = 1$ représente le pas correspondant aux indices de J_{NN}.
Quand à θ_F , il se calcule de façon que le passage du plan x au plan $\overline{x}$ assure une relaxation maximale de la fonction objectif tout en gardant le même signe pour E_j et $\overline{E_j}$, où

$$E_N^T(x) = g^T(x)Z, \overline{E_N}(\overline{x}) = \overline{E_N}(x + \theta^0 \ell),$$

$$\overline{E_N}(\overline{x}) = E_N(x) + \theta^0 M \ell_N = E_N(x) + \theta^0 \delta_N.$$

On posera donc $\theta_F = \sigma_{j_*} = min \ \ \sigma_j, j \in J_{NN}$, avec $\ \delta_j = M(j, J_N)\ell_N$

$$\sigma_j = \begin{cases} -\frac{E_j}{\sigma_j}, & Si \ \ E_j \delta_j < 0, \\ 0, & Si \ \ E_j = 0 \ \ et \ \ \delta_j < 0, \\ \infty, & Sinon. \end{cases}$$

4.7.3 Estimation de suboptimalité

Soit $\beta(\overline{x}, J_B)$ la nouvelle estimation de suboptimalite, nous allons maintenant la calculer en fonction de $\beta(x, J_B)$.

$$\begin{aligned}
\beta(\overline{x}, J_B) &= \sum_{j\in J_N, \overline{E_j}>0} \overline{E_j}(\overline{x_j} - d_j^-) + \sum_{j\in J_N, \overline{E_j}<0} \overline{E_j}(\overline{x_j} - d_j^+) \\
&= \sum_{j\in J_N, \overline{E_j}>0} \overline{E_j}(x_j + \theta^0\ell_j) - d_j^- + \sum_{j\in J_N, \overline{E_j}<0} \overline{E_j}(x_j + \theta^0\ell_j) - d_j^+ \\
&= \sum_{j\in J_N, \overline{E_j}>0} (E_j + \theta^0\delta_j)(x_j + \theta^0\ell_j - d_j^-) + \sum_{j\in J_N, \overline{E_j}<0} (E_j + \theta^0\delta_j)(x_j + \theta^0\ell_j - d_j^+). \\
&= \sum_{j\in J_N, E_j>0} E_j(x_j - d_j^-) + \sum_{j\in J_N, E_j>0} E_j\theta^0\ell_j \sum_{j\in J_N, E_j>0} \theta^0\delta_j(x_j - d_j^-) \\
&+ \sum_{j\in J_N, E_j>0} \theta^0\delta_j\theta^0\ell_j + \sum_{j\in J_N, E_j<0} E_j(x_j - d_j^+) \sum_{j\in J_N, E_j<0} E_j\theta^0\ell_j \sum_{j\in J_N, E_j<0} \theta^0\delta_j(x_j - d_j^+) \\
&+ \sum_{j\in J_N, E_j<0} \theta^0\delta_j\theta^0\ell_j
\end{aligned}$$

En vertu de la relation(4.15), l'estimation deviendra alors :

$$\begin{aligned}
\beta(\overline{x}, J_B) &= \beta(x, J_B)\theta^0 \sum_{j\in J_N, E_j>0} E_j(d_j^- - x_j) + \theta^0 \sum_{j\in J_N, E_j<0} E_j(d_j^+ - x_j) + \sum_{j\in J_N, E_j>0} \delta_j(-\ell_j) \\
&+ \theta^0 \sum_{j\in J_N, E_j<0} \delta_j(-\ell_j) + (\theta^0)^2 + \sum_{j\in J_N, E_j>0} \delta_j\ell_j(\theta^0)^2 \sum_{j\in J_N, E_j<0} \delta_j\ell_j \\
&= \beta(x, J_B) - \theta^0\beta(x, J_B) - (\theta^0\delta_N^T\ell_N + \theta^0)^2\delta_N^T\ell_N
\end{aligned}$$

Finalement, on aura la formule suivante :

$$\beta(\overline{x}, J_B) = (1 - \theta^0)\beta(x, J_B) - \theta^0(1 - \theta^0)\ell^T D\ell. \tag{4.21}$$

4.7.4 Changement de support

Le changement de support s'éffectue lorsque $\theta^0 < 1$ et $\beta(\overline{x}, J_B) > \epsilon$. Dans ce qui suit, on énumère les différents changements de support suivant la valeur que prend θ^0.

1. **Pour** $\theta^0 = 1$, le vecteur $x^0 = \overline{x} = x + \ell$ est une solution optimale du problème (P).
2. **Pour** $\theta^0 = \theta_{j_1}$, dans ce cas le choix de l'indice j_0 n'est pas unique contrairement au simplexe, ce qui est particulier à cette méthode. Lorsque ce cas se réalise pour un indice $j_1 \in J_B$, on a alors

$$\ell_{j_1} = -\sum_{j\in J_N} e_{j_1}^T A_B^{-1} a_j\ell_j = -\sum_{j\in J_N} x_{j_1 j}\ell_j \neq 0$$

où e est un vecteur unitaire de dimension m. Il existe alors $j_0 \in J_N$ tel que $x_{j_1 j_0} \neq 0$. Cette dernière condition nous assure par conséquent que $\overline{J} = (J_B \setminus j_1) \cup j_0$ est bel et bien un support. Si on peut avoir $x_{j_1 j_0} \neq 0$, avec $j_0 \in J_S$ on posera donc

$$\overline{J_B} = (J_B \setminus j_1) \cup j_0, \overline{J_S} = (J_S \setminus j_0).$$

Sinon, on choisira un indice $j_0 \in J_{NN}$ tel que $\ell j_0 \neq 0$ et on posera $\overline{J_B} = (J_B \setminus j_1) \cup j_0$ et $\overline{J_S} = J_S$.

3. **Pour** $\theta^0 = \theta_{j_S}$: on pose :$\overline{J_B} = J_B, \overline{J_S} = J_S \setminus j_s$.
4. **Pour** $\theta^0 = \theta_F$: dans ce cas on aurra $\overline{J_B} = J_B, \overline{J_S} = J_S \cup j_*$ tel que j_* est l'indice correspondant à θ_F.

Une fois ces étapes exécutées, on fera une autre itération avec comme plan de support $\{\overline{x}, \overline{J}_p\}$ si $\beta(\overline{x}, \overline{J}_B) > \epsilon$

4.8 Algorithme de la méthode adaptée pour la programmation quadratique convexe à variables hybrides

Nous commençons d'abord par présenter la méthode de résolution d'un problème quadratique convexe à variables hybrides, qui est représenté sous la forme suivante :

$$F(x) = \frac{1}{2} x^T D x + c^T x \to \min, \tag{4.22}$$

$$Ax = b, \tag{4.23}$$

$$d^- \leq x \leq d^+, \tag{4.24}$$

La méthode est résumée dans l'algorithme suivant :

Algorithme 8 Méthode adaptée pour la programmation quadratique convexe à variables hybrides

Début

1−Soit un plan de support initial$\{x, J_P\}$ du problème (P) ;

2−Calculer le vecteur des estimations $E_N^T = E^T(J_N) = g_N^T - u^T A_N$

si $\beta(x, J_B) \leq \epsilon$ **alors**

x est $\epsilon-$optimal ;

Aller à fin.

finsi

3−**Changement de plan**

Calculer la direction d'amélioration ℓ ;

Calculer le pas $\theta^0 = min(1, \theta_{j1}, \theta_{js}, \theta_F)$.

Calculer $\overline{x} = x + \theta^0 \ell$,

Calculer l'estimation de suboptimalité $\beta(\overline{x}, J_B) = (1 - \theta^0)(\beta - \theta^0 \ell^T D \ell)$

si $\beta(\overline{x}, J_B) \leq \epsilon$ **alors**

Aller à fin.

sinon

Aller à l'étape'4'.

finsi

4−**Changement de support**

si $\theta^0 = 1$ **alors**

$\overline{x} = x + \ell; \beta(\overline{x}, J_B) = 0;$

$\overline{x}$ est optimal

aller à fin.

sinon

si $\theta^0 = \theta_{j_1}$ **alors**

si $j_0 \in J_S$ **alors**

$\overline{J_B} = (J_B \setminus j_1) \cup j_0, \overline{J_S} = (J_S \setminus j_0).$

sinon

si $j_0 \in J_{NN}$ **alors**

$\overline{J_B} = (J_B \setminus j_1) \cup j_0$ et $\overline{J_S} = J_S.$

finsi

sinon

si $\theta^0 = \theta_{j_s}$ **alors**

$\overline{J_B} = J_B, \overline{J_S} = J_S \setminus j_s.$

sinon

si $\theta^0 = \theta_F$ **alors**

$\overline{J_B} = J_B, \overline{J_S} = J_S \cup j_*$

finsi

finsi

finsi

si $\beta(\overline{x}, \overline{J_B}) > \epsilon$ **alors**

aller à l'étape "2",en partant de $\{x, J_P\}$;

finsi

finsi

finsi

fin ;

4.9 Proposition d'une nouvelle méthode de résolution de Problème quadratique biobjectif à variables mixtes

Dans cette partie nous présentons le fonctionnement de notre méthodes adaptée et les interventions faites dans l'algorithme (8) pour atteindre notre but.

Soit : $\overline{x} = x + \theta^0 \ell$, où :

- x a K éléments positifs.
- $\overline{x}$ a K_1 éléments positifs.

4.9.1 Intervention sur la direction d'amélioration

Algorithme 9 Intervention sur la direction d'amélioration

si $K_1 > K$ **alors**

i=indice (max E_j) ;

$\ell_i = -\frac{x_i}{\theta^0}$;

k = indice (minE_j) ;

$\ell_k = \ell_k + \frac{x_i}{\theta^0}$;

sinon

si $K_1 < K$ **alors**

H = indice j : $x_j = 0$;

i = indice (max E_j) ; $j \notin H$;

$\ell_i = -\frac{x_i}{\theta^0} + \frac{\frac{x_i}{K-K_1+1}}{\theta^0}$;

j = les $(K - K_1)$ premiers indices selon l'ordre croissant de E_j ;

$\ell_j = \frac{\frac{x_i}{K-K_1+1}}{\theta^0}$;

finsi

finsi

4.9.2 Intervention dans le changement de plan

Algorithme 10 Intervention dans les changement de plan

si $K_1 > K$ **alors**

i=indice (max E_j) ;

$C = x_i$;

$x_i = 0$;

$k =$ indice(min E_j) ;

$x_k = x_k + C$;

sinon

si $K_1 < K$ **alors**

$H =$ indice j : $x_j = 0$;

$R = (K - K_1)$ premiers indices selon l'ordre croissant de E_j ;$j \in H$;

$j =$ indice (max E_j) ;$j \notin H$;

$C = \frac{x_j}{K - K_1 + 1}$;

$x_j = C$;

$x(i \in R) = C$;

finsi

finsi

4.10 Algotithme général de la méthode

Algorithme 11 Méthode adaptée pour la programmation quadratique bi-objectif en variables mixte

Début
1−Soit un plan de support initial$\{x, J_P\}$ du problème $(P_Q{=}x^t Dx)$;
2−Calculer le vecteur des estimations $E_N^T = E^T(J_N) = g_N^T - u^T A_N$
si $\beta(x, J_B) \leq \epsilon$ **alors**
  x est $\epsilon-$optimal ;
  Aller à (5).
finsi
3−**Changement de plan**
Calculer la direction d'amélioration ℓ ;
Calculer le pas $\theta^0 = min(1, \theta_{j1}, \theta_{j_S}, \theta_F)$.
Calculer $\overline{x} = x + \theta^0 \ell$, avec **l'algorithme (9)** ou **l'algorithme (10)**
Calculer l'estimation de suboptimalité $\beta(\overline{x}, J_B) = (1 - \theta^0)(\beta - \theta^0 \ell^T D\ell)$
si $\beta(\overline{x}, J_B) \leq \epsilon$ **alors**
  Aller à (5).
sinon
  Aller à l'étape'4'.
finsi
4−**Changement de support**
si $\theta^0 = 1$ **alors**
  $\overline{x} = x + \ell$;
  l'algorithme (9) ou **l'algorithme (10)**
  $\beta(\overline{x}, J_B) = 0$;
  $\overline{x}$ est optimal
sinon
  si $\theta^0 = \theta_{j_1}$ **alors**
    si $j_0 \in J_S$ **alors**
      $\overline{J_B} = (J_B \setminus j_1) \cup j_0, \overline{J_S} = (J_S \setminus j_0)$.
    sinon
      si $j_0 \in J_{NN}$ **alors**
        $\overline{J_B} = (J_B \setminus j_1) \cup j_0$ et $\overline{J_S} = J_S$.
      finsi
    sinon
      si $\theta^0 = \theta_{j_s}$ **alors**
        $\overline{J_B} = J_B, \overline{J_S} = J_S \setminus j_s$.
      sinon
        si $\theta^0 = \theta_F$ **alors**
          $\overline{J_B} = J_B, \overline{J_S} = J_S \cup j_*$
        finsi
      finsi
    finsi
    si $\beta(\overline{x}, \overline{J_B}) > \epsilon$ **alors**
      aller à l'étape "2",en partant de $\{x, J_P\}$;
    finsi
  finsi
finsi
Eff=Eff $\cup$ x ;
5−**Soit Z(max) la deuxième fonction objectif .**
calculons $\mathbf{Z}_i(\mathbf{x}){=}\mathbf{C}^t{-}\mathbf{C}_b^t\mathbf{A}_b^{-1}$ **A ;**
si $\exists$ **i :**$\mathbf{Z}_i(\mathbf{x}){>}\mathbf{0}$ **alors**
  Ajouter $\sum_{i \in H} X_i \geq 1$ **;H={i :**$_i$**(x)>0 ;**
  Aller à (2)
sinon
  Arrêter
finsi
fin ;

4.11 Conclusion

Aprés avoir la construction de l'algorithme de la méthode adaptée et les déffirents définitions pour sa comprehension, nous implimenterons cette méthode dans logiciel de programmation Matlab, ce qui est présenté dans le chapitre suivant.

Chapitre 5

Implémentation de la méthode adaptée et application sur les problèmes d'optimisatios des portfeuilles

"Le fondement de la théorie c'est la pratique".
(Mao Tsé-Toung)

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. Dans ce domaine, un logiciel commercial est devenu, ces dernières années, presque incontournable : il s'agit de Matlab de la société The Mathworks.
Ce dernier met à la disposition de l'utilisateur un environnement performant pour mener à bien les calculs numériques.

5.2 Choix du langage

Le choix s'est porté sur l'emploi du langage du logiciel Matlab 2013, car il répond aux critéres suivants :

– La maniabilité du langage : constitué d'un ensemble de possibilités faisant en sorte que

le programmeur travaille avec aisance, assuré d'une part par la syntaxe du langage et d'autre part par un aspect visuel clair représentatif à la fois du détail et du global.
- Le bagage du langage : il contient une interface graphique puissante ainsi qu'une grande variété de méthodes scientifiques implémentées (prédéfinies).

5.2.1 Généralités sur le langage

Matlab est un logiciel parfaitement dédié à la résolution de problèmes d'analyse numérique ou de traitement du signal. Permet d'effectuer des calculs matriciels ou de visualiser les résultats sous forme graphique. La formulation des problèmes s'apparente à la formulation mathématique des problèmes à résoudre. L'utilisation de ce logiciel consiste à lancer des lignes de commandes, qui peuvent le plus souvent ressembler à la programmation en C.

Le nom Matlab vient de MATrix LABoratory, les éléments de données de base manipulés par Matlab étant des matrices (mais pouvant évidemment se réduire à des vecteurs et des scalaires) qui ne nécessitent ni dimensionnement ni déclaration de type. Contrairement aux langages de programmation classiques, les fonctions du Matlab permettent de manipuler directement et intéractivement ces données matricielles, le rendant ainsi particuliérement efficace en calcul numérique, analyse et visualisation de données en particulier. Il existe deux modes de fonctionnement sur Matlab :

Le mode interactif : les instructions sont exécutées au fur et à mesure qu'elles sont données par l'usager.

Le mode exécutif : dans ce cas, l'utilisateur utilise un fichier "M-file" contenant toutes les instructions à exécuter.

5.2.2 Programmation avec Matlab

Il y a deux façons pour écrire des fonctions Matlab :
- Soit directement dans la fenêtre de commandes,
- Soit en utilisant l'éditeur de développement de Matlab, en sauvegardant les programmes dans des fichiers texte avec l'extension ".m".

Fichiers *.m

Les programmes sauvegardés dans les fichiers Matlab (*.m) sont alors directement utilisables comme des fonctions Matlab à partir de la fenétre de commande. Pour cela, le fichier doit se trouver dans le répertoire Matlab, qui est en pratique le dossier Work.

Création d'une fonction

La création d'une fonction dans Matlab se fait par la syntaxe suivante : function[s1, s2,...]=nom-fonction(e1, e2,...). Les variables s1, s2,. . . , sont les arguments de sortie (S) de la fonction et les variables e1,e2,. . . , sont les arguments d'entrée (E).
Attention : le fichier M(M-file) doit avoir le même nom que la fonction qu'il contient.

5.3 Présentation du prologiciel

Pour que le lecteur puisse utiliser notre application dans des bonnes conditions, nous allons consacrer cette partie à la présentation de l'application de notre méthode.
Les photos ci-dessous représentent la démarche à suivre pour utiliser notre application.

FIGURE 5.1 – Première interface du programme

Cette interface contient le titre de notre méthode, les réalisateurs et l'encadreur de ce travail avec un bouton "Entrer" qui nous permet d'accéder à la prochaine interface.

FIGURE 5.2 – Interface du choix de type de problème

Dans cette interface nous avons trois boutons.

1. Résolution de Problèmes de Programmation Quadratique Mono-Objectif.
2. Résolution de Problèmes de Programmation Bi-Objectif Quadratique en Variables Mixtes.
3. Application sur les problèmes d'optimisation des portefeuilles.

En cliquant sur le premier bouton, voici l'interface qui s'affiche :

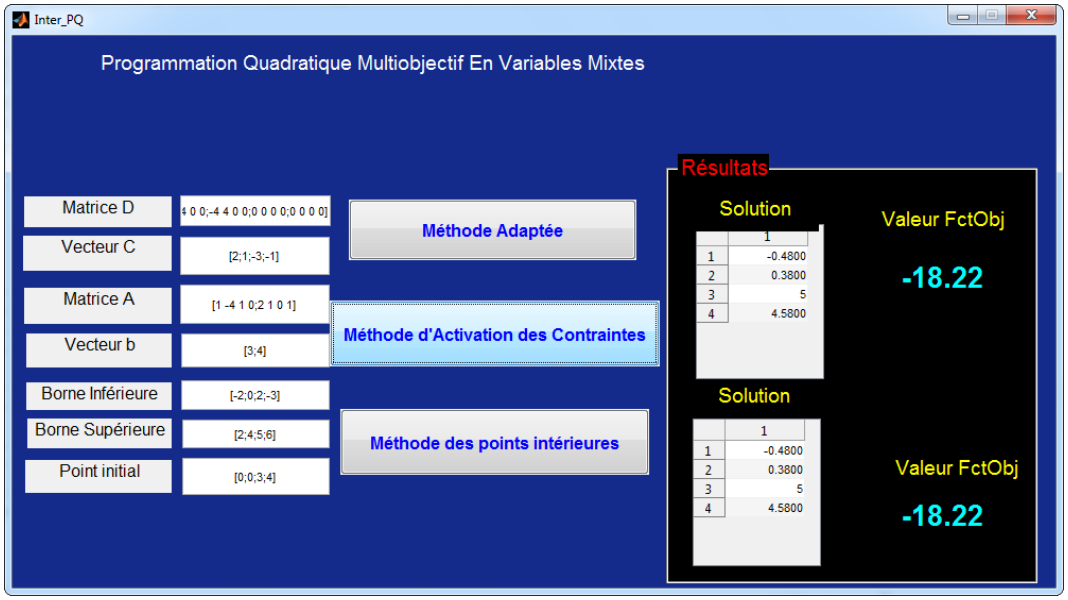

FIGURE 5.3 – Programmation Quadratique Mono-Objectif.

Cette affiche représente trois méthodes de résolution : notre méthode adaptée et deux autres(méthode d'activation des contraintes et la méthode des points intérieures) pour faire une étude comparative.

En faisant rentrer les données necessaires, nous remarquons que les résultats qui s'affichent pour notre méthode adaptée sont les mêmes pour les deux autres méthodes.
Et lorsque on clique sur le deuxième bouton, l'interface qui s'affiche est la suivante :

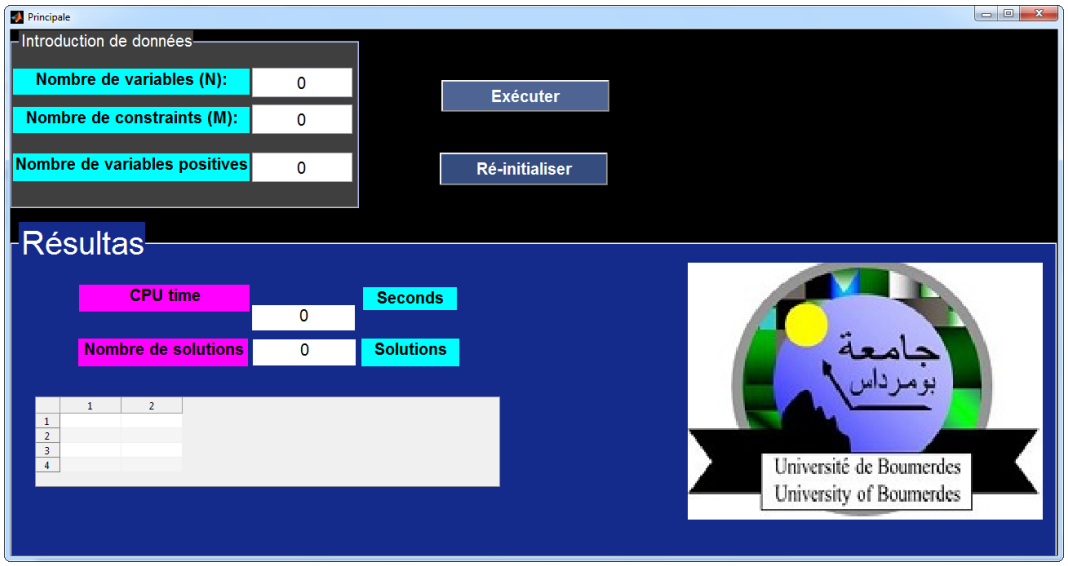

FIGURE 5.4 – Interface de PQBVM

Aprés avoir remplir le nombre de variables, le nombre de contraintes et le nombre de variables positives, nous avons comme résultats : le nombre de solutions efficaces ainsi que le CPU time. le tableau contient la valeur de chaque fonction objectif pour chaque solution efficace et le graphe représente le front de Pareto, ce qui est présenté dans l'interface ci-dessous :

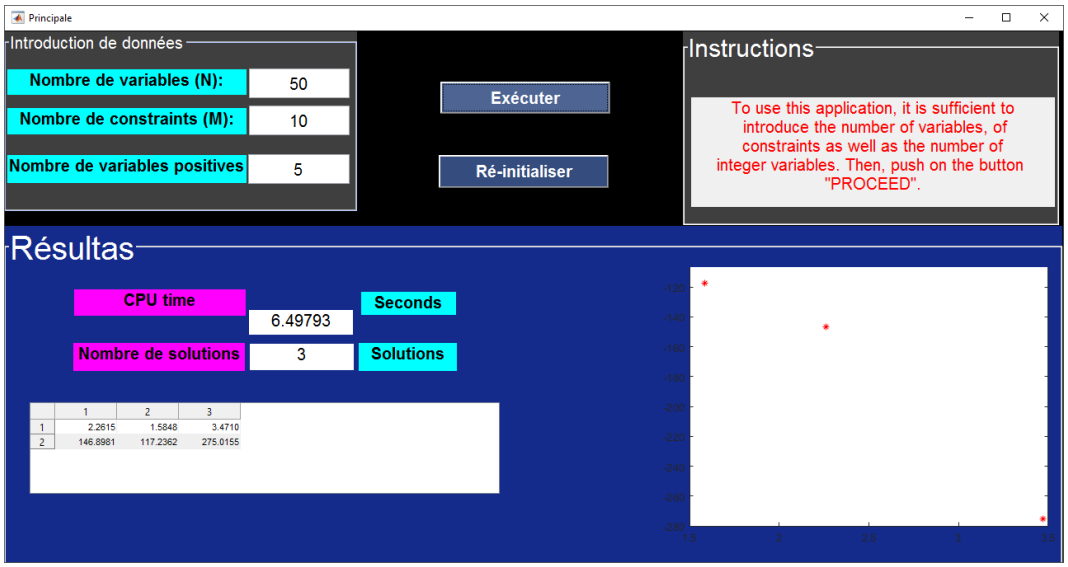

FIGURE 5.5 – Programmation Bi-Objectif Quadratique en Variables Mixtes.

5.4 Application de la méthode sur le problème de portfeuille

La théorie moderne du portefeuille est une théorie financière développée en 1952 par Harry Markowitz. Elle expose comment des investisseurs rationnels utilisent la diversification afin d'optimiser leur portefeuille, et quel devrait être le prix d'un actif étant donné son risque par rapport au risque moyen du marché.

5.4.1 Etat de l'art de l'optimisation de portefeuille avec contrainte de cardinalité

Le travail le plus important dans l'optimisation des portefeuilles revient à Markowitz [1] [48] qui a proposé un modèle "moyenne-variance" pour mesurer le rendement attendu et le risque du portefeuille.

Supposons qu'il existe un marché avec n actifs pour un investisseur avec un capital donné à investir, soit R_i le rendement aléatoire de le i^{eme} actif. Si le capital est investi dans cet actif et μ_i=E(R_i) est l'esperence de R_i pour i=1....n. Désignons R=(R_1, R_2... R_n)^t et μ=(μ_1 μ_2 μ_n) ^{t}, de plus, soit Q = $[Q_{ij}]_{n\times n}$ avec Q_{ij} = E[(R_i - μ_i)(R_j - μ_j)] $\in$ S^n la matrice de covariance de R, où S^n est l'ensemble des matrices symétriques d'ordre n, alors, nous savons que Q $\in$ S_+^n, où S_+^n est l'ensemble des matrices semi-définies positives d'ordre n. Définir $x = (x_1, ..., x_n)^t$ un vecteur de dimension n avec x_i étant la partie du capital total investi dans la i^{eme} actif, par conséquent, le rendement attendu et la variance du rendement du portefeuille sont : μ^t x et x^t Qx respestivement. le modèle classique "moyenne- variance" peut être exprimé sous la forme suivante :

$$(MV)\begin{cases} \min f(x) = x^t Q x \\ s.c.\ \mu^t x \geq \rho \\ (e^n)^t x = 1 \\ x \in X \end{cases}$$

où $\rho \in \Re$ est un niveau de rendement prévu, la contrainte μ^tx $\geq \rho$ nécessite que le rendement du portefeuille soit au moins égale au niveau de rendement prévu, $(e^n) \in \Re^n$ est un vecteur avec chacune de ces composantes égale 1, la contrainte $(e^n)^t x = 1$ indique que tous le capital est investi, et X $\in$ $\Re^n$ est un ensemble formé par d'autres contraintes déterministes sur x tels que la contrainte budgétaire, Contrainte de secteur, bornes inférieures et supérieures .
Au cours des soixante dernières années, différentes extensions du modèle moyenne-variance ont été proposées et sont basées sur le travail de Markowitz[48].

1. Markowitz : economiste américain, prix Nobel d'économie en 1990

Elton et Gruber[55] ont étudié le court cas de vente qui permet à certains de x à être négative. Konno et Yamazaki [46] ont développé un modèle d'écart absolu moyen (MAD). Young [78]a maximisé le rendement du pire scénario dans un modèle de sélection du portefeuille minimax. Des informations sur les modèles d'optimisation traditionnelles de sélection du portefeuille peut être trouvée dans [57]. En outre, en raison des préoccupations des coûts de transaction et de gestion, des contraintes commerciales réelles tels que la contrainte de cardinalité, contraintes maximales de maintien, les seuilles d'achat minimal sont introduits dans les modèles de sélection du portefeuille. Ces contraintes ont les formes suivantes :

$$|Supp(x)| \leq K \;\; et \;\; l_i \leq x_i \leq u_i \;\; , \forall i \in Supp(x) \tag{5.1}$$

où

$$Supp(x) = \{i | x_i \neq 0, 1 \leq i \leq n\} \tag{5.2}$$

K est un entier positif tel que : K < n , 0< l_i < u_i < 1 l_i est le seuil d'achat minimum et u_i est le niveau d' attente maximal pour le i^{eme} actif.
Le problème de la sélection du portefeuille de cardinalité contraintes est exprimé sous la forme suivante :

$$(PSP) \begin{cases} \min f(x) = x^t Q x \\ s.c.\ \mu^t x \geq \rho \\ (e^n)^t x = 1 \\ |Supp(x)| \leq K \\ x_i \geq l_i, \forall i \in Supp(x) \\ 0 \leq x_i \leq u_i, i = 1, ..., n \end{cases}$$

Bienstock [12] a signalé que le problème (PSP) est NP-difficile. Plusieurs algorithmes de Branch and bound basé sur la résolution des différents sous-problèmes ont été proposées pour résoudre le problème de sélection de portefeuille avec contrainte de cardinalité.Xie et al . [77] ont étudié une approche aléatoire pour obtenir une solution d'approximation au modèle moyenne-variance avec contrainte de cardinalité et d'autres contraintes latérales .
Frangioni et Gentile [26][27][28] ont proposé une méthode des coupes(relaxation et coupée)pour une classe de 0-1, le programme quadratique convexe en nombre entier avec des variables semi-continu qui comprend la contrainte de cardinalité est basé sur ce resultat. En outre , Zheng et al . [79]appliqué un schéma de décomposition lagrangienne spéciale du problème (PSP) . Cela conduit à une programmation SDP pour calculer la "meilleure" décomposition en diagonale. Gao et Li [30]ont étudié les structures et propriétés géométriques derrière la formulation mathématique et appliqué la relaxation lagrangienne au problème primal dans une situation de cardinalité pure.

D'autres auteurs ont étudié certaines méthodes heuristiques et la recherche locale dans les modèles de sélection de portefeuille avec contrainte de cardinalité.
autre formulation est proposée pour le problème de sélection de portefeuille avec contrainte de cardinalité, avant de la présenter, nous définissons les variables utiliser dans cette formulation :
n :nombre des actifs valable.
μ_i l'esperence de rendement de l'actif i.
x_i la variable de décision qui représente la proportion de l'investisssement dans chaque actif i.
K :le nombre d'actif souhaité dans le portefeuille.
z_i une variable binaire tq :

$$z_i = \begin{cases} 1 & \text{Si l'actif i est sélectionné} \\ 0 & sinon \end{cases}$$

ϵ_i : la proportion ,minimale qui doit être tenu de l'actif i(i=$\overline{1,n}$).
δ_i la proportion maximale qui doit être tenu de l'actif i (i=$\overline{1,n}$).
où il faut :0$\leq \epsilon_i \leq \delta_i \leq 1$.(i=$\overline{1,n}$).

$$(P)\begin{cases} \min \ X^tQX \\ \max \ \mu^tX \\ S.C \\ \sum_{i=1}^{n} x_i = 1 \quad (i=\overline{1,n}). \\ \sum_{i=1}^{n} z_i = K \quad (i=\overline{1,n}). \\ \epsilon_i z_i \leq x_i \leq \delta_i z_i \quad (i=\overline{1,n}). \\ z_i = \begin{cases} 1 \\ 0 \end{cases} \end{cases}$$

La première fonction objectif minimise le risque de portefeuille.
La deuxième fonction objectif maximise le rendement de portefeuille.
la première contrainte assure que le capital est investi complètement.
la deuxième contrainte représente la contrainte de cardinalité qui assure que le nombre des actifs qui sont sélectionnés égale éxactement à K.
la troisième contrainte assure que si l'actif i est sélectionné donc sa proportion est compris entre ϵ_i et δ_i
Il existe une autre représentation de ce problème qui est mono-objectif (transformation de problème multi-objectif en mono-objectif en se basant sur la méthode d'agrégation) :

$$(P')\begin{cases} \min \ \ \lambda(X^tQX) - (1-\lambda)\mu^tX \\ S.C \\ \sum_{i=1}^{n} x_i = 1 \quad (i = \overline{1,n}). \\ \sum_{i=1}^{n} z_i = K \quad (i = \overline{1,n}). \\ \epsilon_i z_i \leq x_i \leq \delta_i z_i \quad (i = \overline{1,n}). \\ z_i = \begin{cases} 1 \\ 0 \end{cases} \end{cases}$$

5.4.2 Définitions

Définition 31 (La rentabilité[20])

La rentabilité mesure l'appréciation (ou dépréciation) relative de la valeur d'un actif financier ou d'un portefeuille d'actifs financiers entre deux instants successifs.

Soient t et $t+1$ deux instants successifs. Nous notons P_t et P_{t+1} les valeurs (prix) de l'actif aux instants t et t+1, respectivement. Nous calculons la rentabilité réalisée dans l'intervalle $[t, t+1]$ par : $r_t = (P_{t+1} - P_t)/(P_t)$.

Définition 32 (Risque d'un portefeuille)

Tant qu'un portefeuille est composé de titres dont les rentabilités ne varient pas toutes de façon exactement parallèle, son risque est inférieur à la moyenne des risques de ces titres. Autrement dit, la théorie du portefeuille démontre qu'en prenant un échantillon de titres, pour une rentabilité donnée, peut réduire le niveau du risque.
La gestion de portefeuille moyenne-variance fait l'hypothèse que les agents sont rationnels et ont de l'aversion pour le risque. Pour comparer et sélectionner les titres, ils mettent en balance l'intérêt que ces titres procurent avec le risque qu'ils font subir. L'agent essaie d'obtenir un rendement maximum pour un risque minimum. En effet, il est logique qu'un investisseur sacrifie un peu de rentabilité pour diminuer le risque de ses portefeuilles.

Définition 33 (Frontière efficiente)

La frontière qui caractérise le polygone ou la courbe des contraintes s'appelle dans cette situation la "frontière efficiente de Markowitz" et dans le polygone/courbe se situent tous les portefeuilles à rejeter dits "portefeuilles domin´ es". Une autre manière de formuler ceci consiste à dire que les combinaisons (rendement, risque) de cette frontière forment un ensemble d'optima, c'est à dire que si l'un des éléments augmente, l'autre doit augmenter aussi.

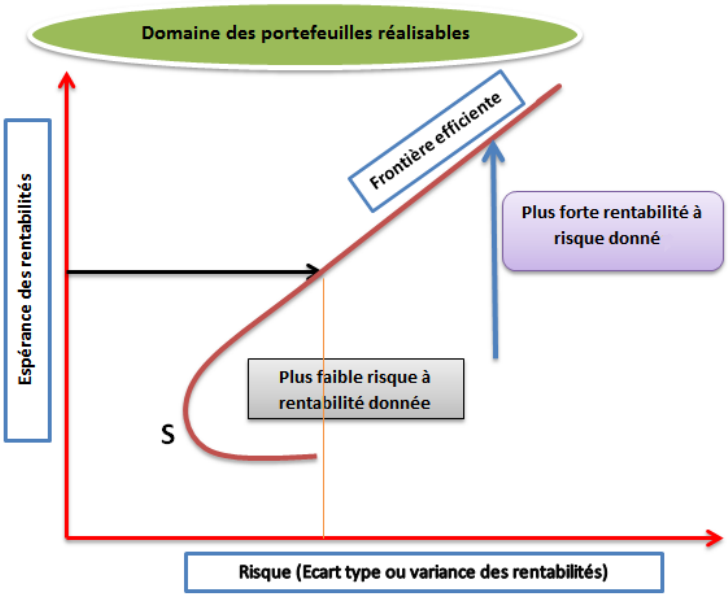

FIGURE 5.6 – Frontières efficaces

5.4.3 Implémentation de la méthode adaptée sur les problèmes des portefeuilles

Comme nous avons présenter dans l'interface (5.2), en cliquant sur le bouton "Application sur les problèmes d'optimisation des portefeuilles", voici l'interface qui s'affiche :

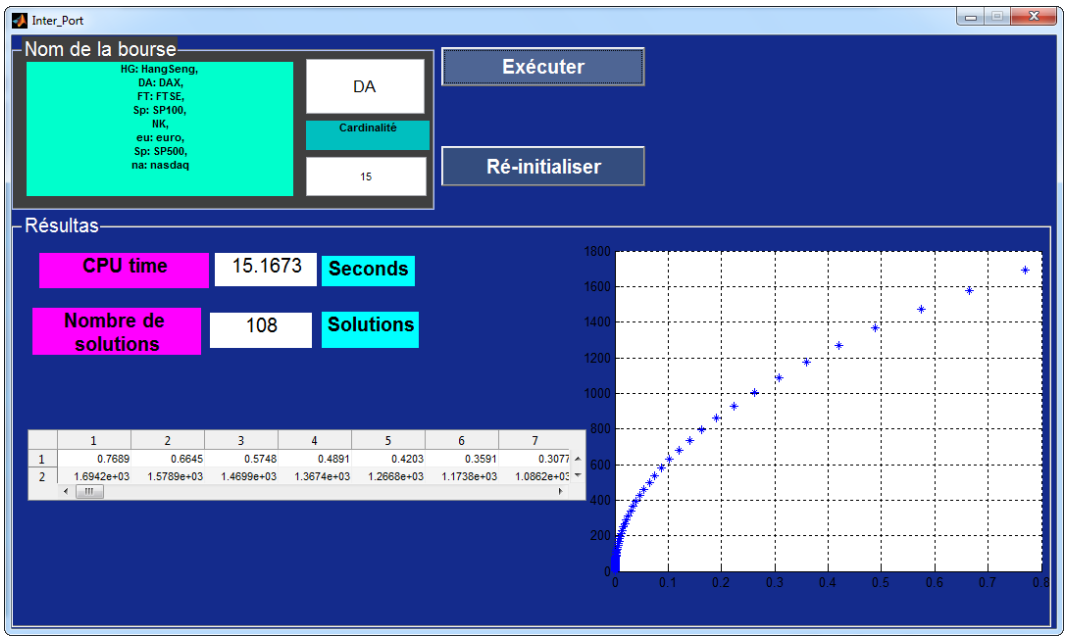

FIGURE 5.7 – Application sur les portefeuilles

Dans cette interface nous avons comme des données : le nom de la bourse et la cardinalité qui est le nombre d'actifs qu'on veut investir, et comme résultats : le temps d'éxécution(secondes) et le nombre des solutions efficaces avec un graphe qui représente le front de pareto.
Dans le tableau suivant nous montrons que, pour chaque problème, la méthode a pu trouver la frontière efficace dans un délais raisonnable.

Remarque 5

L'étude de la simulation a été réalisée sur sept ensembles de données de références, cinq d'entre eux sont dérivées de (Http ://people.brunel.ac.uk/*mastjjb/jeb/orlib/portinfo.html) mis à la disposition dans OU Bibliothèque du Beasley et deux ensembles de données supplémentaires décrit par Cesarane 2014, au (http ://w3.uniroma1.it/Tardella/datasets.html). Ils rapportent les 263 prix faiblement de Mars 2003 à Mars 2008 SP500 (476 actifs) et NASDAQ (2191 actifs) des indices de marché.

		K=10	K=20	K=30	K=50	K=80	K=100	K=N
2*Hang Seng N=37	Number of Eff Sol	119	119	114	-	-	-	105
	Cpu Time	10,08	10,15	15,64	-	-	-	13,57
2*DAX100 N=85	Number of Eff Sol	108	110	111	113	-	-	100
	Cpu Time	10,15	14,51	17,97	27,47	-	-	32.97
2*FTSE100 N=89	Number of Eff Sol	181	178	175	166	144	-	130
	Cpu Time	4,28	5,78	7,74	12,24	14,52	-	14,68
2*S&P100 N=98	Number of Eff Sol	111	111	109	108	98	-	57
	Cpu Time	3,06	3,94	5,35	8,42	10,15	-	7.49
2*Nikkei N=225	Number of Eff Sol	973	959	943	902	801	322	221
	Cpu Time	35,56	82,63	143,33	209,92	394,78	187,01	94,65
2*S&P500 N=478	Number of Eff Sol	95	92	89	79	55	17	22
	S&P100	26,17	27,91	28,56	35,90	40,71	18,48	18,19
2*Nasdaq N=2196	Number of Eff Sol	194	241	265	307	238	159	337
	Cpu Time	1544,21	1155,33	511,74	506,96	437,23	162,27	329,76

5.5 Conclusion

On remarque que notre algorithme, converge en un temps très acceptable, et que le modèle qu'on propose dans ce travail donne plus de choix au décideur qui contrairement aux modèle classique mono-objectif, le décideur aura à choisir son portefeuille directement parmi les solutions efficaces.

Conclusion générale

"La théorie, c'est quand on sait tout et que rien ne fonctionne. La pratique, c'est quand tout fonctionne et que personne ne sait pourquoi. Ici, nous avons réuni théorie et pratique : Rien ne fonctionne... et personne ne sait pourquoi !".

(Albert Einstein)

Dans ce mémoire, nous sommes arrivés à construire une nouvelle méthode qui permet de résoudre un problème quadratique biobjectif en variables mixtes. Pour cela, nous avons commencé par présenter des définitions, des théorèmes et des méthodes de résolutions de la programmation multiobjectifs, la programmation quadratique et la programmation en nombre entier dans les trois premiers chapitres respectivement.

En s'inspirant des méthodes de supports pour la résolution des problèmes linéaires et quadratiques, développées par R. Gabassov et al, ainsi que des travaux de M.O. Bibi, S. Radjef[58] pour la méthode de support pour le cas des variables mixtes, nous avons mis au point dans le quatrième chapitre notre nouvelle méthode adaptée qui est basée sur une métrique différente de celle du simplexe, tout en donnant un algorithme qui explique d'une façon explicite les étapes de la méthodes.

Enfin, afin de tester l'efficacité de cette méthode, nous l'avons implémenter sous le logiciel de programmation Matlab et appliquer sur les problèmes d'optimisation des portefeuilles. Les résultats obtenus par la méthode adaptée proposée sont très encourageants en matière de temps d'exécution.

Comme perspectives, nous proposons une étude plus poussé de la méthode proposée. Ainsi qu'une comparaison avec des méthodes existantes comme celle d'activation de contraintes avec des benchmark et des problèmes tests.

Bibliographie

[1] Abdellaoui, M., and Gonzales, C. Théorie de l'utilité multi-attribut.

[2] Asli, L. *Approche hybride pour les problèmes d'optimisation combinatoire multiobjectif.* PhD thesis, 2010.

[3] Barichard, V. *Approches hybrides pour les problèmes multiobjectifs.* PhD thesis, Université d'Angers, 2003.

[4] Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W., and Vance, P. H. Branch-and-price : Column generation for solving huge integer programs. *Operations research 46*, 3 (1998), 316–329.

[5] Benlahrache, N. Optimisation Multi-Objectif Pour l' Alignement Multiple de Séquences.

[6] Berro, A. *Optimisation multiobjectifs et stratégies d'évolution en environnement dynamique.* PhD thesis, ANRT [diff.], 2001.

[7] Berro, A. Algorithmes évolutionnaires pour l'optimisation multi-objectif. *Technique et Science Informatiques 25*, 8-9 (2006), 991–1021.

[8] Bertsekas, D. P., Bertsekas, D. P., Bertsekas, D. P., and Bertsekas, D. P. *Dynamic programming and optimal control*, vol. 1. Athena Scientific Belmont, MA, 1995.

[9] Bezoui, M. Méthode adaptée de programmation quadratique convexe.

[10] Bezoui, M. Cours deuxième année recherche opérationnelle.

[11] Bezoui, M. Cours master recherche opéretionnelle "Optimisation multiobjectif".

[12] Bienstock, D. Computational study of a family of mixed-integer quadratic programming problems. *Mathematical programming 74*, 2 (1996), 121–140.

[13] Boland, N. L. A dual-active-set algorithm for positive semi-definite quadratic programming. *Mathematical Programming 78*, 1 (1996), 1–27.

[14] Coello, C. A. An updated survey of GA-based multiobjective optimization techniques. *ACM Computing Surveys (CSUR) 32*, 2 (2000), 109–143.

[15] Collette, Y., and Siarry, P. *Optimisation multiobjectif*. Editions Eyrolles, 2002.

[16] Dantzig, G. B. Maximization of a linear function of variables subject to linear inequalities, in Activity Analysis of Production and Allocation.

[17] Dantzig, G. B., and Thapa, M. N. *Linear programming 1 : introduction*. Springer Science & Business Media, 2006.

[18] Dantzig, G. B., and Wolfe, P. Decomposition principle for linear programs. *Operations research 8*, 1 (1960), 101–111.

[19] Deroussi, L. *Heuristiques, métaheuristiques et systèmes de voisinage : application à des problèmes théoriques et industriels de type TSP et ordonnancement*. PhD thesis, Clermont-Ferrand 2, 2002.

[20] DINH, P. T. P., and de Rouen, I. La programmation DC et DCA pour l'optimisation de portefeuille.

[21] Eschenauer, H., Fuchs, W., Post, P. U., Adali, S., Duffy, K. J., Stenvers, K.-H., Koski, J., and Silvennoinen, R. *Structures Made of Advanced Materials*. Springer, 1990.

[22] Fiacco, A. V., and McCormick, G. P. *Nonlinear programming : sequential unconstrained minimization techniques*, vol. 4. Siam, 1990.

[23] Figueira, J., Greco, S., and Ehrgott, M. *Multiple criteria decision analysis : state of the art surveys*, vol. 78. Springer Science & Business Media, 2005.

[24] Fletcher, R. A general quadratic programming algorithm. *IMA Journal of Applied Mathematics 7*, 1 (1971), 76–91.

[25] Fletcher, R. *Practical methods of optimization*. John Wiley & Sons, 2013.

[26] Frangioni, A., and Gentile, C. Perspective cuts for a class of convex 0−1 mixed integer programs. *Mathematical Programming 106*, 2 (2006), 225–236.

[27] Frangioni, A., and Gentile, C. SDP diagonalizations and perspective cuts for a class of nonseparable MIQP. *Operations Research Letters 35*, 2 (2007), 181–185.

[28] Frangioni, A., and Gentile, C. A computational comparison of reformulations of the perspective relaxation : SOCP vs. cutting planes. *Operations Research Letters 37*, 3 (2009), 206–210.

[29] Gabasov, R., Kirillova, F. M., and Tyatyushkin, A. I. Constructive methods of optimization. *PI-University Press, Minsk* (1984).

[30] Gao, J., and Li, D. Optimal cardinality constrained portfolio selection. *Operations research 61*, 3 (2013), 745–761.

[31] Gardeux, V. *Conception d'heuristiques d'optimisation pour les problèmes de grande dimension. Application à l'analyse de données de puces à ADN.* PhD thesis, Université de Paris-Est, 2012.

[32] Garfinkel, R. S., and Nemhauser, G. L. *Integer programming*, vol. 4. Wiley New York, 1972.

[33] Gill, P. E., and Murray, W. Numerically stable methods for quadratic programming. *Mathematical programming 14*, 1 (1978), 349–372.

[34] Glover, F. Future paths for integer programming and links to artificial intelligence. *Computers & operations research 13*, 5 (1986), 533–549.

[35] Glover, F. Tabu search-part I. *ORSA Journal on computing 1*, 3 (1989), 190–206.

[36] Glover, F. Tabu searchâ"part II. *ORSA Journal on computing 2*, 1 (1990), 4–32.

[37] Goldberg, D. E., and Holland, J. H. Genetic algorithms and machine learning. *Machine learning 3*, 2 (1988), 95–99.

[38] Goldfarb, D., and Idnani, A. A numerically stable dual method for solving strictly convex quadratic programs. *Mathematical programming 27*, 1 (1983), 1–33.

[39] Gomory, R. E. Outline of an algorithm for integer solutions to linear programs and an algorithm for the mixed integer problem. In *50 Years of Integer Programming 1958-2008*. Springer, 2010, pp. 77–103.

[40] Gould, N. I. An algorithm for large-scale quadratic programming. *IMA Journal of Numerical Analysis 11*, 3 (1991), 299–324.

[41] Haimes, Y. Y., Hall, W. A., and Freedman, H. T. *Multiobjective optimization in water resources systems : the surrogate worth trade-off method*, vol. 3. Elsevier, 2011.

[42] Hansen, M. P. Tabu search for multiobjective optimization : MOTS. In *Proceedings of the 13th International Conference on Multiple Criteria Decision Making* (1997), Citeseer, pp. 574–586.

[43] Hao, J.-K., Galinier, P., and Habib, M. Métaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes. *Revue d'intelligence artificielle 13*, 2 (1999), 283–324.

[44] Hiriart-Urruty, J.-B. *Optimisation et analyse convexe*. EDP sciences, 2012.

[45] Holland, J. H. *Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence.* U Michigan Press, 1975.

[46] Konno, H., and Yamazaki, H. Mean-absolute deviation portfolio optimization model and its applications to Tokyo stock market. *Management science 37*, 5 (1991), 519–531.

[47] LAOUAR, A. Méthode adaptée pour la résolution d'un problème de programmation quadratique convexe Ã variables mixtes.

[48] Leyton-Brown, K., Nudelman, E., and Shoham, Y. Empirical hardness models : Methodology and a case study on combinatorial auctions. *Journal of the ACM (JACM) 56*, 4 (2009), 22.

[49] Mahdi, S. Optimisation multiobjectif par un nouveau schéma de coopération méta/exacte. *Mémoire de Master, Université Mentouri de Constantine* (2007).

[50] Mamadou, B. Nouvelle méthode de résolution des poblèmes linéaire Ã variables bornées par décomposition.

[51] Merdjaoui, B. *Optimisation multi-objectif par algorithmes génétiques et approche pareto des paramètres d'usinage sous contraintes des limitations de production.* PhD thesis, 2006.

[52] Miettinen, K. *Nonlinear multiobjective optimization*, vol. 12. Springer Science & Business Media, 2012.

[53] Minoux, M., and Balas, E. Programmation mathematique : theorie et algorithmes. Tome 1.

[54] Multisimplex. Manuel d'utilisation du logiciel MultiSimplex.

[55] Myers, S. C. Finance theory and financial strategy. *Interfaces 14*, 1 (1984), 126–137.

[56] Nelder, J. A., and Mead, R. A simplex method for function minimization. *The computer journal 7*, 4 (1965), 308–313.

[57] Pardalos, P. M., Sandstràm, M., and Zopounidis, C. On the use of optimization models for portfolio selection : A review and some computational results. *Computational Economics 7*, 4 (1994), 227–244.

[58] radjef, S., and bibi, M. Nouvel algorithme de résolution d'un programme quadratique convexe Ã variables mixtes.

[59] Radjef, S., and Hamga, M. Etude comparative entre deux méthodes de résolution d'un programme quadratique convexe.

[60] Reardon, B. J. Fuzzy Logic Vs. Niched Pareto Multiobjective Genetic Algorithm Optimization : Part I : Schaffer's F2 Problem,'. *Los Alamos National Laboratory Unclassified Report, LA-UR-97-3675* (1997).

[61] Reardon, B. J. Fuzzy Logic Vs. Niched Pareto Multiobjective Genetic Algorithm Optimization : Part II : A Simplified Born Mayer Problem. *Los Alamos National Laboratory Unclassified Report, LA-UR-97-3676* (1997).

[62] Rozenn, T.-P. CONVEXITé ET APPLICATIONS.

[63] Ruppli, R. Programmation Linéaire Idées et méthodes.

[64] Saadi, L. Optimisation MultiObjectifs par Programmation Génétique, 2007.

[65] Sakarovitch, M. *Optimisation combinatoire : Programmation discrète*, vol. 2. Editions Hermann, 1984.

[66] Sakawa, M. *Genetic algorithms and fuzzy multiobjective optimization*, vol. 14. Springer Science & Business Media, 2012.

[67] Sakawa, M., and Yano, H. An interactive fuzzy satisficing method for multiobjective nonlinear programming problems with fuzzy parameters. *Fuzzy Sets and Systems 30*, 3 (1989), 221–238.

[68] Schaffer, J. D. Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the 1st international Conference on Genetic Algorithms* (1985), L. Erlbaum Associates Inc., pp. 93–100.

[69] Talbi, E.-g. Métaheuristiques pour l'optimisation combinatoire multi-objectif : Etat de l'art. *Rapport CNET (France Telecom) Octobre* (1999).

[70] T'kindt, V., and Billaut, J.-C. *Multicriteria scheduling : theory, models and algorithms*. Springer Science & Business Media, 2006.

[71] Tsoukià s, A. De la théorie de la décision à l'aide à la décision. *Concepts et Méthodes* (2006).

[72] Tzeng, G.-H., and Huang, J.-J. *Multiple attribute decision making : methods and applications*. CRC press, 2011.

[73] VILFREDO, and PARETO. COURS d'Économie POLITIQUE.

[74] Wilbaut, C. *Heuristiques hybrides pour la résolution de problèmes en variables 0-1 mixtes*. PhD thesis, Université de Valenciennes et du Hainaut-Cambresis, 2006.

[75] Wolfe, P. The simplex method for quadratic programming. *Econometrica : Journal of the Econometric Society* (1959), 382–398.

[76] Wolsey, L. A., and Nemhauser, G. L. *Integer and combinatorial optimization*. John Wiley & Sons, 2014.

[77] Xie, J., He, S., and Zhang, S. Randomized portfolio selection with constraints. *Pacific Journal of Optimization 4* (2008), 89–112.

[78] Young, M. R. A minimax portfolio selection rule with linear programming solution.

[79] Zheng, X., Sun, X., and Li, D. Improving the performance of MIQP solvers for quadratic programs with cardinality and minimum threshold constraints : A semidefinite program approach. *INFORMS Journal on Computing 26*, 4 (2014), 690–703.

Résumé : Dans ce travail, nous avons étudié les problèmes de programmation quadratique et mis le point sur le cas d'existence de plusieurs fonctions devant être optimisé simultanément. Une approche originale de ce problème en utilisant une méthode adaptée qui est une amélioration de la méthode de support de Gabassov et Kirilova [29]. Nous avons proposé une nouvelle méthode adaptée pour la résolution des problèmes quadratique multiobjectif en variables mixtes, nous avons aussi appliqué cette méthode pour la résolution de problèmes d'optimisation des portefeuilles de cardinalité et testé sur des cas réèls (Dax, Footzi, Hangseng, Nasdaq, ...).

Mots Clés : Optimisation quadratique, optimisation multiobjectif, optimisation formes quadratiques, méthode adaptée de support, problèmes des portefeuilles avec contrainte de cardinalité,PMQ.

Abstract : In this work, we studied the quadratic programming problems and put an update on the case of existence of several functions to be optimized simultaneously. An original approach to this problem by using an adapted method that is improved Gabassov support method Kirillova [29] . We proposed a new adapted method for solving quadratic multiobjective problems with mixed variables, we also applied this method for solving optimization problems cardinality portfolios and tested on real cases (Dax, Footzi, Hangseng, Nasdaq , ...).

Keywords : Quadratic optimization, portefolio selection with cardinality constraint, portfolio selection model markowitz.