N° D'ORDRE : 34/2011-M/M.T

République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université des Sciences et de la Technologie Houari Boumediene

Faculté de Mathématiques

MÉMOIRE

PRÉSENTÉ POUR L'OBTENTION DU DIPLÔME DE MAGISTER

EN MATHÉMATIQUES

Spécialité : **RECHERCHE OPÉRATIONNELLE (MATHÉMATIQUES DE GESTION)**

Par : **Madani BEZOUI**

Programmation multiobjectif quadratique en variables mixtes : Application à l'optimisation des portefeuilles

Soutenu publiquement le 13 Juillet 2011, devant le jury composé de :

Président	M^r	Moncef	ABBAS	Professeur	USTHB
Directeur de Mémoire	M^r	Mustapha	MOULAÏ	Professeur	USTHB
Examinateur	M^r	Mohamed El Amine	CHERGUI	M. C. A	USTHB
Invité	M^r	Abdelhak	MEZGHICHE	M. A. A	USTHB

Table des matières

Table des figures

Liste des Algorithmes

Notations

Symbole	signification
$F(P)$	Ensemble des solutions efficaces du problème (P).
LP	Programme linéaire.
ILP	Programme linéaire en nombres entiers.
$MILP$	Programme linéaire en variables mixtes.
PQ	Programme quadratique.
IQP	Programme quadratique en nombres entiers.
$MIQP$	Programme Quadratique en variables mixtes.
$MOMIQP$	Programme Quadratique Multiobjectif en variables Mixtes.
$\mathcal{F}$	Espace réalisable dans l'espace des objectifs.

Remerciements

*P*remièrement, je remercie Dieu le Miséricordieux, pour m'avoir donné la volonté et la force pour accomplir ce modeste travail, elhamdou li llah.

*J*e tiens à exprimer ma profonde gratitude au professeur M. MOULAÏ, mon directeur de mémoire, et mon guide, pour la confiance qu'il m'a faite en acceptant de diriger mes recherches, et pour ses précieux conseils et orientations, ainsi que pour l'intérêt particulier qu'il a accordé à ce travail. Je ne le remercierai jamais assez pour la grande contribution et l'aide qu'il m'a apporté pour l'aboutissement de ce travail.

J'adresse mes remerciements au professeur Abbas, pour l'honneur qu'il m'a fait en acceptant de présider le jury de ce mémoire.

*J*e remercie le docteur Chergui pour avoir accepté de juger ce travail.

*J*e remercie également monsieur Mezghiche pour avoir accepté d'assister à ma soutenance.

*J*e remercie tous mes collègues et amis pour leurs aides et soutiens et tous ceux qui ont contribué de près ou de loin à la réalisation de ce travail, Hassane, Salima, Midou, Zoubir ...

*J*e n'oublierai pas de remercier ma famille et ma fiancée qui m'ont toujours encouragé et soutenu.

Je dédie ce travail :

A mes très chers parents.

A mes frères et Sœurs.

A la mémoire de mes grand parents.

A ma fiancée.

A tous mes amis.

Madani.

Introduction générale

"Le commencement est la moitié de tout".

Pythagore(vers 580-495 av. J.-C.)

Longtemps, le problème d'optimisation de portefeuilles est formulé comme un problème d'optimisation mono-objectif, sans tenir compte de tous les critères (simultanément) qui peuvent influencer nos résultats et ne pas avoir la meilleur solution possible. C'est pour cette raison qu'on propose ici un modèle multiobjectif (bi-objectif) pour trouver une solution (ou des solutions) qui satisfait au mieux le décideur (dans notre cas, on parle d'investisseur), pour l'aider à prendre une bonne décision (choisir le bon portefeuille).

En effet, Markowitz[39] a révolutionné l'optimisation de portefeuilles, en proposant le modèle moyen-variance et une méthode de recherche de la frontière efficace. Puis viendra Roy[47] et propose un algorithme pour résoudre le modèle de Markowitz et à partir de ces deux travaux des milliers de travaux sont réalisés. Mais les algorithmes proposés passent tous par une modélisation ou une transformation (avec une fonction d'agrégation ou autre) vers un modèle mono-objectif ou en utilisant des métaheuristiques qui donnent,

certes, de bonnes résultats, mais qui restent toujours des méthodes approximatives.

Pour réaliser notre travail, on va commencer par quelques rappels de la programmation quadratique et l'optimisation multiobjectif [12], pour ensuite donner quelques méthodes de résolution des problèmes de programmation quadratique et des problèmes multiobjectif. Dans le quatrième chapitre, on va traiter la programmation discrète, ainsi que quelques méthodes de résolution de cette classe de problèmes.

Pour finir, dans la partie pratique, on va définir l'optimisation des portefeuilles, ainsi que quelques méthodes classiques de résolution, pour ensuite proposer une méthode exacte de résolution des problèmes de programmation quad-lin[18], ainsi que son implémentation.

Chapitre 1

Programmation quadratique

"Les mathématiques sont une gymnastique de l'esprit et une préparation à la philosophie".

Isocrate(Athènes 436–338 av. J.-C.)

Introduction

La programmation quadratique est une classe importante de programmation mathématique, de part sa vague utilisation dans différents domaines comme les probabilités (avec la régression[54]), la production efficace[20], la sélection de portefeuille [38] (qu'on traitera ultérieurement)...

1.1 Problèmes quadratiques

Définition 1.1 On dit que $f : \mathbb{R}^n \to \mathbb{R}$ est une *fonction quadratique* s'il existe une matrice $D \in \mathbb{R}^{n\times n}$, un vecteur $c \in \mathbb{R}^n$ et un nombre réel α tel que

$$f(x) = \frac{1}{2}x^T Dx + c^T x + \alpha \tag{1.1}$$

pour tout $x \in \mathbb{R}^n$.

Si

$$D = \begin{pmatrix} d_{11} & \dots & d_{1n} \\ \dots & \dots & \dots \\ d_{n1} & \dots & d_{nn} \end{pmatrix}, c = \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix}, x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix},$$

alors (1.1) peut s'écrire comme suit :

$$f(x) = \frac{1}{2}(\sum_{j=1}^{n}\sum_{i=1}^{n} d_{ij}x_i x_j) + \sum_{i=1}^{n} c_i x_i + \alpha.$$

Comme $x^T Dx = \frac{1}{2}x^T(D + D^T)x$ pour tout $x \in \mathbb{R}^n$, la représentation (1.1) reste valide si on remplace D par la matrice symétrique $\frac{1}{2}(D + D^T)$. Pour cette raison, on suppose que la matrice carrée dans la représentation (1.1) est symétrique. L'espace des $n \times n$-matrices symétriques seront notées par $\mathbb{R}_S^{n\times n}$.

Définition 1.2 [35] Un problème (P) est appelé "problème de programmation quadratique" si sa fonction objectif f est une fonction quadratique et l'ensemble de ses solutions réalisables est un polyèdre convexe.

1.2 Convexité

Définition 1.3 (Ensemble convexe) [4].

Un ensemble C est convexe si pour tout $x_1,\ x_2 \in C$, et tout $\theta \in [0, 1]$

$$\theta x_1 + (1 - \theta)x_2 \in C \tag{1.2}$$

Définition 1.4 (Fonction convexe) [4].

Une fonction $f : D \to \mathbb{R}$ est convexe si son domaine de définition D est un ensemble convexe et si pour tout $x_1,\ x_2 \in D$, et tout $\theta \in [0, 1]$

$$f(\theta x_1 + (1-\theta)x_2) \leq \theta f(x_1) + (1-\theta) f(x_2) \tag{1.3}$$

Si cette dernière inégalité est stricte, pour tout $x_1 \neq x_2$, on parlera alors de **convexité stricte**.

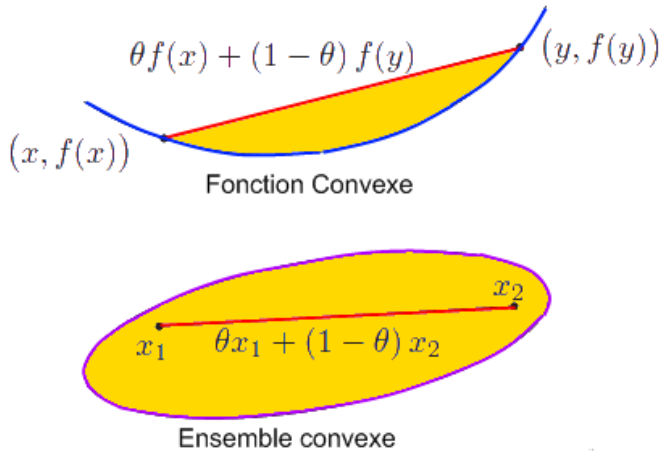

FIG. 1.1 – Représentation géométrique de la notion de convexité

Définition 1.5 (Fonction concave) [4].

Une fonction $f : D \to \mathbb{R}$ est concave si $-f$ est convexe, et strictement concave si $-f$ est strictement convexe.

1.2.1 La convexité dans la programmation quadratique

Définition 1.6 [7] Une matrice $D \in \mathbb{R}^{n\times n}$ est dite *définie positive* (resp., *définie négative*) si $v^T Dv > 0$ (resp., $v^T Dv < 0$) pour tout $v \in \mathbb{R}^n \backslash \{0\}$. Si $v^T Dv \geq 0$ (resp., $v^T Dv \leq 0$) pour tout $v \in \mathbb{R}^n$ alors D est dite *semi-définie positive* (resp., *semi-définie négative*).

Proposition [7] Soit $f(x) = \frac{1}{2}x^T Dx + c^T x + \alpha$ où $D \in \mathbb{R}_S^{n\times n}, c \in \mathbb{R}^n$ et $\alpha \in \mathbb{R}$. Si D est une matrice *semi-définie positive* alors f est une *fonction convexe.*

Preuve [7] Comme $x \mapsto c^T x + \alpha$ est une fonction convexe et la somme des deux fonctions convexe est une fonction convexe, il suffit de montrer que $f_1(x) = x^T Dx$ est une fonction convexe. On sait que D est une matrice définie semi-positive, pour tout $u \in \mathbb{R}^n$ et $v \in \mathbb{R}^n$ on a :

$$0 \leq (u-v)^T D(u-v) = u^T Du - 2v^T Du + v^T Dv.$$

Ceci implique que

$$v^T Dv \leq u^T Du - 2v^T D(u-v). \tag{1.4}$$

Pour un $x \in \mathbb{R}^n$, $y \in \mathbb{R}^n$ et $t \in (0,1)$, on pose $z = tx + (1-t)y$.

D'après 1.4 on a :

$$z^T Dz \leq y^T Dy - 2z^T D(y-z),$$

$$z^T Dz \leq x^T Dx - 2z^T D(x-z).$$

Comme $y - z = t(y-x)$ et $x - z = (1-t)(x-y)$, d'après les deux dernières inégalités on déduit que

$$(1-t)z^T Dz + tz^T Dz \leq (1-t)y^T Dy + tx^T Dx,$$

donc

$$f_1(tx + (1-t)y) = f_1(z) \leq tf_1(x) + (1-t)f(y).$$

Ainsi f_1 est une fonction convexe.

■

Remarque 1 Si D est semi-définie négative, alors la fonction f donnée par 1.1 est concave, i.e.

$$f(tx + (1-t)y) \geq tf(x) + (1-t)f(y)$$

pour tout $x \in \mathbb{R}^n$, $y \in \mathbb{R}^n$ et $t \in (0,1)$.

Remarque 2 Dans le cas où la matrice D n'est ni semi-définie positive, ni semi-définie négative, on dit que $f(x) = \frac{1}{2}x^T Dx + c^T x$, où $c \in \mathbb{R}^n$, est une fonction quadratique indéfinie. Le programme correspondant sera appelé programme quadratique indéfini.

1.3 Gradient d'une forme quadratique

Soit $F(x) = x^T Dx$ une forme quadratique, où D est symétrique (i.e. $x \in \mathbb{R}^n$, $D^T = D = (d_1, .., d_n)$) et d_i est un n-vecteur colonne défini comme suit :

$$d_i = \begin{pmatrix} d_{1i} \\ d_{2i} \\ \vdots \\ d_{ni} \end{pmatrix}.$$

On a alors :

$$Dx = \begin{pmatrix} d_{11}x_1 + d_{12}x_2 + ... + d_{1n}x_n \\ d_{21}x_1 + d_{22}x_2 + ... + d_{2n}x_n \\ \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ d_{n1}x_1 + d_{n2}x_2 + ... + d_{nn}x_n \end{pmatrix} = \begin{pmatrix} d_1^T x \\ d_2^T x \\ \vdots \\ d_n^T x \end{pmatrix}.$$

D'où

$$F(x) = x_1 d_1^T x + x_2 d_2^T x + ... + x_n d_n^T x.$$

Par dérivation de la fonction F par rapport à chaque composante x_j et cela pour $j = \overline{1,n}$, on obtient les équations suivantes :

$$\begin{cases} \frac{\partial F(x)}{\partial x_1} = d_1^T x + x_1 d_{11} + x_2 d_{21} + ... + x_n d_{n1} = 2.d_1^T x, \\ \frac{\partial F(x)}{\partial x_2} = d_2^T x + x_1 d_{12} + x_2 d_{22} + ... + x_n d_{n2} = 2.d_2^T x, \\ \qquad\qquad \vdots \qquad\quad \vdots \qquad\quad \vdots \\ \frac{\partial F(x)}{\partial x_n} = d_n^T x + x_1 d_{1n} + x_2 d_{2n} + ... + x_n d_{nn} = 2.d_n^T x. \end{cases}$$

Finalement, on obtient

$$\nabla F(x) = 2 \begin{pmatrix} d_1^T x \\ d_2^T x \\ \vdots \\ d_n^T x \end{pmatrix} = 2Dx. \tag{1.5}$$

1.4 Théorème de Frank Wolfe

Considérons le programme sous sa forme quadratique suivant :

$$P_1 : \begin{cases} Minimiser & f(x) = \frac{1}{2} x^T D x + c^T x\,; \\ s.t., & x \in \mathbb{R}^n, Ax \geq b, \end{cases}$$

*où*D$\in \mathbb{R}_S^{n\times n}$, $A \in \mathbb{R}^{m\times n}$, $c \in \mathbb{R}^n$ et $b \in \mathbb{R}^m$. Pour l'ensemble des contraintes et la valeur optimale de P_1 on utilisera les abréviations suivantes :

$$\triangle(A, b) = \{x \in \mathbb{R}^n : Ax \geq b\},$$

$$\overline{\theta} = inf\{f(x) : x \in \triangle(A, b)\}.$$

Si $\triangle(A, b) = \emptyset$ alors $\overline{\theta} = +\infty$ par convention. Si $\triangle(A, b) \neq \emptyset$ alors il y a deux situations :

1. $\overline{\theta} \in \mathbb{R}$,
2. $\overline{\theta} = -\infty$.

Si la situation 2. se produit, alors P_1 n'a pas de solutions. Il est naturel de se demander : *Quand est-ce que le problème admet des solutions quand la situation* 1. *se produit.*

***Théorme** 1 (Frank-Wolfe ; [24], p. 108) Si $\overline{\theta} = inf\{f(x) : x \in \triangle(A,b)\}$ est un nombre réel fini alors le problème P_1 admet une solution.*

Pour la démonstration voir [8].

***Théorme** 2 (Eaves [21], page 702) Le problème P_1 admet des solutions si et seulement si les trois conditions suivantes sont satisfaites :*

1. *$\triangle(A,b)$ est non vide.*
2. *Si $v \in \mathbb{R}$ et $Av \geq 0$ alors $v^T Dv \geq 0$;*
3. *Si $v \in \mathbb{R}$ et $x \in \mathbb{R}^n$ sont tels que $Av \geq 0$, $v^T Dv = 0$ et $Ax \geq b$, alors $(Dx+c)^T v \geq 0$.*

Preuve Voir [35].

Corollaire [35] Supposons que D est une matrice semi-définie positive. Alors le problème P_1 admet des solutions si et seulement si $\triangle(A,b)$ est non vide et la condition suivante est vérifiée :

$$(v \in \mathbb{R}^n,\ x \in \mathbb{R}^n,\ Av \geq 0,\ v^T Dv = 0,\ Ax \geq b) \Rightarrow (Dx+c)^T v \geq 0. \tag{1.6}$$

Preuve [35] Notez que la deuxième condition du théorème 2 est satisfaite de par notre supposition, $v^T Dv \geq 0$ pour tout $v \in \mathbb{R}^n$. Ainsi, la conclusion est une implication du théorème 2.

■

Corollaire [35] Si D est une matrice définie positive, alors le problème (P_1) admet des solutions si et seulement si $\triangle(A,b)$ est non vide.

Corollaire [35] Si D est une matrice semi-définie positive, alors le problème (P_1) admet des solutions si et seulement si $\triangle(A,b)$ est non vide et compact.

1.5 Conditions nécessaires et suffisantes pour l'optimalité des programmes quadratiques

1.5.1 Conditions d'optimalité de premier ordre

Théorme *3 ([35]) Soit le vecteur $\overline{x}$ solution du programme quadratique*

$$\min\{f(x) = \frac{1}{2}x^T Dx + c^T x : \ x \in \triangle\}, \tag{1.7}$$

où, $D \in \mathbb{R}_S^{n\times n}$, $c \in \mathbb{R}^n$, et $\triangle \subset \mathbb{R}^n$ est un polyèdre convexe.

☞ *Si $\overline{x}$ est une solution locale de ce problème, alors*

$$\langle D\overline{x} + c, x - \overline{x}\rangle \geq 0 \ \textit{pour tout} \ x \in \triangle. \tag{1.8}$$

☞ *Si*

$$\langle D\overline{x} + c, x - \overline{x}\rangle > 0 \ \textit{pour tout} \ x \in \triangle\backslash\{\overline{x}\}, \tag{1.9}$$

alors $\overline{x}$ est une solution locale de 1.7 et, de plus, il existe $\varepsilon > 0$ et $\rho > 0$ tel que :

$$f(x) - f(\overline{x}) \geq \rho\|x - \overline{x}\| \ \textit{pour tout} \ x \in \triangle \cap B(\overline{x}, \varepsilon) \tag{1.10}$$

Théorme *4 (Farkas, [46], p. 200) Soient $a_0, a_1, ..., a_k$ des vecteurs de $\mathbb{R}^n$. L'inégalité $\langle a_0, x\rangle \leq 0$ est une conséquence du système :*

$$\langle a_i, x\rangle \leq 0, i = 1, 2, ..., k,$$

si et seulement s'il existe des nombres non négatifs réels $\lambda_1, ..., \lambda_k$ tel que :

$$\sum_{i=1}^{k} \lambda_i a_i = a_0.$$

Théorme *5 ([14], p. 118) Si $\overline{x} \in \mathbb{R}^n$ est une solution locale du problème (P_1) alors il existe $\lambda = (\lambda_1, \lambda_2..., \lambda_2) \in \mathbb{R}^m$ tel que :*

$$P_\lambda = \begin{cases} D\overline{x} - A^T\lambda + c = 0, \\ A\overline{x} - b \geq 0, \ \lambda \geq 0, \\ \lambda^T(A\overline{x} - b) = 0. \end{cases}$$

Définition 1.7 Si $(\overline{x}, \lambda) \in \mathbb{R}^n \times \mathbb{R}^m$ vérifie le problème P_λ, alors on dit que $(\overline{x}, \lambda)$ est le couple de Karush Kuhn Tucker du programme quadratique standard. Le point $\overline{x}$ est appelé un "point KKT", et les nombres réels $\lambda_1, \lambda_2, ..., \lambda_m$ sont appelés les multiplicateurs de Lagrange correspondant à $\overline{x}$.

1.5.2 Conditions d'optimalité de second ordre

Le plus important résultat dans ce domaine est publié par Majthay en 1971. Ce résultat est cité dans le théorème suivant.

Théorme *6 ([36, 13]) La condition nécessaire et suffisante pour qu'un point $\overline{x} \in \mathbb{R}^n$ soit une solution locale du problème*

$$\min\{\frac{1}{2}x^T Dx + c^T x : x \in \mathbb{R}^n, Ax \geq b, Cx = d\} \tag{1.11}$$

est qu'il existe un couple de vecteurs

$$(\overline{\lambda}, \overline{\mu}) = (\overline{\lambda}_1, \overline{\lambda}_2, \cdots, \overline{\lambda}_m, \overline{\mu}_1, \cdots, \overline{\mu}_s) \in \mathbb{R}^m \times \mathbb{R}^s$$

tel que

1. *Le système :*

$$P_{\lambda,\mu} = \begin{cases} D\overline{x} - A^T\overline{\lambda} + C^T\overline{\mu} + c = 0, \\ A\overline{x} - b \geq 0, \ C\overline{x} = d, \ \overline{\lambda} \geq 0, \\ \overline{\lambda}^T(A\overline{x} - b) = 0. \end{cases}$$

est satisfait, et

2. *si* $\nu \in \mathbb{R}^n \setminus \{0\}$ *est tel que* $A_{I_1}\nu = 0,\ A_{I_2}\nu \geq 0,\ C\nu = 0,$ *où :*

$$I_1 = \{i : A_i\overline{x} = b_i,\ \overline{\lambda}_i > 0\}, I_2 = \{i : A_i\overline{x} = b_i,\ \overline{\lambda}_i = 0\},$$

alors $\nu^T D\nu \geq 0$.

Considérons le problème 1.11 et l'ensemble :

$$\triangle = \{x \in \mathbb{R} : Ax \geq b, Cx = d\}.$$

Définition 1.8 ([37], p201) Un point $\overline{x} \in \triangle$ est appelé l'unique solution locale du problème $\min\{f(x) :\ x \in \triangle\}$, où $f : \mathbb{R}^n \to \mathbb{R}$ est une fonction réelle et $\triangle \subset \mathbb{R}^n$ est un sous ensemble donné, s'il existe $\varepsilon > 0$ tel que :

$$f(x) > f(\overline{x}),\quad \forall x \in (\triangle \cap B(\overline{x}, \varepsilon))\backslash\{\overline{x}\}.$$

Théorme *7 ([37, 13]) La condition nécessaire et suffisante pour qu'un point* $x \in \mathbb{R}^n$ *soit l'unique solution du problème* (1.11) *est qu'il existe un couple de vecteurs :*

$$(\overline{\lambda}, \overline{\mu}) = (\overline{\lambda}_1, \ldots, \overline{\lambda}_m, \overline{\mu}_1, \ldots, \overline{\mu}_s) \in \mathbb{R}^m \times \mathbb{R}^s$$

tel que :

1. *Le système* $P_{\lambda,\mu}$ *est satisfait, et*
2. *Si* $v \in \mathbb{R}^n \setminus 0$ *est tel que* $A_{I_1}v = 0,\ A_{I_2} \geq 0,\ Cv = 0,$ *où*

   $I_1 = \{i : A_i\overline{x} = b_i,\ \overline{\lambda}_i > 0\},\ I_1 = \{i : A_i\overline{x} = b_i,\ \overline{\lambda}_i = 0\},$ *alors* $v^T Dv > 0$.

1.6 Caractérisation des ensembles de solutions des programmes quadratiques

Considérons le problème suivant :

$$(P)\quad \min\{f(x) = \frac{1}{2}x^T Dx : x \in \mathbb{R}^n, Ax \geq b, Cx = d\},$$

et son problème homogène correspondant :

$$(P_0) \quad \min\{\frac{1}{2}v^T Dv : v \in \mathbb{R}^n, Av \geq 0, Cv = 0\},$$

Où $D \in \mathbb{R}_S^{n\times n}, A \in \mathbb{R}^{m\times n},\ C \in \mathbb{R}^{s\times n},\ c \in \mathbb{R}^n, b \in \mathbb{R}^m, d \in \mathbb{R}^s$.

Soit $\triangle = \{x \in \mathbb{R}^n, Ax \geq b, Cx = d\}$.

Notons par $Sol(P)$, $loc(P)$, $S(P)$, respectivement, l'ensemble de solutions, l'ensemble des solutions locales et l'ensemble de points de KKT du problème (P). Notre but est d'étudier les propriétés de ces ensembles.

Définition 1.9 La demi droite $w = \{\overline{x} + t\overline{v} : t \geq 0\}$, où $\overline{v} \in \mathbb{R}^n \setminus \{0\}$, qui est un sous ensemble de $Sol(P)$ (resp., loc(P), S(P)), est dite rayon de solutions (resp., rayon de solutions locales, rayon de points KKT) de (P).

***Théorme** 8 ([21]) L'ensemble $Sol(P)$ n'est pas borné si et seulement si (P) a un rayon de solutions. Une condition nécessaire et suffisante pour que $Sol(P)$ soit non borné est qu'il existe $\overline{x} \in Sol(P)$ et $\overline{v} \in Sol(P_0) \setminus \{0\}$ tel que :*

$$(D\overline{x} + c)^T\overline{v} = 0 \tag{1.12}$$

Les résultats suivants sont les conséquences directes de ce théorème.

Corollaire [21] Si l'ensemble de solutions $Sol(P_0)$ est vide ou il est réduit au singleton $\{0\}$ alors, pour tout $(c, b, d) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^s$, l'ensemble de solutions $Sol(P)$ est borné. Dans le cas où $Sol(P_0)$ contient un élément non nul, si

$$(D\overline{x} + c)^T\overline{v} > 0, \quad \forall \overline{x} \in Sol(P), \quad \forall \overline{v} \in Sol(P_0) \setminus \{0\},$$

alors $Sol(P)$ est borné.

***Théorme** 9 ([21]) L'ensemble $loc(P)$ n'est pas borné si et seulement si (P) a un rayon de solutions locals.*

Théorme *10 ([21]) L'ensemble $S(P)$ n'est pas borné si et seulement si (P) a un rayon de points KKT.*

1.7 Dualité des problèmes quadratiques

1.7.1 Dualité des programmes quadratiques

Dans plusieurs cas, passer au problème dual s'avère plus facile pour traiter notre problème, ainsi les variables duales ont des interpretation très utiles, exemples : courant/tension dans les réseaux électriques, consommation/prix dans les modèles économiques, tensions/déplacements dans la statique.... etc.

Définition 1.10 (Problème primal) Un problème primal est un problème de minimisation qui consiste à trouver un vecteur x^*, s'il existe , vérifiant :

$$(P)\begin{cases} F(x^*) = min_{x\in X} F(x), \\ x^* \in X = \{x \in \mathbb{R}^n / g(x) \leq 0\}, \end{cases},$$

où F est une fonction réelle, de classe C^1 sur $\mathbb{R}^n$, et une fonction vectorielle de classe C^1, $g : \mathbb{R}^n \to \mathbb{R}^m$.

Définition 1.11 (Problème dual), On définit le problème dual de (P) comme un problème de maximisation qui consiste à trouver deux vecteurs $\lambda^* \in \mathbb{R}^n$ et $y^* \in \mathbb{R}^m$, s'ils existent, tels que :

$$(D)\begin{cases} \zeta(\lambda^*, y^*) = max_{(\lambda,x)\in} \zeta(\lambda, y), \\ (\lambda^*, y^*) \in Y = \{(\lambda, y) \in \mathbb{R}^n \times \mathbb{R}^m / \nabla_\lambda \zeta(\lambda, y) = 0, y \geq 0\}, \end{cases}$$

où $\zeta(\lambda, y) = F(\lambda) + y'g(\lambda)$ est la fonction de Lagrange. Ici l'ensemble des contraintes Y est définit par des égalités et des inégalités linéaires.

1.8 Méthodes de résolution des problèmes quadratiques

Il existe plusieurs méthodes pour résoudre les problèmes de programmation quadratique, parmi elles, citons :

1.8.1 Méthode d'activation de contraintes (ASM)

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

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'égalité, 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.

La méthode ASM pour le cas linéaire est facile à appliquer par rapport au cas quadratique puisque cela dépend du nombre de contraintes actives à l'optimum. D'après la théorie de la programmation linéaire, on sait à l' avance que la solution optimale correspond à un sommet du polyèdre du domaine admissible, contrairement à un problème quadratique où la solution peut être un sommet, une face ou un point de l'intérieur du polyèdre.

1.8.2 Méthodes de points intérieurs

Les méthodes de points intérieurs forment une classe d'algorithmes qui permettent de résoudre des problèmes d'optimisation convexe (linéaires ou non).

Les méthodes de points intérieurs se répartissent en plusieurs familles :

- La méthode "affine scaling" (optimisation sur des ellipsoïdes)
- La méthode de réduction du potentiel (notion de barrière, chemin central, relaxation).

La méthode du chemin central est le représentant le plus important de cette famille. Toutes ces méthodes trouvent leur origine (historique) dans les travaux d'un élève du mathématicien soviétique Andreï Kolmogorov : Dikin. C'est, en effet, à Dikin que l'on doit la méthode des ellipsoïdes sur laquelle se fondent plus ou moins toutes les méthodes de points intérieurs. Arkadi Nemirovski, David B. Yudin, Shor développent, en 1972, la méthode des ellipsoïdes pour des problèmes d'optimisation (non linéaires) convexe. En 1979, Leonid Khachiyan démontre que la méthode des ellipsoïdes, appliquée à la PL a une complexité, dans le pire des cas, polynomiale. Cependant, l'algorithme qu'il propose est beaucoup plus lent que le simplexe. Cette approche, qui s'étend de façon très élégante à des problèmes non linéaires convexes peut être considéré comme l'idée germinale des méthodes de points intérieurs développées par la suite. Les algorithmes développés par la suite ont été inspirés par l'Algorithme de Karmarkar[30], développé en 1984 par Narendra Karmarkar pour l'optimisation linéaire. L'idée de base de la méthode est d'utiliser des fonctions barrières pour décrire l'ensemble des solutions qui est convexe par définition du problème. A l'opposé de l'algorithme du simplexe, cette méthode atteint l'optimum du problème en passant par l'intérieur de l'ensemble des solutions réalisables. Toutefois, on trouve quelques unes des plus importantes contributions dans les travaux de Nesterov et Nemirovski [42], qui stipulent que la méthode peut être appliquée sur d'autres classes de problèmes comme les problèmes quadratiques convexes.

1.8.3 Méthode du Simplexe quadratique de Wolfe (1959)

Durant ces dernières années beaucoup d'algorithmes ont été conçus pour résoudre les problèmes quadratiques convexe, la méthode classique pour la résolution de ce type de problème est celle de Wolfe [24, 52, 53] qui est une modification légère de la méthode du simplexe [16],[17],[25].

Son principe consiste à résoudre le système d'optimalité de "Karush-Kuhn-Tucker"(KKT), théorème [25], 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.

Considérons le problème de programmation quadratique sous la forme standard suivante :

$$\begin{cases} F(x) = \min \frac{1}{2}x'Dx + c'x, \\ Ax = b, \\ x \geq 0, \end{cases} \tag{1.13}$$

où $D' = D \geq 0, c \in \mathbb{R}^n, b \in \mathbb{R}^m, rangA = m < n.$

La fonction de Lagrange associée au problème d'optimisation précédent est donnée comme suit :

$$\zeta(x, \lambda, \delta) = \frac{1}{2}x'Dx + c'x + \lambda_1'(Ax - b) - \delta'x.$$

Le m-vecteur λ représente le multiplicateur de Lagrange associé à la contrainte d'égalité, et le n-vecteur δ représente le multiplicateur de KKT associé à la contrainte d'inégalité du problème (1.13).

Comme le problème (1.13) est convexe alors les contraintes d'optimalité de KKT sont à la fois nécessaires et suffisantes, formulées de cette manière :

x^* est une solution optimale du problème (1.13) si et seulement si il existe deux vecteurs $\lambda^* \in \mathbb{R}^m$ et $\delta^* \geq 0$, appartenant à $\mathbb{R}^n$ vérifiant les relations suivantes :

$$\begin{cases} \frac{\partial \zeta}{\partial x^*} = \text{Dx+c+A'}\lambda^* - \delta^* = 0, & L_1 \\ \frac{\partial \zeta}{\partial \lambda^*} = \text{Ax}^* - b = 0, x^* \geq 0, & L_2 \\ \delta^{*\prime} x^* = 0, & L_3 \\ \lambda^* \in \mathbb{R}^m, \delta^* \geq 0. & L_4 \end{cases} \tag{1.14}$$

Où,

- ✍ la relation (L_1) est appelée condition de stationnarité,
- ✍ la relation (L_2) est la faisabilité de la solution,
- ✍ la relation (L_3) est la condition de complémentarité et la dernière,
- ✍ la relation (L_4) est la non négativité des multiplicateurs de KKT.

À cause de (L_3) ce système n'est pas linéaire, on obtient un système linéaire de $(n+m)$ équations à $(2n+m)$ inconnues, cela en considérant les équations $(L_1) et (L_2)$, avec en plus n inconnues, non linéaires.

$$\delta_j^* x_j^* = 0, \forall j = \overline{1,n}. \tag{1.15}$$

Pour trouver une solution telle que (1.15) soit vérifiée, il suffit d'obtenir une solution réalisable basique du système linéaire $(L_1 - L_2)$, tout en s'assurant que x_j^* soit basique et δ_j^* non basique, ou vice-versa.

Pour appliquer la méthode du simplexe, il faut alors écrire le système $\{(L_1), (L_2)\}$ sous forme standard, ce qui veut dire que le second membre doit être positif ou nul, ainsi que le vecteur λ^* doit être réécrit sous la forme suivante :

$$\lambda_i^* = \alpha_i^* - \alpha_{m+i}^*, \alpha_{m+i}^* \geq 0, \alpha_i^* \geq 0, i = \overline{1,m}.$$

Algorithme de la méthode[23]

Algorithme 1 Algorithme de Wolfe
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 ;
3. Initialiser les vecteurs 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 $u' = c'_B A_B^{-1}$;
   - ➴ Calculer le vecteur des estimations $E'_N = u' A_N - c'_N$;
   - ➴ **Si** $E_N \geq 0$ alors la solution actuelle est optimale ;
   - ➴ **Sinon** Aller à l'étape 5 ;
   - ➴ **FinSi** ;
5. Déterminer la variable qui entre en 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.

fin

1.8.4 Méthode du gradient réduit

Comme pour l'algorithme du simplexe, on utilise une décomposition du vecteur x en variables de base x_B et variables hors base x_N :

$$\begin{aligned} \min_{x_B, x_N} \quad & f(x_B, x_N) \\ & Bx_B + Nx_N = b \\ & x_B, x_N \geq 0, \end{aligned}$$

tel que B : la matrice de variables de base, N : la matrice de variables hors base. En posant $x_B = B^{-1} - B^{-1}Nx_N$ on obtient :

$$\begin{aligned} \min_{x_N \geq 0} g(x_N) &= f(B^{-1}b - B^{-1}Nx_N, x_N) \\ s.\acute{a} &= x_B = B^{-1}b - B^{-1}Nx_N \geq 0. \end{aligned}$$

Le gradient réduit $r(x)$ correspond au vecteur des coûts réduits en programmation linéaire :

$$r(x) = \nabla f(x) - \nabla_B f(x) B^{-1} A.$$

On a : $r_N(x) = \nabla g(x) = \nabla_N f(x) - \nabla_B f(x) B^{-1} N$ et $r_B(x) = 0$. La direction de descente est :

$$\begin{aligned} j \in J_N : \quad d_j(x) = -r_j(x) \quad & sir_j(x) < 0, \\ d_j(x) = -r(x) \quad & six_j > 0, \\ d_j(x) = 0 \quad & sinon, \end{aligned}$$

$$j \in J_B : \quad d_j = -B^{-1} N d_N,$$

tel que J_B, J_N sont respectivement les indices des variables de base et hors base.
Si $r(x) = 0$, la solution x satisfait les conditions de K.K.T. Puis on effectue la minimisation

$$\min_{\alpha \geq 0} f(x + \alpha d) \rightarrow x^+,$$

en s'assurant que le vecteur x demeure non négatif.

Comme dans l'algorithme du simplexe, toute variable de base qui s'annule quitte la base pour être remplacée par une variable hors base. Le choix de la variable hors base n'est pas toujours unique ; en effet, même si la solution de base courante n'est pas dégénérée, il se peut qu'il y ait plusieurs variables hors base positives (ces variables sont appelées superbasiques). Il se peut aussi que le minimum de f dans la direction d soit atteint sans qu'aucune variable de base ne s'annule. On répète alors le processus sans changer de base.

Conclusion

Comme on l'a vu dans ce chapitre, la programmation quadratique est très utilisée et on ne cesse de découvrir beaucoup de champs de ses applications. C'est une classe importante de la programmation non linéaire, en effet, avec l'interpolation quadratique, toutes fonction non linéaire pourrait être transformer en une fonction quadratique (en connaissant préalablement les valeurs de la fonction en deux points), ce qui augmente encore plus le champs d'application de la programmation quadratique.

Chapitre 2

L'optimisation multiobjectif

"Le meilleur moyen de se préparer à atteindre un objectif, c'est de s'imaginer qu'on l'a déjà atteint".

Dominique Glocheux (écrivain et conférencier français)

Introduction

L'optimisation est la tâche de trouver une ou plusieurs solutions qui minimisent (ou maximisent) un ou plusieurs objectifs spécifiés et lesquelles satisfont toutes les contraintes (si il y en a). Un problème d'optimisation mono-objectif implique une seule fonction objectif. En revanche, l'optimisation multiobjectif considère plusieurs objectifs souvent incompatibles à optimiser simultanément. Dans un tel cas, il n'y a habituellement pas qu'une seule solution optimale, mais un ensemble d'alternatives avec différentes valeurs de fonc-

tions coût, appelées solutions Pareto optimales, ou solutions non-dominées. En dépit de l'existence de plusieurs solutions Pareto optimales, en pratique, une seule de ces solutions est choisie. Donc, dans le processus de résolution des problèmes multiobjectif, il y a au moins deux étapes importantes : une première qui consiste à trouver les solutions Pareto optimales (implique une procédure informatisée) et une deuxième étape de la prise de décision pour choisir la meilleure solution parmi celles trouvées dans la première étape. Cette dernière nécessite l'information de préférence du décideur.

Dans ce travail on va traiter la première étape, c'est à dire la recherche de solutions non dominées. Ce chapitre est consacré au rappel de quelques notions de l'optimisation multiobjectif ainsi que les approches de résolution.

2.1 Vocabulaire et définitions

2.1.1 Problème d'optimisation multiobjectif

Définition 2.1 (Optimisation multiobjectif) La phrase "Optimisation multiobjectif", est synonyme avec "Optimisation multicritères" ou "Optimisation multi performances", dans [10] on a défini l'optimisation multiobjectif comme un problème de recherche d'un vecteur de variables de décisions qui satisfait les contraintes et optimise un vecteur ayant comme éléments les fonctions objectifs.Ces fonctions sont souvent en conflit, d'où le terme "Optimise" veut dire trouver une solution qui puisse donner les valeurs de toutes les fonctions objectifs **acceptables au décideur**[43].

Formulation

La formulation générale d'un problème d'optimisation multiobjectif est la suivante :

$$\begin{aligned} &\text{"optimiser"} \quad Z(x) = (z_1(x), z_2(x), ..., z_p(x)) \\ &\text{t.q} \qquad\qquad x \in S \end{aligned} \tag{2.1}$$

Où $p \geq 2$ représente le nombre d'objectifs à optimiser, x représente un vecteur de variables de décision, $S = \{x \in R^n / g_j(x) < 0, x \geq 0\}$ est l'ensemble de solutions réalisables associé à des contraintes d'égalité, d'inégalité et des bornes explicites (espace des décisions) et $Z(x)$ est le vecteur des objectifs à optimiser. z_k et g_j, des fonctions à valeurs réelles du vecteur de décision. $(k = 1, 2, ..., p.\ j = 1, 2, ..., m)$

Dans le cadre de l'optimisation multiobjectif, le plus souvent le décideur raisonne plutôt en termes d'évaluation d'une solution sur chaque critère. L'ensemble $Y = Z(S)$ représente les points réalisables dans l'espace des critères, et $Z = (z_1, z_2, ..., z_p)$ avec $Y_k = z_k(x)$ représente un point de l'espace des critères.

2.1.2 Solutions non dominées et solutions Pareto optimales

La plupart des algorithmes d'optimisation multiobjectif utilisent le concept de dominance dans leur recherche. Ici, nous définissons le concept de dominance et les termes apparentés et plusieurs techniques pour identifier des solutions non dominées dans une population finie de solutions.

Solutions spéciales

Nous définissons en premier quelques solutions spéciales qui sont souvent utilisées dans les algorithmes de l'optimisation multiobjectif.

Vecteur Objectif idéal

Le vecteur idéal Z^* du problème est le vecteur de l'espace des critères dont chaque composante Z_k est la solution du problème d'optimisation de la fonction Z_k sous les contraintes du problème.

Définition 2.2 Dans le problème de maximisation, le vecteur idéal Z^* est le vecteur qui optimise chacune des fonctions objectifs Z^*, $Z_k^* = \max_{x \in S}(Z_k(x))$ $k = 1, ..., p$.

Vecteur Objectif Nadir A la différence du vecteur idéal qui représente les bornes supérieures de chaque objectif dans l'espace faisable, le *vecteur Nadir* Z^{nad} correspond à leurs bornes supérieures sur la surface de Pareto et non pas dans tout l'espace faisable. Pour certains problèmes, la méthode de la table des payoff peut être utilisée, le vecteur nadir peut correspondre à une ou à aucune des solutions existantes, en fonction du problème (notamment, de la convexité du domaine de recherche). Pour normaliser chaque objectif, le vecteur idéal et le vecteur nadir sont en particulier utilisés de la façon suivante :

$$Z_k^{norm} = \frac{Z_k - Z_k^*}{Z_k^{nad} - Z_k^*}$$

Définition 2.3 Soit $\hat{x}^j$ une solution qui optimise le critère Z_j. La matrice $(p \times p)$ formée des éléments de $z_{kj} = Z_k(\hat{x}^j)$ est dite **matrice des gains.**

$$\begin{pmatrix} \bar{z}_1 & \cdots & z_{1j} & \cdots & z_{1p} \\ \vdots & & \vdots & & \vdots \\ z_{k1} & \cdots & \bar{z}_{kj} & \cdots & z_{kp} \\ \vdots & & \vdots & & \vdots \\ z_{p1} & \cdots & z_{pj} & \cdots & \bar{z}_p \end{pmatrix}$$

La diagonale de cette matrice représente les coordonnées du point idéal.

Le vecteur nadir, noté η, peut être défini dans la matrice des gains $\eta \in^r \eta_k = \min_{j=1,...,p}(Z_{kj})$ $k = 1, ..., p$.

Définition 2.4 [Concept de dominance] Le vecteur x_1 est dit dominé par un autre vecteur x_1, si :

- x_1 est au moins aussi bon que x_2 dans tous les objectifs, et,
- x_1 est strictement meilleur que x_2 dans au moins un objectif.

Mathématiquement, cela est expliqué comme suit :

1. $Z_k(x^{(i)}) \leq Z_k(x^{(j)})$, $\forall k \in 1, ..., p$
2. $\exists k \in 1, ..., p$ tel que $Z_k(x^{(i)}) < Z_k(x^{(j)})$.

Si la solution $x^{(i)}$ domine la solution $x^{(j)}$, nous allons écrire $x^{(i)} \prec x^{(j)}$

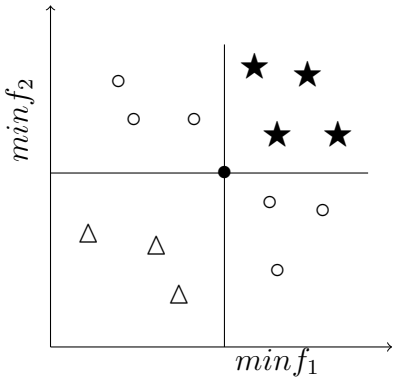

FIG. 2.1 – Le point noir est dominé par chacun des triangles, domine chacun des étoiles et équivalents aux anneaux aux sens de la dominance

Notons que pour toute paire de solution $x^{(1)}$ et $x^{(2)}$, une et seulement une des affirmations suivantes est vraie :

- $x^{(1)}$ domine $x^{(2)}$;
- $x^{(1)}$ est dominée par $x^{(2)}$;
- $x^{(1)}$ et $x^{(2)}$ sont équivalentes au sens de la dominance.

Par la suite, les solutions équivalentes au sens de la dominance seront parfois évoquées comme solutions équivalentes au sens de Pareto ou, encore, comme solutions Pareto équivalentes.

Définition 2.5

1- Soient $Z, Z^{'} \in \mathbb{R}^p$, on dit que Z **domine fortement** $Z^{'}$ si et seulement si $Z_k \succ Z_k^{'}$ $\forall k = 1, ..., P$. (si Z domine fortement $Z^{'}$, alors Z est meilleur que $Z^{'}$ sur tous les critères).

2- Un vecteur Z est dit **faiblement non dominé** s'il n'existe pas de $Z^{'} \in \mathbb{R}^p$ tel que $Z^{'}$ domine fortement Z.

2.1.3 Propriétés de la relation de dominance

La relation binaire de dominance $\prec$, tel qu'elle est définie ci-dessus,

- N'est pas réflexive, car une solution ne se domine pas elle-même ;
- N'est pas symétrique, car on n'a jamais $x^{(1)} \prec x^{(j2)}$ et $x^{(2)} \prec x^{(1)}$;
- N'est pas antisymétrique, du fait de l'existence de solutions Pareto-équivalentes ;
- Est transitive, car $x^{(1)} \prec x^{(k)}$ et $x^{(k)} \prec x^{(2)}$ implique $x^{(1)} \prec x^{(2)}$.

Notons en particulier que $x^{(1)} \not\prec x^{(2)}$ n'implique pas $x^{(2)} \prec x^{(1)}$.

Optimalité de Pareto

Soit P un ensemble de solutions-candidats d'un problème d'optimisation multiobjectif. L'ensemble $P^{'} \subseteq P$, composé de tous les éléments de P qui ne sont dominés par aucun élément de P est dit *sous ensemble non dominé de l'ensemble de solutions* P.

Un sous ensemble de solutions sont Pareto-équivalentes entre elles même s'il existe d'autres solutions qui dominent des solutions de l'ensemble en question.

De façon analogue aux solutions optimales globales et locales dans le contexte de l'optimisation mono-objectif, les notions d'optimum local et d'optimum global au sens de Pareto peuvent être introduites :

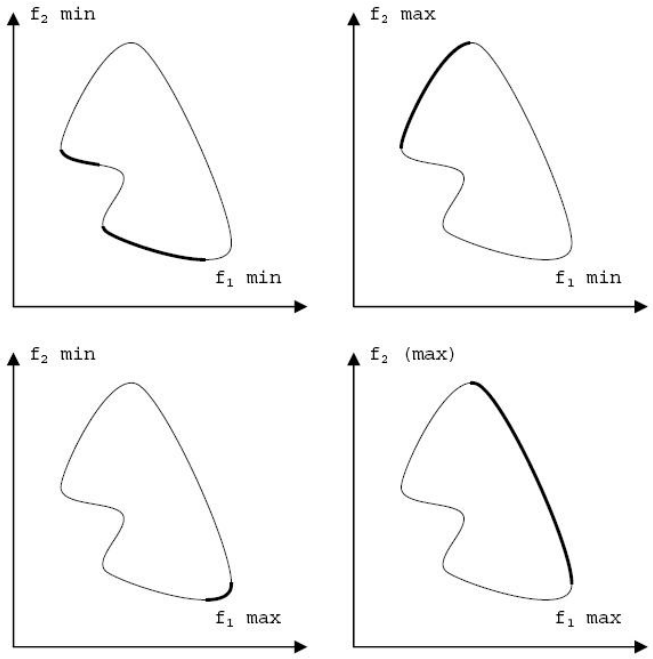

FIG. 2.2 – Les solutions Pareto-Optimales sont marquées avec les courbes continues pour quatre combinaisons de deux types d'objectifs.

- Cet ensemble est constitué des *solutions non dominées du problème d'optimisation* appelées aussi *solution de Pareto* ou *solutions Pareto-optimales* de ce problème.
- L'image de l'ensemble de Pareto dans l'espace des critères est appelée la *surface de Pareto* (ou le *front de Pareto* dans le cas de problème bi-objectif), ou également la *surface des compromis optimaux*.
- *L'ensemble de Pareto* ***global*** du problème d'optimisation multiobjectif, est l'ensemble de points tels qu'aucun autre point de l'espace faisable S ne les domine. Souvent, l'ensemble de Pareto global est évoqué simplement comme *l'ensemble de Pareto* ou

encore *l'ensemble des compromis optimaux*.

– De manière similaire, un sous-ensemble P'_{loc} de l'espace de décision est appelé *ensemble de Pareto* ***local*** s'il existe $\epsilon > 0$ tel que pour tous élément x de P'_{loc} il n'existe pas de solutions y dans S (où ϵ est un petit nombre positif) dominant un élément de P'_{loc} et vérifiant $\|y - x\|_\infty \leq \epsilon$.

2.2 Classification des approches multicritères

2.2.1 Approches a priori

(décideur → recherche) Les solutions les plus intuitives pour résoudre des problèmes multiobjectif consistent souvent à combiner les différentes fonctions objectifs en une fonction d'utilité suivant les préférences du décideur. Dans ce cas le décideur est supposé connaître a priori le poids de chaque objectif et les mettre dans une fonction unique. Cela revient à résoudre un problème mono-objectif. Cependant dans la plupart des cas, le décideur ne peut pas exprimer clairement sa fonction d'utilité, soit par manque d'expérience ou d'information, soit par ce que les différent objectifs sont non commensurable[1].

2.2.2 Approches a posteriori

(recherche→ décideur) Le décideur prend sa décision d'après un ensemble de solutions calculées par un solveur. Dans ce cas la qualité de la décision dépend du choix de la méthode de résolution. Car celle-ci va devoir donner un ensemble plus représentatif de l'espace des objectifs efficaces.

[1]Se dit de deux grandeurs qui ont une mesure commune.

2.2.3 Approches progressives ou interactives

(décideur⇄recherche) Dans ces méthodes, les processus de décision et d'optimisation sont alternés. Par moment, le décideur intervient de manière à modifier certaines variables ou contraintes afin de diriger le processus d'optimisation. Le décideur modifie ainsi interactivement le compromis entre ses préférences et les résultats. Ces méthodes exigent une connaissance approfondie, de la part du décideur, des outils utilisés. [51] présente plusieurs méthodes progressives utilisées en recherche opérationnelle. Il trouve que la relation de dominance est trop pauvre pour être utile et les fonctions d'utilité multiattribut[2] trop riches pour être fiable. Il tente d'enrichir la relation de dominance par des éléments peu discutables : des préférences solidement établies.

2.3 Fonctions scalarisantes

L'ensemble des méthodes agrégés[3] repose sur l'axiome suivant : tout décideur essaye inconsciemment de maximiser une fonction d'utilité U

$$U = U(z_1, z_2, ..., z_k)$$

Où $U : \Re^p \times \Lambda \rightarrow \Re$, $\Lambda = \{\lambda \in \Re^p : \sum_{k=1}^{p} \lambda_k = 1, \lambda_k > 0\}$ qui agrège les valeurs des critères pour chaque solution ($\Lambda \subset \Re^p$ est l'ensemble des paramètres choisis).

Il est possible de définir une fonction croissante dite fonction scalarisante, qui agrège les valeurs des critères pour chaque solution : $U(Z, \lambda) : \Re^p \times \Lambda \rightarrow \Re$

Les fonctions d'utilités les plus employées sont :

– somme pondérée des objectifs (particulièrement utilisée dans le cas linéaire) :

$$U_1(Z, \lambda) = \sum_{k=1}^{p} \lambda_k Z_k,\ \lambda \in \Lambda,$$

[2]Terme utilisé pour identifier les méthodes d'agrégation.

[3]Qu'on traitera en detail dans le chapitre suivant

ou bien,

$$U_2(Z,\lambda) = \sum_{k=1}^{p} \lambda_k \mid Z_k - \bar{Z}_k \mid,\ \lambda \in \Lambda.$$

Où $\bar{Z}_k$ est la k^{eme} composante du point idéal

- Norme L_p pondérée :

$$U_3(Z,\lambda) = \left[\sum_{k=1}^{p} \lambda_k \mid Z_k - \bar{Z}_k \mid^p \right]^{\frac{1}{p}},\ \lambda \in \Lambda,\ p \in_+^*.$$

- Norme L_∞ pondérée de Tchebychev :

$$U_4(Z,\lambda) = \max_{1\leq k\leq p} \left\{ \lambda_k \mid Z_k - \bar{Z}_k \mid \right\},\ \lambda \in \Lambda.$$

- Norme composée (Tchebychev pondérée augmentée) :

$$U_4(Z,\lambda) = \max_{1\leq k\leq p} \{\lambda_k \mid Z_k - \bar{Z}_k \mid\} + \rho \sum_{k=1}^{p} \lambda_k \mid Z_k - \bar{Z}_k \mid,\ \rho > 0.$$

2.4 Solutions efficaces supportées et non supportées

Dans l'ensemble des solutions efficaces d'un problème de programmation linéaire discrète à objectifs multiples, deux types de solutions peuvent être différenciées :

- *Solutions supportées* : Elles appartiennent à l'enveloppe convexe de la frontière Pareto. Chacune de ces solutions peut être trouvée en optimisant une agrégation linéaire des objectifs. Toute fois, différents poids fournissent différentes solutions supportées, une même solution peut être générée en utilisant des poids différents.
- *Solutions non supportées* : Il existe d'autres solutions qui, bien qu'efficaces, ne peuvent être obtenues par la résolution d'un programme paramétrique (agrégé). Ces solutions, dites *non supportées*, sont dominées par certaines combinaisons convexes de solutions supportées ; il s'agit de points de $Z(s)$ à l'intérieur de l'enveloppe convexe de $Z(S)$

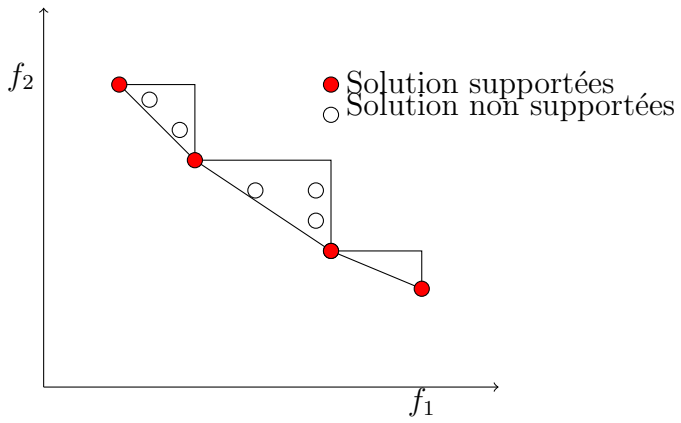

FIG. 2.3 – Solutions supportées et non supportées

Conclusion

Dans ce chapitre, on a rappelé quelques définitions et concepts de base de l'optimisation multiobjectif, qu'on pense nécessaire pour la suite de notre travail. Maintenant, on passe aux méthodes de résolution, qui est le sujet traité dans le chapitre suivant.

Chapitre 3

Méthodes de résolution

"Il n'y a pas une méthode unique pour étudier les choses".

Aristote (philosophe grec , 384-322 av. J.-C)

Introduction

En présence d'un problème d'optimisation concret, le chercheur est confronté à la principale difficulté du choix d'une méthode "efficace" (lorsqu'elles existent), capable de produire une solution "optimale" - ou de qualité acceptable - au prix d'un temps de calcul "raisonnable". Face à ce souci un grand nombre de méthodes ont été développées pour tenter d'apporter une réponse satisfaisante à ces problèmes. Parmi celles-ci, nous distinguons les méthodes dédiées à un problème et les méthodes plus génériques pouvant s'appliquer à un ensemble de problèmes.

Dans ce chapitre, nous essayons de faire un état d'art sur les différentes méthodes d'optimisation, on distingue deux grandes classes à savoir méthodes exactes et méthodes approchées. Les méthodes exactes examinent, souvent de manière implicite, la totalité de l'espace de recherche. Ainsi, elles ont l'avantage de produire une solution optimale lorsqu'aucune contrainte de temps n'est donnée. Néanmoins, le temps de calcul nécessaire pour atteindre une solution optimale peut devenir vite prohibitif ce, malgré les diverses techniques et heuristiques qui ont été développées pour accélérer l'énumération des solutions.

Dans de telles situations (et dans beaucoup d'autres), les méthodes approchées constituent une alternative indispensable et complémentaire. Le but d'une telle méthode n'est plus de fournir une solution optimale au problème donné. Elle cherche avant tout à produire une solution sous-optimale de meilleure qualité possible avec un temps de calcul raisonnable. En général, une méthode approchée examine seulement une partie de l'espace de recherche.

3.1 Classification de méthodes

Il existe un nombre important de méthodes, dont apparaissent plusieurs types de classement selon des critères différents.

Les méthodes de résolution de problèmes multiobjectif peuvent être rangées en ces trois grandes familles :

- **Les méthodes à préférence a priori**

  Dans ces méthodes, l'utilisateur définit le compromis qu'il désire réaliser (il fait part de ses préférences) avant de lancer la méthode d'optimisation. On retrouve dans cette famille la plupart des méthodes par agrégation (où les fonctions objectif sont

fusionnées en une seule).

- **Les méthodes à préférence progressive**

  Dans ces méthodes, l'utilisateur affine son choix de compromis au fur et à mesure du déroulement de l'optimisation. On retrouve dans cette famille les méthodes interactives.

- **Les méthodes à préférence a posteriori**

  Dans ces méthodes, l'utilisateur choisit une solution de compromis en examinant toutes les solutions extraites par la méthode d'optimisation. Les méthodes de cette famille fournissant, à la fin de l'optimisation, une surface de compromis.

Il existe des méthodes d'optimisation multiobjectif qui n'entrent pas exclusivement dans une famille. Par exemple, on peut utiliser une méthode à préférence a priori en lui fournissant des préférences choisies au hasard. Le résultat sera alors un grand nombre de solutions qui seront présentées à l'utilisateur pour qu'il décide de la solution de compromis. Cette combinaison forme alors une méthode à préférence a posteriori. [12] D'autres types de classement classifient les méthodes d'optimisation multiobjectif selon les principes mathématiques, c'est-à-dire est ce que les méthodes utilisent les dérivées des fonctions objectif ou non.

3.2 Méthodes Exactes

3.2.1 Méthode de pondération de fonctions objectif

Cette approche du résolution des problèmes d'optimisation multiobjectif est la plus évidente. D'ailleurs, on appelle aussi cette méthode l'«approche naïve» de l'optimisation multiobjectif [29]. Le but, ici, est de revenir à un problème d'optimisation mono-objectif,

pour lequel existent 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 pondérée des fonctions objectif. On obtient alors une nouvelle fonction objectif[12].

$$(p)\left\|\begin{array}{ll}\text{Minimiser} & f_{eq}(\overrightarrow{x}) = \sum_{i=1}^{k} w_i.f_i(\overrightarrow{x}) \\ \text{et que} & \overrightarrow{g}(\overrightarrow{x}) \leq 0 \\ \text{avec,} & \overrightarrow{h}(\overrightarrow{x}) = 0\end{array}\right.$$

Fréquemment, les poids doivent respecter la relation suivante : $w_i \geq 0$ pour tous $i \in \{1, \ldots, k\}$ et $\sum_{i=1}^{k} w_i = 1$.

3.2.2 Méthode de Keeney-Raiffa

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

$$\left\|\begin{array}{ll}\text{Minimiser} & f_{eq}(\overrightarrow{x}) = \prod_{i=1}^{k} w_i.f_i(\overrightarrow{x}) \\ \text{et que} & \overrightarrow{g}(\overrightarrow{x}) \leq 0 \\ \text{avec,} & \overrightarrow{h}(\overrightarrow{x}) = 0\end{array}\right.$$

On retrouve cette fonction dans la théorie de la fonction utilité multi-attribut exposée dans (M.A.U.T Multi Attribute Utility Theory)[31]. Cette théorie issue des sciences d'aide à la décision, traite des propriétés de la manière de créer une « fonction d'utilité ». [12]

3.2.3 Méthode de la distance à un objectif de référence

Cette méthode permet de transformer un problème d'optimisation multiobjectif en un problème d'optimisation mono-objectif. La somme que l'on va utiliser ici prendra la forme d'une distance ($\sqrt[k]{()^k}$) ou la racine $k^{\text{ème}}$ de la somme d'éléments élevés à une certaine puissance k. De plus, dans certains cas on normalisera les éléments par rapport à une valeur avant d'en faire la somme [5],[40].

Nous commençons à partir du problème p où le vecteur $\overrightarrow{F}$ (composant $\overrightarrow{F}_i, i \in \{1, \ldots, k\}$) est un vecteur dont les composants correspond à un objectif idéal (ou objectif de référence) dans une certaine mesure définis par le décideur (il peut être un point idéal, ou un point qui représente la meilleure préférence du décideur). Le vecteur $\overrightarrow{F}$ est choisi par le décideur. Nous pouvons par exemple, employer la distance absolue. la définition de cette distance est

$$L_r\big(\overrightarrow{f}(\overrightarrow{x})\big) = \big[\sum_{i=1}^{k} |\overrightarrow{F}_i - f_i(\overrightarrow{x})|^r\big]^{\frac{1}{r}}$$

avec $0 \leq r \leq \infty$

3.2.4 Méthode de compromis (L'approche par ϵ-contrainte)

Une autre façon de transformer un problème d'optimisation multiobjectif en un problème simple objectif est de convertir $m-1$ des m objectifs du problème en contraintes et d'optimiser séparément l'objectif restant[12].

La démarche est la suivante :

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

On appelle aussi cette méthode la méthode de la ϵ-contrainte [40]. Le problème peut être reformulé de la manière suivante[6] :

$$
\begin{array}{ll}
\text{Minimiser} & f_i(\overrightarrow{x}) \\
\text{tel que,} & f_1(\overrightarrow{x}) \leq \epsilon_1 \\
 & \vdots \\
 & f_{i-1}(\overrightarrow{x}) \leq \epsilon_{i-1} \\
 & f_{i+1}(\overrightarrow{x}) \leq \epsilon_{i+1} \\
 & \vdots \\
 & f_m(\overrightarrow{x}) \leq \epsilon_m \\
\text{et que} & \overrightarrow{g}(\overrightarrow{x}) \leq 0 \\
\text{avec,} & \overrightarrow{x} \in \mathbb{R}^n, \overrightarrow{f}(\overrightarrow{x}) \in \mathbb{R}^m, \overrightarrow{g}(\overrightarrow{x}) \in \mathbb{R}^q
\end{array}
$$

L'approche par ϵ-contrainte doit aussi être appliquée plusieurs fois en faisant varier le vecteur $\overrightarrow{\epsilon}$ 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. Ainsi la figure 5.2 illustre, en dimension 2, le cas où un point $(\epsilon; f_{1_{min}})$, de la partie non convexe, est trouvé. La figure 5.2 montre aussi comment cette approche procède. En transformant des fonctions objectif en contraintes, elle diminue la zone réalisable par paliers. Ensuite, le processus d'optimisation trouve le point optimal sur l'objectif restant.

L'inconvénient de cette approche réside dans le fait qu'il faille lancer un grand nombre de fois le processus de résolution. De plus, pour obtenir des points intéressants et bien répartis sur la surface de compromis, le vecteur $\overrightarrow{\epsilon}$ doit être choisi judicieusement. Il est clair qu'une bonne connaissance du problème a priori est requise.

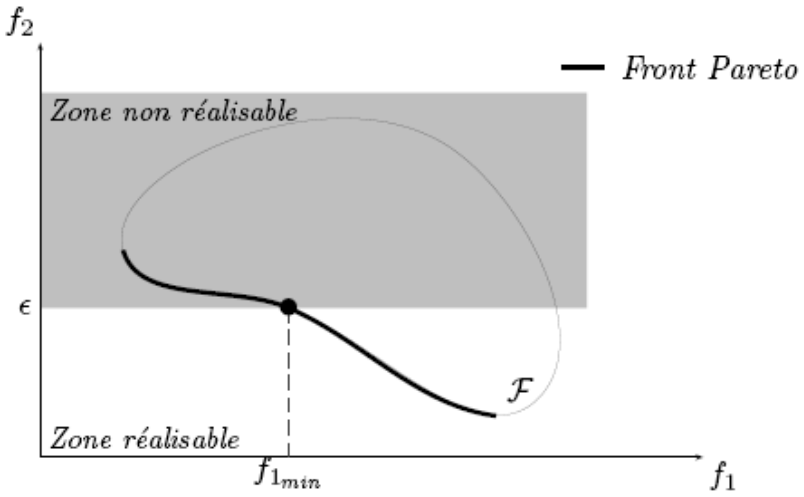

FIG. 3.1 – Interprétation graphique de l'approche par ϵ-contrainte.

3.2.5 Méthode de programmation de but (Goal programming method)

Cette méthode est proche de la méthode de but à atteinte. La différence principale est que, après avoir transformé le problème d'optimisation, nous avons des contraintes d'égalité au lieu des contraintes d'inégalité. Cette approche est la suivante [12] :

- Nous choisissons un vecteur initial des fonctions objectif $\overrightarrow{F}$,
- À chaque objectif, nous associons deux nouvelles variables "slack" liées au vecteur initial des fonctions objectif que nous avons choisi.
- Après, nous minimisons une des deux variables.

Présentation de la méthode Nous commençons à partir du problème P. Nous choisissons un vecteur initial des fonctions objectif $\overrightarrow{F} \in \mathbb{R}^k$. Nous associons également un ensemble de variables "slack" d_i^+ et d_i^- à chaque fonction objectif $f_i(\overrightarrow{x}), i \in \{1, \ldots, k\}$.

Ainsi, nous obtenons le problème suivant :

$$\left\| \begin{array}{ll} \text{Minimiser} & (d_1^+ \ \ ou \ \ d_1^-, \ldots, d_k^+ \ \ ou \ \ d_k^-) \\ \text{tel que,} & f_1(\overrightarrow{x}) = F_1 + d_1^+ - d_1^- \\ & \vdots \\ & f_k(\overrightarrow{x}) = F_k + d_1^+ - d_k^- \\ \text{et que} & \overrightarrow{g}(\overrightarrow{x}) \leq 0 \\ \text{avec,} & \overrightarrow{h}(\overrightarrow{x}) = 0 \end{array} \right.$$

Les variables "slack" que nous essayons de minimiser doivent respecter quelques contraintes : $d_1^+ \geq 0$ et $d_k^- \geq 0$ et $d_1^+ \cdot d_k^- = 0$ avec $i \in \{1, \ldots, k\}$

Nous pouvons essayer de minimiser les divers combinaisons des coefficients d_i^+ et d_i^-, selon une manière, pour atteindre le but $\overrightarrow{F}$.

3.2.6 Méthode lexicographique

Cette méthode est très intuitive. Elle consiste à considérer les fonctions objectif une après l'autre et minimiser un problème d'optimisation mono-objectif. Au fur et à mesure, on lui ajoute graduellement des contraintes[11].

Présentation de la méthode Nous commençons à partir du problème P. Nous procédons en k étapes (il y a autant d'étapes qu'il y a de fonctions objectif). Commençons par la première fonction objectif. Nous résolvons :

$$\left\| \begin{array}{ll} \text{Minimiser} & f_1(\overrightarrow{x}) \\ \text{tel que,} & \overrightarrow{g}(\overrightarrow{x}) \leq 0 \\ \text{avec} & \overrightarrow{h}(\overrightarrow{x}) = 0 \end{array} \right.$$

Nous notons par f_1^* la solution de ce problème.

Après résolution, nous transformons la première fonction objectif en contraintes d'égalité. De ce fait, nous prenons la seconde fonction objectif et on résout le problème.

Nous répétons cette approche jusqu'à atteindre les K fonctions objectif. Pour finir, nous obtenons le problème à résoudre :

$$\left\| \begin{array}{ll} \text{Minimiser} & f_k(\overrightarrow{x}) \\ \text{tel que} & f_1(\overrightarrow{x}) = f_1^*, \ldots, f_{k-1}^* \\ & \overrightarrow{g}(\overrightarrow{x}) \leq 0 \\ \text{avec} & \overrightarrow{h}(\overrightarrow{x}) = 0 \end{array} \right.$$

La dernière valeur de $\overrightarrow{x}$ est celle qui réduit au minimum toutes les fonctions objectif.

Remarque 3 : Dans la littérature, il existe d'autres méthodes qu'on n'a pas développé comme :

- Méthode appropriée de contraintes d'égalité ou Proper-equality-constraints-method ;
- Méthode appropriée de contraintes d'inégalité ou Proper-inequality-constraints-method ;
- Algorithme de Lin-Tabak ;
- Algorithme de Lin-Giesy ;
- etc...

3.2.7 Méthode de compromis par substitution

Cette méthode (Surrogate Worth Tradeoff (SWT)) ou «méthode de compromis par substitution» a été très utilisée pour l'optimisation de ressources en eau [29]. Elle est basée sur la méthode de compromis, à laquelle on a ajouté un processus interactif, pour que la méthode aboutisse à la solution la plus susceptible de satisfaire le décideur [12].

Présentation de la méthode La méthode comprend sept étapes (voir algorithme 2) :

Il est difficile de comprendre clairement la signification du coefficient W_{1j}.

Algorithme 2 Surrogate Worth Tradeoff (SWT)-algorithm

étape 1 Trouver le minimum de la fonction f_j par la résolution de :

$$\left\| \begin{array}{ll} \text{Minimiser} & f_j(\overrightarrow{x}) \\ \text{tel que} & \overrightarrow{g}(\overrightarrow{x}) \leq 0 \\ \text{avec} & \overrightarrow{h}(\overrightarrow{x}) = 0 \end{array} \right.$$

La solution de ce problème devient la $j^{\grave{e}me}$ coordonnée du vecteur f_{min}, qui recueille l'ensemble des $k-1$ valeurs minimum des f_j, $j = 1, \ldots, k$. Si possible, nous recherchons pendant cette étape les coordonnées du vecteur f_{max}, qui recueille l'ensemble des $k-1$ valeurs maximum des f_j, $j = 2, \ldots, k$. également.

étape 2 Choisir la valeur du départ de $\epsilon \geq f_{j_{min}}$, $j = 2, \ldots, k$.

étape 3 Résoudre le problème suivant :

$$\left\| \begin{array}{ll} \text{Minimiser} & f_1(\overrightarrow{x}) \\ \text{Minimiser} & f_2(\overrightarrow{x}) \geq \epsilon_2, \ldots, f_k(\overrightarrow{x}) \geq \epsilon_k \\ \text{tel que} & \overrightarrow{g}(\overrightarrow{x}) \leq 0 \\ \text{avec} & \overrightarrow{h}(\overrightarrow{x}) = 0 \end{array} \right.$$

Si certaines contraintes de ce problème ne sont pas respectées, nous plaçons $\epsilon_j = f_j(\overrightarrow{x})$ quand j correspond à l'index de la contrainte qui n'est pas respectée (nous détachent les contraintes). Nous devons reprendre cette étape. Si les contraintes sont respectées, nous appelons $\overrightarrow{x^*}$ le vecteur solution et nous allons à la prochaine étape.

étape 4 Si les contraintes sont suffisamment relâchées, nous allons à l'étape 5 ; autrement, nous choisissons une nouvelle valeur pour $\epsilon_j \geq f_{j_{min}}$, pour tout $j = 2, \ldots, k$ et retourner à l'étape 3. Une méthode de choix de la nouvelle valeur pour ϵ est de choisir une valeur très grande pour ϵ et puis de diminuer chaque ϵ_j, pour tout $j = 2, \ldots, k$, en utilisant un certain $\Delta_j > 0$, chaque fois qu'une contrainte n'est pas respectée. Si les contraintes ne sont pas respectées, nous plaçons $\epsilon_j = f_j(\overrightarrow{x})$ où j est chaque index de la contrainte qui n'est pas respectée (ici, $\overrightarrow{x}$ correspond à la valeur courante de la solution).

étape 5 Nous demandons au décideur de choisir les coefficients W_{1j}, pour tout $j = 2, \ldots, k$. Ces coefficients représentent l'opinion du décideur au sujet d'une augmentation de la fonction f_j par des unités λ_j (la manière que nous calculons λ_{1j} est présentée ci-dessous). Le coût est mesuré sur une échelle de -10 à +10, où -10 correspond à l'opinion défavorable au sujet de cette augmentation et +10 correspond à un avis favorable. 0 correspond à l'indifférence. Cette opération est répétée pour j variant de 2 à k.

étape 6 Nous répétons l'étape 5 jusqu'à ce que nous trouvons des valeurs nulles pour les coefficients W_{1j}, $j = 2, \ldots, k$. Nous notons par f_j^*, pour tout $j = 2, \ldots, k$, les valeurs de la fonction objectif correspondant à ces coefficients.

étape 7 Le vecteur solution $\overrightarrow{x^*}$ préférée est déterminé en résolvant le problème suivant :

$$\left\| \begin{array}{ll} \text{Minimiser} & f_1(\overrightarrow{x}) \\ \text{tel que} & f_2(\overrightarrow{x}) = f_2^*, \ldots, f_k(\overrightarrow{x}) = f_k^* \\ \text{et} & \overrightarrow{g}(\overrightarrow{x}) \leq 0 \\ \text{avec} & \overrightarrow{h}(\overrightarrow{x}) = 0 \end{array} \right.$$

3.2.8 Méthode de Fandel

Le but de cette méthode est d'aider le décideur dans le choix des poids [22]

Présentation de la méthode Nous partons du problème P, que nous modifions de cette manière :

$$P = \left\| \begin{array}{ll} \text{Minimiser} & \sum_{i=1}^{k} w_i.f_i(\overrightarrow{x}) \\ \text{tel que} & \overrightarrow{g}(\overrightarrow{x}) \leq 0 \\ \text{avec} & \overrightarrow{h}(\overrightarrow{x}) = 0 \end{array} \right.$$

Nous avons $w_i \geq 0$ et $\sum_{i=1}^{k} w_i = 1$. Nous trouvons cette condition dans la méthode de la somme-pondérée-des-fonctions-objectif.

Comme avec la somme pondérée des fonctions objectif, la variation des coefficients w_i nous permet de déterminer des points sur la surface de compromis. Néanmoins, dans la méthode de Fandel, nous supposons également que le décideur recherche une solution qui est proche de la solution idéale.

3.2.9 Méthode de Geoffrion

Cette méthode repose sur un algorithme nommé : l'algorithme de Frank-Wolfe [22]

Présentation du la méthode Nous transformons le problème P de la façon suivante :

$$\left\| \begin{array}{ll} \text{Minimiser} & \left(\nabla_{\overrightarrow{x}} p \cdot [\overrightarrow{f}\,\overrightarrow{x}]\right)^t \cdot \overrightarrow{\delta} \\ \text{avec} & \overrightarrow{h}(\overrightarrow{x}) = 0 \\ & \overrightarrow{g}(\overrightarrow{x}) \leq 0 \end{array} \right. ,$$

avec $\overrightarrow{\delta} \in \mathbb{R}^k$.

La fonction de préférence $p \cdot [\overrightarrow{f}(\overrightarrow{x})]$ n'est pas nécessairement connue par le décideur.

Deux méthodes pour déterminer le point de départ $\overrightarrow{x^0}$ sont possibles : soit, le décideur choisit un vecteur suivant la méthode de Jahn[3], soit le vecteur est calculé en résolvant le problème de

substitution (P') suivant :

$$\left\| \begin{array}{ll} \text{Minimiser} & \sum_{i=1}^{k} w_i \cdot f_i(\overrightarrow{x}) \\ \text{avec} & \overrightarrow{h}(\overrightarrow{x}) = 0 \\ \text{et} & \overrightarrow{g}(\overrightarrow{x}) \leq 0 \end{array} \right.$$

Nous identifions ici la méthode de somme-pondéré-des-fonctions-objectif en résolvant ce problème ; $\overrightarrow{x^0} = \overrightarrow{x^*}$.

Dans ce cas, nous pouvons résoudre le problème P'. La solution nous donne une direction de recherche δ. Des informations sur la fonction de préférence sont maintenant exigées. Nous devons transformer le problème P' en employant la relation suivante :

$$\nabla_{\overrightarrow{x}} p \cdot \big[\overrightarrow{f}(\overrightarrow{x})\big] = \sum_{i=1}^{k} \big[\frac{\partial p}{\partial f_i}\big]_{\overrightarrow{x}} \cdot \nabla_{\overrightarrow{x}} \cdot f_i(\overrightarrow{x})$$

On obtient le problème suivant :

$$\left\| \begin{array}{ll} \text{Minimiser} & \sum_{i=1}^{k} w_i \cdot (\nabla_{\overrightarrow{x}} f_i(\overrightarrow{x^k}))^t \cdot \overrightarrow{\delta} \\ \text{avec} & \overrightarrow{h}(\overrightarrow{x}) = 0 \\ \text{et} & \overrightarrow{g}(\overrightarrow{x}) \leq 0 \end{array} \right.$$

On obtient

$$w_i = \frac{\big[\frac{\partial p}{\partial f_i}\big]_{\overrightarrow{x}}}{\big[\frac{\partial p}{\partial f_1}\big]_{\overrightarrow{x}}}$$

avec $i \in \{1, \ldots, k\}$, et la direction de l'étape est donné par $\overrightarrow{\delta} = \overrightarrow{x^{k+1}} - \overrightarrow{x^k}$.

Les coefficients de poids reflètent la différence exécutée par le décideur entre la fonction objectif f_i et la fonction objectif f_1 de référence.

Après la résolution du problème de substitution ci-dessus, la longueur de l'étape λ doit être déterminée par le décideur lui-même, qui doit considérer le but suivant :

$$\left\| \begin{array}{ll} \text{Minimiser} & p\big[f(\overrightarrow{x_k} + \overrightarrow{\lambda^k} \cdot \overrightarrow{\delta^k})\big] \\ \text{avec} & \overrightarrow{h}(\overrightarrow{x_k} + \overrightarrow{\lambda^k} \cdot \overrightarrow{\delta^k}) = 0 \\ \text{et} & \overrightarrow{g}(\overrightarrow{x_k} + \overrightarrow{\lambda^k} \cdot \overrightarrow{\delta^k}) \leq 0 \end{array} \right.$$

tout en appliquant l'hypothèse suivante :

$$0 \leq \lambda^k \leq 1$$

Une fois la longueur de l'étape déterminée, le nouveau vecteur fonction objectif peut être calcul :

$$\overrightarrow{f^{k+1}} = \overrightarrow{f}(\overrightarrow{x_k} + \overrightarrow{\lambda^k} \cdot \overrightarrow{\delta^k})$$

Le décideur évalue le résultat, et décide si le processus devrait continuer ou si la solution peut être acceptée comme solution de différence. si le décideur décide de continuer, il peut couvrir la surface de différence.

3.2.10 Méthode de Simplexe

Cette méthode n'a rien à voir avec la méthode du simplexe utilisée en PL. Il s'agit là d'une méthode de recherche séquentielle de l'optimum d'un problème. Le logiciel [multi simplex] réalise cette optimisation pour un problème d'optimisation multiobjectif de manière interactive.

Cette méthode utilise $k+1$ essais (où k représente la dimension de la variable de décision $\overrightarrow{x}$) pour définir les améliorations des fonctions objectif. Ces améliorations sont obtenues en utilisant une méthode d'agrégation floue.

On commence par choisir, d'un façon aléatoire, $k+1$ valeurs pour la variable de decision $\overrightarrow{x}$. L'algorithme évalue alors les $k+1$ points et supprime le point le moins "efficace". Il crée alors un nouveau point à partir du point supprimé et recommence l'évaluation.
L'algorithme comporte quelques règles, permettant le choix des points en évitant de tourner auteur d'une mauvaise solution. Les deux principales règles sont :

Règle 1 : rejeter les pires solutions. La nouvelle position de la variable de decision $\overrightarrow{x}$ est calculée par réflexion de la position rejetée (voir la figure 3.3). Après cette transformation, on recherche le nouveau pire point. La méthode est alors répétée avec ce point en éliminé, ... etc. A chaque étape, on se rapproche de la zone où se trouve l'optimum recherché.

Règle 2 : ne jamais revenir sur un point qui vient juste d'être rejeté. Sans cette règle, l'algorithme pourrait osciller entre Deux « mauvais » points. [12]

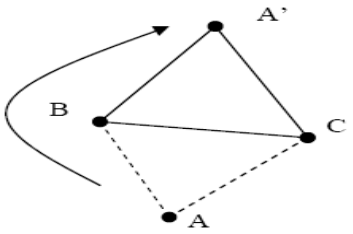

FIG. 3.2 – Choix d'un nouveau point par symétrie.

W : point le moins favorable ou point venant d'être rejeté.

B : le point le plus favorable.

R : le deuxième point le plus favorable.

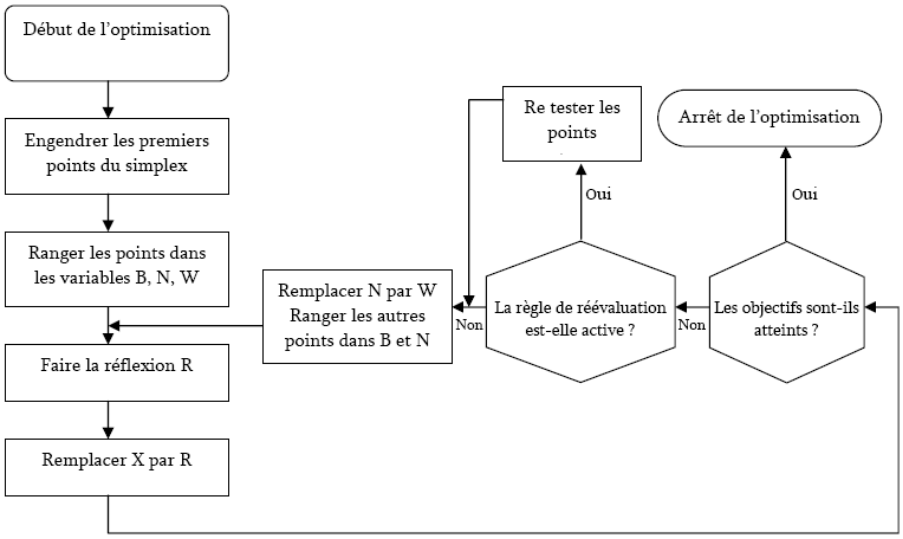

FIG. 3.3 – L'algorithme de la méthode de Simplex.

3.3 Méthodes floues

3.3.1 Méthode de Sakawa

Cette méthode fait intervenir la logique floue à tous les niveaux (sur les paramètres du problème ainsi que sur les paramètres des contraintes). Ces solutions auront un niveau d'appartenance. Ce seront des solutions qui auront une corrélation variable avec l'objectif initial [49] (le niveau de corrélation avec l'objectif initial est fixé par l'utilisateur)[12].

Nous commençons à partir du problème :

$$\begin{Vmatrix} \min f_1(\overrightarrow{x}), \ldots, f_k(\overrightarrow{x}), \\ \overrightarrow{g}(\overrightarrow{x}) \leq 0, \end{Vmatrix}$$

3.3.2 Méthode de Reardon

Nous présentons une méthode simplifiée qui emploie la logique floue pour résoudre un problème à objectifs multiples. Quelques exemples qui emploient cette méthode peuvent être trouvés dans [45]

Présentation de la méthode Nous commençons à partir du problème P. Pour chaque fonction objectif, nous définissons une fonction d'adhésion, qui a la forme dessinée dans la figue 3.4.

Cette fonction d'adhésion permet d' "informer" l'algorithme que la fonction objectif a sa valeur située à l'intérieur de l'intervalle $[O_i - E_i, O_i + E_i]$ (le secteur où la valeur d'adhésion est 0). Si la valeur de la fonction objectif est située en dehors de de cet intervalle, nous pénalisons la fonction objectif de plus en plus. Dans ce cas, la pénalité varie entre 0 et S_{min} et S_{max}.

S_{min} **et** S_{max} **:** facteurs de proportionnalité brouillés.

f_{min} **:** valeur minimum de fonction objectif numéro i.

f_{max} **:** valeur maximum de fonction objectif numéro i.

Algorithme 3 Algorithme de la méthode de Sakawa

étape 1 : sous la contrainte $\overrightarrow{g}(\overrightarrow{x}) \leq 0$, nous optimisons, chaque fonction objectif séparément, afin de trouver l'intervalle de variation de la fonction d'adhésion de la fonction objectif.

étape 2 : nous définissons la fonction d'adhésion de cette façon :

$$\mu_i(\overrightarrow{x}) = \frac{f_{i1} - f_i(\overrightarrow{x})}{f_{i1} - f_{i0}} \tag{3.1}$$

où

- f_{i0} est la moindre valeur intéressante de la fonction d'adhésion,
- f_{i1} est la valeur la plus intéressante de la fonction d'adhésion.

étape 3 : nous définissons la fonction DM_i de prise de décision comme suit :

$$DM_i(\overrightarrow{x}) = \begin{cases} 0, & \text{si } \mu_i(\overrightarrow{x}) \leq 0\,; \\ \mu_i(\overrightarrow{x}), & \text{si } 0 \leq \mu_i(\overrightarrow{x}) \leq 1\,; \\ 1, & \text{si } \mu_i(\overrightarrow{x}) \geq 1. \end{cases} \tag{3.2}$$

où

- $DM_i(\overrightarrow{x}) = 0$, quand la fonction objectif est non atteinte,
- $DM_i(\overrightarrow{x}) = 1$, quand la fonction objectif est atteinte.

étape 4 : nous maximisons la fonction DM_i.

étape 5 : s'il n'y a aucune solution, ou si la solution ne satisfait pas le décideur, alors nous devons modifier la fonction d'adhésion et passer en arrière à l'étape 3.

étape 6 : Arrêter et afficher les résultats.

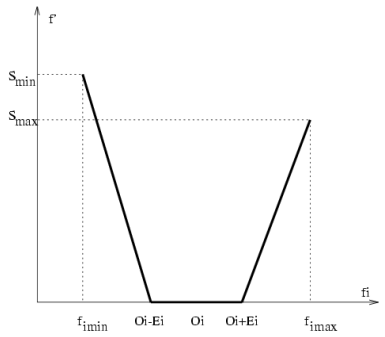

FIG. 3.4 – Fonction d'adhésion de Reardon

O_i **:** valeur expérimentale de fonction objectif numéro i. Nous voudrions $f_i = O_i$.

E_i **:** marge d'erreur acceptable (l'objectif O_i a un intervalle de satisfaction $2 \cdot E_i$).

- si $f_i \leq (O_i - E_i)$ alors $f'(f_i) = [\frac{S_{max}}{f_{min} - (O_i - E_i)}] \cdot (f_i - (O_i - E_i))$;
- si $(O_i - E_i) \leq f_i \leq (O_i + E_i)$ alors $f'(f_i) = 0$;
- si $f_i \geq (O_i + E_i)$ alors $f'(f_i) = [\frac{S_{min}}{(O_i + E_i) - f_{max}}] \cdot (f_i - (O_i + E_i))$.

3.4 Méthodes Approchées

Les métaheuristiques, sont des méthodes générales de recherche dédiées aux problèmes d'optimisation difficiles [48]. Ces méthodes sont, en général, présentées sous la forme de concept d'inspiration. Comme nous le verrons plus tard, elles reprennent des idées que l'on retrouve parfois dans la vie courante. Ces méthodes ont des inspirations de l'éthologie comme les colonies de fourmis, de la physique comme le recuit simulé, et de la biologie comme les algorithmes évolutionnaires. Dans la figure suivante 3.5, on voit un classement de ces méthodes selon le principe d'inspiration utilisé, est ce qu'il est basé. Plusieurs définition ont été données pour clarifié le concepts de l'heuristique, parmi les quels on trouve les définitions suivantes :

Définition 3.1 (Heuristique selon Reeves) *Une heuristique est une technique trouvant de bonnes solutions (c-à-d proches de l'optimum) pour un coût de calcul raisonnable, sans pouvoir*

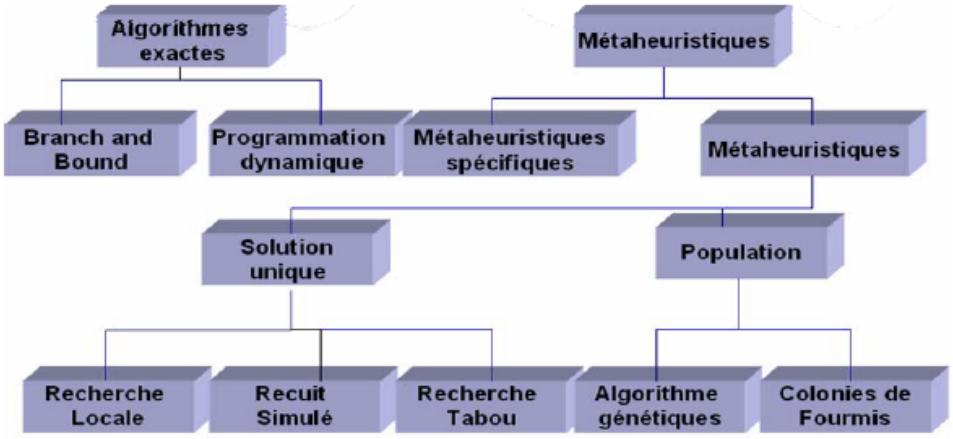

FIG. 3.5 – Schéma des méthodes basées sur les métaheuristiques

garantir l'admissibilité ou l'optimalité, ou même dans de nombreux cas, préciser la distance à l'optimum d'une solution particulière.[44]

Ce type de méthodes est particulièrement utile pour les problèmes nécessitant une solution en temps réel (ou très court) ou pour résoudre des problèmes difficiles sur des instances numériques de grande taille. Elles peuvent aussi être utilisées afin d'initialiser une méthode exacte (Branch and Bound par exemple).

D'autres méthodes sont apparues, qualifiées de métaheuristiques. Plusieurs définitions ont été proposées pour décrire leurs particularités par rapport aux heuristiques.

Définition 3.2 (Métaheuristique selon Osman et Laporte) *Une métaheuristique est un processus itératif de génération guidant une heuristique subordonnée en combinant intelligemment différents concepts pour explorer et exploiter l'espace de recherche en utilisant des stratégies pour structurer l'information de manière à trouver efficacement des solutions proches de l'optimum.*

Cependant, cette définition ne fait pas clairement apparaître l'indépendance des métaheuristiques actuelles vis-à-vis des problèmes à traiter. Nous préférerons la définition de Pirlot [44] (définition 3.3) qui assimile les métaheuristiques à des stratégies de recherche. Ainsi, il faut distinguer les

heuristiques ciblées sur un problème particulier et les métaheuristiques plus générales et adaptables pour résoudre un grand nombre de problèmes. Cependant, pour être suffisamment performante sur un problème donné, une métaheuristique nécessitera une adaptation intelligente aux caractéristiques de ce problème. Une même métaheuristique peut d'ailleurs être à l'origine d'heuristiques différentes pour un même problème.

Définition 3.3 (Métaheuristique selon Pirlot) *Les métaheuristiques ne sont pas à proprement parler des heuristiques, mais des schémas généraux, des "moules" à heuristiques ; le schéma général doit être adapté à chaque type particulier de problème.[44]*

Malgré la grande diversité de métaheuristiques (algorithmes génétiques, méthode de bruitage, méthode GRASP, recherche à voisinage des variable, recherche dispersée, recherche tabou, recuit simulé, systèmes de fourmis, ...), leurs principes sont souvent assez proches. Taillard distingue trois éléments principaux constitutifs de toutes les métaheuristiques :

- la structure de voisinage permettant de modifier une solution,
- la mémoire des solutions déjà obtenues,
- la construction de solutions entièrement nouvelles.

Ainsi, chaque métaheuristique utilise au moins l'un de ces trois éléments. De plus, l'hybridation de différentes métaheuristiques est une pratique de plus en plus courante permettant d'obtenir des heuristiques performantes. Taillard propose d'ailleurs une unification de la plupart des métaheuristiques telles qu'elles sont implantées actuellement sous la dénomination de programmation à mémoire adaptative.[19]

Une métaheuristique est donc une méthode très générale, qui nécessite quelques transformations (mineure en générale) avant de pouvoir être appliquée à la résolution d'un problème particulier. Si l'on considère les différentes métaheuristiques, on constate que celles-ci se répartissent en trois familles :

3.4.1 Quelques méthodes métaheuristiques

✣ **Recuit simulé** Cette méthode de recherche a été proposée par les chercheurs d'IBM qui étudiaient les verres de spin. Ici, on utilise un processus métallurgique (le recuit) pour trouver un minimum. En effet, pour qu'un métal retrouve une structure proche du cristal parfait (l'état cristallin correspond au minimum d'énergie de la structure atomique du métal), on porte celui-ci à une température élevée, puis on le laisse refroidir lentement de manière à ce que les atomes aient le temps de s'ordonner régulièrement.

Ce processus métallurgique a été transposé à l'optimisation et a donné une méthode simple et efficace. Le fonctionnement de cet algorithme est le suivant :

- On commence par choisir un point de départ au hasard ;
- On calcule un voisin de ce point ($\gamma = \mathcal{V}(x)$) ;
- On évalue ce point voisin et on calcule l'écart par rapport au point d'origine ($\Delta C = C(\gamma) - C(x)$) ;
- Si cet écart est négatif, on prend le point γ comme nouveau point de depart, s'il est positif on peut quand même accepter le point γ comme nouveau point de depart, mais avec une probabilité $e^{-\frac{\Delta C}{T}}$ (qui varie en sens inverse de la temperature T) ;
- Au fur et a mesure du déroulement de l'algorithme, on diminue la temperature $T(T = \alpha(T))$, souvent par paliers ;
- On répète toutes ces étapes tant que le système n'est figé (par exemple, tant que la température n'a pas atteint un seuil minimal).

Au début de la simulation, les points ont une grande capacité d'exploration de l'espace d'état car l'algorithme accepte des déplacements très importants. Au fur à mesure que T diminue P augmente, la capacité de déplacement d'un point diminue et les points améliorant leur valeur sont de plus en plus nombreux, la chance d'une solution négative d'être acceptée diminue. Quand T est proche de zéro, seuls les points dont les valeurs sont améliorées seront acceptés. A la fin il y a une probabilité nulle que n'importe quel changement négatif de température peut être réalisé. A ce point de "refroidissement", le système doit converger

vers la solution optimale. Dans la littérature des méthodes d'optimisation multiobjectif, on trouve deux importantes extensions de la méthode Recuit Simulé basées sur les frontières de Pareto, La méthode P.A.S.A (Pareto Archived Simulated et La méthode M.O.S.A (Multiple objectif Simulated Annealing) Annealing).

✣ **La recherche tabou** Cette méthode, mise au point par [F. Glover 86] est conçue en vue de surmonter les minima locaux de la fonction objectif. C'est une technique d'optimisation combinatoire que certains présentent comme une alternative au recuit simulé.

A partir d'une configuration initiale quelconque, Tabou engendre une succession de configurations qui doivent aboutir à la configuration optimale. A chaque itération, le mécanisme de passage d'une configuration $\overrightarrow{x_n}$, à la suivante $\overrightarrow{x_{n+1}}$, est le suivant :

– on construit l'ensemble des «voisins» de $\overrightarrow{x_n}$, c'est-à-dire l'ensemble des configurations accessibles en un seul «mouvement» élémentaire à partir de $\overrightarrow{x_n}$ (si cet ensemble est trop vaste, on en extrait aléatoirement un sous-ensemble de taille fixée) : soit le Voisinage $(\overrightarrow{x_n})$ l'ensemble (ou le sous-ensemble) envisagé ;

– on évalue la fonction objectif f du problème de chacune des configurations appartenant au Voisinage $(\overrightarrow{x_n})$. La configuration $\overrightarrow{x_{n+1}}$, qui succède à la configuration $\overrightarrow{x_n}$ dans la chaîne de Markov construite par Tabou, est la configuration de Voisinage $(\overrightarrow{x_n})$ en laquelle f prend sa valeur minimale.

Notons que la configuration $\overrightarrow{x_{n+1}}$ est adoptée même si $f(\overrightarrow{x_{n+1}}) > f(\overrightarrow{x_n})$: c'est grâce à cette particularité que Tabou permet d'éviter les minima locaux de f.

Cependant, telle quelle la procedure ne fonctionne généralement pas, car il y a un risque important de retourner à une configuration déjà retenue lors d'une iteration précédente, ce qui provoque l'apparition d'un cycle. Pour éviter ce phénomène, on met à jour, à chaque itération, une liste tabou de mouvements interdits ; cette liste qui a donné son nom à la méthode contient les mouvements inverses $(\overrightarrow{x_{n+1}} \rightarrow \overrightarrow{x_n})$ des m derniers mouvements $(\overrightarrow{x_n} \rightarrow \overrightarrow{x_{n+1}})$ effectues (typiquement $m = 7$). La recherche du successeur de la configuration courante $\overrightarrow{x_n}$ est alors restreinte aux voisins de $\overrightarrow{x_n}$ qui peuvent être atteints sans utiliser

un mouvement de la liste tabou. La procedure peut être stoppée dès que l'on effectue un nombre donné d'itérations, sans améliorer l'actuelle meilleure solution.

- **La méthode GRASP** La métaheuristique *GRASP* a été proposée initialement par Féo et Resende. C'est une méthode multi-départ en deux phases pour la résolution approchée de problèmes difficiles de l'optimisation combinatoire. La première phase correspond à la construction d'une solution initiale à l'aide d'une procédure gloutonne aléatoire. L'incorporation d'une composante aléatoire permet d'obtenir des solutions dans des zones diverses de l'espace des solutions. La seconde phase correspond à une recherche locale améliorant ces solutions. Ce processus est répété un grand nombre de fois.
- **Méthode de Colonie de Fourmis** La recherche guidée par Marco Dorigo produit un nouveau membre de cette classe d'algorithmes : l'algorithme de «système de fourmis».

  Algorithme de base : Le point de départ de cet algorithme se base sur l'observation des fourmis qui construisent des chemins entre une source de nourriture et leur nid. Les fourmis sont capables de déposer sur le sol une certaine quantité d'une substance chimique volatile (le pheromone) qu'elles peuvent détecter ensuite. Les fourmis se déplacent au hasard, dans ce cas, on peut dire qu'elles sont aveugles, mais sont attirées par les chemins de pheromone déposées par d'autres fourmis. Ainsi, plus les fourmis empruntent un chemin, plus il y aura des fourmis attirées par cet itinéraire.
- **Monté Carlo** Les méthodes Monté Carlo consistent en des simulations expérimentales ou informatiques des problèmes mathématiques ou physiques, basées sur le tirage des nombres aléatoires. Généralement on utilise en fait des séries de nombres pseudo-aléatoires générées par des algorithmes spécialisés. Les propriétés de ces séries sont très proches de celles d'une véritable suite aléatoire.

  Le grand avantage de cette méthode est sa simplicité. Elle permet entre autres de visualiser l'effet de différents paramètres et de donner ainsi des orientations, d'étudier des structures

intéressantes qui auraient été a priori écartées et de trouver facilement des structures que l'on n'aurait pas aussi bien optimisées «à la main».

Les méthodes de types Monté Carlo recherchent l'optimum d'une fonction en générant une suite aléatoire de nombres en fonction d'une loi uniforme.

- **Les algorithmes évolutionnaires** On peut distinguer trois grandes classes d'algorithmes évolutionnaires : les algorithmes génétiques [Holland, 1975 ; Goldberg, 1989], les stratégies d'évolution [Schwefel, 1981] et la programmation évolutive [Fogel, 2000]. Ces méthodes se différencient par leur manière de représenter l'information et par leur façon de faire évoluer la population d'une génération à l'autre. Un algorithme évolutionnaire est typiquement composé de trois éléments fondamentaux :
  - une ***population*** constituée de plusieurs individus représentant des solutions potentielles (configurations) du problème donné,
  - un ***mécanisme d'évaluation des individus*** permettant de mesurer l'adaptation de l'individu à son environnement,
  - un ***mécanisme d'évolution de la population*** permettant, grâce à des opérateurs pré-définis, d'éliminer certains individus et d'en créer de nouveaux.

  Parmi les composants d'un algorithme évolutionnaire, l'***individu*** et la ***fonction d'évaluation*** correspondent respectivement à la notion de configuration et à la fonction d'évaluation dans les méthodes de voisinage. Le mécanisme d'évolution est composé de plusieurs opérateurs tels que la sélection, la mutation et le croisement.

  La ***sélection*** a pour objectif de sélectionner des individus qui vont pouvoir se reproduire pour transmettre leurs caractéristiques à la génération suivante. Le ***croisement*** ou recombinaison est un opérateur permettant de construire des nouveaux individus enfants à partir des caractéristiques d'individus parents sélectionnés. La ***mutation*** effectue des légères modifications de certains individus. Comme exemple des algorithmes évolutionnaires, nous présentons maintenant les algorithmes génétiques[6].

- **Les algorithmes génétiques** Les algorithmes évolutionnaires sont parmi les métaheuristiques à base de population, ils sont inspirés de la biologie toute en introduisant le théorème de Darwin dans l'évolution des espèces. Les algorithmes génétiques (AGs) ont été introduits par Holland [Holland, 1975] comme un modèle de méthode adaptative. Ils s'appuient sur un codage de l'information sous forme de chaînes binaires de longueur fixe et d'un ensemble d'opérateurs génétiques : la sélection, la mutation, le croisement,... Un individu sous ce codage, appelé un chromosome, représente une configuration du problème[6].

  L'évaluation de cet individu consiste à transformer la chaîne 0/1 du chromosome en une valeur réelle, appelée valeur d'adaptation. La fonction d'évaluation dépend principalement de l'objectif du problème. Si cette fonction est uniquement basée sur la fonction objectif du problème, on utilisera le terme de *fonction de coût* pour la désigner.

  La valeur d'adaptation obtenue pour chaque chromosome lors de l'évaluation est utilisée, notamment lors des opérations de sélection, pour choisir les individus amener à se reproduire par des croisements ou à être utilisés par d'autres opérateurs génétiques.
  Le croisement permet de produire deux nouveaux individus, appelés les enfants, à partir de deux individus, appelés les parents.
  La mutation consiste à changer la valeur de certaines variables du chromosome. Les variables à modifier sont le plus souvent choisies aléatoirement.
  Ainsi, la nouvelle population construite à partir des opérateurs génétiques présentés devient la population de référence de la prochaine génération. Ce cycle d'opérations continue tant que la méthode n'a pas rencontré une condition d'arrêt définie préalablement, un nombre maximal de générations par exemple.

  Un mécanisme de sélection Pareto Un des principaux avantages des algorithmes évolutionnaires pour l'optimisation multiobjectif est qu'ils permettent non seulement la

mise en oeuvre d'approches non Pareto (agrégation des objectifs,...), mais aussi l'implementation d'approches Pareto. En effet, certains mécanismes de sélection implémentent la relation de dominance réalisant ainsi une sélection Pareto.

Une sélection Pareto utilise la relation de dominance pour affecter des rangs aux individus de la population, faisant apparaître la notion de front.[6]

L'élitisme L'élitisme permet de conserver les meilleurs individus dans les générations futures[6].

Maintenir la diversité

La diversité est une notion déjà importante lors de l'optimisation de problèmes à un objectif, elle devient prépondérante lorsque l'on traite de problèmes multiobjectifs. Maintenir un certain degré de diversité dans la population d'un algorithme révolutionnaire consiste à éviter que la population ne converge prématurément vers une petite zone de l'espace de recherche ou de l'espace des objectifs. En effet, s'il n'existe pas de mécanisme de contrôle de la diversité, les opérations de sélection vont privilégier trop vite certains individus meilleurs à cette étape de la recherche. Cette convergence prématurée a pour effet de limiter la recherche à un sous-ensemble plus restreint de l'espace de recherche, qui peut ne contenir aucune solution optimale. Dans le cas de problèmes multiobjectif, converger prématurément rend impossible la découverte de l'intégralité du front Pareto. En effet, les individus de la population se focaliseront sur une partie du front Pareto, et ne se répartiront pas sur la totalité de ce front Pareto[6].

Le"sharing"

Le sharing consiste à modifier la valeur de coût d'un individu (calculée uniquement à partir de la fonction objectif du problème). C'est cette nouvelle valeur qui sera utilisée comme valeur d'adaptation par l'opérateur de sélection[6].

La réinitialisation La réinitialisation est une technique largement utilisée par toutes les métaheuristiques, les algorithmes évolutionnaires n'échappant pas à la règle. Lors d'approches par population, elle consiste à réinitialiser un certain nombre d'individus de la population, par exemple de manière aléatoire. En introduisant de manière régulière certains individus générés aléatoirement, de nouvelles zones de l'espace de recherche, peut être inexplorées jusqu'ici, peuvent être découvertes[6].

Le "crowding" L'approche par "crowding" consiste à déterminer un représentant par niche découverte. À la différence du "sharing", où tous les individus sont susceptibles d'être sélectionnés et de participer aux phases de croisement, mutation et sélection, avec le "crowding", seuls les représentants participeront aux différentes étapes de l'algorithme[6].

Conclusion

Dans ce chapitre, nous avons fait une étude récapitulative est menée sur les méthodes de résolution des problèmes d'optimisation multiobjectif, tout en montrant la manière de définir un problème pareil, ses contraintes, ses objectifs, tout en respectant le concept de compromis et les frontières de Pareto. Nous avons parlé de méthodes utilisées dans le domaine de l'optimisation multiobjectif à savoir exactes, flous et métaheuristiques.

Chapitre 4

Programmation discrète

"Il est faux qu'en divisant un espace on puisse arriver à une partie indivisible, c'est-à-dire qui n'ait aucune étendue ..."

Blaise Pascal (Mathématicien, physicien et philosophe français 1623-1662)

Introduction

La programmation discrète est un cas spéciale de la programmation mixte, et la méthode de résolution des problèmes à variables mixtes n'est pas très différente avec celle des problèmes à variables entières. En effet, les deux problèmes ont presque le même processus de résolution, pour le cas entier :

1. relaxation,
2. résolution du problème relaxé,

3. vérification d'intégrité de la solution (si elle existe),
4. coupe d'intégrité si nécessaire,
5. jusqu'à se que toutes les solution soient entières.

Pour le cas mixte, il suffit d'intervenir au niveau de la troisième et de la cinquième étape, et d'appliquer la vérification et le critère d'arrêt juste pour les variable astreintes à être entières.

4.1 Programmation Linéaire mono-objectif en nombres entiers

La forme générale d'un problème de programmation linéaire mono-objectif est définie par :

$$(LP) \quad \begin{array}{ll} max & Z = Cx \\ \text{t.q} & x \in S \end{array}$$

Où $C \in R^{1\times n}$, $S = \{x \in R^n / Ax = b, x \geq 0\}$, $A \in^{m\times n}$ et $b \in R^{m\times 1}$.

Dans le cas où seulement quelques variables sont entières, on a alors un problème de programmation linéaire en variables mixtes s'écrit comme suit :

$$(MILP) \quad \begin{array}{ll} max & Z = (Cx + hy) \\ \text{t.q} & Ax + Gy = b \\ & x \geq 0 \\ & y \in \mathbb{N} \end{array} \tag{4.1}$$

$$(ILP) \quad \begin{array}{ll} max & Z = Cx \\ \text{t.q} & x \in D \end{array} \tag{4.2}$$

Où $\mathrm{D} = \mathrm{S} \cap \mathrm{Z}^n$, est l'ensemble des nombres entiers relatifs appartenant à S.

4.1.1 Méthodes de résolution d'un programme linéaire en nombres entiers

Les deux principales familles de méthodes actuellement connues pour résoudre les problèmes de programmation linéaire en nombres entiers sont les méthodes des coupes et les méthodes arborescentes.

Coupe fractionnaire de Gomory [27]

Soit à résoudre le programme linéaire en nombres entiers suivant :

$$(ILP) \quad \begin{array}{ll} max & Z = Cx \\ \text{t.q} & x \in D \end{array}$$

Dont le problème relaxé est : $(LP) Max\{Z = Cx \mid x \in S\}$.

Dans cette méthode, le programme linéaire (LP) est résolu en première étape en utilisant la méthode du simplexe. A l'optimum on a les conditions :

1. $\hat{c} \leq 0$
2. $\hat{b} \geq 0$

qui sont vérifiées. Si de plus :

3. $\hat{b}$ est entier, alors la solution optimale de (LP) est aussi optimale pour (ILP). Sinon, une ou plus des variables de base soumises à l'intégrité sont à valeurs fractionnaires. L'idée principale de l'algorithme de Gomory [27] est de maintenir les conditions (1) et (2) satisfaites et de rajouter des contraintes dites *coupes de Gomory* [27] une par une jusqu'à ce que la troisième condition soit vérifiée.

Définition 4.1 Étant donné un programme linéaire en nombres entiers (ILP), on dit que l'inéquation

$$\sum_{j=1}^{n} \alpha_j x_j \leq \beta$$

est valide si elle est satisfaite par tout point de D. Une coupe est une inéquation valide qui n'est pas satisfaite pour tout point de S.

Définition 4.2 Soit α un scalaire quelconque, on désigne par :

- $\lfloor \alpha \rfloor$ le plus grand entier inférieur ou égal à α
- $\lceil \alpha \rceil$ le plus petit entier supérieur ou égal à α
- $\langle \alpha \rangle = \alpha - \lfloor \alpha \rfloor$

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

Génération de la coupe de Gomory :

Soit le tableau optimal issu de la résolution de (LP) par la méthode du simplexe où la solution optimale est supposée non entière (voir tab4.1). Sans perte de généralités, on suppose qu'à l'optimum, il y a un total de (m + n) variables où m représente le nombre de variables de base notées x_i, $(i = 1, m)$ et n représente le nombre de variables hors base notées y_j , $(j = 1, n)$.

x_B	$\hat{A} = B^{-1}A$	$\hat{b} = B^{-1}b$
$-Z$	$\hat{c} = c - \pi A$	$z - c_B B^{-1} b$

TAB. 4.1 – Tableau optimal associé à la base B.

Où :

$\pi = c_B B^{-1}$: est dit vecteur multiplicateur relatif à la base B.

$\hat{c} = c - \pi A$: est dit vecteur coût réduit relatif à la base B, avec $\hat{c}_B = 0$.

A partir du tableau optimal, choisir une variable de base dont la valeur est fractionnaire, soit x_i. La i^{me} équation du tableau est donnée par :

$$x_i + \sum_{j=1}^{n} \hat{a}_{ij} y_j = \hat{b}_i \tag{4.3}$$

Où : $\hat{a}_{ij}$: est un élément de la matrice optimale des contraintes $\hat{A}$.

En décomposant chaque coefficient en la somme de sa partie entière et de sa partie fractionnaire, on obtient :

$$x_i + \sum_{j=1}^{n} \langle \hat{a}_{ij} \rangle y_j + \sum_{j=1}^{n} \lfloor \hat{a}_{ij} \rfloor y_j = \langle b_i \rangle + \lfloor \hat{b}_i \rfloor$$

Puisque $0 \leq \langle \hat{a}_{ij} \rangle < 1$, ceci implique :

$$x_i + \sum_{j=1}^{n} \lfloor \hat{a}_{ij} \rfloor y_j \leq \langle b_i \rangle + \lfloor \hat{b}_i \rfloor$$

Or le membre gauche est entier et $0 \leq \langle \hat{b}_i \rangle < 1$. Par conséquent cette dernière équation implique :

$$x_i + \sum_{j=1}^{n} \lfloor \hat{a}_{ij} \rfloor y_j \leq \lfloor \hat{b}_i \rfloor$$

et en soustrayant cette dernière inéquation de (4.3), on obtient :

$$\sum_{j=1}^{n} \langle \hat{a}_{ij} \rangle y_j \geq \langle \hat{b}_i \rangle \tag{4.4}$$

qui est bien valide. De plus (4.4) n'est pas satisfaite par la solution optimale de (LP), donc c'est bien une coupe. En ajoutant une variable d'écart x_s à (4.4), on obtient la coupe de Gomory définie par

$$x_s - \sum_{j=1}^{n} \langle \hat{a}_{ij} \rangle y_j = -\langle \hat{b}_i \rangle \tag{4.5}$$

Une fois la coupe générée, les coefficients de (4.5) sont insérés dans une nouvelle ligne du tableau (4.1).

Pour ce nouveau programme, la condition (1) est toujours satisfaite mais pas la condition (2). On effectue donc une ou plusieurs itérations de l'algorithme dual du simplexe, jusqu'à ce que la condition (2) soit satisfaite. Si $\hat{b}$ est entier, la solution courante est optimale pour (4.2), sinon

on recommence le processus. Après un nombre fini d'itérations, ou bien on obtient une solution optimale entière, ou bien le problème devient impossible.

Méthode Branch & Bound

La méthode Branch & Bound [33, 41] (par séparation et évaluation) est très efficace pour la résolution des problèmes de programmation linéaire en nombres entiers. Elle a été à l'origine par Land et Doig [34] pour résoudre un problème de programmation linéaire en nombres entiers.

Une approche simple pour résoudre un problème de programmation linéaire en nombres entiers est d'énumérer tous les points entiers réalisables du problème, d'évaluer la fonction objectif en chaque point et d'identifier celui qui a la meilleure valeur de fonction objectif. Bien qu'une recherche si approfondie dans l'espace des solutions réalisables soit simple de mettre en oeuvre, elle sera très coûteuse en terme de temps de calcul même pour des problèmes de taille réduite.

Son principe est de découper (branche) l'espace initial de recherche en domaines de plus en plus restreints afin d'isoler l'optimum global (principe de séparation). L'algorithme de recherche forme ainsi un arbre dont chaque nœud représente une partie de l'espace. Ensuite chaque nœud est évalué de façon à déterminer sa borne (bound) inférieure en fonction d'un critère d'évaluation. Si cette borne n'est pas meilleure que la solution courante alors la recherche sur ce nœud est stoppée, sinon la séparation continue.

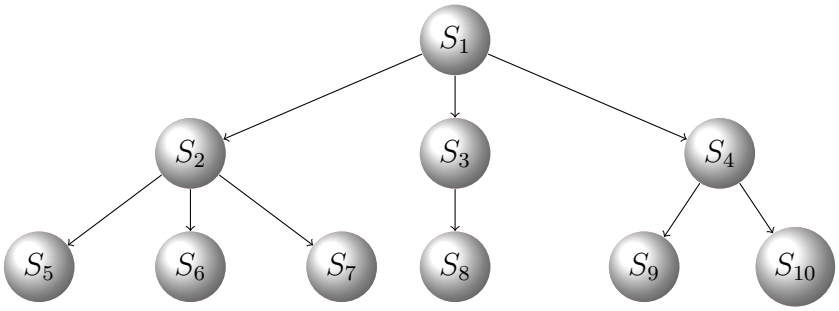

FIG. 4.1 – Principe de Branch and Bound

4.2 Programmation linéaire multiobjectif en nombres entiers

Un programme linéaire multiobjectif en nombres entiers est constitué d'un système de contraintes linéaires définissant un domaine discret de solutions réalisables, et d'un ensemble de fonctions linéaires à maximiser ou à minimiser définissant des objectifs conflictuels. Le problème consiste à déterminer toute solution réalisable entière telle qu'il n'existe aucune autre solution réalisable entière qui fournisse des valeurs au moins aussi bonnes que celles de chaque objectif et même meilleure pour au moins un objectif.

Un problème (MOILP) peut être formulé par :

$$(MOILP) \quad \begin{array}{ll} \text{"optimiser"} & Z_k = C_k x \\ \text{t.q} & x \in D \end{array} \tag{4.6}$$

Où $D{=}S{\cap}^n$ étant l'ensemble des nombres entiers relatifs, solutions de S.
$C \in \mathbb{R}^{1\times n}$, $S = \{x \in \mathbb{R}^n / Ax < b, x \geq 0\}$, $A \in^{m\times n}$ et $b \in \mathbb{R}^{m\times 1}$

4.2.1 Détection graphique de l'efficacité

Soit le problème de programmation linéaire multiobjectif en nombres entiers (MOILP). Pour tester l'efficacité en un point $x^* \in D$, Steuer [50] a introduit le concept d'ensemble dominant qui est principalement basé sur la notion du cône.

Définition 4.3 (cône) Soit le V $\in \mathbb{R}^n$, V $\neq \emptyset$, V est un Cône si et seulement si $\alpha v \in$ V pour tout scalaire $\alpha \geq 0$ et tout $v \in$ V.Le vecteur d'origine $0 \in \mathbb{R}^n$ est contenu dans chaque Cône.

Définition 4.4 (cône polaire) Soit V $\in \mathbb{R}^n$ un Cône. Le cône polaire non négatif de V (noté $V^{\geq}$) est le cône convexe $V^{\geq} = \{y \in \mathbb{R}^n \mid y^t v \geq 0 \forall v \in V\}$

C'est-à-dire, tous les vecteurs de $V^{\geq}$ font un angle inférieur ou égal à 90° avec chaque vecteur de V.

Définition 4.5 (cône semi-polaire positif) SoitV un cône convexe généré par $v^1, v^2, ..., v^P$. Alors, le cône semi-polaire positif (noté $V^{>}$) est le cône convexe $V^{>} = \{y \in \mathbb{R}^n \mid y^t v^i \geq 0$, pour tout i et $y^t v^i > 0$ pour au moins un $i\} \cup \{0_{\mathbb{R}_n}\}$

Définition 4.6 (Générateurs.) Considérons $v_1, v_2, ..., v_p$, un ensemble de p vecteurs de $\mathbb{R}^n$ et l'ensemble V tel que :
$V = \{v \in \mathbb{R}^n \mid v = \sum_{i=1}^{p} \alpha_i v^i, \alpha_i > 0\}$

V est l'ensemble de toutes les combinaisons linéaires à coefficients non négatifs des v^i, $i = 1, ..., p$ et est le cône convexe engendré par l'ensemble $v_1, v_2, ..., v_p$.

Définition 4.7 (Ensemble dominant.)
$x^* \in D$ et $C^{>}$ le cône polaire semi positif du cône C généré par les gradients des p fonctions objectif i.e. : $V^{>} = \{y \in \mathbb{R}^n \mid c^i y \geq 0$, pour tout i et $c^i y > 0$ pour au moins un $i\} \cup \{0_{\mathbb{R}_n}\}$
On définit l'ensemble de dominance de x^* noté ED_{x^*}, comme étant la somme des ensembles x^* et $C^{>}$:

$$ED_{x^*} = x^* \bigoplus C^{>}$$

C'est à dire :

$$ED_{x^*} = \{x \in \mathbb{R}^n \mid x = x^* + y, y \in C^{>}\}$$

L'ensemble dominant ED_{x^*} contient tous les points dont les vecteurs critères dominent le vecteur critère de $x^* \in D$. Notons que la somme des ensembles x^*et $C^{>}$ effectue une translation du cône polaire semi positif de l'origine vers le point en question. Le théorème suivant montre l'importance de cet ensemble dans la détection des solutions efficaces.

Théorme *11 ([50]) Soit ED_{x^*} l'ensemble dominant en $x^* \in D$. Alors x^* est efficace si et seulement si : $ED_{x^*} \cap D = x^*$.*

4.2.2 Exemple illustrant ces définitions

Considérons le programme linéaire bi-objectifs en nombres entiers suivant :

$$(MOILP)\begin{cases} \text{" max "} & (x_1, -x_1 + x_2) \\ \text{t.q} & x_1 + 2x_2 \leq 10 \\ & x_1 + x_2 \leq 6 \quad x_1, x_2 \in_+ \\ & x_1 \leq 4 \end{cases}$$

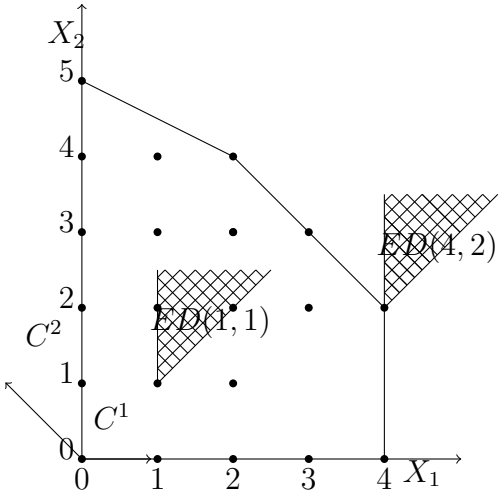

FIG. 4.2 – Espace des critères

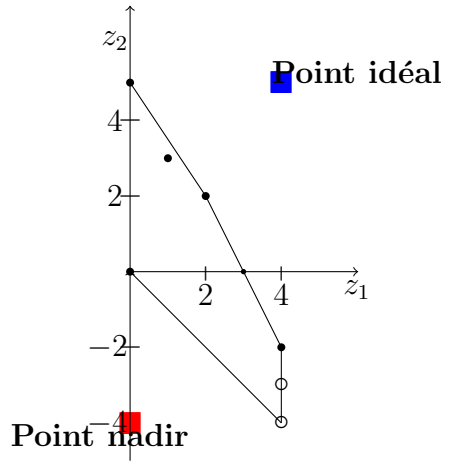

FIG. 4.3 – Espace des décisions

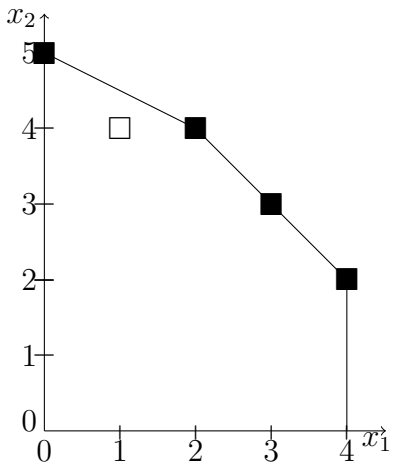

FIG. 4.4 – Solutions supportées et non supportées

- Les solutions efficaces du programme (P) sont $\{(4,2),(3,3),(2,4),(1,4),(0,5)\}$. Par exemple le point $x^{'} = (4,\ 2)$ est efficace car $ED_{x'} \cap ED = x^{'}$, tandis que le point $x^{*}= (1,\ 1)$ n'est pas efficace car $ED_{x^*} \cap ED \neq x^*$
- Deux solutions faiblement non dominées sont détectées, c'est les points (4,-4) et (4,-3).
- Les solutions efficaces supportées sont : (4, 2), (3, 3), (2, 4), (0, 5). Une solution efficace non supportée est détectée, c'est le point (1, 4).
- Le point idéal est le point : (4, 5).
- La matrice des gains ici est unique : $\begin{pmatrix} 4 & 0 \\ -4 & 5 \end{pmatrix}$
- Le point nadir est le point de coordonnées : η = (0,-4)

4.2.3 Quelques méthodes de résolution

Il existe plusieurs recherches dans ce domaine, citons en particulier, Steuer et Choo [50], Klein et Hannan [32], Crema et Sylva [15], Gupta et Malhotra [28], Abbas et Moulaï [2], Abbas et Chaabane [1], motivés par de nombreuses stratégies (applications), se sont intéressés à caractériser totalement ou partiellement l'ensemble des solutions efficaces du problème de programmation linéaire multiobjectif en nombres entiers. Dans cette partie, quelques méthodes de résolution des

problèmes MOILP sont exposées. Quelques-unes nécessitent la présence du décideur (méthodes interactives), et d'autres donnent l'ensemble des solutions efficaces sans intervention de ce dernier.

Nous présentons quelques unes, dans cette section.

Méthode de D. Klein & E. Hannan

La technique proposée par D. Klein & E. Hannan [32] peut être utilisée aussi bien pour identifier l'ensemble de toutes les solutions efficaces que pour en caractériser une partie seulement. Elle consiste à résoudre progressivement une séquence de programmes linéaires mono-objectif en nombres entiers avec des contraintes ajoutées à chaque étape. Les contraintes supplémentaires éliminent les solutions efficaces déjà trouvées, et font en sorte que les nouvelles solutions générées soient efficaces.

Algorithme

*É*tape1

Résoudre le problème (P_1) défini comme suit(l'indice i est pris arbitrairement dans $\{1, \ldots, p\}$) :

$$(P_1) : Max\{Z^i = c^i x : x \in D\}$$

Cas 1. Si la solution optimale de (P_1), soit x^1, est unique alors elle est efficace pour (P).

Cas 2. Sinon, déterminer toutes les solutions alternatives et à x^1 et par comparaison deux à deux des vecteurs critère associés, garder uniquement celles qui sont efficaces pour construire l'ensemble ESE(P_1) des solutions efficaces générées à l'étape 1.

*É*tape générale j

A l'étape j, c'est le problème (P_j) qui est résolu, il est définit comme suit :

$$(P_j)\begin{cases} max & Z_i = C^i x \\ \text{t.q} & x \in D \\ & \bigwedge_{k=1}^{r}(\bigvee_{s=1,s\neq i}^{p}(C^s x \geq C^i \tilde{x}_k + \epsilon^s)) \end{cases} \tag{4.7}$$

Avec $0 < \epsilon^s \leq 1$ et $\tilde{x}_k$ (k=1,...,r) les points efficaces obtenus aux étapes $1, ..., j-1$.

Si ESE(P_j) est l'ensemble des solutions efficaces obtenues à l'étape j et Y^j l'ensemble

des points efficaces accumulés à la fin de l'étape j, alors $Y^j = Y^{j-1} \bigcup ESE(Y^j)$ pour $j \geq 2$ avec $Y^1 = ESE(P_1)$.

Étape finale n

La procédure s'arrête lorsque le problème (P_n) est irréalisable.

Les auteurs R. Gupta et R. Malhorta[28] ont proposé deux procédures pour résoudre le problème (MOILP). La première est basée sur une méthode de coupe. Mais il s'est avéré qu'une erreur au niveau du second test d'arrêt empêche dans certains cas l'algorithme de donner tous les points efficaces du problème étudié. Un contre exemple a été présenté par M. Abbas et M. Moulaï [2] ainsi que par D. Chaabane. La deuxième est une variante de la méthode de Klein et Hannan[32].

M. Abbas et M. Moulaï ont proposé dans [2] une Méthode d'optimisation discrète linéaire Multicritère "MODILIM", pour la détermination de toutes les solutions efficaces du problème de programmation linéaire multiobjectif en nombres entiers. Elle peut être vue comme une alternative à celle de Gupta et Malhotra [28] (première procédure), où les auteurs ont proposé un autre test d'arrêt permettant à l'algorithme de fournir toutes les solutions efficaces. Elle est basée sur une technique de coupes planes et est décrite en plusieurs étapes :

1. L'approche est appropriée pour résoudre le problème de programmation linéaire entière à objectifs multiples.

2. Considérant une seule fonction objectif, la première itération est de déterminer une solution entière réalisable optimale de cette fonction objectif sous les contraintes du problème (MOILP).
3. La seconde itération nous permet de trouver toutes les solutions alternatives (alternatives) de la solution entière optimale obtenue ci-dessus, si elles existent, et déduire les premières solutions entières efficaces du problème principal.
4. On introduit une coupe plane pour dénombrer toutes les autres solutions efficaces restantes en utilisant la méthode duale du simplexe.
5. Pour l'étape finale, l'algorithme aura exploré toutes les solutions efficaces du problème étudié.

M. Abbas et D. Chaabane ont proposé dans [1] la méthode de détermination des Solutions Efficaces dans l'Espace des Variables Discrètes "SEEVD", cette méthode est une forme améliorée de la méthode de Gupta et Malhotra où le test d'arrêt est corrigé pour déterminer toutes les solutions efficaces du problème MOILP sans n'en manquer aucune.
Dans une première étape, une solution initiale est déterminée en résolvant un problème unicritère en nombre entier, cette solution constitue le premier élément de la liste des solutions efficaces, puis une séquence de coupes est appliquée après avoir exploré les arêtes incidentes à cette solution selon une direction précise définie par un ensemble d'indices hors-base, ceci nous permet de générer une nouvelle solution efficace ajoutée à la liste précédente de solutions efficaces et de réduire le domaine de recherche jusqu'à ce qu'il devient vide. Dans ce cas le processus s'arrêt avec une liste finale contient toutes les solutions efficaces du problème traité.

Méthode de A.Crema et J.Sylva

La méthode développée par Crema et Sylva [15] est une variante de celle de Klein et Hannan étudiée précédemment. Son principe repose sur la résolution d'une succession de programmes linéaires en nombres entiers optimisant à chaque étape une combinaison positive des critères. Un ensemble de contraintes est rajouté à chaque fois assurant la détection d'une nouvelle solution efficace. A la fin, la méthode fournit l'ensemble de toutes les solutions non dominées du problème de programmation linéaire discrète à objectifs multiples.

Proposition 4.1 [15] Soient $x^1, x^2, ..., x^l$ des solutions efficaces du problème 4.6. Posons $D_s = \{x \in^n | C^k x \leq C^k x^s, \forall s = 1, ..., l\}$. Si x^* solution efficace pour le problème :

$$(P_l) \begin{cases} \text{Max} & C^k x \\ \text{tq} & x \in (D \bigcup_{s=1}^{l} D_s) \end{cases} \tag{4.8}$$

alors x^* est une solution efficace pour le problème 4.6. De plus, si le problème 4.8 est irréalisable, alors $\{(C^k x^s)_{s=1}^{l}\}$ est l'ensemble de toutes les solutions non dominées du problème 4.6.

Proposition 4.2 [15] Soient $x^1, x^2, ..., x^l$ des solutions efficaces du problème 4.6. Posons $D_s = \{x \in Z^n | C^k x \leq C^k x^s, \forall s = 1, ..., l\}$. Si x^* est une solution optimale pour le problème mono objectif :

$$(P_l^{\lambda}) \begin{cases} \text{Max} & \sum_{k=1}^{p} \lambda_k C^k x \\ \text{tq} & x \in (D \bigcup_{s=1}^{l} D_s) \end{cases}$$

pour certaines valeurs du vecteur $\lambda \in \mathbb{R}^p$ $\lambda > 0$, alors x^* est une solution efficace pour le problème 4.6.

Théorme *12 ([26]) Si x^* une solution optimale du problème mono-objectif*

$$(P_\lambda) \begin{cases} \text{" max "} & \lambda^T C x \\ t.q & x \in S \end{cases}$$

pour un certain $\lambda \in \Lambda = \left\{ \lambda \in \mathbb{R}^p : \sum_{k=1}^{p} \lambda_k = 1, \lambda_k = 1 > 0 \right\}$ *alors,* x^* *est une solution efficace du problème multiobjectif* "max"$\{Z(x) : x \in S\}$.

Algorithme

Pour les problèmes de grande taille, la détermination de l'ensemble de toutes les solutions non dominées devient très coûteuse en terme de temps de calcul. Pour cela, une étape d'interaction avec le décideur peut être intégrée à la procédure. Cette étape a pour objectif d'éliminer des solutions efficaces que le décideur juge insatisfaisantes. Dans ce cas le problème (P_{l+1}) devient :

$$(P_{l+1}) \begin{cases} \text{Max} & \sum_{k=1}^{p} \lambda_k C^k x \\ \text{tq} & x \in D \\ & C^k x \geq (C^k x^s + f_k) y_k^s - M_k(1 - y_k^s) \\ & \sum_{k=1}^{p} y_k^s \geq 1 \qquad y_k^s \in \{0,1\} \; pour k = 1, ..., p, \; s = 1, ..., l \end{cases}$$

Où f_k représente l'amélioration minimale dans la $i^{ème}$ fonction objectif fixée par le décideur, $f_k > 1$ (entier).

Exemple

Pour illustrer la technique, nous considérons le MOLIP à deux fonctions objectifs suivant :

$$(P) \begin{cases} \text{Max} & Z_1 = x_1 - 2x_2 \\ \text{Max} & Z_2 = -x_1 + 3x_2 \\ \text{tq} & x_1 - 2x_2 \leq 0, \\ & x_1, x_2 \in \{0, 1, 2\}. \end{cases}$$

On pose $-M_1 = -4, -M_2 = -2$, les bornes inférieures des fonctions objectifs Z_1, $Z_2, f_k = 1, \forall k$ et $\lambda = (4, 3)$. La fonction scalarisante est donnée par : $Z_\lambda = x_1 + x_2$.

Soit :

$$(P_0) \begin{cases} \text{Max} & Z_1 = x_1 + x_2 \\ \text{tq} & x_1 - 2x_2 \leq 0, \\ & x_1, x_2 \in \{0, 1, 2\}. \end{cases}$$

Algorithme 4 Algorithme de Sylva & Crema

Début *Étape1*

Après avoir fixer le vecteur poids $\lambda = (\lambda_1, \lambda_2, ..., \lambda_p)$ à des valeurs strictement positives, la première étape de l'algorithme consiste en la résolution du problème :

$$(P_1) : Max\{\sum_{k=1}^{p} \lambda_k C^k x | x \in D\}$$

Deux cas se présentent :

Cas 1. Si (P_1) n'admet pas de solutions, alors 4.6 l'est aussi.

Cas 2. Sinon, une solution x^1 est trouvée et elle est efficace.

Ensuite, une suite de programmes linéaires en nombres entiers augmentés par certaines contraintes sont résolus progressivement.

Après k étapes du processus :

Cas 1. Si (P_l) est irréalisable, alors l'algorithme prend fin.

Cas 2. Sinon, une nouvelle solution efficace, soit x^l, est trouvée et le nouveau problème (P_{l+1}) est définit à partir de (P_l) en lui éliminant toutes les solutions vérifiant $C^k x \leq C^k x^l, \forall k = 1, ..., p$. Ceci peut être traduit par le rajout de contraintes suivantes :

$$\begin{cases} C^k x \geq (C^k x^l + 1) y_k^l - M_k(1 - y_k^l), & \text{k=1,...,p} \\ \sum_{k=1}^{p} y_k^l \geq 1,\ y_k^l \in \{0, 1\}, & \text{k=1,...,p} \end{cases}$$

Où $-M_k$ est la borne inférieure pour les valeurs réalisables de la $k^{ème}$ fonction objectif (Par exemple, dans le cas où la matrice des critères C est positive, M_k peut être fixée à 0 pour tout k).

Étape générale $(l+1)$**. Résoudre le problème** (P_{l+1})

$$(P_{l+1})\begin{cases} \text{Max} & \sum_{k=1}^{p}\lambda_k C^k x \\ \text{tq} & x \in D \\ & C^k x \geq (C^k x^s + 1)y_k^s - M_k(1-y_k^s) \\ & \sum_{k=1}^{p} y_k^s \geq 1 \qquad y_k^s \in \{0,1\}\ pour k = 1,...,p,\ s = 1,...,l \end{cases}$$

Étape finale n La procédure continue jusqu'à ce que le problème (P_n)devient irréalisable. A la fin, on obtient l'ensemble de toute les solutions efficaces ou seulement une partie qui intéresse le décideur.

fin

La résolution de problème (P_0) donne $x_1 = (2,2)$ de valeur optimale $Z_\lambda(P_0) = 4$. Donc, x_1 est une solution efficace du problème initial (p). Correspondant au vecteur non dominée (-2,4) (Voir figure 4.5).

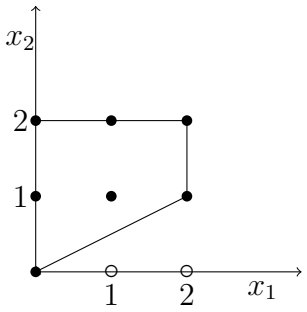

FIG. 4.5 – Espace des décision de P

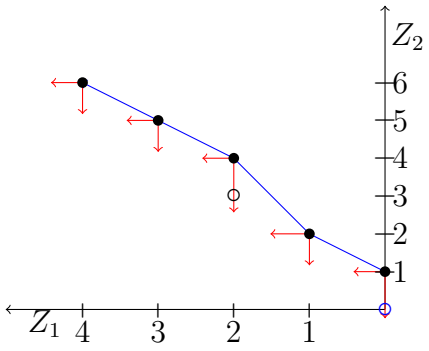

FIG. 4.6 – Espace des critères de P

Le deuxième problème à résoudre est :

$$(P_1)\begin{cases} \text{Max} & Z_1 = x_1 + x_2 \\ \text{tq} & x_1 - 2x_2 \geq -y_1^1 - 4(1 - y_1^1), \\ & -x_1 + 3x_2 \geq 5y_2^1 - 2(1 - y_2^1), \\ & y_1^1 + y_2^1 \geq 1, \\ & x_1, x_2 \in \{0, 1, 2\}, \\ & y_1^1, y_2^1 \in \{0, 1\}. \end{cases}$$

La résolution de (P_1) donne $x_1 = 2, x_2 = 1$, $y_1^1 = 1$, $y_2^1 = 0$, avec valeur optimale $Z_\lambda(P_1) = 3$. Donc, x_2=(2,1) est un autre point efficace correspondant au couple non-dominée (0,1), (4.5).

Formons le problème suivant (P_2) en ajoutant au domaine précédent D_1 de (P_1) cinq contraintes :

$$(P_2)\begin{cases} \text{Max} & Z_1 = x_1 + x_2 \\ \text{tq} & x \in D_1 \\ & x_1 - 2x_2 \geq y_1^2 - 4(1 - y_1^2), \\ & -x_1 + 3x_2 \geq 2y_2^2 - 2(1 - y_2^2), \\ & y_1^2 + y_2^2 \geq 1, \\ & y_1^2, y_2^2 \in \{0, 1\}. \end{cases}$$

Une solution optimale à ce problème est $x_1 = 1, x_2 = 2$, $y_1^1 = y_1^2 = 0$, $y_2^1 = y_2^2 = 1$ avec valeur optimale $Z_\lambda(P_2) = 2$. Cela correspond à un nouveau point efficace x_3=(1,2) avec un vecteur de la valeur de la fonction objectif égale à (-3,5).

Le pas suivant en ajoutant des contraintes qui effacent seulement le point efficace x^3, on

obtient le problème (P_3) :

$$(P_3)\begin{cases} \text{Max} & Z_1 = x_1 + x_2 \\ \text{tq} & x \in D_2 \\ & x_1 - 2x_2 \geq y_1^3 - 4(1 - y_1^3), \\ & -x_1 + 3x_2 \geq 2y_2^3 - 2(1 - y_2^3), \\ & y_1^3 + y_2^3 \geq 1, \\ & y_1^3, y_2^3 \in \{0, 1\}. \end{cases}$$

Une solution optimale au dernier problème est $x_1 = 0, x_2 = 2$, $y_1^1 = y_1^2 = y_1^3 = 0$, $y_2^1 = y_2^2 = y_2^3 = 1$ avec $Z_\lambda(P_3) = 2$. Cela correspond à une solution efficace x_3=(0,2) Avec une valeur de la fonction objective égale au vecteur à (-4,6).

Maintenant, problème (P_4) est défini :

$$(P_4)\begin{cases} \text{Max} & Z_1 = x_1 + x_2 \\ \text{tq} & x \in D_3 \\ & x_1 - 2x_2 \geq -3y_1^4 - 4(1 - y_1^4), \\ & -x_1 + 3x_2 \geq 7y_2^4 - 2(1 - y_2^4), \\ & y_1^4 + y_2^4 \geq 1, \\ & y_1^4, y_2^4 \in \{0, 1\}. \end{cases}$$

Ce problème a un seul point réalisable $x_1 = x_2 = 1$, $y_1^1 = y_1^3 = y_1^4 = 1$, $y_1^2 = 0$, $y_2^1 = y_2^3 = y_2^4 = 0$, $y_2^2 = 1$ avec $Z_\lambda(P_4) = 2$. Cela correspond à un nouveau point efficace x_4=(1,1) Avec une valeur de la fonction objective égale au vecteur à (-1,2),(4.6).

Comme ce n'est pas optimal dans D pour toute combinaison positive des fonctions objectives de

problème P. Le prochain problème à résoudre est le suivant :

$$(P_4)\begin{cases} \text{Max} & Z_1 = x_1 + x_2 \\ \text{tq} & x \in D_4 \\ & x_1 - 2x_2 \geq -4(1 - y_1^5), \\ & -x_1 + 3x_2 \geq 3y_2^5 - 2(1 - y_2^5), \\ & y_1^5 + y_2^5 \geq 1, \\ & y_1^5, y_2^5 \in \{0, 1\}. \end{cases}$$

Comme ce problème est irréalisable, le processus s'arrête, et nous avons l'ensemble complet de vecteurs objectifs non dominé aussi bien qu'un point efficace de la décision pour chacun de ceux de (4.2).

(x_1, x_2)	(z_1, z_2)
(2,2)	(-2,4)
(2,1)	(0,1)
(1,2)	(-3,5)
(0,2)	(-4,6)
(1,1)	(-1,2)

TAB. 4.2 – Les solutions de problème P

Méthode de Chergui, Moulaï & Ouaïl [9]

Cette méthode est basée sur le principe de "branch and cut", et celui de recherche d'un ensemble d'indices des coûts réduits, appelé H_l. En suite, à l'aide d'une coupe d'efficacité (définie comme la somme des variables hors base dont les indices appartiennent à H_l qui doit être supérieure à l'unité). Dans ce qui suit, une présentation plus détaillée de la méthode :

Soit le programme linéaire (P_l) défini comme suit :

$$P_0 = \begin{cases} \max Z_i = C_i x \quad i = \overline{1,k} \\ x \in \mathcal{S} \cap \mathbb{Z} \end{cases}$$

Avec $\mathbb{S} = \{Ax \leq b, x \geq 0\}, A \in \mathbb{Z}^{m \times n}, b \in \mathbb{Z}^m, C_i \in \mathbb{R}^{1,N}$.

On suppose que $\mathbb{X}$ est un polyèdre non vide et compact.

Définissons le problème (P_l) suivant :

$$P_l = \begin{cases} \max Z = \sum_{i=1}^{k} \lambda_i C_i x \\ x \in \mathcal{S}_l \end{cases}$$

Avec $\lambda_i \geq 0 \, \forall i = 1, \ldots, k$; $\mathcal{S}_0 = \mathcal{S}, \mathcal{X}_0 = \mathcal{X}$

À chaque étape qu'une solution entière x_l^* est obtenue. On note respectivement par : $\mathcal{B}_l$, $\mathcal{N}_l$ les ensembles d'indices des variables de base et celui des variables hors base de x_l^*.

$H_l = \{j \in \mathcal{N}_l / \exists i \in \{1, , k\}; \hat{c}_j^i > 0\} \cup \{j \in \mathcal{N}_l / \hat{c}_j^i = 0, \forall i \in \{1, \ldots, k\}\}$ où $\hat{c}_j^i$ est la $j^{\text{ème}}$ composante du vecteur coût réduit du critère Z_i. On définit les deux sous-ensembles de $\mathcal{S}_l$ comme suit :

$$\mathcal{S}_{l+1} = \{x \in \mathcal{S}_l / \sum_{j \in H_l} x_j \geq 1\}$$

$$T_{l+1} = \{x \in \mathcal{S}_l / \sum_{j \in \mathcal{N}_l / H_l} x_j \geq 1\}$$

•Principe de la méthode :

La où la plupart des méthodes de la littérature commencent avec une solution optimale d'un programme linéaire en nombres entiers, cette méthode a l'avantage de commencer avec une solution optimale d'un programme linéaire dont l'objectif est une combinaison linéaire positive des critères, et emploie un procédé arborescent pour générer l'ensemble des solutions admissibles entières.

Lorsqu'une telle solution est trouvée, une coupe est construite de manière à supprimer certaines des solutions non-efficaces, sans les calculer. La méthode est basée sur le concept de la recherche arborescente en programmation linéaire. À chaque nœud, nous avons à résoudre un programme linéaire (P_l). Le nœud l de l'arbre est stérilisé si le programme correspondant (P_l)

n'est pas réalisable ou si $H_l = \varnothing$. Si la solution optimale x du programme (P_l) n'est pas entière, soit x_j une coordonnée de telle que $x_j = \alpha_j$ où α_j est un nombre fractionnaire. Le nœud l est séparé en deux nœuds avec les contraintes supplémentaires : $x_j \leq \lfloor \alpha_j^* \rfloor$ et $x_j \geq \lfloor \alpha_j^* \rfloor + 1$, respectivement.

Le cas correspondant à une solution entière x est résolu en utilisant les directions croissantes des critères afin d'éviter d'explorer les régions non-efficaces de l'espace des solutions réalisables. Seule la partie du domaine des solutions réalisables dans laquelle au moins l'un des objectifs du problème peut être amélioré est traitée. Ceci est rendu possible par l'ajout de la contrainte valide (et efficace) suivante :

$$\sum_{j \in H_l} x_j \geq 1$$

Algorithme 5 Algorithme de Chergui Moulaï Ouaïl

Étape 1.(Initialisation) $\mathcal{S}_0 := \mathcal{S}, l := 0$ *et* $\boldsymbol{Eff(P_0)} := \varnothing$

Soit $\lambda_i \geq 0$ pour tout $i \in \{1, \ldots, k\}$, avec au moins une inégalité stricte, résoudre le programme linéaire (P_0) au nœud 0 et soit x sa solution optimale. Si x n'est pas entière, passer à **l'étape 2a**, sinon passer à **l'étape 2b**.

Étape 2. (étape général) Tant qu'il y a un nœud non stérilisé dans l'arbre, faire :

Choisir le premier nœud l créé dans l'arbre, pas encore stérilisé et résoudre le programme linéaire correspondant (P_l). Si le programme (P_l) n'a pas de solutions réalisables, alors il est stérilisé. Sinon, soit x sa solution optimale. Si x n'est pas entière, passer à **l'étape 2a**. Sinon, passer à **l'étape 2b**.

Étape 2a. Choisir l'une des coordonnées x_j de x telle que $x_j := \alpha_j$, avec α_j fractionnaire, et séparer le nœud l actuel en deux nœuds : ajouter la contrainte $x_j \leq \lfloor \alpha_j^* \rfloor$ au premier nœud, et la contrainte $x_j \geq \lfloor \alpha_j^* \rfloor + 1$ au deuxième nœud et passer à **l'étape 2.**

Étape 2b. Si Cx n'est pas dominé par Cy, pour toute solution $y \in \boldsymbol{Eff(P_l)}$, alors $\boldsymbol{Eff(P_l)} := \boldsymbol{Eff(P_l)} \cup \{x\}$. S'il existe $y \in \boldsymbol{Eff(P_l)}$ telle que Cy est dominé par Cx, alors $\boldsymbol{Eff(P_l)} := \boldsymbol{Eff(P_l)}/\{y\} \cup \{x\}$.

Déterminer les ensembles $\mathcal{B}_l$, $\mathcal{N}_l$ et H_l, Si $H_l = \varnothing$, alors le nœud l correspondant est stérilisé, passer à **l'étape 2.** Sinon, ajouter la contrainte $\sum_{j \in H_l} x_j \geq 1$ pour obtenir l'ensemble $\mathcal{S}_{l+1}$, résoudre le programme correspondant en utilisant la méthode duale du simplexe et soit $x$la solution optimale obtenue, si x est une solution entière, passer à **l'étape 2b**. Sinon, passer à **l'étape 2a.**

Conclusion

Comme on la vue dans ce chapitre, des méthodes de programmation mono et multi objectifs en variables entières sont définies. La programmation en variables mixtes n'est pas si différentes avec la programmation discrète, il suffit juste de bien savoir fixer les conditions d'arrêt de notre algorithme ainsi que le processus de branchement.

Chapitre 5

Application à l'optimisation des portefeuilles

"Le fondement de la théorie c'est la pratique".

Mao Tsé-Toung (politicien chinois, 1893-1976.)

Introduction

Un problème permanent dans la finance est "comment combiner les investissements pour former un portefeuille ?" répondre à cette question, serait faire de la "Sélection de portefeuille". En 1952, Harry Markowitz a développé sa formulation de la moyenne-variance, puis en 1956, il a introduit un algorithme pour trouver la "frontière efficiente", et s'est basé sur la selection d'un portefeuille optimal sur la frontière efficiente. Il y a aussi l'algorithme de Roy (1952), pour le

calcul direct d'un premier portefeuille sûr.

La gestion d'actifs financiers, aussi appelée gestion de portefeuille ou asset management, consiste à gérer les capitaux confiés en respectant les contraintes réglementaires et contractuelles en appliquant les politiques d'investissement définies en interne, pour en tirer le meilleur rendement possible en fonction du risque choisi.

Optimiser ces portefeuilles gérés, revient à minimiser le risque au lieu de le choisir auparavant, toute en maximisant le rendement. C'est pour ça que la meilleure modélisation de ce genre de problème est multiobjectif.

5.1 Définitions

Définition 5.1 (Rendement) Le rendement d'un portefeuille est une combinaison linéaire des actifs qui le composent, pondérés par leur poids w_i.

Définition 5.2 (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.

5.2 La théorie de Harry Markowitz

Markowitz (économiste américain, prix Nobel d'économie en 1990) a eu l'idée de mesurer la rentabilité d'un portefeuille par l'espérance de rendement et le risque par sa variance. Il postule

dans sa "théorie de portefeuille" que lorsque l'on cherche à répartir son épargne entre différents placements ou types de placements, deux, et deux seulement, critères de choix des supports d'investissements qu'il faut revoir, sont la rentabilité espérée du placement, et le risque associé, mesuré par la variance (ou écart-type) de cette rentabilité. A partir de ces deux critères, ce qui est appelé le modèle de Markowitz fournit la répartition dite optimale des actifs donc "le portefeuille". Les deux étapes du raisonnement sont :

1. les préférences des individus admettent une représentation dans le plan espérance-variance,
2. le modèle de Markowitz permet de construire l'ensemble des portefeuilles optimaux selon ces préférences.

Le choix de ces deux critères n'a rien d'évident comme le reconnaît Markowitz lui-même et repose sur des hypothèses assez restrictives. Malgré son fondement assez fragile, l'analyse de Markowitz a connu un grand succès car elle est intuitivement compréhensible, techniquement réalisable et donne lieu à une profusion d'application en finance de marché et en théorie financière.

But du modèle de Markowitz

Le modèle de Markowitz vise à l'aide d'une méthode mathématique à construire un portefeuille assurant :

- soit le meilleur rendement à risque donné,
- soit le plus petit risque à rendement donné.

Notations

Considérons un investisseur désirant répartir une somme initial $W(0)$ entre un actif financier (actions, obligations, ..etc). Nous supposons ses préférences représentées par l'espérance d'une fonction d'utilité u croissante, strictement concave et differentiable sur $\mathbb{R}$.

Chaque portefeuille est modélisé par :

- $W(t)$: la valeur du portefeuille à l'instant t,

– $x = \begin{pmatrix} x_1 \\ \vdots \\ \vdots \\ x_n \end{pmatrix}$, où x_i représente la part de l'investissement initial $W(0)$ affecté au $i^{ème}$ actif.

Chaque actif du portefeuille peut être acquis à la date $t = 0$ en quantité illimitée au prix unitaire $V_i(0)$. Chaque actif est caractérisé par deux variables aléatoires $V_i(t)$ et $R_i(t)$ définies sur l'ensemble $\Omega(t)$ des mondes possibles la date t telles que :

– $V_i(t)$ est la valeur du $i^{ème}$ actif à l'instant t,

– $R_i(t)$ est la rentabilité du $i^{ème}$ actif à l'instant t avec :

  – $V_i(t) = (1 + R_i(t)V_i(0))$ et donc

  – $R_i(t) = -1 + \frac{V_i(t)}{V_i(0)}$.

Principe du modèle de Markowitz

Entre deux portefeuilles par leurs rendements (supposés aléatoires), on retient :

– *à risque identique celui qui a l'espérance de rendement la plus élevée,*

– *à espérance de rendement identique, celui qui présente le risque le plus faible.*

Ce principe conduit à éliminer un certain nombre de portefeuilles, moins efficients que d'autres.

La courbe qui relie l'ensemble des portefeuilles efficients s'appelle la *frontière efficiente.*

En dessous de cette courbe, tous les portefeuilles rejetés sont dits dominés.

Définition 5.3 (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 dominés". 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.

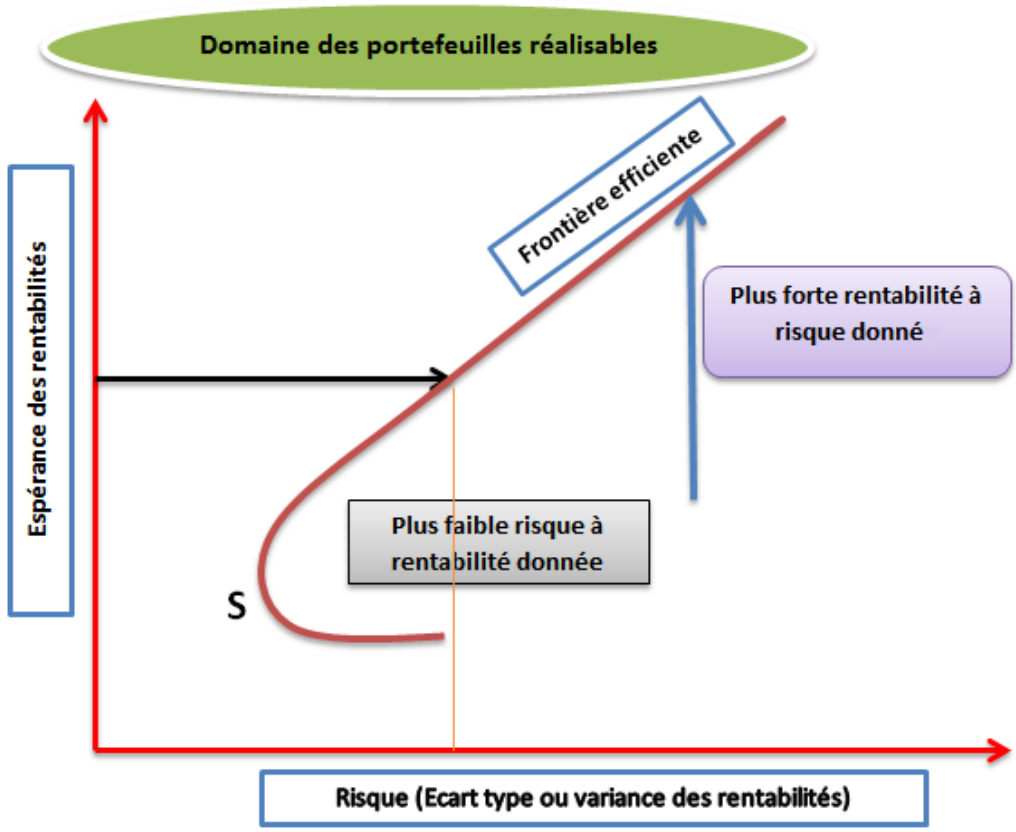

FIG. 5.1 – Frontières efficaces

Sélection d'un portefeuille situé sur la frontière efficiente

Soit R_p le rendement du portefeuille composé de n actifs caractérisés par leur rendement respectif $R_1, R_2, .., R_n$. On suppose, en outre, que chaque actif i entre pour une proportion X_i dans la composition du portefeuille P.

En d'autre termes : $R_p = \sum_{i=1}^{n} X_i R_i$.

Dès lors :

$$\begin{aligned} E(R_p) &= E(\sum_{i=1}^{n} X_i R_i) = \sum_{i=1}^{n} X_i E(R_i) \\ V(R_p) &= \sum_{i=1}^{n} \sum_{j=1}^{n} X_i X_j cov(X_i, X_j) \end{aligned}$$

Nous pouvons également écrire cette dernière relation sous forme matricielle si nous notons X le vecteur des parts d'actifs et X^T le même vecteur transposé :

$$X = \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{pmatrix} \quad X^T = \begin{pmatrix} X_1 & X_2 & \cdots & X_n \end{pmatrix}$$

et c_{ij}, la matrice des covariance :

$$c_{i,j} = \begin{bmatrix} cov(R_1, R_1) & cov(R_1, R_2) & \cdots & cov(R_1, R_n) \\ cov(R_2, R_1) & cov(R_2, R_2) & \cdots & cov(R_2, R_n) \\ \vdots & \vdots & \ddots & \vdots \\ cov(R_n, R_1) & cov(R_n, R_2) & \cdots & cov(R_n, R_n) \end{bmatrix}$$

matrice qui se simplifie directement en :

$$c_{i,j} = \begin{bmatrix} \sigma_1^2 & cov(R_1, R_2) & \cdots & cov(R_1, R_n) \\ cov(R_2, R_1) & \sigma_2^2 & \cdots & cov(R_2, R_n) \\ \vdots & \vdots & \ddots & \vdots \\ cov(R_n, R_1) & cov(R_n, R_2) & \cdots & \sigma_n^2 \end{bmatrix}$$

nous obtenons finalement la relation de la variance sous forme matricielle condensée,

$$V(R_p) = X^T(c_{i,j}.X).$$

Pour revenir à la forme algébrique du modèle, puisque la covariance est symétrique :

$$cov(R_i, R_j) = cov(R_j, R_i)$$

et que :

$$cov(R_j, R_i) = V(R_j)$$

Nous pouvons simplifier et écrire la variance sous la forme algébrique suivante :

$$V(R_p) = \sum_{i=1}^{n}\sum_{j=1}^{n} X_i X_j cov(R_i, R_j)$$

Sélectionner un portefeuille revient à choisir celui tel que :

- $E(R_p)$ soit maximal,
- $V(R_p)$ soit minimal,
- sous la contrainte que $\sum_{i=1}^{n} X_i = 1$.

Il s'agit donc d'un problème de maximisation d'une fonction économique sous contrainte.

Soit Z cette fonction économique.

$Z = \Phi E(R_i) - V(R_p)$ qui doit être maximiser sous la contrainte que $\sum_{i=1}^{n} X = 1$.

Où Φ est un paramètre qui représente le degré d'aversion au risque des investisseurs. Autrement dit, il s'agit du taux marginal de substitution du rendement et du risque qui exprime dans quelle mesure l'investisseur est d'accord pour supporter un risque accru en contrepartie d'un accroissement de son espérance de rendement.

En utilisant le lagrangien de cette expression, le problème de maximisation sous contrainte consiste à déterminer le maximum de la fonction Z définie par :

$$Z = \Phi \sum_{i=1}^{n} X_i E(R_i) - \sum_{i=1}^{n}\sum_{j=1}^{n} X_i X_j cov(R_i, R_j) + \lambda \left(\ 1 - \textstyle\sum_{i=1}^{n} X_i \ \right).$$

Cette fonction de $n+1$ variables $(X_1, X_2, ..., X_n, \lambda)$ est maximisée si sa dérivée partielle par rapport à chacune de ces variables est nulle, ce qui revient à poser le système suivant :

$$\begin{cases} \frac{\partial Z}{\partial X_1} = \Phi E(R_1) - 2X_1 cov(R_1, R_1) - 2X_2 cov(R_1, R_2) - ... - 2X_n cov(R_1, R_n) - \lambda = 0 \\ \frac{\partial Z}{\partial X_2} = \Phi E(R_2) - 2X_1 cov(R_2, R_1) - 2X_2 cov(R_2, R_2) - ... - 2X_n cov(R_2, R_n) - \lambda = 0 \\ \vdots \\ \frac{\partial Z}{\partial X_n} = \Phi E(R_n) - 2X_1 cov(R_n, R_1) - 2X_2 cov(R_n, R_2) - ... - 2X_n cov(R_n, R_n) - \lambda = 0 \\ \frac{\partial Z}{\partial \lambda} = 1 - X_1 - X_2 - .... - X_n = 0 \end{cases}$$

On peut alors écrire :

$$\begin{cases} 2X_1\sigma_{11} + 2X_2\sigma_{12} + ... + 2X_n\sigma_{1n} + \lambda = \Phi E(R_1) \\ 2X_2\sigma_{21} + 2X_2\sigma_{22} + ... + 2X_n\sigma_{2n} + \lambda = \Phi E(R_2) \\ \vdots \\ 2X_n\sigma_{n1} + 2X_2\sigma_{n2} + ... + 2X_n\sigma_{nn} + \lambda = \Phi E(R_n) \\ X_1 + X_2 + ... + X_n = 1 \end{cases}$$

L'écriture matricielle sera alors :

$$\begin{pmatrix} 2\sigma_{11} & 2\sigma_{12} & \cdots & 2\sigma_{1n} & 1 \\ 2\sigma_{21} & 2\sigma_{22} & \cdots & 2\sigma_{2n} & 1 \\ \cdots & \vdots & \ddots & \vdots & \vdots \\ 2\sigma_{n1} & 2\sigma_{n2} & \ldots & 2\sigma_{nn} & 1 \\ 1 & 1 & \cdots & 1 & 0 \end{pmatrix} \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \\ \lambda \end{pmatrix} = \begin{pmatrix} \Phi E(R_1) \\ \Phi E(R_2) \\ \vdots \\ \Phi E(R_n) \\ 1 \end{pmatrix}.$$

Alors,

$$A = \begin{pmatrix} 2\sigma_{11} & 2\sigma_{12} & & 2\sigma_{1n} & 1 \\ 2\sigma_{11} & 2\sigma_{12} & & 2\sigma_{1n} & 1 \\ \vdots & \ldots & \ddots & \ldots & \vdots \\ 2\sigma_{11} & 2\sigma_{12} & & 2\sigma_{1n} & 1 \\ 1 & 1 & \cdots & 1 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} \Phi E(R_1) \\ \Phi E(R_2) \\ \vdots \\ \Phi E(R_n) \\ 1 \end{pmatrix}, \quad X = \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \\ \lambda \end{pmatrix}$$

Dans ce cas, le système d'équations à résoudre peut se résumer sous la forme $A.X = B$.

Par conséquent : $X = A^{-1}B$

La détermination du poids de chacun des n actifs susceptibles d'enter dans la composition d'un portefeuille passe donc par l'inversion d'une matrice carrée de $n+1$ lignes et $n+1$ colonnes.

Après une description rapide du modèle et des hypothèses de Markowitz, nous allons nous appuyer sur ce dernier pour introduire notre modèle qui va prendre les deux objectifs en considération, en effet notre modèle sera un modèle multiobjectif.

Modélisation mathématique

Il s'agit maintenant d'obtenir des proportions X_i du portefeuille obligataire, tout en optimisant soit le risque ou bien le rendement, suivant le modèle de Markowitz.

Cas (1) : 0n choisit d'optimiser (Minimiser) le risque, on aura pour contraintes en plus de celles qui existent déjà, la contrainte de rendement.

$$\begin{cases} \min Var(Rnd) = \sum_{i=1}^{N}\sum_{i=1}^{N} x_i x_j \sigma_{ij} \\ \sum_{i=1}^{N} x_i R_i \geq R^*_{fix} \\ \sum_{i=1}^{us} x_i \geq 50 \\ \sum_{i=1}^{ge} x_i \geq 30 \\ \sum_{i=1}^{uk} x_i \leq 10 \\ \sum_{i=1}^{N} x_i = 100 \\ 0 \leq x_i \leq 1, \quad i = \overline{1,N} \\ x_j \in Z, \forall j \in J. \end{cases} \qquad .....(1)$$

Cas 2 : On choisit maintenant, de maximiser le rendement en ayant le risque comme contrainte :

$$\begin{cases} \max E(Rnd) = \sum_{i=1}^{N} x_i E(R_i) \\ \sum_{i=1}^{N}\sum_{j=1}^{N} x_i x_j \sigma_{ij} \leq V^*_{fix} \\ \sum_{i=1}^{us} x_i \geq 50 \\ \sum_{i=1}^{ge} x_i \geq 30 \\ \sum_{i=1}^{uk} x_i \leq 10 \\ \sum_{i=1}^{N} x_i = 100 \\ 0 \leq x_i \leq 1, \quad i = \overline{1,N} \\ x_j \in Z, \forall j \in J. \end{cases} \qquad .....(2)$$

Avec,

- x_i : est la proportion de portefeuille investi dans le titre i, i=1,...,N,
- σ_{ij} : la covariance entre le titre i et le titre j,

- R^*_{fix} : un rendement du portefeuille fixé,
- V^*_{fix} : un risque du portefeuille fixé,
- us, ge, uk : sont respectivement le nombre total d'obligation du marché respectivement américain, allemand et anglais.

Pour le modèle (1), la fonction objectif traduit la minimisation du risque qui est mesuré par la variance du rendement. Quand aux contraintes, le rendement du portefeuille doit être supérieur ou égal à certain rendement minimum R^*_fix fixé, on doit aussi respecter les contraintes du volume du portefeuille comme les obligations américaines, allemandes et anglaises doivent respecter une certaine proportion et les contraintes du portefeuille (La somme des proportions du portefeuille investie dans les différents titres doit être égale à 1 et ces proportions entre 0 et 1).

Pour le modèle (2), la fonction objectif, traduit la maximisation du rendement, qui est mesuré par l'espérance du rendement, le risque du portefeuille doit être inférieur à un certain risque maximum V^*_{fix} fixé, on doit aussi respecter les contraintes du volume du portefeuille et les contraintes du portefeuille comme dans le modèle (1).

Les modèle (1) et (2) sont une projection du modèle de Markowitz de notre problème, en effet selon ce modèle moyenne-variance, tout investisseur poursuit deux objectifs conflictuels qui sont la maximisation du rendement espéré et la maximisation du risque mesuré par la variance du rendement.

Pour obtenir une solution, on doit tenir compte des deux modèles à la fois et avoir un rendement minimum et un risque maximum à ne pas dépasser.

En plus du risque et du rendement on pourrai parler de dividendes, de liquidité, des responsabilités sociales, de l'inflation...etc. On constate ainsi que l'optimisation multicritère n'ai plus une option mais bien une obligation pour une bonne modélisation économique.

5.3 Introduction d'un modèle multiobjectif

Les deux derniers modèles traités, ne peuvent pas trouver la meilleure solution ou les meilleures solutions de notre portefeuille. En effet, si l'on modélise notre problème comme un problème mono objectif on risque d'avoir deux défauts :

1. La borne de l'espérance ou de la variance est trop large : c'est a dire que nous pouvons faire mieux. Il y a une solution qui maximise (resp. minimise) mieux, l'espérance (resp. la variance).
2. La borne de l'espérance ou de la variance est trop restreinte : on risque de ne pas avoir de solution réalisable.

Donc, dans ce qui suit une proposition d'un modèle multiobjectif pour résoudre les problèmes d'optimisation de portefeuille :

$$(PQ_1) : \begin{cases} \text{"}min\text{"} & Z_2 = r'x \\ \text{"}min\text{"} & Z_1 = x^T \sigma x \\ \text{t.q} & Ax \geq b, \\ & Aeqx = beq, \\ & x \geq 0, \\ & x_j \in N, \forall j \in J. \end{cases} \tag{5.1}$$

Tel que σ est une matrice symétrique, qui représente la variance. $r \in \mathbb{R}^n$ représente le rendement. Quand aux contraintes linéaires $Aeqx \geq beq$, elles peuvent représenter la satisfaction des demandes. $x_j \in N, \forall j \in J.$ cette contrainte est pour le cas où l'investisseur aura à choisir des lots discrets d'actions dans un stock donné.

5.4 Méthode exacte de résolution

On propose ici une nouvelle approche, qui est une combinaison de la méthode du simplexe[1] on ayant comme fonction objectif la fonction de rendement à maximiser (qui est linéaire), avec la méthode [9]. présentée dans la section 4.2.3 pour trouver l'ensemble les solutions non dominés, avec une stratégie modifiée de séparation-évaluation en utilisant l'algorithme de Branch and cut sélectif[2].

5.4.1 Description de la méthode

La méthode se base sur le principe de Dantzig [25], et de recherche de direction de descente (cas de minimisation), et de faire une coupe pour suivre cette direction, cela nous y assuré par les coupes de Chergui. Pour le cas non linéaire, on a introduit ici un moyen d'adaptation.

5.4.2 Notations

B_l, ensemble d'indices basiques de la solution optimale x_l^*.

N_l, ensemble d'indices hors-base de la solution optimal x_l^*.

$\mathrm{H}_l = \{j \in N_l \ tq. \ \exists q \in \{1,2\}; C^{jql}_{réduit} < 0\}$, tel que $C^{jql}_{réduit}$ est la j^{eme} composante du coût réduit[3] de la q^{eme} fonction objectif à la l^{eme} itération.

Où $C^{jql}_{réduit}$=

$c'^1_{jl} - Z'^1_{jl}$ est la j^{eme} composante du coût réduit du vecteur de la fonction objectif Z'^1 (de la fonction linéaire, récupérés dans le dernier tableau du simplexe) ;

$\nabla f^l_j(x_N) - \nabla f(x_B) A_B^{-1} A_j$ est le j^{eme} coût réduit de la fonction quadratique, x_B, x_N représentent respectivement les variables de base et les variables hors base. A_j est la j^{eme} colonne de la matrice des contraintes A. Voir 1.8.4, pour plus de détails.

[1] Si on prend comme fonction objectif le risque, on choisira de la résoudre avec le simplexe de Dantzig-Wolfe [17, 52, 53].

[2] Le test d'intégrité et d'arrêt est restreint aux variables appartenant à J.

[3] s'il est négatif, on améliore la fonction objectif dans le sens de la maximisation

Soit le problème P_l suivant :

$$P_l : \begin{cases} r^T x \\ x \in S_l, \end{cases}$$

tel que $S_0 = \{x \in R^n \backslash Ax = b \; et \; x \geq 0\}$.

5.5 Algorithme de la méthode

Algorithme 6 Algorithme de résolution de problèmes quad-lin mixtes

Étape 1 : Initialisation $Eff = \emptyset$;

Étape 2 : Résoudre le problème relaxé P_l

Étape 3 : Branchement ☞ **Si** le problème relaxé n'a pas de solution : Stop.

☞ **Sinon**

➴ Si il existe une composante $x_i \notin N, i \in I$,

➴ Séparer le problème P_l en deux sous problèmes,

✂ $S_{l+1} = \{S_l \cap \{x_i \leq \lfloor x_i \}\}$, $l = l + 1$;

✂ $S_{l+1} = \{S_{l-1} \cap \{x_i \geq \lceil x_i \rceil\}\}$,

✂ Aller à l'étape 2.

➴ Sinon Aller à l'étape 4.

Étape 4 : Test d'efficacité et Mise à jour de Eff Test d'efficacité :

✍ Si la solution trouvée x^l est non dominée par aucune solution de Eff,

✍ $Eff = Eff \cup x^l$;

✍ si x^l domine une solution $x^{dominé}$ dans l'ensemble Eff

✍ $Eff = Eff \setminus x^{dominé}$

Étape 5 : Coupe d'efficacité ☞ Déterminer l'ensemble H_l,

☞ introduire la coupe d'efficacité $\sum_{i \in H_l} x_i \geq 1$,

☞ Aller à l'étape 2.

L'algorithme s'arrête s'il n'existe pas de problème réalisable parmi les problèmes générés ou si $H_l = \emptyset$

5.6 Implémentation de la méthode

5.6.1 Choix du langage

Le choix s'est porté sur l'emploi du langage du logiciel Matlab 2010a, car il répond aux critères suivants :

- Les performances du langage : concernent la taille du code généré (le langage le plus apprécié est celui qui généré des exécutables optimisés) et l'exploitation efficace des ressources (logique ou physique).
- La maniabilité du langage : constitué d'un ensemble de possibilités faisant de telle 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).

5.6.2 Généralités sur le langage

Matlab est un logiciel parfaitement dédié à la résolution de problèmes d'analyse numérique ou de traitement du signal qui 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 resembler 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 interactivement ces données matricielles, le rendant ainsi particulièrement 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.

5.6.3 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 fenêtre 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 ce fait par la syntaxe suivante : function[s1, s2,...]=nom-fonction(e1, e2,...). Tel que 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.

5.6.4 Implémentation

Pour les besoins de la programmation de la méthode, on a utilisé et programmé les sous fonctions suivantes :

- ✍ linprog : Qui est une modification de celle qui existe déjà sous matlab de telle sorte de récupérer les variables de base de la solution optimale.
- ✍ branch and bound, mixte : C'est comme la méthode classique, mais on restreint la vérification de l'intégrité de la solution aux variables entières seulement.
- ✍ test d'efficacité : Pour tester l'efficacité d'une solution, on a utiliser la définition.
- ✍ calcul du coût réduit : Un petit programme nous permettant de récupérer le coût réduit de la fonction linéaire et le gradient réduit de la fonction quadratique par rapport aux variable hors base de de la solution optimale.
- ✍ Un générateur de problèmes test suivant une loi uniforme entre deux valeurs fixées par l'utilisateur.

5.6.5 Résultats

Le tableau suivant est un résumé de résultats trouvés on exécutant le programme sur une machine ayant les caractéristiques suivantes :

CUP Core 2 Duo, 2. GHz,

RAM 3 GO,

cela 5 fois pour chaque valeur de m : nombre lignes de la matrice des contraintes, n : dimension des variables de décision, n_0 : cardinale de J qui représente l'ensemble des indices des variables entières. On désigne aussi pour chaque valeur de m, n, n_0 les notations suivantes : '061

"nbr Moyen de sol eff" : nombre moyen des solutions efficaces trouvées,

"nbr max de sol eff" : nombre maximal de solutions efficaces trouvées,

"Mean CPU time" : Le temps moyen de la durée de l'exécution,

"Max CPU time" : Le temps maximal de la durée de l'exécution.

m	n	n_0	nbr Moyen de sol eff	nbr max de sol eff	Mean CPU time	Max CPU time
5	10	2	3.5	10	2.05849	3.21579
	20	5	3	11	2.14081	2.96114
	30	10	5.25	10	3.73760	4.19708
	50	10	4.5	8	4.51672	6.18912
	100	30	3.75	6	9.87357	12.4102
10	20	5	8	14	4.67177	6.24194
	50	10	2.25	3	1.91525	4.76012
	100	20	3	5	5.93728	7.40183
20	100	20	7.5	11	7.71783	12.1096
	150	20	5	8	7.45040	11.2910

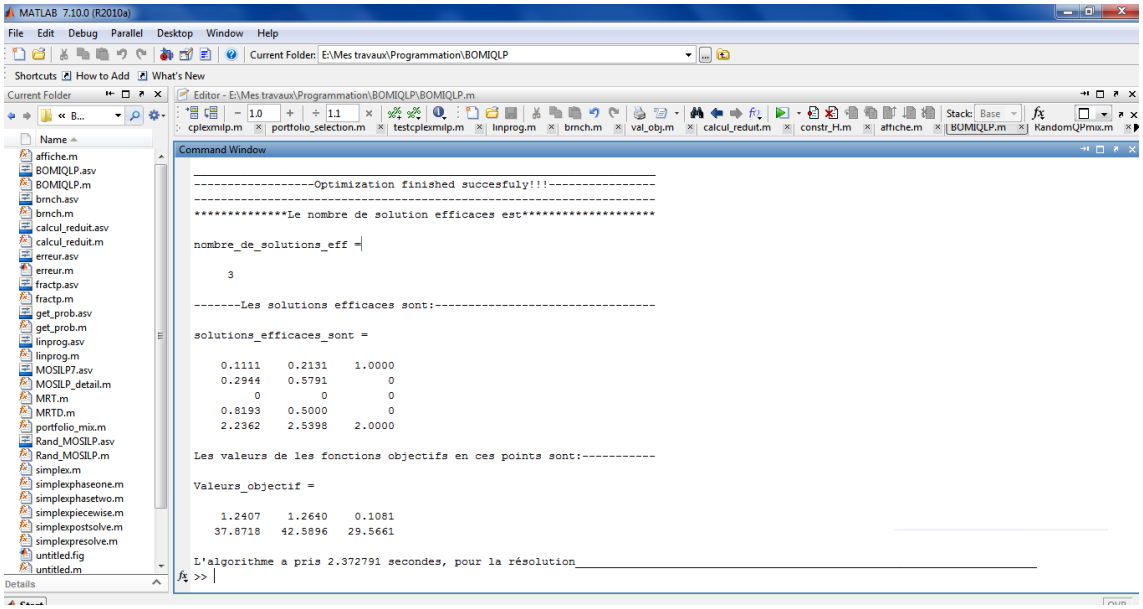

FIG. 5.2 – Une capture d'écran d'une exécution avec le générateur.

On a exécuter l'application avec l'exemple suivant : en se référant au problème PQ_1 défini précédemment, soient :

```
    R = [0.1 0.2 0.15];
V = [0.005   -0.010    0.004
    -0.010    0.040   -0.002
    0.004   -0.002    0.023];
A=[1 2 2;2 4 2;1 1 1];
b=[7;11;3];
```

Après exécution, on a les résultats suivants :

```
------------------------------------------------------------------------
------------------Optimization finished successful!!!-----------------
------------------------------------------------------------------------
**************Le nombre de solutions efficaces est********************

nombre_de_solutions_eff =

     4

-------Les solutions efficaces sont:---------------------------------

solutions_efficaces_sont =

         0    0.5000    1.0000    2.0000
    2.0000    1.5000    1.0000    1.0000
    1.0000    1.0000    1.0000         0

Les valeurs des fonctions objectif en ces points sont:-----------
```

```
Valeurs_objectif =


    0.5500    0.5000    0.4500    0.4000

    0.1750    0.0973    0.0520    0.0200


L'algorithme a pris 1.714111 secondes, pour la résolution_____________________________________
```

5.7 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 (physicien d'origine allemande, 1879-1955)

Dans ce travail, nous avons étudié la programmation quadratique multiobjectif (bi-objectif), côté théorie et pratique. Nous nous sommes intéressés au modèle quad-lin [18] pour la sélection de portefeuilles. Et pour cela, on a suivi le plan suivant :

Dans le premier chapitre, on a présenté quelques définitions et concepts de base de la programmation quadratique, puis, dans le deuxième c'est les concepts de l'optimisation multiobjectif qui sont mets en exergue, dans le troisième une étude bibliographique des méthodes de résolution de problèmes multiobjectif est menée. Ensuite, au quatrième chapitre on a traité la programmation discrète.

Dans le cinquième chapitre, une modélisation multiobjectif pour le problème de l'optimisation des portefeuilles, puis une proposition d'une méthode exacte de résolution bi-objectif quadratique.

On a aussi programmé la méthode et tester avec des problèmes-test, et constaté la nécessite d'une modélisation multiobjectif au modèle Markowitz.

Comme perspectives :

- traiter le modèle moyenne-variance avec des variables incertaines (Programmation quadratique multiobjectif stochastique en variables mixtes),
- faire de même pour des variables floues, (Programmation quadratique multiobjectif floues),
- étudier le modèle mean-variance-skewness pour les investisseurs non standards.

Bibliographie

[1] ABBAS M., AND CHAABANE D. An algorithm for solving multiple objective integer linear programming problem. *RAIRO Operations Research 36* (2002), 351–364.

[2] ABBAS M., AND MOULAÏ M. Journal of the italian operations research society (ricerca. *Journal of the Italian Operations Research Society (Ricerca Operativa) 29* (1999), 15–38.

[3] ASLI L. Approche hybride pour la résolution des problèmes multiobjectif, cas des problèmes du sac-à-dos. Master's thesis, mémoire de Magistère, USTHB, 2010.

[4] AXEHILL D. *Applications of Integer Quadratic Programming in Control and Communication.* LiU-Tryck, Linköping, Sweden, 2005.

[5] AZARM S. *multiobjective Optimum Desig.* www.glue.umd.edu/azarm/optimum_notes/multi/multi.html, 1996.

[6] BARICHARD V. *Approches hybrides pour les problèmes multiobjectifs.* Ecole Doctorale d'Angers, 2003.

[7] BEZOUI M. *Méthode adaptée de programmation quadratique convexe, théorie et applications.* Editions Universitaires Européennes, 2011.

[8] BLUM E., AND OETTLI W. Direct proof of the existence theorem for quadratic programming. *Operations Research 20* (1972), 165–167.

[9] CHERGUI M.E-A. MOULAÏ M., AND OUAÏL F.Z. Solving the multiple objective integer linear programming problem. *Modelling, Computation and Optimization in Information Sys-*

tems and Management Sciences Communications in Computer and Information Science 14 (2008), 69–76.

[10] COELLO COELLO C., AND BECERA R. Evolutionary multiobjective optimization using acultural algorithm. *IEEE Swarm Intelligence Sumposium Proceedings* (2003), 6–13.

[11] COELLO COELLO C. A. *An Updated Survuey of G.A Based multiobjective optimization techniques.* Technical report Lania-RI-98-08 Laboratorio Nacional Avanzada , Xalapa, Veracruz, Mexico, 1998.

[12] COLLETTE Y., AND SIARRY P. *Multiobjective Optimization.* Springer, 2003.

[13] CONTESSE L. Une caractkrisation complkte des minima locaux en programmation quadratique. *Numerische Mathematik 1980*, 34 (315-332).

[14] COTTLE R. W., P. J. S., AND STONE R. E. The linear complementarity problem. *Academic Press* (1992).

[15] CREMA A., AND SYLVA J. A method for finding the set of non-dominated vectors for multiple objective integer linear programs. *European Journal of Operational Research 2003*.

[16] DANTZIG G. B. Maximization of a linear function of variables subject to linear inequalities : in t.c. *Koopmans (ed.) Activity Analysis of. John Wiley Sons, New York* (1951), 339–347.

[17] DANTZIG G.B., AND WOLFE P. Decomposition principle for linear programs. *Operations Research 8* (1960), 101–111.

[18] DEB K., STEUER R. E., T. R., AND TEWARI RAHUL. Bi-objective portfolio optimization using a customized hybrid nsga-ii procedure. *Springer-Verlag Berlin Heidelberg* (2011).

[19] DELORME X. *Modélisation et résolution de problèmes liés à l'exploitation d'infrastructures ferroviaires.* l'université de Valenciennes et du Hainaut-Cambrésis, 2003.

[20] DORFMAN R. Application of linear programming to the theory of the firm. *University of California Press* (1951).

[21] EAVES B. Quadratic programming and affine variational inequalities. *On quadratic programming, Management Science*, 17 (1971), 698–71.

[22] ESCHENAUER H., K. J., AND OSYCZKA A. *Multicriteria design optimisation : Procedures and Applications.* Springer, Berlin, Heidelberg, 1990.

[23] FLETCHER R. *Practical Methods of Optimization.* John Wiley Sons Ltd, USA, 1981.

[24] FRANK M., AND WOLFE P. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3 (1956), 95–110.

[25] FRANK M., AND WOLFE P. An algorithm for quadratic programming. *Naval Research Logistics Quarterly 3* (1956), 95–110.

[26] GEOFFRION A. M. Proper efficiency and the theory of vector. *Journal of Mathematical Analysis and Applications 22* (1968), 618–630.

[27] GOMORY R. E. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the AMS 64* (1958), 275–278.

[28] GUPTA R., AND MALHOTRA R. Multi-criteria integer linear programming problem. *Cahiers de CERO 34* (1992).

[29] HAIMES Y. Y., H. W. A., AND FREEDMAN H. T. *A multiobjective optimization in water resources systems.* Elsevier Scientific, 1975.

[30] KARMARKAR N. K. A new polynomial-time algorithm for linear programming. *Combinatorica* (1984), 373–395.

[31] KEENEY R. L., AND RAAIFFA H. *Decision with Multiple Objective : Preference and value Tradoff.* Cambridge University Press, 1993.

[32] KLEIN D., AND HANNAN E. An algorithm for multiple objective integer linear programming problem. *European Journal of Operational Research 9* (1982), 378–385.

[33] L. ABBACI, AND M. MOULAÏ. *Optimisation sur l'ensemble efficient d'un problème stochastique multiobjectif discret.* Mémoire de magistère, U.S.T.H.B, Alger, Algerie, 2010.

[34] LAND A.H., AND DOIG A.G. An automatic method for solving discrete programming problems. *Econometrica 28* (1960), 497–520.

[35] LEE G. M., T. N. N., AND YEN N. D. *Quadratic programming and affine variational inequalities*, vol. 28. Science+Business Media, Inc, 2005.

[36] MAJTHAY A. Optimality conditions for quadratic programming. *Mathematical programming* (1971), 359–365.

[37] MANGASARIAN O. L. Locally unique solutions of quadratic programs, linear and nonlinear complementarity problems. *Mathematical Programming*, 19 (1980), 200–212.

[38] MARKOWITZ H. Portfolio selection : Efficient diversification of investments. *John Wiley and Sons, New York* (1959).

[39] MARKOWITZ H. M. Portfolio selection. *Journal of Finance 7* (1952), 77–91.

[40] MIETTIENEN K. M., AND MÄKELÄ M. M. *Proper pareto Optimality in Nonconvex Problems-characterization with Tangent and Normal Cones, Technical report.* Jyväskylä University, 1998.

[41] MOULAÏ M., AND ABBAS M. Integer linear fractional programming with multiple objective. *Journal of the Italian Operations Research Society 1* (2002), 15–38.

[42] NESTEROV Y., AND NEMIROVSKI A. *Interior-Point Polynomial Algorithms in Convex Programming.* Philadelphia, PA : SIAM, 1994.

[43] OSYCZKA A. Multicritaria optimization for engineering design. *Gero JS, Design optimization. Academic press, Cambrige* (1985), 193–227.

[44] PIRLOT M. *Métaheuristiques pour l'optimisation combinatoire.* Hermés Science Publications, Paris, 2002.

[45] REARDON B. J. *Fuzzy logic us Niched pareto muli-objective Genetic Algorithm Optimization : Part I : Schaffer's Problem.* Los Alamos National Laboratory, 1997.

[46] ROCKAFELLAR R. T. *Convex Analysis.* Princeton University Press, Princeton, New Jersey, 1970.

[47] ROY, A. D. Safety first and the holding of assets. *Econometrica 20*, 3 (1952), 431–449.

[48] SAIT S. M., AND YOUSSEF H. iterative computer algorithms with applications iningineering : solving combinatorial optimization problems. *IEEE computerssociety* (1999).

[49] SAKAWA M., AND YANO H. An interactive fuzzy satisfcing method for multiobjective linear programming problems witth fuzzy parametres. *Fuzzy sets and Systems 28* (1988), 129–144.

[50] STEUER R. *Multiple Criteria Optimization : Theory, Computation and Applications.* John Wiley and Sons, New-York, 1985.

[51] VINCKE P. *L'Aide Multicritère à la Décision : Statistique et mathématique appliquées.* Ellipses, Paris, 1988.

[52] WOLFE P. A duality theorem for nonlinear programming. *Quarterly of Applied Mathematics 19* (1961), 239–244.

[53] WOLFE P. Some simplex-like nonlinear programming procedures. *Operations Research 10* (1962), 438–447.

[54] ZELLNER A. Linear regression with inequality constraints on the coefficients : An application of quadratic programming and linear decision rules. *Econ Inst Netherlands School of Economics*, 6109 (1961).

Résumé

Résumé

L'objectif de ce travail est l'étude de problèmes quadratiques multiobjectif en variables mixtes qu'on notera MIMOQP (Mixed Integer Multi Objectif Quadratic Problem), et l'élaboration d'une méthode d'optimisation de MIMOQP. Il s'ensuit une adaptation de la méthode au problème d'optimisation de portefeuilles basé sur un modèle "Moyenne-Variance" initié par Markowitz (1959) avec une implémentation sur un problème test.

Mots clés : *problèmes d'optimisation multiobjectif, programmation quadratique, Branch and Bound, variables mixtes, activation des contraintes, selection de portefeuilles, modèle de Markowitz, modèle quad-lin.*

Abstract

The objective of this work is the study of Mixed Integer Multi Objective Quadratic Problems, that we will note MIMOQP, and the development of a method of optimization of MIMOQP. It follows an adaptation of the method to the problem of portfolio optimization based on a "Mean-Variance" model initiated by Markowitz (1959) with an implementation on a test problem.

Keywords : *Multiobjective optimization problems, quadratic programming, Branch and Bound, mixed numbers, active set method, portfolio selection, Markowitz model, quad-lin model.*