MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIPHIQUE

UNIVERSITE MOULOUD MAMMERI

FACULTE DES SCIENCES
DEPARTEMENT DE MATHEMATIQUES

Mémoire de fin d'études pour l'obtention du diplôme
Master 2 en Recherche Opérationnelle
Option : Méthodes et Modèles d'aide à la décision

Thème :

Optimisation Linéaire Multicritère Discrète: Implémentation de la Méthode de Sylva & Crema

Présenté par : *M^elle KAHIL O.*
et
M^elle SMAIL M.

Devant le jury composé de :

Mr KASDI K., Président

Mme RABIA F., Rapporteur

Melle BELLAHCENE S., Examinatrice

Melle KARA F., Examinatrice

Promotion : 2011/2012

Table des matières

Introduction

Les racines de la prise de décision multicritère et de l'optimisation multicritère ou multiobjectif ont été étendues par Pareto, à la fin du XIX^{éme} siècle, et depuis lors la discipline : optimisation multiobjectif s'est developpé et a prospéré. Aujourd'hui, beaucoup de systèmes interactifs d'aide à la décision incorporent de nouvelles méthodes pour traiter des objectifs contradictoires.

Depuis les commencements de l'optimisation multiobjectif, un grand nombre d'ouvrages, de numéros de revues et d'articles ont été publiés. Au cours des années, des enquètes et des bibliographies ont été éditées. Tous les travaux dans ce domaine et le développement des sous-champs ont été consacrés aux aspect particuliers de l'optimisation multicritère : programmation multiobjectif en nombres entiers, optimisation combinatoire multiobjectif, méthodes évolutionnaires et programmation par but "Goal Programming". Par conséquent le besoin de vue d'ensemble compréhensive de la littérature dans l'optimisation multiobjectif peut servir d'aperçu et de guide du dernier cri de la vaste quantité de publications.

Dans de nombreuses situations réelles modélisables par la programmation mathématique, certaines variables de décision peuvent être soumises à être entières. Dans ce cas, on parle de programmation mixte notée MIXLP (Mixed Linear Programming).

La présence de variables discrètes dans un modèle de programmation linéaire multiobjectif modifie considérablement la structure mathématique de ce dernier. Donc, produire des méthodes spécifiques à ce problème est nécessaire.

Notre travail est structuré de la façon suivante :

Dans le premier chapitre, nous rappelons les notions fondamentales liées à la théorie de la décision multiobjectifs, et nous présentons quelques méthodes d'optimisation multiobjectif.

Dans le deuxième chapitre, on s'intéressera aux problèmes de décision multiobjectifs en nombres entiers. Ce chapitre est partagé en deux parties :

Dans la première, on parlera de la programmation mono-objectif en nombres entiers, en donnant des définitions, quelques notions concernant ces problèmes et quelques méthodes de résolution (comme la méthode de Gomory à la quelle on s'intéressera dans la partie pratique de notre travail).

Dans la deuxième partie, on s'intéressera aux méthodes de résolution des problèmes linéaires multiobjectifs en nombres entiers.

Le troisième chapitre est consacré à l'implémentation d'une de ces méthodes de résolution à savoir la méthode de Sylva & Crema [22], en utilisant le logiciel MATLAB 7.0 [17].

On terminera notre travail par une conclusion générale.

Chapitre 1

Programmation Linéaire Multiobjectif

1.1 Introduction

Dans ce chapitre nous présenterons les définitions fondamentales ainsi que quelques méthodes existantes traitant les problèmes de programmation mathématique multiobjectif.

Un problème multiobjectif (MO) consiste à optimiser (maximiser ou minimiser) p fonctions objectifs $\{f_k\}_{k=1,2...,p}$ simultanément $(p \geq 2)$. Il est défini par :

$$\begin{cases} "OPTIMISER" (f_1(x), f_2(x), ..., f_p(x)) \\ tq \quad x \in S \end{cases} \tag{MO}$$

Où $S = \{x \in R^n \ / \ g_i(x) \leq 0, \ i = 1, 2, ..., m\}$ sont des fonctions à valeurs réelles du vecteur de décision $x \in R^n, et \ g_j$ sont des fonctions réelles. Lorsque les objectifs f_i et les fonctions g_j sont linéaires, on obtient un problème de programmation linéaire multiobjectif (MOLP) écrit comme suit :

$$\begin{cases} "OPTIMISER" z_k = c^k x \quad k = 1, 2, ..., p \\ tq \ x \in S \end{cases} \tag{MOLP}$$

$c^k \in R^n, \ k = 1, 2, ..., p$ et l'ensemble S est constitué de contraintes (inéquations, équations) linéaires.

$$S = \{x \in R^n / Ax \leq b, \ x \geq 0\} \ ; \ A \in R^{m \times n} \ , \ b \in R^m.$$

Le mot optimiser est bien écrit entre guillemets car la position du problème (MO) n'a pas de sens en réalité, contrairement au problème unicritère de programmation mathématique classique.

Résoudre le problème (MO) consiste à trouver une solution (action) $\bar{x} \in S$ qui est bonne, i.e qui réalise un meilleur compromis dit solution optimale au sens de Pareto.

Illustration

Nous présentons un exemple d'application où il est demandé à minimiser trois fonctions objectifs sous des contraintes linéaires.

Nous considérons le réseau de gaz simplifié de la Hollande.

On note par :

g_1 : le gaz naturel Allemand.

g_2 : le gaz importé.

k_1, k_2 : le charbon.

g_3 g_4 : le charbon-gaz.

g_5 : le gaz total.

Les paramètres dans ce modèle sont μ_1, μ_2 : efficacité des deux types des usines de gazéification de charbon.

d : la demande totale du gaz.

c : capacité limite de gaz importé .

s_1, s_2 , s_3 : les coefficients d'émission.

c_1 , c_2, c_3..., c_6: les coefficients coûts.

On a trois objectifs :

- minimisation du coût total.
- minimisation de la pollution.
- minimisation du gaz naturel Allemand utilisé.

Le modèle est le suivant :

$$\begin{cases} \text{"}min\text{"}\ z_1 = c_1g_1 + c_2g_2 + c_3k_1 + c_3k_2 + c_4g_3 + c_5g_4 + c_6g_5 \\ \text{"}min\text{"}\ z_2 = s_1g_1 + s_1g_2 + s_2k_1 + s_3k_2 \\ \text{"}min\text{"}\ z_3 = g_1 \\ t.q \\ g_3 - \mu_1 k_1 = 0 \\ g_1 - \mu_2 k_2 = 0 \\ g_1 + g_2 + g_3 + g_4 - g_5 = 0 \\ g_5 \geq d \\ g_2 \leq c \\ g_1, g_2, g_3, g_4, g_5, k_1, k_2 \geq 0 \end{cases}$$

1.2 Concepts de base

La difficulté d'un problème multiobjectif réside dans le fait qu'il soit un problème mathématique mal posé (il n'admet pas de solution optimal). En effet, certains objectifs sont en général en conflit dans la nature, il n'éxiste pas alors une solution meilleure que toutes les autres simultanément pour tous les critères. Le concept de solution optimale n'a donc pas de sens dans ce contexte. Nous allons donc générer l'ensemble des solutions efficaces ou l'ensemble des solutions optimales au sens de Pareto.

1.2.1 Dominance et efficacité

Nous avons utilisé le terme solution optimale au sens de Pareto, il nous faut en donner une définition précise. Reprenons celle du Franco-Italien Vilfredo Pareto concernant le bien être des individus dans [économie mathématique dans l'encyclopédie des sciences mathématiques TOME, p. 624]. « Considérons une position quelconque et supposons qu'en s'en éloigne d'une quantité très petite, compatible avec les liaisons : si en faisant cela, on augmente le bien être de tous les individus de la collectivité, il est évident que la nouvelle position est plus avantageuse à chacun d'eux, et vice versa, elle est moins avantageuse si on diminue le bien être de tous les individus. Le bien être de certains individus peut demeurrer constant sans que les conclusions changent. Mais, si au contraire ce petit mouvement fait augmenter le bien être de certains individus et diminue celui des autres, on ne peut plus affirmer qu'il est avantageux à la collectivité d'effectuer ce mouvement ». Il suffit de remplacer la notion de bien être des individus pour obtenir la définition des solutions efficaces. Plus précisément, soit le problème (MOLP)

$$\begin{cases} \text{“max”}\ z_k = c^k x,\ k = 1, 2, ..., p \\ tq\ x \in S \end{cases} \qquad \text{(MOLP)}$$

Considérons une application linéaire Z qui associe à chaque vecteur de décision $x \in S$ son image $Z(x)$ dans l'espace des critères. On définit une relation d'ordre “ $>$” sur $\mathbb{R}$ par :

$\forall z, z^{'} \in Z(S)$ avec $z \neq z^{'}$; $z > z^{'}$ si et seulement si $z_k \geq z_k^{'}$ $\forall k$ et $\exists k_0$, $1 \leq k_0 \leq p$ tel que $z_{k_0} > z_{k_0}^{'}$. On dira que $z^{'}$est dominé par z.

Définition 1.2.1 *Un vecteur $z^{'} \in Z(S)$ est dit non dominé si et seulement s'il n'existe pas $z \in Z(S)$ et $z \neq z^{'}$ tel que $z > z^{'}$. L'image réciproque par Z d'un vecteur non dominé est dite solution efficace (ou encore optimale au sens de Pareto).*

Définition 1.2.2 *un vecteur z^1 domine fortement z^2 si et seulement si $z^1 > z^2$ (i.e. $z_k^1 > z_k^2$ pour tout k).*

On note par E l'ensemble des solutions efficaces du problème (MOLP).

1.2.2 Point idéal

Un point idéal est un vecteur $\bar{z} \in \mathbb{R}^p$ dont les coordonnées sont les valeurs optimales des objectifs pris séparément. Il est donné par :

$$\bar{z} = (\max_{x \in S} z_1(x), \max_{x \in S} z_2(x), .., \max_{x \in S} z_p(x))$$

1.2.3 Matrice des gains et point Nadir

La matrice des gains est une matrice carrée G de dimension p dont les termes g_{ij}, *pour* $i, j = 1, 2, .., p$ sont donnés par

$$g_{ij} = \begin{cases} \overline{z}_j = \max\limits_{x \in S} \{z_i(x)\} = z_i(\overline{x}_j) \ si \ i = j \\ z_{ij} = z_i(\overline{x}_j) \ si \ i \neq j \end{cases}, \ pour \ i, j = 1, 2, .., p$$

Le vecteur $\underline{z}$ de $\mathbb{R}^p$ de coordonées $\min\limits_{i=1,..,p} \{g_{ki}\}, k \in \{1, 2, .., p\}$ est appelé point nadir.

Définition 1.2.3 *Un sous ensemble F de $Z(S)$ est dit une face de dimension p si la plus petite variété linéaire L vérifie $L \cap Z(S) = F$. Une face de dimension 1 est une arrête de $Z(S)$.*

Définition 1.2.4 *Une face (arrête) F de $Z(S)$ est dite non dominée si tout $z \in F$ est non dominé. Une face F de S est dite efficace si tout x dans F est efficace.*

1.2.4 Fonctions scalarisantes

Le choix d'une solution efficace parmi toutes les solutions réalisables exige une certaine connaissance de la structure de préférence. Cette information est obtenue directement ou indirectement et peut parfois se traduire en terme de paramètre de préférence. Les plus courantes sont :

1) Les poids λ_k , $k = 1, 2, ..., p$ qui reflètent l'importance relative de chaque critère.

2) Le point de référence qui est défini par des niveaux de réservation (valeurs souhaitables) sur chaque critère .

3) Le point de réservation qui est défini par des niveaux (valeurs non souhaitables) sur chaque critère.

La fonction scalarisante est define par $s(z, \lambda) : \mathbb{R} \times \Lambda \to \mathbb{R}$, où Λ est l'ensemble des paramètres. Les fonctions scalarisantes les plus courantes sont les suivantes :

Fonctions linéaires : $s_1(z, \lambda) = \sum\limits_{k=1}^{p} \lambda_k z_k$ avec $\sum\limits_{k=1}^{p} \lambda_k = 1, \ \lambda_k > 0, \ \forall k$

$$\text{et} \quad s_2(z, \lambda) = \sum_{k=1}^{p} \lambda_k \left| z_k - \overline{z}_k \right|$$

Norme L_p pondérée : $s_3(z, \lambda) = \left[\sum\limits_{k=1}^{p} \lambda_k \left| z_k - \overline{z}_k \right|^q \right]^{\frac{1}{q}}$, $q \in \mathbb{Z}^*_{\div}$.

Norme L_∞ de Tchebycheff pondérée : $s_4(z, \lambda) = \max\limits_{1 \leq k \leq p} \left\{ \lambda_k \left| z_k - \overline{z}_k \right| \right\}$.

Norme Tchebycheff pondérée augmentée :

$$s_5(z, \lambda) = \max_{1 \leq k \leq p} \left\{ \lambda_k \left| z_k - \overline{z}_k \right| \right\} + \rho \sum_{k=1}^{p} \lambda_k \left| z_k - \overline{z}_k \right| ; \ \rho > 0.$$

Ici $\overline{z}_k$désigne une valeur de référence de z_k.

Les fonctions scalarisantes ne peuvent engendrer que des vecteurs non dominés. Dans ce cas, elles caractérisent complètement l'ensemble des solutions efficaces.

1.3 Classification des approches multicritères

Dans la programmation mathématique multiobjectif, on distingue trois principales approches

1.3.1 Approche à priori (décideur→recherche)

Dans ce type de méthode le problème multiobjectif est remplacé par un problème unicritère. Dans ce cas le décideur est supposé connaître à priori le poids de chaque objectif afin de les mélanger dans une fonction scalarisante unique :

$$\max_{x \in S} s(z, \lambda)$$

1.3.2 Approche à postériori des préférences (recherche →décideur)

Dans cette approche, le décideur prend sa décision d'après un ensemble de solutions efficaces 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 de résultats le plus représentatif de l'espace des objectifs efficaces.

1.3.3 Approche intéractive (recherche←→décideur)

C'est une approche itérative qui consiste en une alternance de deux types étapes.

a) Les étapes de calcul exécutées par l'analyste

b) Les compromis fournis par la première étape sont présentés aux décideurs qui réagissent surtout en apportant des informations complémentaires sur les préférences. Cette quantité d'information est injectée dans le modèle utilisé et permet de construire de nouveaux compromis. Le processus continu après jusqu'à satisfaction du décideur ou après la satisfaction d'un certain test d'arrêt indiquant l'étape finale (comme le nombre d'itérations fixé par le décideur).

1.4 Caractérisation des solutions efficaces

Les solutions efficaces peuvent être caractésés de plusieurs façons. Nous présentons ici, les caractérisations générales.

Soit $\Lambda = \left\{ \lambda \mid \lambda \in \mathbb{R}^p,\ \lambda_k \geq 0,\ \sum_{k=1}^{p} \lambda_k = 1,\ k = 1, 2, ..., p \right\}$

Théorème 1.4.1 (Geoffrion 10) *: Un vecteur $z^* \in Z(S)$ est dit non dominé si et seulement si z^*est une solution optimale du problème paramétrique $\max_{z \in Z(S)} s_1(z, \lambda)$ avec $\lambda \in \Lambda$.*

Théorème 1.4.2 *(Bowman [6]) : Un vecteur $z^* \in Z(S)$ est dit non dominé si et seulement si z^*est une solution optimale unique du*

problème paramétriqueSoit le problème :
$\max_{z\in Z(S)} s_2(z,\lambda)$ avec $\lambda \in \Lambda$.

Théorème 1.4.3 *(Soland [20]) : Un vecteur $z^* \in Z(S)$ est dit non dominé si et seulement si z^*est une solution optimale du problème paramétrique* $\begin{cases} \max\limits_{z\in Z(S)} s_1(z,\lambda) \\ z_k(x) \geq \alpha_k \end{cases}$ *avec $\lambda \in \Lambda$ et $\alpha_k \in \mathbb{R}$ pour $k = 1, 2, ..., p$*

1.5 Quelques méthodes d'optimisation multiobjectif

Reprenons le problème (MOLP)

$$\begin{cases} \text{“max”}\ z_k = c^k x,\ k = 1, 2, ..., p \\ tq\ x \in S \end{cases} \quad ou \quad \begin{cases} \text{” max”} z = Cx \\ tq\ x \in S \end{cases} \qquad \text{(MOLP)}$$

Tout algorithme déterminant les points efficaces extrêmes ou les arrêtes efficaces passent par les points suivants :

- Obtention d'une base efficace initiale ou terminer la méthode avec $E = \varnothing$ s'il n'existe aucune solution.
- Calcul de tous les points efficaces extrêmes ou toutes les arrêtes efficaces.

1.5.1 Méthode de la somme pondérée [02]

Cette méthode consiste à associer un poids à chaque objectif. Ce poids représente l'importance relative que le décideur attribue à l'objectif. Cela modifie un problème multiobjectif en un simple objectif ;

$$\max \left\{ \lambda' Cx \mid x \in S \right\} \qquad (P_\lambda)$$

où $\lambda \in \Lambda = \left\{ \lambda \in \mathbb{R}^p \mid \lambda_i > 0 \text{ et } \sum_{i=1}^{p} \lambda_i = 1 \right\}$. Si le problème (P_λ) admet une solution, alors elle est efficace pour (MOLP) et la méthode donne bien une solution extrême efficace si $S \neq \varnothing$ et borné.

Cette méthode est simple à mettre en œuvre et elle est d'une grande efficacité. Les difficultés essentielles de cette approche sont :

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

1.5.2 Méthode lexicographique[02]

Fourman propose une méthode dans laquelle les objectifs sont préablement rangés par ordre d'importance par le décideur. En suite, l'optimum est obtenu en minimisant (ou

maximisant) tout d'abord la fonction objectif la plus importante puis la deuxième et ainsi de suite.

Soient donc les fonctions objectifs z_i avec $i = 1, ..., p$. Supposons que

$$z_1 > z_2 > z_3 > ... > z_p$$

Résoudre le programme $\max_{x \in S} z_1(x)$. Soit x_1^* la solution optimale trouvée avec $z_1^* = z_1(x_1^*)$
Résoudre le nouveau problème $\max_{x \in S} z_2(x)$ augmenté de la contrainte $z_1^* = z_1(x_1^*)$.

Soit x_2^*,la solution de ce problème.
A la $i^{ème}$ itération on résout le problème $\max_{x \in S} z_i(x)$ augmenté des contraintes
$z_1^* = z_1(x_1^*),\ z_2(x) = z_2^*, ...,\ z_{(i-1)}(x) = z_{(i-1)}^*$.

La procédure est répétée jusqu'à ce que tous les objectifs soient traités. La solution obtenue à l'étape p sera la solution efficace du problème (MOLP).

Le risque essentiel de cette méthode est la grande importance attribuée aux objectifs classés en premier. La meilleure solution z_i^* trouvée pour l'objectif le plus important va faire converger l'algorithme vers une zone restreinte de l'espace d'état et enfermer les points dans cette zone.

1.5.3 Méthode d'Ecker et Kouada[09]

Soit le problème (MOLP) suivant :

$$\begin{cases} \max Z = C^0 x \\ x \in S = \{x \mid A^0 x = b^0,\ x \geq 0\} \end{cases} \qquad \text{(MOLP)}$$

Où A^0est une $m \times n$ matrice, b^0 un m vecteur et C^0 une $r \times n$ matrice la ci-dessous.

0	-C^0
b^0	A^0

Tableau T^0

Si la solution efficace initiale n'est pas un sommet, alors c'est l'intérieur d'une face F^0. Donc on a seulement besoin de faire une simple opération pivot dans la matrice A^0 du tableau T^0 pour se déplacer vers une solution extrême de F^0. Donc, avec une telle opération appropriée dans A^0, on peut représenter un sommet efficace avec le tableau suivant :

-C	0
A	I

Tableau T

Théorème 1.5.1 *: Etant donné le tableau efficace non dégénéré ci-dessus T . Soit le programme linéaire suivant :*

$$\begin{cases} \max \Psi = e's \\ t.q.Cx = s + C^j \\ x \succeq 0; s \succeq 0 \end{cases} \qquad (Q^j)$$

où C^j est la colonne de C correspondante à l'ensemble des indices hors base N. Soit E l'ensemble des points efficaces, alors $F^j \subset E$ si et seulement si $\Psi_{\max} = 0$ dans le problème (Q^j).

Théorème 1.5.2 *: Soit $x^0 \in S$ et le problème*

$$\begin{cases} \max \ \Psi = e's \\ t.q.Cx = Is + Cx^0 \\ Ax = b \\ 0 \leq x \in \mathbb{R}^n \\ 0 \leq s \in \mathbb{R}^k \end{cases} \qquad (E-K)$$

Si $(\bar{x}, \bar{s})$ est une solution optimale du problème (E-K) et si la fonction objectif Ψ a une valeur positive non bornée, alors $E = \phi$.

Ce résultat (théorème d'Ecker et Kouada) nous offre un outil de détermination d'une solution efficace quand $E \neq \phi$ et indique que E est vide quand il n'y a pas de solutions efficaces. L'inconvénient avec cette méthode est qu'une fois qu'une solution est obtenue on ne sait pas si c'est une solution extrême ou non.

1.5.4 Méthode du "Goal Programming"[02]

Cette méthode est également appelée **Target vector optimisation**. Le décideur fixe un but T_k à atteindre pour chaque objectif z_k.Ces valeurs sont ensuite ajoutées au problème comme des contraintes supplémentaires. La nouvelle fonction objectif est modifiée de façon à minimiser la somme des écarts entre les résultas et les but à atteindre :

$$\min \sum |z_k(x) - T_k \mid \ avec\ x \in S$$

T_k représente la valeur à atteindre pour le k^{ème}objectif.

La méthode est facile à mettre en oeuvre mais la définition des poids et des objectifs à atteindre est une question délicate qui détermine l'efficacité de la méthode. Cette méthode a l'avantage de fournir un résultat même si un mauvais choix initial a conduit le décideur a donner un ou plusieurs but(s) T_k non réalisable (s).

Différentes variantes et applications de cette technique ont été proposées [Ignizio (1981),Van Veldhuizen (1999)].

1.5.5 Méthode STEM[04]

Cette méthode décrite dans est intéractive. Elle a été introduite dans le cadre de la résolution des problèmes (MOLP). A chaque itération elle réduit $\Psi(S)$ progressivement par l'addition des contraintes sur les valeurs des critères. A chaque étape interactive, un nouveau compromis est trouvé en optimisant une norme de type $s_5(z,\lambda)$ sur une partie de $\Psi(S)$ selon une direction correspondant à la relaxation d'un critère fixé par le décideur.

Les différentes étapes sont résumées comme suit :

Etape 1. On détermine la matrice des gains ainsi que les coefficient orthonormés suivants :

$\lambda_j = \frac{\alpha_j}{\sum\limits_i \alpha_i}$ où $\alpha_j = \frac{\bar{z}_j - n_j}{\max\left\{|\bar{z}_j|,|n_j|\right\}}$ pour $j = 1, ..., p$ et n_j est le point nadir.

Soit $\Psi(S') = \Psi(S)$ et $l = 1$.

Etape 2. Une solution de compromis z^l est calculée en résolvant le problème

$$\begin{cases} \min\left(\mu - \sum\limits_{i=1}^{p} \rho_j z_j\right) \\ t.q \;\; \mu \geq \lambda_j \left(\bar{\bar{z}}_j - z_j\right) \text{ pour } j = 1, ...p \end{cases}$$

avec $\bar{\bar{z}}_j = \bar{z}_j + \epsilon_j$ et ϵ_j, ρ_j sont positifs et très petits. $\bar{z}_j$ est le $j^{ème}$ élément de la diagonale principale de la matrice des gains.

Etape 3. La solution obtenue z^l est présentée au décideur.

· S'il est satisfait avec z^l alors la procédure prend fin.

· Autrement, lui demander d'indiquer sur quel critère (indice) i il est prêt à faire une concession et quel quantité maximum Δ_i il accepte de concéder.

Etape 4. L'ensemble des résultats potentiels est réduit à :

$$\Psi(S^{l+1}) = \left\{z \in \Psi(S^l) \mid z_i \geq z_i^l - \Delta_i \;\text{ et } z_j \geq z_j^l, \; \forall j \neq i\right\}$$

Faire $\lambda_i = 0$, $l = l + 1$ et aller à l'étape 1.

1.5.6 Méthode de vincke[25]

Cette méthode exécute une analyse interactive de sensibilité en utilisant les classiques de la méthode du simplexe. Elle est restreinte aux problèmes (MOLP) et ressemble à la méthode STEM dans ses premières étapes. Le décideur peut revenir sur un choix déjà formulé (il a la possibilité de relaxer une contrainte ou un des critères) et de nouveau un autre compromis est déterminé.

1.5.7 Méthode de Vanderpooten[24]

Cette méthode est applicable sans restriction y compris pour des problèmes possédant une liste explicite d'alternatives. La méthode propose des comparaisons par paires

entre l'alternative préférée et une autre qui représente une amélioration potentielle. La réaction du décideur est analysée afin de dégager une région intéressante où de nouvelles propositions peuvent émerger. Elle est caractérisée par le fait qu'il n'y a pas d'hypothèse d'existence d'une fonction d'utilité. Elle est construite dans une perspective de processus d'apprentissage, sans aucune restriction sur l'ensemble des solutions et avec peu de questions au décideur. Elle est simple et comporte assez peu de calculs. Les différentes étapes sont résumées ci-dessous.

Etape 1. Dans la première étape, $\bar{z}$ est calculé comme dans la méthode de STEM. On considère $\bar{\bar{z}} = \bar{z} + \varepsilon$ et le vecteur $\lambda > 0$. Le premier point de compromis est calculé par la résolution du problème suivant :

$$\begin{cases} \min s(z, \bar{\bar{z}}, \lambda^1) \\ t.q \quad z \in \Psi(S) \end{cases}$$

$s(z, \bar{\bar{z}}, \lambda^1)$ est une fonction scalarisante dépendante de la nature du problème et de $\Psi(S)$. Généralement on prend $s(z, \bar{\bar{z}}, \lambda^1) = s_5(z, \bar{\bar{z}}, \lambda^1)$.

Soit $\Psi(\bar{S}^0) = \Psi(S)$ et $k = 1$.

Etape 2. Considérons z^{k-1}, demander au décideur d'indiquer les critères qui devraient être améliorés.

Soit J l'ensemble des indices des critères correspondants.

(i) Si $J = \varnothing$, fin et z^{k-1} est la prescription finale.

(ii) Sinon, on détermine la nouvelle région d'intérêt :

$$\Psi(S^k) = \left\{ z \in \Psi(\bar{S}^{k-1}) \mid \ z_j \geq z_j^{k-1}, \forall j \in J \right\}$$

Soit $\bar{z}^k = z^{*k} + \varepsilon$ où z^{*k}est le poit idéal relatif à $\Psi(S^k)$.

Etape 3. Calculer une nouvelle solution de compromis z^k par la résolution du problème suivant :

$$\begin{cases} \min s(z, \ \bar{z}, \ \lambda^k) \\ t.q. \ z \in \Psi(S^k) \end{cases}$$

Etape 4. Présenter au décideur z^{k-1} et z^k et lui demander d'indiquer celui qu'il préfère.

(a) Déterminer une première approximation de la prochaine région d'intérêt :

$$\Psi(\bar{S}^k) = \left\{ z \in \Psi(S) \mid z_j \geq z_j^{k-1}, \forall j \in J \right\}$$

et construire un nouveau vecteur λ^{k+1}(direction de préférence) déterminé par

$$\lambda^{k+1} = \frac{1}{\bar{z}_j - z_j^k} \ , \ (j = 1, ..., p)$$

(b) Si z^{k-1} est le plus préféré, demander au décideur d'indiquer quels critères (dont les valeurs ont diminué) sont les plus responsables de ce jugement.

Soit Ω l'ensemble des indices des critères correspondants. Une première approximation de la prochaine région d'intérêt est déterminée par :

$$\Psi(\bar{S}^k) = \left\{ z \in \Psi(S) | z_j \geq z_j^k, \ \forall j \in \Omega \right\}.$$

Soit $z^k = z^{k-1}$ et $\lambda^{k+1} = \lambda^k$.

Etape 5. Soit $k = k + 1$ et aller à l'étape 2 (étape (ii))

Chapitre 2

Programmation Linéaire multi-objectif en nombres entiers

Un programme linéaire multi-objectif 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 toutes les solutions réalisables efficaces entières.

Dans ce chapitre, nous allons présenter un bref aperçu sur les notions de base de la programmation linéaire en nombres entiers ainsi que quelques méthodes de résolution de ces problèmes.

Un problème (MOILP) peut être formulé par

$$\begin{cases} "OPTIMISER" z_k = c^k x \quad k = 1, 2, ..., p \\ tq \quad x \in D \end{cases} \tag{MOILP}$$

où $D = S \cap \mathbb{Z}_+^n$ et $S = \{x \in \mathbb{R}^n \ / \ Ax \leq b, x \geq 0\}$; $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$.

Une écriture matricielle est parfois utilisée :

$$\begin{cases} " \max" ou" \min" Z = Cx \\ tq \ x \in D \end{cases} \tag{MOILP}$$

où la k^{ième} ligne de C est c^k.

2.1 Programmation mono-objectif en nombres entiers(ILP)

Une variété très large de problèmes est formulée et résolue en utilisant la programmation en nombres entiers. Parmi les problèmes connus, on note le problème de transport, de planification de la production, de télécommunication et autres applications récentes dans la biologie moléculaire et les statistiques.

$$\begin{cases} \max \quad z = cx \\ tq \quad x \in D \end{cases} \tag{PLNE}$$

La nature des problèmes de programmation linéaire en variables continues et les problèmes de programmation linéaire en variables discrètes est différente. Le problème (PLNE) est beaucoup plus difficile et par conséquent la caractérisation d'une solution optimale et le problème d'existance d'une solution $x \in D$, $D = S \cap \mathbb{Z}_+^n$ est NP-complet.

2.1.1 Méthodes de résolution d'un problème de programmation linéaire en nombres entiers

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

Méthode Branch & Bound[01]

Une approche naive 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 à mettre en oeuvre, elle sera très coûteuse en terme de temps de calcul même pour des problèmes de taille réduite.

La méthode Branch & Bound est très efficace pour la résolution des problèmes de programmation linéaire en nombres entiers. Son principe est de découper l'espace initial de recherche en domaines de plus en plus restreints afin d'isoler l'optimum global. L'algorithme de recherche forme ainsi un arbre dont chaque noeud représente une partie de l'espace. Ensuite chaque noeud 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 noeud est stoppée, sinon la séparation continue.

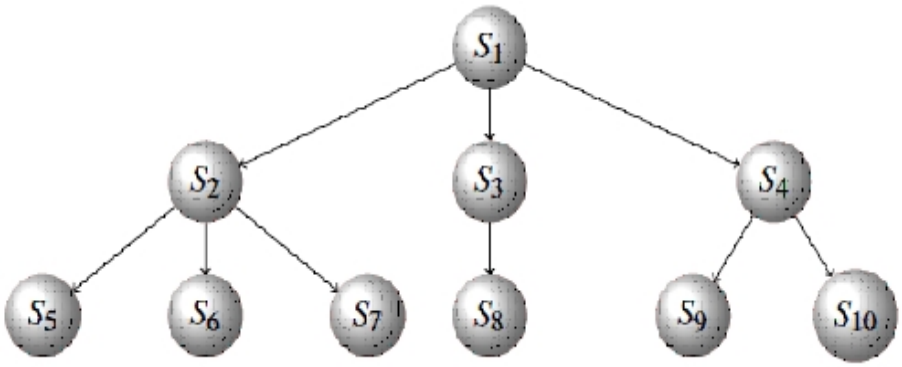

Fig 2.1 Principe de Branch & Bound

Méthodes de Coupes

Coupe de Dantzig[07] Historiquement, Dantzig est le premier chercheur a avoir proposé une coupe qui porte son nom. Cette coupe

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

a été proposée sur la base que dans le problème (PL) $\{$"max$\rfloor$ $z = cx,\ x \in S\}$, où le vecteur second membre de l'équation contrainte est supposé positif mais non entier, une des variables hors base est strictement positive ; donc supérieure ou égale à 1.

Méthode de Gomory[11] La méthode des troncatures ou ‹‹méthode des coupes›› développée par Ralph Gomory (1958) fournit une méthode de résolution des programmes linéaires en nombres entiers (PLNE). On peut résumer la méthode de résolution d'un problème PLNE avec l'algorithme de Gomory comme suit :

Algorithme de Gomory

1. Etant donné un (PLNE), résoudre le problème relaxé (PL)

$$\begin{cases} \max & Z = cx \\ tq & x \in S \end{cases} \qquad \text{(PL)}$$

correspondant à l'aide de l'algorithme du simplexe. Si la solution optimale du PL contient uniquement des valeurs entières, elle est également une solution optimale du (PLNE). La résolution est terminée. A l'optimum on a les conditions suivantes :

a) $\hat{c} \leq 0$

b) $\hat{b} \geq 0$

2. Si une ou plusieurs variables de base dans la solution optimale du problème relaxé (PL) ne sont pas entière, on doit alors générer, à partir d'une des lignes du tableau (celle dont la partie fractionnaire pour la variable de base correspondante est la plus élevée) une contrainte supplémentaire dite coupe de Gomory (nous indiquons comment construire cette contrainte séquentiellement). Cette contrainte (sous forme d'équation) est ajoutée au tableau optimal du (PL) et on détermine le nouveau tableau optimal à l'aide de la méthode dual du simplexe .

3. Si les variables de base dans le nouveau tableau optimal sont entières , nous avons obtenu également la solution optimal du (PLNE). La résolution est terminée.

4. Sinon, on doit générer (à partir du dernier tableau optimal) une nouvelle coupe de Gomory, l'ajouter au dernier tableau optimal pour trouver la nouvelle solution optimal à l'aide de la méthode dual du simplexe. Si la solution optimale obtenue est à valeur entières, la résolution est terminée. Sinon, on répète la procédure jusqu'à l'obtention d'une solution optimale à valeur entière.

Avant d'appliquer l'algorithme de Gomory à la résolution d'un (PLNE), introduisons les notions de partie entière et partie fractionnaire d'un nombre.

Pour un nombre a, la somme de sa partie entière $\lfloor a \rfloor$ et de sa partie fractionnaire $\langle a \rangle$ s'écrit

$$a = \lfloor a \rfloor + \langle a \rangle$$

où $N = \lfloor a \rfloor$ est le plus grand nombre entier inférieur ou égal au nombre a ($N \leq a$) et $0 < \langle a \rangle < 1$. la partie fractionnaire d'un nombre réel "a" est donc la différence entre ce nombre et sa partie entière .

Génération de la coupe de Gomory

Soit le tableau optimal issu de la résolution de (PL) par la méthode du simplexe où la solution optimale est supposée non entière (voir tab1.1). Sans perte de généralités, on suppose qu'à l'optimum, il ya 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	$\hat{A} = B^{-1}A$	$\hat{b} = B^{-1}b$
$-Z$	$\hat{c} = c - \pi A$	$z = c_B B^{-1} b$

Tab1.1 :tableau final associé à la solution optimale

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, on choisit une variable de base dont la valeur est fractionnaire, soit x_i. La $i^{ème}$ équation du tableau est donnée par :

$$x_i + \sum_{j=1}^{n} \hat{a}_{ij} y_j = \hat{b}_i \tag{1.1}$$

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 :

$$x_i + \sum_{j=1}^{n} \langle \hat{a}_{ij} \rangle y_j + \sum_{j=1}^{n} \lfloor \hat{a}_{ij} \rfloor y_j = \left\langle \hat{b}_i \right\rangle + \left\lfloor \hat{b}_i \right\rfloor \tag{1.2}$$

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

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

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

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

En soustrayant cette dernière inéquation de (2.4), on obtient :

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

qui est bien valide. De plus (2.6) n'est pas satisfaite par la solution optimale de (PL), donc (2.6) est bien une coupe.

En ajoutant une variable d'écart x_s à (2.6), on obtient la coupe de Gomory définie par

$$x_s - \sum_{j=1}^{n} \lfloor \hat{a}_{ij} \rfloor \, y_j = - \left\lfloor \hat{b}_i \right\rfloor \tag{1.3}$$

Une fois la coupe est générée, les coefficients de (2.8) sont insérés dans une nouvelle ligne du tableau (Tab1.1).

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

Application de l'algorithme de Gomory

On veut résoudre le programme linéaire en nombres entiers suivant :

$$\begin{cases} max \; z = 3x_1 + 6x_2 + 2x_3 \\ \text{avec les contraintes :} \\ 2x_1 + x_2 + 3x_3 \leq 10 \\ x_1 + 2x_2 + 3x_3 \leq 10 \\ x_1 \geq 0 \; , x_1 \geq 0 \; , x_3 \geq 0 \;\; \text{et entiers} \end{cases} \tag{PLNE}$$

Le tableau suivant présente les solutions qu'on obtient en résolvant notre problème (sans restriction de variables entières) selon la méthode de simplexe, les valeurs entières arrondies, avec B&B et les solutions optimales entières obtenues en utilisant la méthode des trancatures.

[Tab 1.2]. Tableau comparatifs des solutions

Solution simplexe	solution entière arrondie	solution B&B	solution Gomory
$x_1 = 3,33$	$x_1 = 3$	$x_1 = 0$	$x_1 = 2$
$x_2 = 3,33$	$x_2 = 3$	$x_2 = 5$	$x_2 = 4$
$x_3 = 0$	$x_3 = 0$	$x_3 = 0$	$x_3 = 0$
$Z = 30$	$Z = 27$	$Z = 30$	$Z = 30$

Conclusion : Le tableau ci-dessus nous montre clairement que les solutions obtenues avec la méthode de Gomory et avec B&B sont meilleures.

2.2 Programmation linéaire multi-objectif en nombres entiers (MOILP)

Notons par E l'ensemble de toutes les solutions efficaces du problème (MOILP).

E est bien caractérisée par la solution du problème (P_λ) comme défini précédemment. Ce principe souvent dit principe de Geoffrion n'est plus valable lorsqu'on traite les problèmes MOILP car l'ensemble d'admissibilité n'est pas convexe. Sur cette base, l'idée d'efficacité peut être élargie en répartissant l'ensemble des points efficaces en deux sous-ensembles. L'ensemble des solutions efficaces supportées note par ES et l'ensemble des solutions efficaces non supportées noté par ENS.

2.2.1 Types de solutions recherchées

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. Cette dernière, consiste à transformer un problème multiobjectif en un problème mono-objectif combinant les différentes fonctions objectif en une seule fonction F de façon linéaire :

$$F(x) = \sum_{i=1}^{r} \lambda_i C^i x \qquad où \quad \lambda_i \in [0,1] \quad et \quad \sum_{i=1}^{r} \lambda_i = 1$$

Différents poids fournissent différentes solutions supportées ; une même solution pouvant être générée en utilisant des poids différents.

Solutions non supportées

Néanmoins, il existe d'autres solutions qui bien qu'efficaces ne peuvent être obtenues par la résolution d'un programme paramétrique. 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 $Z(S)$

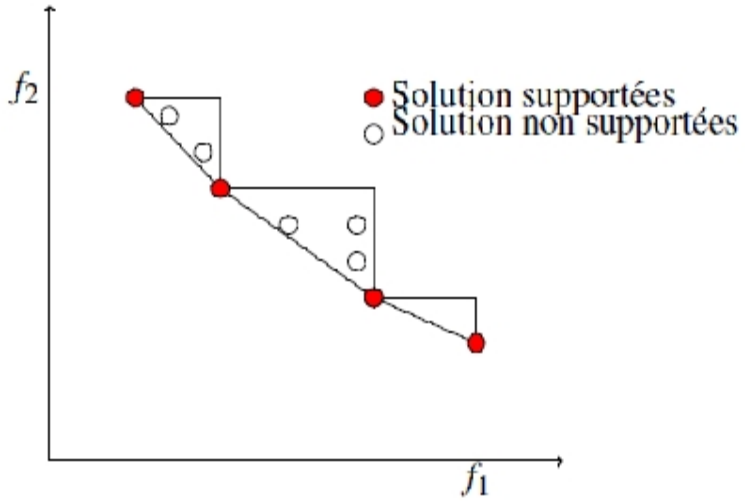

Fig 2.1 Solutions supportées et non supportées

Illustrons par un exemple numérique la notion de solutions supportées et non supportées.

Exemple 2.2.1 *Soit :*

$$(MOILP) \qquad \begin{cases} \text{" max "} z_1 = 6x_1 + 3x_2 + x_3 \\ \text{" max "} z_2 = x_1 + 3x_2 + 6x_3 \\ t.q \quad x_1 + x_2 + x_3 \leq 1 \\ x_1, x_2, x_3 \in \{0, 1\} \end{cases}$$

Il est trés clair d'aprés le thèorème de Geoffrion que les solutions optimales du problème :

$$(P_\lambda) \qquad \begin{cases} \text{" max "} z_\lambda = \lambda(6x_1 + 3x_2 + x_3) + (1 - \lambda)(x_1 + 3x_2 + 6x_3) \\ t.q \quad x_1 + x_2 + x_3 \leq 1 \\ x_1, x_2, x_3 \in \{0, 1\} \qquad \text{avec} \qquad 0 < \lambda < 1 \end{cases}$$

sont des solutions efficaces.

$X_1 = (1, 0, 0)$ avec $Z_1 = (6, 1)$ et $X_2 = (0, 0, 1)$ avec $Z_2 = (1, 6)$ sont des solutions efficaces, solutions optimales de (P_λ) mais, la solution $X' = (0, 1, 0)$ avec $Z' = (3, 3)$ qui est aussi efficace n'est pas une solution optimale du problème paramétrique

2.2.2 Résolution dans l'espace des critères

La méthode de Klein-Hannan [15]

La technique de Klein-Hannan est une méthode interactive, pour générer séquentiellement l'ensemble des solutions efficaces du problème (MOILP). Elle consiste à résoudre progressivement des programmes linéaires mono-critère avec des contraintes ajoutées à chaque itération. La méthode génère des solutions efficaces de façon que l'utilisateur ne soit pas obligé de déterminer l'ensemble des solutions efficaces entièrement s'il ne s'intéresse qu'à quelques solutions.

Une brève présentation de la méthode est résumée par l'algorithme ci-dessous.

Algorithme

Etape1 : Résoudre le problème (P_1) défini comme suit (l'indice i est pris arbitrairement dans $(1, ..., p)$) :

$$(P_1) : Max\{Z_i = c_i x : x \in D\}$$

Cas1. Si la solution optimale de (P_1), soit x_1, est unique alors elle est efficace pour (MOILP).

Cas 2. Sinon, déterminer toutes les solutions alternatives à x_1 et par comparaison deux à deux des vecteurs critères associés, garder uniquement celles qui sont efficaces pour construire l'ensemble des solutions efficaces générées à l'étape 1.

Etape générale j :

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

$$(Pj)\begin{cases} \text{" max "} \ Z_i = C^i x \\ t.q \quad x \in D \\ \bigcap\limits_{k=1}^{r} \left(\bigcup\limits_{s=1, s\neq i}^{p} (C^s x \ \geq C^I \overset{*}{x}_k + \varepsilon^s) \right) \end{cases}$$

avec $0 \leq \varepsilon^s \leq 1$ et $\overset{*}{x}_k (k = 1, ..., r)$ les points efficaces obtenus aux étapes $(1, ..., j-1)$.

Si $E(P_j)$ est l'ensemble des solutions efficaces obtenues à l'étape j et Y^j est l'ensemble des points accumulés à la fin de l'étape j,alors $Y^{j-1} \cup E(Y^j)$ pour $j \geq 2$ avec $Y^1 = E(P_1)$.

Etape finale n :

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

Méthode de Chalmet et al. [08]

Cette méthode a été proposée par Chalmet et al. pour résoudre un problème bicritère en variables discrètes. Le problème $\max\limits_{x \in D} \{\lambda \ z_1(x) + (1-\lambda) \ z_2(x)\}$ est résolu afin de générer l'ensemble des solutions supportées (ES). Puis l'ensemble des solutions non supportées (NES) est déterminé comme suit :

Soient $z^1 = (z_1^1, z_1^2)$ et $z^2 = (z_1^2, z_2^2)$ deux points non dominés adjacents parmis les points déjà obtenus. On cherche s'il y'a une solution non dominée entre ces points en résolvant le problème

$$(P') \begin{cases} \text{" max "} \ \lambda z_1(x) + (1-\lambda) z_2(x) \\ \text{t.q.} \ \ z_1(x) \geq z_1^1 \\ z_2(x) \geq z_2^2 \\ x \in D \\ \lambda \in [0, 1] \end{cases}$$

Soit z^l la solution efficace (s'il ya des solutions à ce problème).

La même procédure est appliquée ensuite avec les paires $(z^1, z^l), (z^l, z^2)$.

S'il n'existe pas de solution pour le problème $(P)'$ alors z^1et z^2 sont adjacents. La méthode prend fin lorsque toutes les paires ont été examinées.

Méthode de Gonzales, Reeves & Franz [12]

La méthode est principalement composée de deux phases :

La première phase consiste à sélectionner un sous ensemble $\tilde{S} \subset E$ de solutions préférées par le décideur.

Dans la deuxième phase, les solutions non supportées seront considérées et seulement les préférées par le décideur seront introduites dans l'ensemble $\tilde{S}$.

Donc, à chaque étape, une nouvelle solution efficace (supportée ou non) est présentée au décideur qui doit la comparer avec les éléments de $\tilde{S}$ pour actualiser cet ensemble.

Phase 1

Etape 1 : On résout les p problèmes unicritères suivants :

$\left\{\max_{x\in D} c^k x\right\}$;pour tout $k = 1, 2, ..., p$.

Soit $\tilde{S} = \{\tilde{x}_k; k = 1, 2, ..., p\}$; où $\tilde{x}_k$ est la solution optimale du $k^{iéme}$critère.

Soit $\tilde{Z} = \{\tilde{z}_k = C\tilde{x}_k \ / \ k = 1, 2..., p\}$ l'ensemble des p points dans l'espace des critères.

Etape j : Une fois déterminé l'hyperplan passant par les p points $\tilde{z}_k$ dans l'espace des critères $\mathbb{R}^p$,l'image réciproque $G(x)$ (la direction linéaire de recherche) peut être construite. Puis une nouvelle solution est déterminée par la résolution du problème $\left\{\max_{x\in D} G(x)\right\}$. Soit $(x^*_G \ , \ Z^*_G)$ la solution de ce problème.

Trois éventualités se présentent :

(i) $Z^*_G \notin \tilde{Z}$ et le dcideur préfère x^*_G à au moins une solution de $\tilde{S}$; dans ce cas x^*_G remplace dans $\tilde{S}$ la solution la moins préférée. La direction $G(x)$ est actualisée, par la suite on passe à l'étage $j + 1$.

(ii) $Z^*_G \notin \tilde{Z}$ mais le décideur préfère toutes les solutions de $\tilde{S}$ à x^*_G;dans ce cas aller au point 2 de la deuxième phase.

(iii) $Z^*_G \in \tilde{Z}$,dans ce cas l'ensemble $\tilde{Z}$ définit une face efficace . Aller au point 1 de la deuxiéme phase.

Phase 2 :

Etape j

(1) Une solution $\tilde{x}_{(j)}$ non supportée est déterminée par la résolution du problème suivant

$$(\ P_G) \ \begin{cases} \max G(x) \\ t.q.x \in D \\ G(x) \leq \tilde{G} - f.....(*) \end{cases}$$

où $\tilde{G} = G(\tilde{x}_{(j)})$. A l'étape 0, $\tilde{G}$ est la valeur de $G(x)$ dans le cas (iii). La contrainte $(*)$ retire toutes les solutions précédentes de la région d'admissibilité.

- Si $\tilde{x}_{(j)}$ est jugée préférable à une des solutions de $\tilde{S}$, alors $\tilde{x}_{(j)}$ remplace la solution la moins préférée dans $\tilde{S}$.

- Sinon, la procédure s'arrête et le décideur choisit la solution qu'il préfère dans $\tilde{S}$. $G(x)$ est recalculée et on pose $j = j + 1$.

(2) Dans ce cas le décideur n'est pas intérésé par les solutions différentes de celle de $\tilde{S}$; avant d'arrêter la procédure, on entame néanmoins une recherche de solution non

supportées dans le voisinage de la solution préférée dans $\tilde{S}$, par résolution du problème (P_G) où $G(x)$ est la fonction objectif qui a engendré ce point.

Cette technique est une méthode interractive assez lourde, néaumoins elle permet de déterminer les solutions efficaces supportées séparement.

Méthode de Steuer & Choo [21]

Cette méthode est générale, elle peut être appliquée à tout problème de programmation mathématique multiobjectif linéaire ou non avec le cas échéant des variable entières.

Etape 1 : Calculer d'unpoint cible, par exemple le point idéal. La recherche des solutions efficaces se fait par "cadrillage"de l'ensemble S au moyen d'un ensemble diversifié des valeurs du vecteur paramètre λ.Pour chaque valeur λ, la distance de Tchebycheff calculée par apport au point cible est minimisée, ceci donne un sous ensemble de solutions efficaces. Soit x^1la solution efficace choisie par le décideur dans ce sous ensemble.

Etape 2 : Déterminer ensuite un deuxième ensemble de valeurs du vecteur λ de façon à quadriller le voisinage élargi de x^1. Une nouvelle solution efficace, soit x^2,est désignée par le décideur comme en étape 1.

Etape 3 : Continuer la procédure mais en focalisant le paramètre λ dans le voisinage restrient du nouveau compromis.

Méthode de Marcott & Soland [16]

Cette méthode utilise le concepte de "Branch and Bound interactif ". Le problème (MOILP) est analysé à l'aide d'une arborescence Branch and Bound au sien de laquelle on progresse en fonction des préférences du décideur. Donc à chaque noeud le décideur choisit le sous noeud à examiner. Malgré que l'idée soit très intéressante, l'implémentation est plus difficile pour les raison suivantes :

-Les ensembles de solutions réalisables correspondant aux sous noeud ne sont pas disjoints.

-L'ordre d'amélioration des critères est fixé une fois pour toutes, indépendamment du compromis proposé.

-La gestion de la progression de l'arborescence est très lourde et nécessite une capacité de stockage importante.

Méthode MOMIX[23]

Momix est une abréviation de (Multi-Objective MIXTE), c'est une méthode introduite par Tghem & Kunsch où l'idée de Branch and Bround interactive est modifiée et un agencement différent de séparation et de progression est proposée. La méthode comporte principalement deux phases.

Phase 1 : C'est une phase d'initialisation, il s'agit de déterminer le premier point efficace x^0 en minimisant la norme de Tchebycheff. Le problème suivant est donc résolu

pour $m = 0$.

$$\begin{cases} \min\ \delta \\ t.q \quad \beta_{m,k}\left(M_{m,k} - c^k x\right) \leq \delta\ ,\ k = 1, 2..., p \\ x \in S_m \end{cases}$$

où $S_0 = S$; $M_{m,k}$ est la valeur optimale de la k^{ième} fonction objectif sur S_m et $\beta_{m,k} = \frac{\alpha_{m,k}}{\sum_{i=1}^{p} \alpha_{m,i}}$ le poids de la k^{ième} fonction objectif par exemple $\alpha_{m,k} = \frac{M_{m,k} - n_{m,k}}{\max\left(\left|n_{m,k}\right|, \left|M_{m,k}\right|\right)} \frac{1}{\left\|c^k\right\|}$; où $n_{m,k}$ est la $k^{ième}$ composante du point nadir sur S_m.

Phase 2 : La méthode Branch and Bound interactive est utilisée en deux étapes :

Première étape (procédure descendante "depth first")

Supposons que nous somme à la m^{ième}itération. Soit x^{m-1}le m^{iéme}compromis et $z_k = c^k x^{m-1}$ valeur correspondante pour la k^{iéme} fonction objectif. Le DM indique le critère $l_m(1) \in \{1, 2, ..., p\}$ à améliorer en priorité. Un nouveau compromis est obtenu en résolvant le problème $(P_{(m)})$ avec

$$S_m = S_{m-1} \cap \left\{ x \mid z_{l_{m(1)}}(x) > z_{l_m(1),(m-1)} \right\}$$

de façon que le critère $l_m(1)$ soit amélioré.

Tests d'arrêt

Des tests d'arrêts ont été définis, on cite quelque uns :

a) $S_m = \emptyset$

b) $M_{k,m} - m_{k,m} \leq \varepsilon_k$ pour toutes les valeurs k ($\varepsilon_k > 0$,fixé)

c) $\tilde{z}$,le vecteur des meilleures valeur trouvées précédemment est préféré au vecteur idéal M_m relatif à D_m.

d) Un nombre Q d'itérations fixé par le décideur a été effectué et aucune amélioration n'a été apporté à $\tilde{z}$.

Deuxième étape : Procédure remontante "backtracking"

La région d'admissibilité négligée à chaque itération de la procedure descendante est scannée pour un éventuel meilleur compromis que $\tilde{z}$ dans cette région. Les différentes opérations sont :

a) Le noeud correspondant au compromis x^{m-1} est séparé en p branches.

b) Soit $l_m(k)$; $k = 1, 2, ..., p$ l'ordre de priorité dans lequel le décideur veut voir améliorer les p critères par rapport à ce compromis. Les p sous noeuds sont obtenus par adjonction respective des contraintes suivantes :

$z_{l_m(1)} > z^{m-1}_{l_m(1)}$

$z_{l_m(2)} > z^{m-1}_{l_m(2)}$ et $z_{l_m(1)} \leq z^{m-1}_{l_m(1)}$

.....

$z_{l_m(p)} > z^{m-1}_{l_m(p)}$ et $z_{l_m(k)} \leq z^{m-1}_{l_m(k)}$; $k = 1, 2, ..., p$.

L'ensemble des solutions admissibles est partitionné et pour chaque sous noeud on résoud le problème (P_m). Les mêmes tests d'arrêt sont utilisables.

Méthode de J.Sylva & A.Crema [22]

La méthode développée par Crema et Sylva 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 (MOILP)

Proposition 2.2.2 *: Soient x_1, $x_2, ..., x_l$ des solutions efficaces du problème*

$$\begin{cases} "OPTIMISER" \ z_k = c^k x \quad k = 1, 2, ..., p \\ tq \ x \in D \end{cases} \qquad \text{(MOILP)}$$

Où $D = S \cap Z^n$ est l'ensemble des nombre entiers relatifs et $S = \{x \in R^n / Ax \leq b, x \geq 0\}$; $A \in R^{m \times n}, x \in R^n, b \in R^m$, $c \in R$, $k = 1, 2, ..., p$.
Posons $D_S = \{x \in Z^n | C^k x \leq C^k x_s \ , \forall s = 1, ..., l\}$. Si x^* est une solution efficace pour le problème :

$$(P_l) \begin{cases} " \max " \quad C^k x \\ tq \quad x \in (D - \bigcup_{s=1}^{l} D_s) \end{cases}$$

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

Proposition 2.2.3 *: Soient $x_1, x_2, ..., x_l$ des solutions efficaces du problème (MOILP). Posons $D_s = \{x \in Z^n \mid C^k x \leq C^k x_s, \forall s = 1, ..., l\}$. Si x^* est une solution optimale pour le problème mono-objectif*

$$(P_l^\lambda) \begin{cases} " \max " \quad \sum_{k=1}^{p} \lambda_k C^k x \\ tq \quad x \in (D - \bigcup_{s=1}^{l} D_s) \end{cases}$$

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

Théorème 2.2.4 *: Si x^* est une solution optimale du problème mono-objectif*

$$(P_l) \qquad \begin{cases} \max \quad \lambda^t C x \\ t.q \quad x \in S \end{cases}$$

pour un certain $\lambda \in \Lambda = \{\lambda \in \mathbb{R}^p / \sum_{k=1}^{p} \lambda_k = 1 \ / \ \lambda_k \geq 0\}$ alors, x^* est une solution efficace du problème multi-objectifs (MOILP)

Algorithme

Etape1 : Après avoir fixé 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 :

Cas1. Si (P_1) n'admet pas de solutions, il est en est de même pour (MOILP) .

Cas2. 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 l étapes du processus on peut avoir deux cas :

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 défini 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 :

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

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

Etape (l+1) : Résoudre le problème (P_{l+1})

$$(P_{l+1}) \begin{cases} \max \sum_{k=1}^{p} \lambda_k C^k x \\ \text{t.q} \quad x \in D \\ C^k x \leq (C^k x^s + 1) y_k^s - M_k(1 - y_k^s); \quad k = 1, ..., p \\ \sum_{k=1}^{p} y_k^s \geq 1 \ ; \quad y_k^s \in \{0, 1\}; \quad k = 1, ..., p \ ; s = 1, ..., l \end{cases}$$

Etape 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. 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 :

$$(P_{l+1}) \begin{cases} \max \sum_{k=1}^{p} \lambda_k C^k x \\ \text{t.q} \quad x \in D \\ C^k x \leq (C^k x^s + f_k) y_k^s - M_k(1 - y_k^s); \quad k = 1, ..., p \\ \sum_{k=1}^{p} y_k^s \geq 1 \ ; \quad y_k^s \in \{0, 1\}; \quad k = 1, ..., p \ ; s = 1, ..., l \end{cases}$$

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

Exemple 2.2.5 *: Pour illustrer la technique, nous considérons le (MOLIP) à deux fonctions objectifs suivant :*

$$(MOLIP) \begin{cases} \max \quad Z_1 = x_1 - 2x_2 \\ \max \quad Z_2 = -x_1 + 3x_2 \\ tp \quad x_1 - 2x_2 \leq 0 \\ x1, x2 \in \{0, 1, 2\} \end{cases}$$

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 :

$$(P_0) \begin{cases} \text{" max "} \quad Z_\lambda = x_1 + x_2 \\ tq \quad x_1 - 2x_2 \leq 0 \\ x_1, x_2 \in \{0, 1, 2\} \end{cases}$$

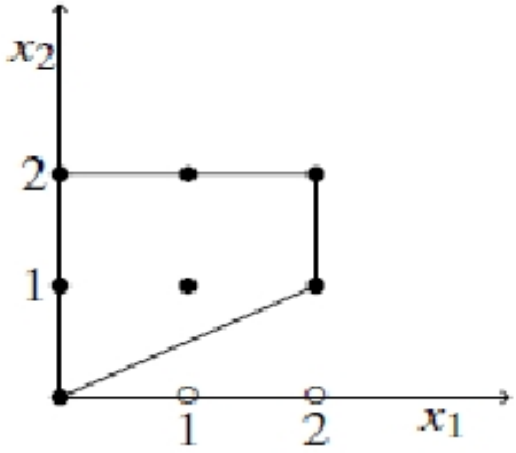

Fig 2.1 Espace de décision pour P_0

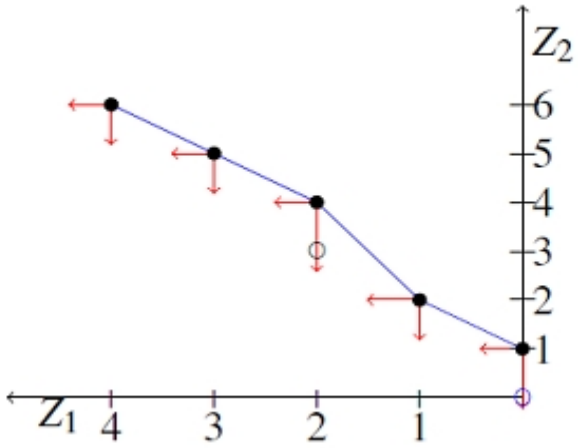

Fig 2.2 Espace des critères

La résolution du problème (P_0) donne $X_0 = (2, 2)$ de valeur optimale $Z_l(P_0) = 4$. Donc, X_0 est une solution efficace du problème initial (MOLIP) correspondant au vecteur non dominé $(-2, 4)$ (Voir figure 2.1).

Le deuxième problème à résoudre est :

$$(P_1) \begin{cases} Max \quad Z_\lambda = x_1 + x_2 \\ tq \quad x_1 - 2x_2 \leq 0 \\ x_1 - 2x_2 \geq -y_1^1 - 4(1 - y_1^1) \\ -x_1 + 3x_2 \geq 5y_2^1 - 2(1 - y_2^1) \\ y_1^1 + y_2^1 \geq 1 \\ x_1, \ x_2 \in \{0, 1, 2\} \\ y_1^1, \ y_2^1 \in \{0, 1\} \end{cases}$$

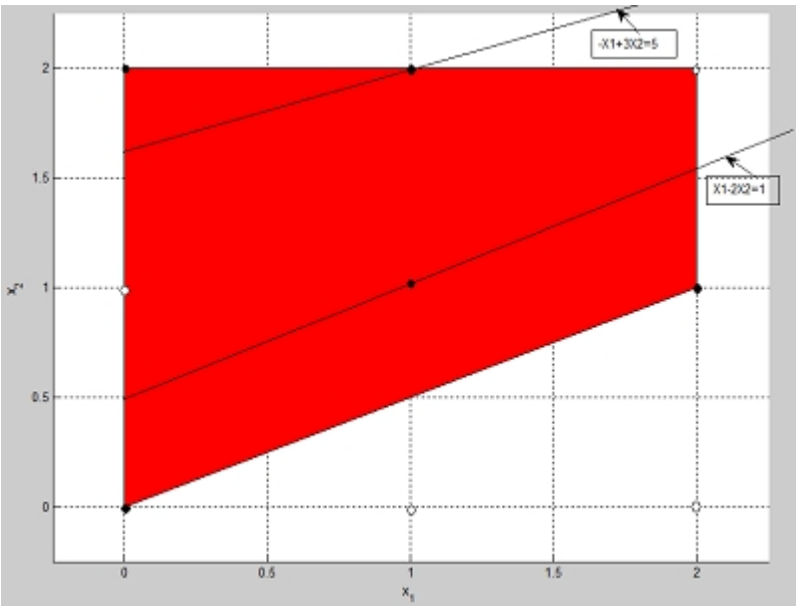

Fig 2.3 Espace de décision pour P_1

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, $X_1 = (2, 1)$ est un autre point efficace correspondant au couple non-dominé $(0, 1)$.(Fig2.3)

Formons le problème suivant (P_2) en ajoutant cinq contraintes à (P_1) :

$$(P_2) \begin{cases} Max \;\; Z_\lambda = x_1 + x_2 \\ tq \;\; x_1 - 2x_2 \leq 0 \\ x_1 - 2x_2 \geq -y_1^1 - 4(1 - y_1^1) \\ -x_1 + 3x_2 \geq 5y_2^1 - 2(1 - y_2^1) \\ x_1 - 2x_2 \geq y_1^2 - 4(1 - y_1^2) \\ -x_1 + 3x_2 \geq 2y_2^2 - 2(1 - y_2^2) \\ y_1^1 + y_2^1 \geq 1 \\ y_1^2 + y_2^2 \geq 1 \\ x_1, \; x_2 \in \{0, 1, 2\} \\ y_1^1, \; y_2^1 \in \{0, 1\} \\ y_1^2, \; y_2^2 \in \{0, 1\} \end{cases}$$

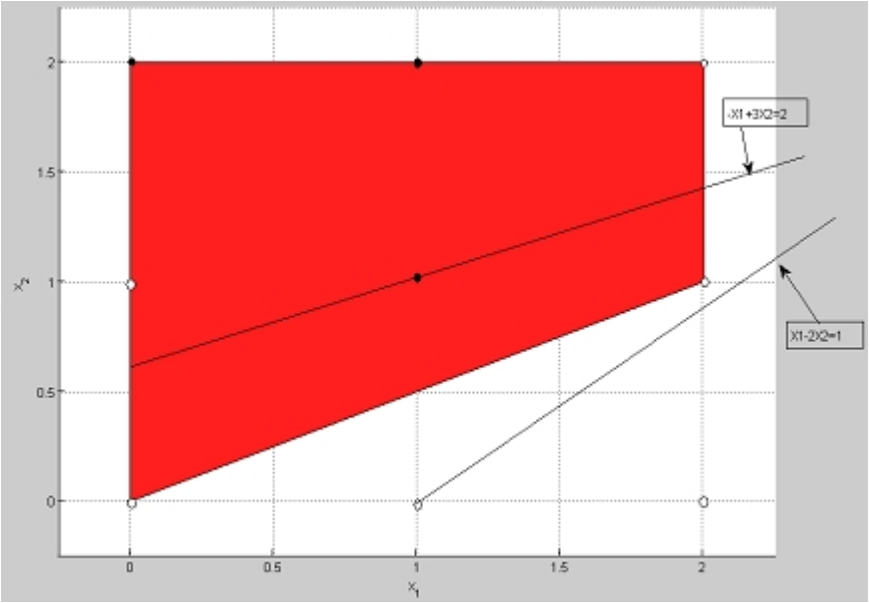

Fig 2.4 Espace de décision pour P_2

La solution optimale de 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 $X_2 = (1, 2)$ avec un vecteur de la valeur de la fonction objectif égale à $(-3, 5)$. (Fig 2.4)

Le pas suivant est d'ajouter des contraintes qui effacent seulement le point efficace X_2, on obtient le problème (P_3) :

$$(P_3)\begin{cases} Max \;\; Z_\lambda = x_1 + x_2 \\ tq \;\; x_1 - 2x_2 \leq 0 \\ x_1 - 2x_2 \geq -y_1^1 - 4(1 - y_1^1) \\ -x_1 + 3x_2 \geq 5y_2^1 - 2(1 - y_2^1) \\ x_1 - 2x_2 \geq y_1^2 - 4(1 - y_1^2) \\ -x_1 + 3x_2 \geq 2y_2^2 - 2(1 - y_2^2) \\ x_1 - 2x_2 \geq -2y_1^3 - 4(1 - y_1^3) \\ -x_1 + 3x_2 \geq 6y_2^3 - 2(1 - y_2^3) \\ y_1^1 + y_2^1 \geq 1 \\ y_1^2 + y_2^2 \geq 1 \\ y_1^3 + y_2^3 \geq 1 \\ x_1, \; x_2 \in \{0, 1, 2\} \\ y_1^1, \; y_2^1, y_1^2, \; y_2^2, y_1^3, \; y_2^3 \in \{0, 1\} \end{cases}$$

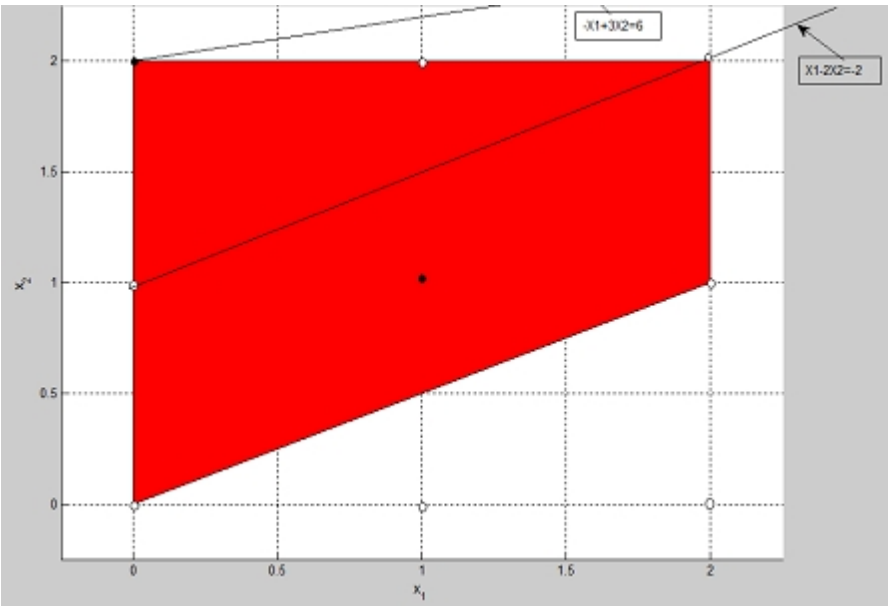

Fig 2.5 Espace de décision pour P_3

La solution optimale du problème (P_3) 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 objectif donnée par le vecteur $(-4, 6)$.(Fig 2.5)

Maintenant, problème (P_4) est défini par :

$$(P_4)\begin{cases} Max \;\; Z_\lambda = x_1 + x_2 \\ tq \quad x_1 - 2x_2 \leq 0 \\ x_1 - 2x_2 \geq -y_1^1 - 4(1 - y_1^1) \\ -x_1 + 3x_2 \geq 5y_2^1 - 2(1 - y_2^1) \\ x_1 - 2x_2 \geq y_1^2 - 4(1 - y_1^2) \\ -x_1 + 3x_2 \geq 2y_2^2 - 2(1 - y_2^2) \\ x_1 - 2x_2 \geq -2y_1^3 - 4(1 - y_1^3) \\ -x_1 + 3x_2 \geq 6y_2^3 - 2(1 - y_2^3) \\ x_1 - 2x_2 \geq -3y_1^4 - 4(1 - y_1^4) \\ -x_1 + 3x_2 \geq 7y_2^4 - 2(1 - y_2^4) \\ y_1^1 + y_2^1 \geq 1 \\ y_1^2 + y_2^2 \geq 1 \\ y_1^3 + y_2^3 \geq 1 \\ y_1^4 + y_2^4 \geq 1 \\ x_1,\ x_2 \in \{0, 1, 2\} \\ y_1^1,\ y_2^1,\ y_1^2,\ y_2^2,\ y_1^3,\ y_2^3,\ y_1^4,\ y_2^4 \in \{0, 1\} \end{cases}$$

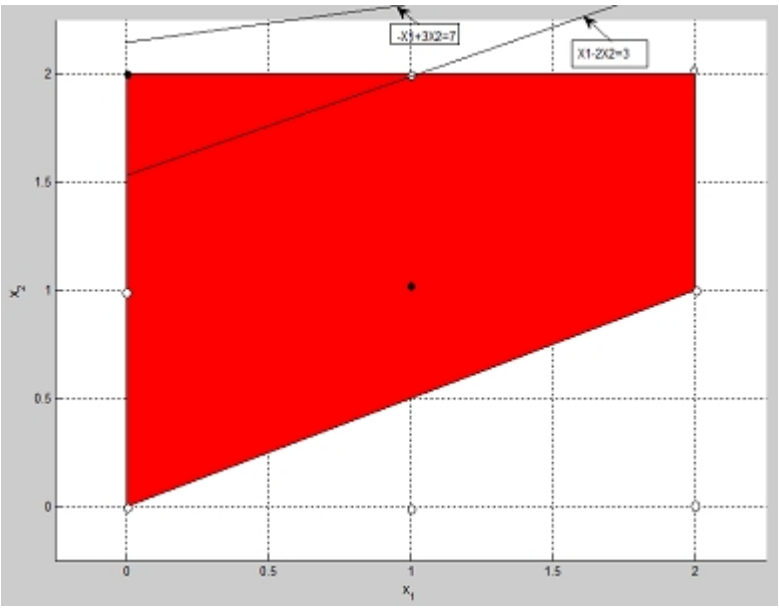

Fig 2.6 Espace de décision pour P_4

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 objectif égale au vecteur $(-1, 2)$. (Fig 2.6)

Le prochain problème à résoudre est le suivant :

$$(P_5)\begin{cases} Max \;\; Z_\lambda = x_1 + x_2 \\ tq \;\; x_1 - 2x_2 \leq 0 \\ x_1 - 2x_2 \geq -y_1^1 - 4(1 - y_1^1) \\ -x_1 + 3x_2 \geq 5y_2^1 - 2(1 - y_2^1) \\ x_1 - 2x_2 \geq y_1^2 - 4(1 - y_1^2) \\ -x_1 + 3x_2 \geq 2y_2^2 - 2(1 - y_2^2) \\ x_1 - 2x_2 \geq -2y_1^3 - 4(1 - y_1^3) \\ -x_1 + 3x_2 \geq 6y_2^3 - 2(1 - y_2^3) \\ x_1 - 2x_2 \geq -3y_1^4 - 4(1 - y_1^4) \\ -x_1 + 3x_2 \geq 7y_2^4 - 2(1 - y_2^4) \\ x_1 - 2x_2 \geq -4(1 - y_1^5) \\ -x_1 + 3x_2 \geq 3y_2^5 - 2(1 - y_2^5) \\ y_1^1 + y_2^1 \geq 1 \\ y_1^2 + y_2^2 \geq 1 \\ y_1^3 + y_2^3 \geq 1 \\ y_1^4 + y_2^4 \geq 1 \\ y_1^5 + y_2^5 \geq 1 \\ x_1, \; x_2 \in \{0, 1, 2\} \\ y_1^1, \; y_2^1, \; y_1^2, \; y_2^2, \; y_1^3, \; y_2^3, \; y_1^4, \; y_2^4, \; y_1^5, \; y_2^5 \in \{0, 1\} \end{cases}$$

Conclusion : Comme le problème (P_5) est irréalisable, le processus s'arrête. L'ensemble des points efficaces est $E = \{(2, 2)^t, (2, 1)^t, (1, 2)^t, (0, 2)^t, (1, 1)^t\}$ et l'ensemble des

solutions non dominées est $Opt = \{(-2,4)^t, (0,1)^t, (-3,5)^t, (-4,6)^t, (-1,2)^t\}$.

Dans le chapitre suivant nous allons introduire des méthodes nouvelles de résolution des problèmes MOILP dans l'espace des variables de décision.

2.2.3 Résolution dans l'espace de décision

Cette vue des problèmes (MOILP) du côté de ses variables de décisions permet de développer d'autres approches basées sur les méthodes de coupes évitant le problème de solutions non sopportées.

Depuis plus d'une vingtaine d'années, les chercheurs essayent de réadapter les techniques utilisées pour le cas continu dans les problèmes à variables discrètes. Cette "projection" de méthode du cas continu au cas discret est très dangereuse si elle n'est pas tout à fait éronée (les solutions efficaces non supportées sont souvent ratées)

Les méthodes décrites précédemment sont développées en tenant compte de cette problèmatique, mais malheureusement dans l'architecture des méthodes est toujours présentée l'idée de base utilisée dans la solution des problèmes (MOLP).

Les premiers auteurs, à notre connaissance, qui ont essayé de se libérer de cette" dépendance" sont Gupta et Malhotra [Gupta & Malhotra] qui ont introduit une nouvelle méthode exacte de caracterisation de l'ensemble de toutes les solutions efficaces d'un problème (MOILP) en un nombre fini d'itérations. La méthode qu'ils ont développée donne effectivement pour quelque exemple toutes les solutions efficaces, mais elle échoue dans quelques cas. Un contre exemple sera présenté pour localiser la défaillance de leur algorithme.

Notations

Les notations que nous présentons sont propres aux auteurs et afin de reporter exactement leur approche nous préférons les conserver. Considerons le problème

$$(P_1) \quad \begin{cases} \max \ z_1 = C^1 X \\ \text{t.q} \quad X \in D \end{cases}$$

On pose :
$z_1^1 = z_1^*$ la valeur optimale de z_1 obtenue dans le problème (P_1).
X_1^* la solution optimale entière de z_1^*.
$(z_1^1, z_2^1, ..., z_p^1)$ le premier p-uplet non dominé, avec $z_1^* = z_i(X_1^*)$ pour $i = 2, 3, ..., 3$
X_1 la solution correspondant à $(z_1^1, z_2^1, ..., z_p^1)$.
B_1 la base associée avec X_1.
$a_{1,j}$ le vecteur d'activité de $x_{1,j}$ par rapport à la region tronquée .
$y_{1,j} = (B_1)^{-1} a_{1,j}$.
$I_1 = \{j / a_{1,j} \in B_1\}$.
$N_1 = \{j / a_{1,j} \notin B_1\}$.

$j_1 = \left\{ j/j \in N_1 \text{et } z^1_{1,j} - c^1_j = 0 \right\}$,où $z^1_{1,j} = C^1_{B_1} y_{1,j}$ et c^1_j est la j^{ème} composante du vecteur C^1; $C^1_{B_1}$est le vecteur des coefficients coûts des variables de base associées à B_1

$\Gamma_1 = \left\{ j \in N_1 \ / \ z^1_{1,j} - c^1_j > 0 \text{ et } \ z^i_{1,j} - c^1_j < 0 \text{ pou au moins un } i \in \{2,3,...,p\} \right\}$ où, $z^i_{1,j} = C^1_{B_1} y_{1,j}$ et c^i_j est la j^{éme}composante du vecteur C^i et $C^i_{B_1}$ est le vecteur des coefficients coûts des variables de base associées à B_1 du vecteur C^i.

Pour $k \geq 2$;

$X_k = (x_{k,j})$ est la solution optimale entière obtenue dans la régoin tronquée S' aprés avoir appliqué la coupe $\sum_{j \in N_{K=1} \setminus \{j_{k-1}\}} \geq 1$.

B_k est base associée à la solution X_k.

$a_{k,\,j}$ est le vecteur d'activité de $x_{i,\,j}$ par rapport à la région tronquée.

$y_{k,\,j} = (B_1)^{-1} a_{k,\,j}$

$I_k = \{ j/a_{k,\,j} \in B_k \}$.

$N_k = \{ j/a_{k,\,j} \notin B_k \}$.

$\Gamma_k = \left\{ j \in N_1 \ / \ z^1_{k,j} - c^1_j > 0 \text{ et } \ z^i_{k,j} - c^1_j < 0 \text{ pou au moins un } i \in \{2,3,...,p\} \right\}$

où $z^i_{k,\,j} = C^1_{B_k} y_{k,j}$ et c^i_j est la j^{ème}composante du vecteur C^i et $C^i_{B_k}$ est le vecteur des coefficients coûts des variables de base associées à B_k du vecteur C^i.

$$E_{j_k} = \left\{ \begin{array}{c} (x_i) \in \mathbb{R}^n | x_i = x_{ki} - \theta_{j_k} y_{k,\,ij_k} \\ x_{jk} = \theta_{j_k} \\ x_\alpha = 0 \text{ pour tout } \alpha \in N_k \setminus \{j_k\} \end{array} \right\} (*)$$

où θ_{jk} est la valeur entière comprise entre 0 et $\min_{i \in I_k} \left\{ \frac{x_{ki}}{y_{k,\,i\,j_k}}; y_{k,\,i\,j_k} > 0 \right\}$; θ_{j_k} , $\theta_{jk} \times y_{k,\,ij_k}$ sont des entiers pour $i \in I_k$.

Les résultats suivants suggèrent une coupe qui peut être vue comme une généralisation de la coupe classique de Dantzig de façon que toute l'arrête est tronquée au lieu d'une seule solution.

Théorème 2.2.6 *: [08] Toutes les solutions entières réalisables du problème (P_1) alternative à X_1 sur l'arrête E_{j_1}de la région D appartiennent au demi-espace* $\sum_{j \in N_1 \setminus \{j_1\}} x_j < 1$.

Démonstration. Soit X_1 une solution réalisable de (P_1).

$AX_1 = b, AX_1 = \sum_{i \in I_1} a_{1,\,i} x_{1,\,i} = b$.Pour $j_1 \in J_1$,on peut écrire

$\sum_{i \in I_1} a_{1,\,i} x_{1,\,i} - \theta_{j_1} a_{1,\,j_1} + \theta_{j_1} a_{1,\,j_1} = b$ où θ_{j_1} est un scalaire positif.

$\sum_{i \in I_1} a_{1,\,i} x_{1,\,i} - \theta_{j_1} \left(\sum_{i \in I_1} a_{1,\,i}\, y_{1,i\,j_1} \right) + \theta_{j_1} \sum_{i \in I_1} a_{1,\,j_1} = b$;

$\sum_{i \in I_1} a_{1,\,i} \ (x_{1,\,i} - \theta_{j_1} y_{1,i\,j_1}) + \theta_{j_1} a_{1,\,j_1} = b$;

pour $0 < \theta_{j_1} < \min_{i \in I_1} \left\{ \frac{x_{ki}}{y_{1,\,i\,j_1}}; y_{1,\,i\,j_1} > 0 \right\}$, X_2 est défini par :

$$X_2 = \left\{ \begin{array}{c} x_{2,i} = x_{1,i} - \theta_{j_1} \times y_{1,\,i\,j_1} \ i \in I_1 \\ x_{2,i} = \theta_{j_1} \\ x_{2,i} = 0 \text{ pour tout } \alpha \in N_1 \setminus \{j_1\} \end{array} \right.$$

est une nouvelle solution réalisable entière de (P_1) avec θ_{j_1}une quantité entière positive et $\theta_{j_1} \times y_{1,\,i\,j_1}$sont des entiers pour tout $i \in I_1$. On montre que $Z_1(X_2) = Z_1(X_1)$.

En effet, $Z_1(X_2) = C^1X_2 = \sum_{i\in I_1} c_i^1 x_{2,i} + c_{j_1}^1 x_{2,j_1} + \sum_{\alpha\in N_1\setminus\{j_1\}} c_\alpha^1 x_{2,\alpha}$

$= \sum_{i\in I_1} c_i^1 \left(x_{1i} - \theta_{j_1} \times y_{1,\,i\,j_1}\right) + c_{j_1}^1\theta_{j_1} = \sum_{i\in I_1} c_i^1 x_{1i} - \sum_{i\in I_1} c_i^1\theta_{j_1} \times y_{1,\,i\,j_1} + c_{j_1}^1\theta_{j_1}$

$= \sum_{i\in I_1} c_i^1 x_{1i} - \theta_{j_1}\sum_{i\in I_1} c_i^1 y_{1,\,i\,j_1} + c_{j_1}^1\theta_{j_1} = \sum_{i\in I_1} c_i^1 x_{1i} - \theta_{j_1}\left(\sum_{i\in I_1} c_i^1 \times y_{1,\,i\,j_1} - c_{j_1}^1\right)$

$= Z_1(X_1) - \theta_{j_1}(z_{1,j}^1 - c_{j_1}^1)$.

Comme $j_1 \in J_1$, alors $z_{1,j}^1 - c_{j_1}^1 = 0$ .Donc $Z_1(X_2) = Z_1(X_1)$.X_2 est une solution réalisable de (P_1), alternative à X_1, sur l'arrête E_{j_1} définie par :

$$E_{j_1} = \left\{ \begin{array}{c} (x_1) \in \mathbb{R}^{(|I_1|+|N_1|)} / x_{2,i} = x_{1,i} - \theta_{j_1} \times y_{1,\,i\,j_1}\ i \in I \\ x_{2,i} = \theta_{j_1} \\ x_{2,\alpha} = 0 \text{ pour tout } \alpha\ N_1\setminus\{j_1\} \end{array} \right.$$

On a , $\sum_{j\in N_1\setminus\{j_1\}} x_{2,j} < 1$; et comme $x_{2,\,j} = 0\ \forall\ j \in N_1\setminus\{j_1\}$. Donc, le point X_2 appartient au demi- espace $\sum_{j\in N_1\setminus\{j_1\}} x_j < 1$. ■

Corollaire 2.2.7 *: Une solution réalisable du problème (P_1) qui n'est pas sur l'arrête E_{j_1}, $j_1 \in \Gamma_1$ et qui émane de la solution optimal du problème (P_1) appartient au demi-espace* $\sum_{j\in N_1\setminus\{j_1\}} x_j \geq 1$.

Démonstration. Soit $\tilde{X} = (\tilde{x}_j)_{i\in I_1\cup N_1}$une solution réalisable du problème (P_1) qui n'est pas sur l'arrête E_{j_1} tel que :

$\sum_{j\in N_1\setminus\{j_1\}} \tilde{x}_j < 1$. Alors $\tilde{x}_j = 0\ \forall \in N_1\setminus\{j_1\}$. Pour $j = j_1$ on a

(*) $\tilde{x}_{j_1} > 0, (\tilde{X}$ alternative à $X_1)$, alors $\tilde{X}$ doit être sur une arrête dans la direction du vecteur $a_{1j_1}, j_1 \in \Gamma_1$.

Si $\tilde{x}_{j_1} > \min_{i\in I_1}\left\{\frac{x_{1i}}{y_{1,i\,j_1}}; y_{k,ij_1} > 0\right\} = \frac{x_{1q}}{y_{1,i\,j_1}}$(par exemple), alors $\tilde{x}_q = \tilde{x}_{1q} - \tilde{x}_{j_1}y_{1,qj_1} < 0$, ceci implique l'infaisabilité de $\tilde{X}_1$ en contradiction avec l'hypothèse.

Si $\tilde{x}_{j_1}$ est un entier positif tel que $\tilde{x}_{j_1} \leq \min_{i\in I_1}\left\{\frac{x_{1i}}{y_{1,i\,j_1}}; y_{k,ij_1} > 0\right\}$,ceci aussi implique que $\tilde{X}$ appartient à l'arrête E_{j_1},ce qui n'est pas vrai.

Donc, $\tilde{x}_j > 0$ pour au moins un $j \in N_1\setminus\{j_1\}$.

(*) $\tilde{x}_{j_1} = 0$,($\tilde{X}$ n'est pas alternative à X_1) alors, l'ensemble des indices des variables hors base dans le tableau optimal correspondant à $\tilde{X}$ est le même que N_1 car $N_1 = N_1\setminus\{j_1\}\cup\{j_1\}$,donc, l'ensemble des indices des variables de base dans le tableau optimal correspondant à $\tilde{X}$ est le même que B. Par conséquent $\tilde{X} = X$, ce qui n'est pas le cas. Donc, $\tilde{x}_j > 0$ pour au moins $j \in N_1\setminus\{j_1\}$. Il est montré alors que $\tilde{x}_j > 0$ pour au moins un indice $j \in N_1\setminus\{j_1\}$. Ce qui implique que $\tilde{X}$ appartient au demi-espace fermé $\sum_{j\in N_1\setminus\{j_1\}} x_j \geq 1$. ■

Procédure de Gupta & Malhotra

Etape 1 : Résoudre le problème (P_1)

Case 1. Si la solution optimale X_1^* de (P_1) est unique, on prend Z_1^1 la valeur optimale de Z_1 est $Z_i^1, i \in \{2, 3, ..., p\}$ les valeurs des fonctions objectifs au point X_1^*,donc on a un p-uplet $(Z_1^1, Z_2^1, ..., Z_p^1)$.

Case 2. Si le problème (P_1) possède des solutions alternatives $(J_1 \neq \emptyset)$, on détermine toutes les solutions alternatives à X_1^* et pour chacune d'elles on enregistre le p-uplet correspondant. On élimine toutes les solutions dominées et on garde seulement celles qui sont non dominées.

Considérons le premier $p - uplet$ avec plus grande valeur Z_2,si toutes les secondes composantes sont identiques choisissons le $p - uplet$ avec plus grande valeur dans la troisième composante, et le processus continu de la même manière jusqu'à ce qu'un noeud se présente. Soit $(Z_1^1, Z_2^1, ..., Z_p^1)$ le premier $p - uplet$.

Etape 2 : Choisir $j_1 \in \Gamma_1$.Trouver le rapport minimum θ de l'opération pivot.

Case 3. $\theta < 1$ dans ce cas, il est ignoré, car si pour un $j_1 \in \Gamma_1 \neq \varnothing$ on a $\theta < 1$,alors aucune solution entière réalisable ne peut être obtenue sur l'arrête E_j et ceci est pareil dans toutes les étapes de la procédure.

Case 4. $\theta \geq 1$, on détermine toutes les solutions réalisables entières sur l'arrête E_{j_1} chacune des solutions donne un $p - uplet$ potentiellement non dominé de la forme $(Z_1, Z_2, ..., Z_p)$ avec $Z_1 < Z_1^1$ et $Z_i > Z_i^1$ pour au moins un i. Soit Opt_1 l'ensemble de tous ces $p - uplets$.

Tronquer l'arrête E_{j_1}par la coupe suivante : $\sum_{j \in N_1 - \{j_1\}} x_j \geq 1$ et appliquer la méthode dual du simplexe et la coupe de gomory si nécessaire pour obtenir une solution réalisable entière X_2 dans la région tronquée.

Etape 3 : Soit $j_2 \in \Gamma_2$, scaner l'arrête E_{j_2} pour chercher d'éventuelles solutions réalisables entières. Enregistrer tous les $p - uplets$ non dominés et définir de nouveau l'ensemble dans la liste Opt_1 (garder seulement les solutions non dominées) et définir Opt_2.

Tronquer l'arrête E_{j_2} par la coupe $\sum_{j \in N_2 - \{j_2\}} x_j \geq 1$ et chercher de nouveau la solution optimale entière dans la région tronquée.

Etape générale k : Choisir $j_{k-1} \in \Gamma_{k-1}$ et explorer l'arrête $E_{j_{k-1}}$. Evaluer tous les critères sur chacune des solutions trouvées et éliminer les $p-uplet$ dominés. Former le nouvel ensemble des solutions non dominés Opt_{k-1}à partir de Opt_{k-2} en ajoutant les nouveaux

vecteurs non dominés.Tronquer ensuite l'arrête $E_{j_{k-1}}$ par la coupe $\sum_{j \in N_{k-1}-\{j_{k-1}\}} x_j \geq 1$ et déterminer la solution optimale entière X_k dans la région tronquée.

Etape finale n : Le processus prend fin à l'étape n quand :
a) $\Gamma_n = \varnothing$ et $z_{nj}^{C^1} - c_j^1 > 0, \forall j \in N_n$;

b) $\Gamma_n \neq \varnothing$ mais pour tout $j \in \Gamma_n$ les solutions entières réalisables sur l'arrête E_{j_n} ne sont pas efficaces.

Illustration Considérons le prolème multiobjectif en nombres entiers suivant :

$$\begin{cases} \max\ Z_1 = x_1 - 3x_2 \\ \max\ Z_2 = x_1 + 3x_2 \\ \text{t.q.} \\ x_1 + 2x_2 \leq 8 \\ 2x_1 + x_2 \leq 7 \\ x_1 - 2x_2 \leq 1 \\ x_1, x_2 \geq 0 \text{ et entiers} \end{cases}$$

La résolution de (P_1) donne le tableau optimal suivant :

Tableau I

Base	valeur de variables de base	x_1	x_2	x_3	x_4	x_5
x_3	7	0	4	1	0	-1
x_4	5	0	5	0	1	-2
x_1	1	1	-2	0	0	1
$Z_J^{C^1} - C_j^1$	1	0	1	0	0	1
$Z_J^{C^2} - C_j^2$	1	0	1	0	0	1

La solution optimale $X_1 = X_1^* = (1, 0)$ est unique et le premier couple non dominé est $(Z_1^1, Z_2^1) = (1, 1)$.$Opt_0 = \left\{ \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right\}$.

$k = 1$,On a :
$I_1 = \{1, 3, 4\}$, $N_1 = \{2, 5\}$ et
$\Gamma_1 = \{2\}$
$j_1 = 2; \theta_{12} = \min\left\{\frac{7}{4}, \frac{5}{5}\right\} = \frac{5}{5} \Rightarrow \theta_{12}^0 = 1$. Il y a une seule solution alternative entière sur l'arrête E_5 donnée par :

$$\begin{cases} x_3^1 = 7 - (4) = 3 \\ x_4^1 = 5 - (5) = 0 \\ x_1^1 = 1 - (-2) = 3 \\ x_2^1 = 1 \\ x_5^1 = 0 \end{cases}$$

Donc $Z^{1,1} = \begin{pmatrix} 0 \\ 6 \end{pmatrix}$qui est non dominée comparée avec la premiére solution dans Opt_0.

Poser $Opt_1 = Opt_0 \cup \left\{ \begin{pmatrix} 0 \\ 6 \end{pmatrix} \right\}$ et tronquer par $x_5 \geq 1 \Rightarrow -x_5 + x_6 = -1$.

Cette contrainte est ajoutée au bas du tableau optimal précédant et la méthodes duale du simplexe et la coupe de Gomory sont appliquées. On obtient le tableau optimal suivant :

Tableau II

Base	valeur de variables de base	x_1	x_2	x_3	x_4	x_5	x_6
x_3	8	0	4	1	0	0	-1
x_4	7	0	5	0	1	0	-2
x_1	0	1	-2	0	0	0	1
x_5	1	0	0	0	0	1	-1
$Z_j^{C^1} - C_j^1$	0	0	1	0	0	0	1
$Z_j^{C^2} - C_j^2$	0	0	-5	0	0	0	1

La solution optimale entière est $X_2 = (0, 0)$ et $(Z_1^2, Z_2^2) = (0, 6)$ qui est non dominée par rapport aux solutions précédentes dans Opt_1.

$k = 2$,

$I_2 = \{1, 3, 4, 5\}$, $N_2 = \{25, 6\}$, $\Gamma_2 = \{2\}$ et $\theta_{22}^0 = 1$.

La solution alternative sur l'arrête E_2 est : $\{x_3^1 = 4, x_4^1 = 2, x_2^1 = 1, x_5^1 = 1, x_6^1 = 0\}$ .Ceci donne un couple $\begin{pmatrix} -1 \\ 5 \end{pmatrix}$ qui est dominé par la solution $\begin{pmatrix} 0 \\ 6 \end{pmatrix}$. Donc, pour tout $j \in \Gamma_2$ les solutions sont dominées . D'après Gupta et Malhotra la procédure s'arrête. Or à cette étape il y'à d'autre solutions réalisables entières qui sont efficaces mais pas détectées par cette procédure. A titre d'exemple, les solution $(1, 0), (2, 2), (0, 4)$.

2.2.4 Problème à variables binaires

Un nombre considérable de problèmes en nombres entiers formulés comme des problèmes à variables qui ne peuvent prendre que deux valeurs 0 ou 1. Ce type de problème est noté par (MOBLP) ; citons par exemple le problème d'affectation d'une personne à une tâche, production d'une pièce,...etc.

La formulation mathématique est donnée par :

$$\left\{ \text{" max"} \quad z_k = c^k x \,;\, k = 1, 2, ..., p \right\} \qquad \text{(MOBLP)}$$

où $S^{'} = \{x \in B^n |\ Ax \leq b\}\,;\ B = \{0, 1\}$. Il est clair que le nombre de solutions varie exponentiellement en fonction de n , et pour identifier les solutions non dominées il est nécessaire de développer les méthodes d'énumération implicite. Ce qui sera vu dans cette section avec une des principales méthodes de la littérature.

Méthode de Bitran [05]

Bitran a considéré d'abord le problème :

(P_{B_1}) $\left\{\text{" max"}\ z_k = c^k x\,;\ \ k = 1, 2, ..., P \right.$ avec $S_1 = B^n$, l'ensemble des sommets de l'hypercube unité dans $\mathbb{R}^n$. On a $x \in E(P_{B_1}) \cap S \Rightarrow x \in E$(MOBLP)

Ceci veut dire qu'une solution efficace pour le problème (P_{B_1}) est réalisable pour le problème (MOBLP) est aussi efficace pour le problème (MOBLP).

$x \notin E(P_{B_1}) \nRightarrow x \notin E$(MOBLP).

Une solution qui n'est pas efficace pour le problème (P_{B_1}) n'est pas nécessairement non efficace pour le problème (MOBLP). L'algorithme de Bitran peut être décrit en trois étapes .

1. Caratérisation de l'ensemble des solutions efficaces du problème (P_{B_1})

2. Parmi ces solutions, on détermine celles qui sont réalisables pour le problème (MOBLP).

3. Adjoindre ensuite les autres solutions efficaces de (MOBLP) non trouvées dans 1.

Caractésation de (P_{B_1})

(a) Considérons l'ensemble $V = \left\{v^t \in \mathbb{R}^n, t \in T\ |Cv^t \geq 0, v_j^t = 0, 1 \text{ ou} - 1\ \forall j\right\}$ dit ensemble de direction de préférence. On dit alors que x domine $x^{'}$dans la direction v^t si et seulement si $x^{'} = x + v^t$, donc $Cx^{'} \geq Cx$.

Bitran a montré que l'ensemble des points dominés de S_1 dans la direction v^t par un autre point de S_1 noté par $M(v^t) = \{x^{t_r}\ |r = 1, 2, ..., R\}$ avec :

$$x^{t_r} = \begin{cases} 0 \ \text{ si } v_j^t = 1 \\ 1 \ \text{ si } \ v_j^t = -1 \\ 0 \text{ ou } 1 \text{si } v_j^t = 0 \end{cases}$$

peut être déterminé par l'ensemble des solutions optimales du problème $\min_{x \in S'_1} v^t x$ où $S'_1 = \{x \in \mathbb{R}^n \mid 0 \leq x \leq 1, \forall j\}$. L'ensemnble E (P_{B_1}) est déterminé à partir de la résolution successive des problèmes relaxés notés (PR_{B_1}) pour différentes directions $v^t, t \in T$. Ce dernier permet de trouver des ensembles $M(v^t)$ puis $E(P_{B_1}) = \overline{\underset{t \in T}{\cup} M(v^t)}$. ($\overline{A}$ veut dire complément de l'ensemble A). Notons que quelques directions seulement sont examinées .

(b) Détermination de l'ensemble $E_1 = E(P_{B_1}) \cap S$. Pour cela, il suffit de tester l'admissibilité des solutions de $E(P_{B_1})$.

(c) Caractérisation de l'ensemble E(MOBLP). Pour déterminer l'ensemble des solutions non efficaces dans (P_{B_1}) et efficaces dans (MOBLP) noté E_2, on utilise la propriété suivante.

$x \in E_2 \Rightarrow x + v^t \notin S$, $\forall\ v^t$, $t \in T : x \in M(v^t)$

Alors E(MOBLP) $= E_1 + E_2$.

Chapitre 3

Implémentation de la méthode de Sylva & Crema

3.1 Introduction

Pour montrer l'efficacité de la méthode de J.Sylva et A.Crema, nous avons implémenté la méthode en utilisant la programmation MATLAB 7.0.

MATLAB (« matrix laboratory ») est un langage de programmation de quatrième génération et un environnement de développement ; il est utilisé à des fins de calcul numérique. Développé par la société The MathWorks, MATLAB permet la manipulation de matrice, afficher des courbes et des données, mettre en œuvre des algorithmes, créer des interfaces utilisateurs, et peut s'interfacer avec d'autres langages comme le C, C++, Java, et Fortran. Les utilisateurs de MATLAB (environ un million en 2004[1]) sont de milieux très différents comme l'ingénierie, les sciences et l'économie dans un contexte aussi bien industriel que pour la recherche. Matlab peut s'utiliser seul ou bien avec des toolbox (boite à outils).

3.2 Illustration

On a appliqué la méthode sur l'exemple précédent (Illustration de la méthode de J.Sylva & A.Crema)

$$(\text{MOILP}) \begin{cases} \max \quad Z_1 = x_1 - 2x_2 \\ \max \quad Z_2 = -x_1 + 3x_2 \\ tp \quad x_1 - 2x_2 \leq 0 \\ x1, x2 \in \{0, 1, 2\} \end{cases}$$

Avec $-M_1 = -4, -M_2 = -2$, 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 $Z_\lambda = x_1 + x_2$.

Sur MATLAB :

On saisit le problème comme suit :

```
type= 'max';
k = [1 -2];
e = [-1 3];
lambda = [4 3];
c =lambda(1).*k +lambda(2).*e;
A =[1 -2;1 0;0 1];
b =[0 2 2];
rel =('<<<');
M1 = 4;
M2 = 2;
>> SCrema(type, c, A,rel, b,k ,e ,M1,M2)
```

Programme de la fonction SCrema :

```
function  SCrema1(type, c, A,rel, b,k ,e ,M1,M2)
str = 'Voulez-vous dériger la progression des calculs ?';
question_ans = questdlg(str,'Faire un choix Window','Oui','Non','No');
 if  strcmp(question_ans,'Oui')
    p1 = 'O';
 else
    p1 = 'N';
 end
 if  type == 'min'
    tp = -1;
 else
    tp = 1;
 end
   [m,n] = size(A);
   nlv = n;
   b = b( :);
   c = c( :)';
   A1 = A;
%Faire appel à la mèthode de simplexe
 if p1 == 'O'
    [A,subs] = simplexe(type,c,A,b,p1);
 else
    [A,subs] = simplexe(type,c,A,b);
 end
%Le programme d"ajout de contraintes :
 disp('La coupe de Gomory');
 disp('_____________________________')
 disp('La contraintes ajoutée :');
```

```
 [m,n] = size(A) ;
 d = A(1 :m-1,end) ;
 pc = fractp(d)
 tbn = 0 ;
%Résoudre le nouveau pb avec la méthode dual de simplexe :
  if p1 == 'O'
disp(sprintf('_____________________________________________________________'))
disp(sprintf('\n Les tableaux de la Methode du Dual de Simplexe'))
disp(sprintf('_____________________________________________________________'))
  end
% Faire des itérations jusquà ce qu"il n"y est plus de partie fractionnaire
 while norm(pc,'inf') > eps
   [el,i] = max(pc) ;
    nr = A(i,1 :n-1) ;
    nr = [-fractp(nr) 1 -el] ;
    B = [A(1 :m-1,1 :n-1) zeros(m-1,1) A(1 :m-1,end)] ;
    B = [B ;nr ;[A(m,1 :n-1) 0 A(end,end)]] ;
    A = B ;
   [m,n] = size(A) ;
   [bmin, row] = min(A(1 :m-1,end)) ;
  while   bmin < 0 & abs(bmin) > eps
          col = MRTD(A(m,1 :n-1),A(row,1 :n-1)) ;
     if  p1 == 'O'
         disp(sprintf('\n pivot row-> %g pivot column->row,col')) ;
          tbn = tbn + 1 ;
          disp(sprintf('\n Tableau %g',tbn)) ;
         A ;
         disp(sprintf(' Appuyez sur une touche pour continuer ...\n')) ;
        pause
      end
     if isempty(col)
        disp(sprintf('\n L"algorithme ne trouve pas la solution optimale.')) ;
       fin=1 ;
     return
      end
  A(row, :)= A(row, :)./A(row,col) ;
  subs(row) = col ;
for i = 1 :m
   if i ~= row
       A(i, :)= A(i, :)-A(i,col)*A(row, :) ;
   end
end
   [bmin, row] = min(A(1 :m-1,end)) ;
end
```

```
  d = A(1 :m-1,end) ;
  pc = fractp(d) ;
end
if p1 == 'O'
  disp(sprintf('\n tableau final '))
  A
  disp(sprintf(' Appuyez sur une touche pour continuer ...\n')) ;
pause
 x = zeros(n-1,1) ;
 x(subs) = A(1 :m-1,end) ;
 x = x(1 :nlv) ;
  disp(sprintf('\n Le problème admet une solution optimale finie \n\n')) ;
 disp(sprintf('\n Valeurs des variables :\n')) ;
for i=1 :nlv
  disp(sprintf(' x(%d)= %g ',i,x(i)))
end
  disp(sprintf('Valeur des fonctions objectifs Z1 et Z2 :')) ;
    z1= fonc1(x(1),x(2))
    z2= fonc2(x(1),x(2))
  disp(sprintf('\n valeur optimale de la fonction objectif :\n')) ;
  disp(sprintf(' z(lambda)= %f',tp*A(m,n)))
  disp(sprintf(' Appuyez sur une touche pour continuer ...\n')) ;
pause
end
% programme de la méthode de J.Sylva & A.Crema
% On fait appel au pb de départ :
% La matrice A, les couts C, etc
% On rajoute les contraintes de la méthode
% Tant que le nouveau pb est réalisable faire une nouvelle itération
I=0 ;
fin=0 ;
k=-k ;
e=-e ;
while fin ~=1
type= 'max' ;
A1
A = A1 ;
[m,n]= size(A) ;
A =[A zeros(m,2)] ;
L1= z1+1+M1
L2= z2+1+M2
k=k( :)' ;
e=e( :)' ;
b=b( :)' ;
```

```
d1=[k zeros(1,I) L1 0]
d2=[e zeros(1,I) 0 L2]
d3=[0 0 zeros(1,I) 1 0]
d4=[0 0 zeros(1,I) 0 1]
d5=[0 0 zeros(1,I) 1 1]
A = [A ; d1 ; d2 ; d3 ; d4 ; d5]
A1=A ;
b =[b M1 M2 1 1 1] ;
rel =('<<<<<<<<><<<<><<<<><<<<><<<<><<<<>') ;
disp('Application de la méthode de deux phases') ;
if p1 == 'O'
  [A, subs,n1] = simplex2p(type,c,A,rel,b,p1) ;
else
 [A, subs,n1] = simplex2p(type,c,A,rel,b) ;
end
  disp(sprintf(' Appuyez sur une touche pour continuer ...\n'))
pause
%Le programme d'ajout de contraintes :
  disp(sprintf(' La coupe de Gomory :')) ;
  [m,n] = size(A) ;
   d = A(1 :m-1,end) ;
  pc = fractp(d)
   tbn = 0 ;
%Résoudre le nouveau pb avec la méthode du dual de simplexe :
if p1 == 'O'
 disp(sprintf('_____________________________________________________
  disp(sprintf('\n Application de la Methode du Dual de Simplexe '))
 disp(sprintf('_____________________________________________________
end
% Faire des itérations jusquà ce qu'il n'y est plus de partie fractionnaire
while norm(pc,'inf') > eps
   [el,i] = max(pc) ;
   nr = A(i,1 :n-1) ;
   nr = [-fractp(nr) 1 -el] ;
   B = [A(1 :m-1,1 :n-1) zeros(m-1,1) A(1 :m-1,end)] ;
   B = [B ;nr ;[A(m,1 :n-1) 0 A(end,end)]] ;
   A = B ;
   [m,n] = size(A) ;
  [bmin, row] = min(A(1 :m-1,end)) ;
while bmin < 0 & abs(bmin) > eps
   col = MRTD(A(m,1 :n-1),A(row,1 :n-1)) ;
if p1 == 'O'
 disp(sprintf('\n pivot row-> %g pivot column->row,col')) ;
  tbn = tbn + 1 ;
```

```
 disp(sprintf('\n Tableau %g',tbn)) ;
 A ;
  disp(sprintf(' Appuyez sur une touche pour continuer ...\n')) ;
pause
end
if isempty(col)
    disp(sprintf('\n L"algorithme ne trouve pas la solution optimale .')) ;
    I ;
return
end
  A(row, :)= A(row, :)./A(row,col) ;
  subs(row) = col ;
for i = 1 :m
if i ~= row
    A(i, :)= A(i, :)-A(i,col)*A(row, :) ;
end
end
   [bmin, row] = min(A(1 :m-1,end)) ;
end
  d = A(1 :m-1,end) ;
   pc = fractp(d) ;
end
if p1 == 'O'
   disp(sprintf('\n tableau final '))
    A
    disp(sprintf(' Appuyez sur une touche pour continuer ...\n')) ;
pause
x = zeros(n-1,1) ;
x(subs) = A(1 :m-1,end) ;
x = x(1 :n1) ;
x1=x(1) ;
x2=x(2) ;
disp(sprintf('\n Le problème admet une solution optimale finie \n\n')) ;
disp(sprintf('\n Valeurs des variables :\n')) ;
for i=1 :n1
disp(sprintf(' x(%d)= %g ',i,x(i))) ;
end
z1= fonc1(x(1),x(2))
z2= fonc2(x(1),x(2))
disp(sprintf('\n valeur optimale de la fonction objectif :\n'))
disp(sprintf(' z= %f',tp*A(m,n)))
  fin ;
  I ;
  disp(sprintf(' Appuyez sur une touche pour continuer ...\n'))
```

```
pause
end
  I=I+2;
end
  I;
   A;
  fin;
  n1;
end
```

Fin du programme

Résultats obtenus :

Soit :

$$(P_0)\begin{cases} \text{" max "} \quad Z_\lambda = x_1 + x_2 \\ tq \qquad x_1 - 2x_2 \leq 0 \\ x_1, x_2 \in \{0, 1, 2\} \end{cases}$$

Les tableaux de l'Algorithme de Simplexe

Tableau initial

```
A =
     1 -2 1 0 0 0
      1  0 0 1 0 2
      0 1 0 0 1 2
    -1 -1 0 0 0 0

z = 4          x =  2
                    2
```

Tableau final

```
A =
    1 0 0  1 0 2
    0 1 0  0 1 2
    0 0 1 -1 2 2
    0 0 0  1 1 4
```

Appuyez sur une touche pour continuer ...

La coupe de Gomory

La contraintes ajoutée :

pc = 0 0 0

Les tableaux de la Methode du Dual de Simplexe

---

tableau final

```
A =
    1 0 0  1 0 2
```

0 1 0 0 1 2
0 0 1 -1 2 2
0 0 0 1 1 4
Le probleme admet une solution optimale finie
Valeurs des variables :
x(1)= 2 x(2)= 2
Valeurs des fonctions objectifs Z1 et Z2 : z1 = -2 z2 = 4

valeur optimale de la fonction objectif : z(lambda) = 4.000000

Le deuxième problème à résoudre est :

$$(P_1) \quad \begin{cases} Max \;\; Z_\lambda = x_1 + x_2 \\ tq \;\; x_1 - 2x_2 \leq 0 \\ x_1 - 2x_2 \geq -y_1^1 - 4(1 - y_1^1) \\ -x_1 + 3x_2 \geq 5y_2^1 - 2(1 - y_2^1) \\ y_1^1 + y_2^1 \geq 1 \\ x_1,\; x_2 \in \{0, 1, 2\} \\ y_1^1,\; y_2^1 \in \{0, 1\} \end{cases}$$

A =
1 -2 0 0
1 0 0 0
0 1 0 0
-1 2 3 0
1 -3 0 7
0 0 1 0
0 0 0 1
0 0 1 1
Application de la méthode des deux phases :
Fin de la Phase 2
****
Le problème admet une solution optimale finie
Valeur de la solution de base :
x(1)= 2.000000
x(2)= 2.000000
x(3)= 0.142857
x(4)= 0.857143
Valeur optimale de la fonction objectif :
z= 4.000000

La coupe de Gomory :
pc =
0 0 0 0.5714 0.8571 0.1429 0.1429 0.8571

Application de la Methode du Dual de Simplexe

Le problème admet une solution optimale finie
Valeurs des variables :
x(1)= 2 ; x(2)= 1 ; x(3)= 1 ; x(4)= 0

Valeurs des fonctions objectifs Z1 et Z2 : z1 = 0 ; z2 = 1

valeur optimale de la fonction objectif : z(lambda)= 3
Formons le problème suivant (P_2) en ajoutant cinq contraintes à (P_1) :

$$(P_2) \quad \begin{cases} Max \;\; Z_\lambda = x_1 + x_2 \\ tq \;\; x_1 - 2x_2 \leq 0 \\ x_1 - 2x_2 \geq -y_1^1 - 4(1 - y_1^1) \\ -x_1 + 3x_2 \geq 5y_2^1 - 2(1 - y_2^1) \\ x_1 - 2x_2 \geq y_1^2 - 4(1 - y_1^2) \\ -x_1 + 3x_2 \geq 2y_2^2 - 2(1 - y_2^2) \\ y_1^1 + y_2^1 \geq 1 \\ y_1^2 + y_2^2 \geq 1 \\ x_1,\ x_2 \in \{0, 1, 2\} \\ y_1^1,\ y_2^1 \in \{0, 1\} \\ y_1^2,\ y_2^2 \in \{0, 1\} \end{cases}$$

A =

1	-2	0	0	0	0
1	0	0	0	0	0
0	1	0	0	0	0
-1	2	3	0	0	0
1	-3	0	7	0	0
0	0	1	0	0	0
0	0	0	1	0	0
0	0	1	1	0	0
-1	2	0	0	5	0
1	-3	0	0	0	4
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	0	1	1

Application de la méthode des deux phases

Fin de la Phase 2
****
Le problème admet une solution optimale finie
Valeur de la solution de base :

x(1)= 2
x(2)= 2
x(3)= 0.142857
x(4)= 0.857143
x(5)= 0
x(6)= 1
Valeur optimale de la fonction objectif : z(lambda)= 4
Le problème admet une solution optimale finie
Valeurs des variables :
x(1)= 1
x(2)= 2
x(3)= 0
x(4)= 1
x(5)= 0
x(6)= 1
Valeurs des fonctions objectifs Z1 et Z2 : z1 = -3 z2 = 5
valeur optimale de la fonction objectif : z(lambda) = 3

Le pas suivant est d'ajouter des contraintes qui effacent seulement le point efficace x_3, on obtient le problème (P_3) :

$$(P_3)\begin{cases} Max \;\; Z_\lambda = x_1 + x_2 \\ tq \;\; x_1 - 2x_2 \leq 0 \\ x_1 - 2x_2 \geq -y_1^1 - 4(1 - y_1^1) \\ -x_1 + 3x_2 \geq 5y_2^1 - 2(1 - y_2^1) \\ x_1 - 2x_2 \geq y_1^2 - 4(1 - y_1^2) \\ -x_1 + 3x_2 \geq 2y_2^2 - 2(1 - y_2^2) \\ x_1 - 2x_2 \geq -2y_1^3 - 4(1 - y_1^3) \\ -x_1 + 3x_2 \geq 6y_2^3 - 2(1 - y_2^3) \\ y_1^1 + y_2^1 \geq 1 \\ y_1^2 + y_2^2 \geq 1 \\ y_1^3 + y_2^3 \geq 1 \\ x_1, \; x_2 \in \{0, 1, 2\} \\ y_1^1, \; y_2^1, y_1^2, \; y_2^2, y_1^3, \; y_2^3 \in \{0, 1\} \end{cases}$$

A =

```
 1  -2  0  0  0  0
 1  0  0  0  0  0
 0  1  0  0  0  0
-1  2  3  0 0  0
 1 -3  0  7 0  0
 0   0  1  0 0 0
 0  0  0  1 0  0
 0  0   1  1  0  0
-1  2  0  0  5  0
 1  -3  0  0  0  4
```

0 0 0 0 1 0
0 0 0 0 0 1
0 0 0 0 1 1
Application de la méthode des deux phases
Fin de la Phase 1
****
Fin de la Phase 2
****
Le probleme admet une solution optimale finie

Valeur de la solution de base :
x(1)= 2.000000
x(2)= 2.000000
x(3)= 0.142857
x(4)= 0.857143
x(5)= 0.000000
x(6)= 1.000000
x(7)= 1.000000
x(8)= 0.000000
Valeur optimale de la fonction objectif : z(lambda) = 4.00000
La coupe de Gomory :
pc =
0 0 0 0.5714 0.8571 0.1429 0.1429 0.8571 0 0 0 0 0 0 0 0 0 0 0 0

Application de la Methode du Dual de Simplexe

Le problème admet une solution optimale finie
Valeurs des variables :
x(1)= 0
x(2)= 2
x(3)= 0
x(4)= 1
x(5)= 0
x(6)= 1
x(7)= 0
x(8)= 1
Valeurs des fonctions objectifs Z1 et Z2 : z1 = -4.0000 z2 = 6.0000
valeur optimale de la fonction objectif : z= 2.000

Maintenant, le problème (P_4) est défini par :

$$(P_4)\begin{cases} Max \;\; Z_\lambda = x_1 + x_2 \\ tq \quad x_1 - 2x_2 \leq 0 \\ x_1 - 2x_2 \geq -y_1^1 - 4(1-y_1^1) \\ -x_1 + 3x_2 \geq 5y_2^1 - 2(1-y_2^1) \\ x_1 - 2x_2 \geq y_1^2 - 4(1-y_1^2) \\ -x_1 + 3x_2 \geq 2y_2^2 - 2(1-y_2^2) \\ x_1 - 2x_2 \geq -2y_1^3 - 4(1-y_1^3) \\ -x_1 + 3x_2 \geq 6y_2^3 - 2(1-y_2^3) \\ x_1 - 2x_2 \geq -3y_1^4 - 4(1-y_1^4) \\ -x_1 + 3x_2 \geq 7y_2^4 - 2(1-y_2^4) \\ y_1^1 + y_2^1 \geq 1 \\ y_1^2 + y_2^2 \geq 1 \\ y_1^3 + y_2^3 \geq 1 \\ y_1^4 + y_2^4 \geq 1 \\ x_1,\ x_2 \in \{0,1,2\} \\ y_1^1,\ y_2^1,\ y_1^2,\ y_2^2,\ y_1^3,\ y_2^3,\ y_1^4,\ y_2^4 \in \{0,1\} \end{cases}$$

A =
1 -2 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0
-1 2 3 0 0 0 0 0 0 0
1 -3 0 7 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0
0 0 1 1 0 0 0 0 0 0
-1 2 0 0 5 0 0 0 0 0
1 -3 0 0 0 4 0 0 0 0
0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0
0 0 0 0 1 1 0 0 0 0
-1 2 0 0 0 0 2 0 0 0
1 -3 0 0 0 0 0 8 0 0
0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 1 1 0 0
-1 2 0 0 0 0 0 0 1 0
1 -3 0 0 0 0 0 0 0 9
0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 1 1
Application de la méthode de deux phases

Le probleme admet une solution optimale finie
Valeur de la solution de base :
x(1)= 2.000000
x(2)= 2.000000
x(3)= 0.142857
x(4)= 0.857143
x(5)= 0.000000
x(6)= 1.000000
x(7)= 1.000000
x(8)= 0.000000
x(9)= 0.500000
x(10)= 0.500000
Valeur optimale de la fonction objectif : z(lambda)= 4.000000
La coupe de Gomory :
pc =
0 0 0 0.5714 0.8571 0.1429 0.1429 0.8571 0.5000 0 0 0 0 0 0 0 0 0 0 0 0.5000 0.5000 0.5000 0.5000

Application de la Methode du Dual de Simplexe

Le problème admet une solution optimale finie
Valeurs des variables :
x(1)= 1
x(2)= 1
x(3)= 1
x(4)= 0
x(5)= 0
x(6)= 1
x(7)= 1
x(8)= 0
x(9)= 1
x(10)= 0
Valeurs des fonctions objectifs Z1 et Z2 : z1 = -1.00 z2 = 2.00
valeur optimale de la fonction objectif : z(lambda)= 2.000
Le prochain problème à résoudre est le suivant :

$$(P_5)\begin{cases} Max \;\; Z_\lambda = x_1 + x_2 \\ tq \;\; x_1 - 2x_2 \leq 0 \\ x_1 - 2x_2 \geq -y_1^1 - 4(1 - y_1^1) \\ -x_1 + 3x_2 \geq 5y_2^1 - 2(1 - y_2^1) \\ x_1 - 2x_2 \geq y_1^2 - 4(1 - y_1^2) \\ -x_1 + 3x_2 \geq 2y_2^2 - 2(1 - y_2^2) \\ x_1 - 2x_2 \geq -2y_1^3 - 4(1 - y_1^3) \\ -x_1 + 3x_2 \geq 6y_2^3 - 2(1 - y_2^3) \\ x_1 - 2x_2 \geq -3y_1^4 - 4(1 - y_1^4) \\ -x_1 + 3x_2 \geq 7y_2^4 - 2(1 - y_2^4) \\ x_1 - 2x_2 \geq -4(1 - y_1^5) \\ -x_1 + 3x_2 \geq 3y_2^5 - 2(1 - y_2^5) \\ y_1^1 + y_2^1 \geq 1 \\ y_1^2 + y_2^2 \geq 1 \\ y_1^3 + y_2^3 \geq 1 \\ y_1^4 + y_2^4 \geq 1 \\ y_1^5 + y_2^5 \geq 1 \\ x_1,\; x_2 \in \{0, 1, 2\} \\ y_1^1,\; y_2^1,\; y_1^2,\; y_2^2,\; y_1^3,\; y_2^3,\; y_1^4,\; y_2^4,\; y_1^5,\; y_2^5 \in \{0, 1\} \end{cases}$$

Application de la méthode de deux phases
L'algorithme ne trouve pas la solution optimale.

3.3 Conclusion

Le problème (P_5) est irréalisable. Comme on l'a montré ci-dessus, la méthode de J.Sylva & A.Crema implémentée sur le logiciel MATLAB 7.0, nous donne l'ensemble des points efficaces E avec :

$E = \{(2,2)^t, (2,1)^t, (1,2)^t, (0,2)^t, (1,1)^t\}$ et l'ensemble des solutions non dominées

$Opt = \{(-2,4)^t, (0,1)^t, (-3,5)^t, (-4,6)^t, (-1,2)^t\}$. La vérification des résultats a été faite sur le logiciel Lingo.

L'inconvénient principal de cette méthode est qu'à chaque itération on ajoute plusieures contraintes et plusieures variables en même temps, ce qui implique que la taille du problème augmente d'une itération à une autre.

Dans cette étude, nous nous sommes intéressés aux problèmes de programmation linéaire multiobjectifs en nombres entiers. L'intérêt de tels problèmes résulte du fait que dans de nombreuses situations réelles modélisables par la programmation mathématique, les variables de décision ne peuvent prendre que des valeurs entières. Le décideur doit réaliser une analyse de l'ensemble des solutions efficaces du problème pour sélectionner les solutions efficaces préférées, et cela s'avère impossible lorsque cet ensemble est très large.

Nous avons commencé ce travail par introduire les notions de base concernant l'optimisation multiobjectif. Par la suite, dans le premier chapitre, nous avons réalisé une description détaillée de quelques méthodes de résolution existantes dans la littérature. Ensuite nous avons réalisé une description sur l'optimisation multiobjectif en nombres entiers. On peut conclure que l'optimisation discrète de manière générale est basée sur l'optimisation continue, vu l'utilisation successives des algorithmes de la méthode du simplexe et la méthode duale du simplexe, ce qui revient à augmenter la complexité des algorithmes applicables dans le cas discret.

Pour les algorithmes d'optimisation uni-critère dans le cas linéaire, nous avons programmé la méthode de Gomory. Nous avons donné un exemple numérique et comparé les résultats avec les deux méthodes (simplexe, Branch and Bound) pour montrer l'efficacité de cette méthode.

Dans le troisième chapitre (partie pratique), nous nous sommes intéressés à la programmtion de la méthode Sylva & Crema.

La perspective envisagée pour la suite de ce travail est d'essayer de programmer l'une des méthodes citée au chapitre II, pour un problème multiobjectif non linéaire.

Pour finir, on peut dire que l'optimisation discrète d'une manière générale n'est pas une tache facile, surtout l'optimisation multiobjectif et la difficulté revient aux variables qui sont discrètes ainsi qu'aux objectifs conflictuels.

Bibliographie

[1] G. Baillargeon, Programmation Linear Appliquéé, SGM,(1954).

[2] A. Berro,Optimisation multiobjectif et stratégies d'évolution en environement dynamique, thèse doctorat en informatique ,Toulose I,(2001).

[3] H. P. Benson , Finding an initial efficient extreme point for a multiple objective program, Journal of the Operational Research Society, 32 (1981) 495-498.

[4] R. Benayoun, J. de Montgolfier, J. Tergny, and O. Larichev, Linear Programming with Multiple Objective Functions, Step Method (STEM), Mathematical Programming, 1 (2),(1971), 366-373.

[5] A. Biran et M. Breiner, MATLAB pour ingénieur,version 6 et 7.

[6] V.J. Bowman , On the Relationship of the Tchebytcheff Norm an the Efficient Frontier of Multiple-Criteria Objectives, in Thiriez H. & Zionts S. (eds), MCDM, Springer-Verlag, Berlin (1976), pp. 76-85.

[7] D. Chaabane, Optimisation multicritère en nombres entiers, Thèse de doctorat, US-THB, (2OO5).

[8] L.G. Chalmet, L. Lemondis and D.J. Elzinga., An algorithm for bi-criterion integer proramming, European Journal of Operational Research 25 (1986), pp. 292-300.

[9] J. G. Ecker and Kouada I. A., Finding all efficient extreme points for multiple objective programs, Mathematical Programming , 14 (1978) 249-261.

[10] A.M. Geoffrion, Proper Efficiency and the theory of Vector Maximization, Journal of Mathematical and Advances Algebra 22 (1968), pp. 618-630.

[11] R.E. Gomory, Outline of an algorithm for integer solutions to linear programs, Bulletin of the AMS 64, pp. 275-278, (1958).

[12] J.J. Gonzales, G.R. Reeves, L.S. Frans, An interactive procedure for solving multiple objective integer linear programming problems, Decision making with multiple objectives, Springer Verlag, Berlin (1985), pp 250-260.

[13] R. Gupta, R. Malhotra, Multi-criteria integer linear programming problem. Cahiers du CERO 34 (1992) 51_68.

[14] J.P. Ignizio and L.C. Thomas, An Enhanced Conversion Scheme for Lexicographic, multiobjective Integer Programs, European Journal of Operational Research 18 (1984), pp. 51-61.

[15] D. Klein and E. Hannan, An algorithm for multiple objective integer linear programming problem, European journal of operational research 9, pp. 152-159, (1982).

[16] O. Marcotte and R.M. Soland, An interactive Branch and Bound algorithm for multiple criteria optimization, Management Science 32, 1, (1986), pp. 1231-1240.

[17] E. Neuman, Linear Programming With MATLAB,Math 472/CS 472.

[18] A. Pareto , Cours d'économie politique, vol. 1 et 2, F. Rouge, Lausanne, (1896).

[19] J. Philip, Algorithms for the Vector Maximization Problem, Mathematical Programming 2 (1972), pp. 207-229.

[20] R.M. Soland , Multicriteria Optimization : A General Characterization of Efficient Solutions, Decision Science 10, 1 (1979), pp. 26-38.

[21] R. Steuer and E.U. Choo, An interactive method weighted Tchebytcheff procedure for multiple objective programming, mathematical programming 26 (1983), pp. 326-344.

[22] J. Sylva and A. Crema, A method for finding the set of non-dominated vectorsfor multiple objective integer linear programs, European Journal of Operational Research, 158 (1), (2004), pp. 46-55.

[23] J. Teghem and P. L. Kunsch, Interactive Methods for Multi-Objective Integer Linear Programming, in Fandel G. and al. (eds), LNMS 273, Springer-Verlag, (1986a), 75-87.

[24] D. Vanderpooten and Ph. Vincke , Description and Analysis of Some Representative Interactive Multicriteria Procedures, Mathematical and Computer Modeling 12, 10/11 (1989), pp. 1221-1238.

[25] Ph. Vincke, L'Aide Multicritère à la Décision, Ellipses, Paris (1989).

[26] M. Zeleny , Multiple Criteria Decision Making, Mc Graw-Hill, New-York (1982).