N° d'ordre : 12/2010-E/MT

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE DES SCIENCES ET DE LE TECHNOLOGIE HOUARI BOUMEDIENE

FACULTE DES MATHEMATIQUES

THESE

Présentée pour l'obtention du grade de Docteur d'Etat

En : MATHEMATIQUES

Spécialité : Recherche Opérationnelle

Par : Mohamed El-Amine CHERGUI

Sujet

Contribution à la programmation non linéaire multicritère

Soutenue publiquement le 21/12/2010, devant le jury composé de :

Mr KHELLADI Abdelkader	Professeur	U.S.T.H.B	Président
Mr ABBAS Moncef	Professeur	U.S.T.H.B	Directeur de thèse
Mr AIDER Méziane	Professeur	U.S.T.H.B	Examinateur
Mr BERRACHEDI Abdelhafid	Professeur	U.S.T.H.B	Examinateur
Mr BLIDIA Mostefa	Professeur	U.S.D/BLIDA	Examinateur
Mr MOULAI Mustapha	Professeur	U.S.T.H.B	Examinateur
Mr CHAABANE Djamal	M. C. (A)	U.S.T.H.B	Examinateur

Table des matières

Remerciements

J'exprime d'abord ma profonde gratitude à mes enseignants de tous les paliers d'enseignements pour m'avoir éclairci le chemin du savoir et inculqué en moi l'esprit de la persévérance et la continuité.

C'est ici pour moi l'occasion d'exprimer ma profonde reconnaissance à mon Directeur de thèse, Professeur Moncef Abbas, pour avoir accepter de diriger ce travail, pour sa disponibilité à mon égard et pour ses précieux conseils tout au long de ce projet. Je lui adresse mes sincères remerciements.

Mon profond respect va à Monsieur le Professeur Abdelkader Khelladi qui m'a honoré en acceptant de présider mon jury malgré ses nombreuses occupations. Je veux lui exprimer toute ma reconnaissance.

Que les Professeurs Méziane Aider, Abdelhafid Berrachedi, Mostafa Blidia, Djamal Chaabane et Mustapha Moulaï trouvent ici l'expression de ma profonde reconnaissance pour l'intérêt qu'il ont porté à mon travail en acceptant de l'examiner, ainsi que pour leur aide et leur soutien indéfectibles.

Je n'aurai sans doute jamais achevé mes études sans le soutien et les encouragements permanents de tous mes collègues de la faculté des mathématiques et particulièrement du département de recherche opérationnelle avec qui j'ai eu des discussions enrichissantes et fructueuses.

Qu'il me soit aussi permis de remercier tous ceux qui, par leur présence et leur amitié, ont contribué à cette longue épopée ; la liste est longue et je risque d'oublier des noms.

A ma mère,

A la mémoire de mon père,

A ma femme et à mes enfants,

A mes frères et à mes soeurs,

A tous les membres de ma famille,

A tous mes amis.

Introduction

Depuis longtemps, l'énoncé d'un problème d'optimisation ne considérait qu'une seule fonction objectif qui fusionne les aspects multidimensionnels de la situation de décision afin d'en faire une appréciation simple de la mesure de la qualité des solutions. Cependant, dans la pratique, il existe une multitude d'applications pour lesquelles un modèle ne prenant en compte qu'un seul objectif d'évaluation des solutions possibles, ne répond en aucun cas à la réalité. En effet, de nombreux problèmes rencontrés quotidiennement dans divers domaines (transport, télécommunication, économie ...etc) nécessitent la considération de plusieurs objectifs contradictoires simultanément. Jusqu'au milieu des années soixantes du siècle dernier, l'absence d'outils théoriques et de méthodes numériques n' a pas permis de prendre en charge les problèmes pratiques prenant en compte plusieures objectifs simultanément. Ce n'est que vers la fin des années soixantes que des travaux aussi bien théoriques que pratiques commencent à être publiers Geoffrion (1967) et (1968), Philip (1972). Le développement des techniques interactives pour les problèmes multicritères a été l'un des domaines les plus actifs de la recherche et plusieurs techniques ont été développés, parmi lesquels on cite ; la méthode STEM de Benayoun et *al.* (1971), la méthode de Dyer (1972), la méthode développée par Geoffrion et *al.* (1971) et la méthode de Ziont et Wallenious (1976).

L'optimisation multi-objectif a pour objet le développement d'outils adéquats, aussi bien théoriques que pratiques, pour la résolution de ce type de problèmes. Les méthodes pour l'optimisation multi-objectif ont connu un intérêt croissant ces vingt dernières années.

De nombreuses méthodologies, exactes ou approchées, ont été proposées pour les problèmes modélisés sous la forme de programmes linéaires mono-objectif en nombres entiers et il est aujourd'hui possible de résoudre des problèmes de grande taille. La prise en compte simultanée de plusieurs objectifs contradictoires est de plus en plus considérée et forme l'objet d'étude de l'optimisation combinatoire multi-objectif. Cependant, à la lecture des écrits publiers, alors qu'on recense des centaines d'études dédiées aux méta-heuristiques multi-objectif basées sur le recuit simulé, la recherche tabou et plus particulièrement les algorithmes évolutionnaires entre autres, le nombre de méthodes portant sur les méthodes exactes multi-objectif est peu important. Si les méthodes méta-heuristiques multi-objectif sont parfois nécessaires pour débloquer des situations complexes et de grandes instances, il n'en demeure pas moins qu'un effort reste à fournir pour amener les méthodes multi-objectif exactes au même niveau d'utilisation et de performance que les méthodes classiques en optimisation, et ainsi développer l'adoption et l'utilisation de l'optimisation en général et l'optimisation combinatoire multi-objectif en particulier, en leur permettant de s'attaquer à des problèmes de grande taille.

Le travail présenté dans cette thèse s'inscrit dans cette optique. Il s'agit essentiellement de l'exploration et l'adaptation de techniques classiques et l'élaboration de techniques nouvelles, utilisant la programmation mathématique pour l'optimisation multi-objectif, en se focalisant sur les problèmes linéaires en nombres entiers.

C'est ainsi que l'essentiel des éléments de la programmation linéaire en nombres entiers est relaté dans le chapitre un. Les principales méthodes de résolution y sont rapportées. Il s'agit de méthodes utilisants des coupes (Gomory, Chvátal, etc...) et des méthodes connues sous le vocable anglais de "branch and bound" et "branch and cut". Le chapitre deux est consacré à la problématique du multi-objectif. Les différentes notions et concepts sont présentés dans une première étape, ensuite des méthodes récentes dédiées au problème multi-objectif linéaire en nombres entiers ($MOILP$) sont développées. Notre première contribution consiste en l'élaboration de deux méthodes pour le problème $MOILP$. La première détermine l'ensemble des solutions entières efficaces dans l'espace des décisions et est présentée dans le

chapitre trois, alors que la deuxième génère toutes les solutions non dominées dans l'espace des critères et fait l'objet du chapitre quatre. Les éléments fondamentaux à même de permettre une bonne acquisition de la problèmatique de la programmation fractionnaire sont présentés au chapitre cinq. Notre deuxième contribution en programmation linéaire fractionnaire en nombres entiers est alors, exposée au chapitre six. Le travail est achevé par une conclusion et des perspectives ouvrant la voie à de futurs travaux de recherche sur des classes de problèmes de programmation multi-objectif.

Chapitre 1

Eléments de la programmation linéaire

1.1 Introduction

"La programmation linéaire est l'une des plus importantes techniques d'optimisation utilisées en recherche opérationnelle" [98].

La programmation linéaire a depuis longtemps prouvé sa valeur comme un modèle important pour de nombreux problèmes d'allocation et des phénomènes économiques. Les applications en expansion continue dans la littérature, a démontré à maintes reprises l'importance de la programmation linéaire comme un cadre général pour la formulation de problèmes.

Un programme linéaire (PL) est un problème d'optimisation dans lequel la fonction objectif est linéaire en les inconnues ou variables, et les contraintes se composent d'égalités et inégalités linéaires. La forme exacte de ces contraintes peut différer d'un problème à l'autre, cependant, tout programme linéaire peut être transformé sous

la forme standard suivante :

$$(PL)\begin{cases} \max z = c^t x \\ Ax = b \\ x \geq 0 \end{cases} \tag{1.1}$$

où c et x sont deux n-vecteurs réels, A est une $m \times n$-matrice réelle et b un m-vecteur réel. L'inégalité vectorielle $x \geq 0$ signifie que chaque composante du vecteur x est positive ou nulle.

L'ensemble des contraintes

$$S = \{x \in \mathbb{R}^n \mid Ax = b,\ x \geq 0\} \tag{1.2}$$

représente l'ensemble des solutions réalisables de (PL).

Si les variables sont astreintes à ne prendre que des valeurs entières, on parle de programme linéaire en nombres entiers $(PLNE)$, et si les variables ne peuvent prendre que les valeurs 0 ou 1, le programme est linéaire à variables bivalentes. Il existe aussi des problèmes dans lesquels une partie seulement des variables ne peut prendre que des valeurs entières ; on parle alors, de programme linéaire mixte.

Il existe une abondante littérature consacrée à des problèmes particuliers de programmation entière : voyageur de commerce, sac-à-dos, recouvrement, stable...etc Les méthodes spécifiques à un type de problème particulier peuvent s'avérer plus efficaces que les algorithmes génériques, et nombre d'entre elles ont été développées durant ces dernières décennies. Cependant, nous ne décrirons pas ces techniques et nous nous limitons à des méthodes générales, celles qui s'appliquent à n'importe quel problème de programmation entière.

La difficulté d'un problème découle de l'efficacité des algorithmes de résolution qu'on peut lui appliquer. La complexité d'un algorithme évalue un majorant du nombre d'opérations élémentaires, comme une fonction de la taille des données, qu'on doit effectuer, dans le pire des cas, pour obtenir le résultat escompté. Un algorithme est dit polynomial s'il est de complexité polynomiale. Un algorithme qui n'est pas

polynomial est dit exponentiel. Un problème de reconnaissance est un problème dont les résultats ne peuvent prendre que deux valeurs Vrai ou Faux. A chaque problème d'optimisation combinatoire on peut associer un problème de reconnaissance. On montre qu'un problème d'optimisation combinatoire est au moins aussi difficile que le problème de reconnaissance associé [92].

On distingue deux types de problèmes de reconnaissance :

- ceux qui peuvent être résolus en temps polynomial en la taille des données (classe P).

- ceux pour lesquels on peut vérifier un certificat donnant la solution, en temps polynomial en la taille des données (classe NP : Non déterministe Polynomial).

On a alors : $P \subseteq NP$.

Les problèmes NP-complets sont les problèmes les plus difficiles de la classe NP et sont équivalents entre eux quant à l'existence d'un algorithme polynomial : si on connaissait un algorithme polynomial pour l'un d'entre eux, on en connaîtrait un pour tout problème de la classe, et on aurait l'égalité $P = NP$ (ce qui est très improbable !).

Contrairement au problème (PL) qui est dans la classe P [67], [66], le problème de la programmation linéaire en nombres entiers ($PLNE$) est dans la classe NP-complet [31], [96].

1.2 Résultats fondamentaux de la programmation linéaire

Les éléments de base de la programmation linéaire ont fait l'objet d'une littérature abondante aussi riche que diversifiée. On retiendra essentiellement les ouvrages spécialisés suivants : [72], [81], [91], [98], [109] et [102].

1.2.1 Solution de base

Considérons le système d'égalité suivant :

$$Ax = b \tag{1.3}$$

où x est un n-vecteur réel, b un m-vecteur réel, et A une $m \times n$-matrice réelle de rang m. Une base de ce système d'équations est un ensemble $B \subset \{1, 2, ..., n\}$ d'indices de colonnes tel que la sous matrice A_B de A engendrée par les colonnes d'indices de B, soit carrée d'ordre m non singulière.

Nous pouvons ainsi, décomposer le système $Ax = b$ comme suit :

$$A_B x_B + A_N x_N = b \tag{1.4}$$

où $N = \{1, 2, ..., n\} \backslash B$, A_N est la matrice engendrée par les $n-m$ colonnes d'indices dans N, x_B est le m-vecteur des indices des colonnes de la base B, et x_N est le $n-m$-vecteur des indices de N.
Si on pose $x_N = 0$, nous devons juste résoudre le système

$$A_B x_B = b \tag{1.5}$$

pour le m-vecteur x_B. En posant $x = (x_B, 0_{\mathbb{R}^{n-m}})$, on obtient une solution de *base* de $Ax = b$.

Plus généralement, on a :

Définition 1.1 *Soit A_B une $m \times m$-matrice inversible extraite de A. Si toutes les $n - m$ composantes de x qui ne sont pas associées aux colonnes de la base B sont égales à zéro, la solution du système $Ax = b$ est appelée solution de base relativement à la base B. Les composantes de x associées à B sont appelées variables de base, et celles associées à N sont appelées variables hors base.*

Dans une solution de base, les variables de base ne sont pas nécessairement toutes non nulles. Ceci est noté par la définition suivante :

Définition 1.2 *Si une variable de base prend la valeur zéro dans une solution de base, cette solution est appelée solution de base dégénérée.*

Considérons maintenant le système

$$\begin{cases} Ax = b \\ x \geq 0 \end{cases} \tag{1.6}$$

qui représente l'ensemble des contraintes S d'un programme linéaire sous forme standard.

Définition 1.3 *Une solution x vérifiant les contraintes de l'ensemble S est appelée solution réalisable ou admissibleadmissible. Une solution réalisable qui est basique est appelée solution de base réalisable et si de plus elle est dégénérée, on l'appelle solution de base dégénérée.*

Le théorème suivant établit clairement l'importance des solutions réalisables de base dans la résolution des programmes linéaires.

Théorème 1.4 *Etant donné un programme linéaire écrit sous forme standard, avec une matrice A de rang m.*
i) S'il admet une solution réalisable, il admet aussi une solution réalisable de base,
ii) S'il admet une solution optimale, il admet une solution optimale de base,
iii) S'il admet une solution réalisable et si la valeur de la fonction objectif est bornée, il admet une solution optimale de base.

Ce théorème connu sous le nom de théorème fondamental de la programmation linéaire, ramène le problème de résolution d'un programme linéaire à celui de recherche des solutions réalisables de base. Pour un problème ayant n variables et m contraintes, il y a un nombre fini de possibilités qui est théoriquement donné par au plus $\complement_n^m = \frac{n!}{m!(n-m)!}$ solutions réalisables de base.

Dans ce qui suit, l'équivalence entre solution réalisable de base et point extrême du polytope convexe des solutions réalisables S d'un (PL) est établie. Rappelons

qu'un point extrême d'un ensemble est un point qui ne se trouve pas strictement à l'intérieur d'un segment reliant deux autres points de l'ensemble.

Soit A la $m \times n$-matrice de rang m et b un m-vecteur.

Théorème 1.5 *Un vecteur x est un point extrême de S si et seulement si x est une solution réalisable de base du système de contraintes définissant S.*

Corollaire 1.6 *Si l'ensemble S est non vide, alors il admet au moins un point extrême.*

Corollaire 1.7 *Si le programme linéaire (PL) défini par [?] admet une solution optimale finie, alors il admet une solution optimale finie qui est un point extrême de S.*

Corollaire 1.8 *L'ensemble S posséde un nombre fini de points extrêmes.*

1.2.2 Caractérisation d'une base optimale

Soit B une base réalisable de (PL) et $N = \{1, \ldots, n\} \setminus B$. Supposons que B se compose des indices des m premières colonnes de A. On partitionne A, x et c comme suit :

$$\begin{aligned} A &= [A_B, A_N] \\ x &= [x_B, x_N] \\ c^t &= [c_B^t, c_N^t] \end{aligned} \tag{1.7}$$

Le programme linéaire (PL) s'écrit alors :

$$\begin{cases} \max z = c_B^t x_B + c_N^t x_N \\ A_B x_B + A_N x_N = b \\ x_B \geq 0,\ x_N \geq 0 \end{cases} \tag{1.8}$$

et les variables de base s'écrivent en fonction des variables hors base comme suit :

$$x_B = A_B^{-1}b - A_B^{-1}A_N x_N \tag{1.9}$$

En remplaçant cette expression dans la fonction objectif, on obtient l'expression de la fonction objectif en fonction des variables hors base seulement :

$$z = c_B^t A_B^{-1} b + (c_N^t - c_B^t A_B^{-1} A_N) x_N \tag{1.10}$$

Aussi, • $\pi = c_B^t A_B^{-1}$ est dit vecteur multiplicateur relatif à la base B,
• $\widehat{c}_N^t = c_N^t - \pi A_N$ est dit vecteur coût réduit relatif à la base B. C'est les composantes de ce vecteur qui sont utilisées pour déterminer quelle variable rentre en base.

Théorème 1.9 *Si le vecteur coût $\widehat{c}_N$ est négatif ou nul, la solution de base correspondante est solution optimale de (PL). La base B est alors une base optimale.*

Si une solution de base optimale x^* est dégénérée, alors elle n'est pas unique. Les autres solutions de bases optimales sont appelées solutions alternatives à x^*.

1.2.3 Algorithme du simplexe

L'algorithme du simplexe a été développé par Dantzig [36] pour la résolution d'un PL. Partant d'une solution réalisable de base correspondant à un sommet du polyèdre S, il recherche par itérations succesives des solutions meilleures correspondant à d'autres sommets, en parcourant les arêtes de S. L'expérience montre que la résolution par la méthode du simplexe de problèmes de divers domaines, et diverses valeurs de n et de m, ne fait en général, que m ou $3m/2$, itérations au plus, même si le nombre de points extrêmes est $\frac{n!}{m!(n-m)!}$ [72]. En termes de compléxité des algorithmes, l'algorithme du simplexe est théoriquement exponentiel alors qu'il se comporte dans la pratique comme un algorithme polynômial.

Algorithme

Soit B une base réalisable de départ, alors l'écriture canonique de (PL) par rapport à B est :

I_m	$\widehat{A}_N = A_B^{-1}A_N$	$\widehat{b} = A_B^{-1}b$
$0_{\mathbb{R}^m}$	$\widehat{c}_N^t = c_N^t - c_B^t A_B^{-1}A_N$	$z - c_B^t A_B^{-1}b$

(1.11)

Etape 1. Si $\widehat{c}_N \leq 0$, terminer ; la solution courante est optimale.

Etape 2. Déterminer quelle variable x_s, $s \in N$, rentre en base en sélectionnant $\widehat{c}_s = \max_{j\in N}\{\widehat{c}_s\}$.

Etape 3. Si $\forall i = 1,\ldots,m$, $\widehat{a}_{is} < 0$, terminer ; la fonction objectif n'est pas bornée.

Sinon, choisir une ligne r pour déterminer la variable qui doit quitter la base telle que :

$$\frac{\widehat{b}_r}{\widehat{a}_{rs}} = \min_{i=1,\ldots,m}\left\{\frac{\widehat{b}_i}{\widehat{a}_{is}} \mid \widehat{a}_{is} > 0\right\} \tag{1.12}$$

$$B \longleftarrow B \cup \{s\} \setminus \{f(r)\} \tag{1.13}$$

où $f(i) \in \{1,\ldots,n\}$ est l'indice de la variable de base associée à la ligne i, $i \in \{1,\ldots,m\}$. L'élément $\widehat{a}_{rs}$ est appelé pivot.

Etape 4. Les m premières lignes du tableau 1.11 ainsi que la dernière ligne sont alors calculées de la façon suivante :

$$\begin{aligned}
l_i &\longleftarrow l_i - \frac{l_r\widehat{a}_{is}}{\widehat{a}_{rs}}; \quad i = 1,\ldots,m,\ i \neq r \\
l_r &\longleftarrow l_i - \frac{l_r}{\widehat{a}_{rs}} \\
l_{m+1} &\longleftarrow l_{m+1} - \frac{l_r\widehat{c}_s}{\widehat{a}_{rs}}
\end{aligned}$$

Aller à l'étape 2.

Fin algorithme.

D'autres algorithmes ont été développés pour la résolution des programmes linéaires : la méthode de l'ellipsoïde [67], construit itérativement une suite d'ellipsoïdes de volume décroissant jusqu'à ce que le centre du dernier ellipsoïde soit un point intérieur du polyèdre S. Une solution optimale est alors déterminée à partir de ce point.

Les algorithmes projectifs (méthodes de points intérieurs ou méthodes de barrières) [66] aboutissent à un point intérieur de la région du domaine des solutions réalisables proche de l'optimum, mais d'une autre manière. Brièvement, il s'agit de construire successivement des points du domaine satisfaisant chaque fois plus de contraintes en considérant leur projection sur les plans délimitant le polyèdre S, c'est-à-dire supportés par $A_i x = b_i$ pour $i \in \{1, ..., m\}$.

Ces deux types de méthodes convergent en temps polynômial et présentent donc pour cela une importance théorique capitale. De plus, les algorithmes projectifs donnent maintenant de bons résultats calculatoires et peuvent être plus rapides que la méthode du simplexe pour les problèmes de grande taille. Cependant, dans la pratique, l'algorithme du simplexe est à ce jour encore le plus utilisé pour son efficacité depuis longtemps éprouvée.

1.2.4 Dualité

Considérons le programme linéaire s'écrivant sous forme standard **??**.

La dualité en programmation linéaire découle de dualité définie pour un programme mathématique général. En effet, la fonction de Lagrange associée à (PL) s'écrit :

$$\begin{aligned} L(x, y) &= c^t x + y^t(-Ax + b) \\ &= (c^t - y^t A)x + y^t b \end{aligned} \tag{1.14}$$

et admet un maximum fini par rapport à x si et seulement si

$$y \in Y = \left\{ y \mid c^t - y^t A \leq 0 \right\} \tag{1.15}$$

Le minimum de la fonction de Lagrange pour $y \in Y$ est atteint lorsque $c^t - y^t A = 0$, ce qui signifie que :

$$\min_{\{x \setminus Ax = b,\ x \geq 0\}} L(x, y) = y^t b \tag{1.16}$$

Le *dual* de (PL) est alors le programme linéaire suivant :

$$(D) \left\{ \begin{array}{l} \min w = y^t b \\ A^t y \geq c \end{array} \right\} \tag{1.17}$$

(PL) est appelé problème *primal*.

Le dual de (D) est (PL) et donc, (PL) et (D) sont duaux l'un de l'autre.

Propriété 1.10 *(dualité faible)*

Si x et y sont des solutions réalisables respectivement du primal (PL) et de son dual (D), elle vérifient : $c^t x \leq y^t b$.

Corollaire 1.11 *Soient x et y des solutions réalisables respectivement du primal et du dual. Si $c^t x = y^t b$ alors, x et y sont solutions optimales.*

Corollaire 1.12 *Si un problème possède une valeur optimale infinie, son dual n'a pas de solutions.*

Propriété 1.13 *(dualité forte)*

Si le problème primal possède une solution optimale, alors il en est de même pour le problème dual et de plus, les deux problèmes ont la même valeur optimale $z^* = w^*$.

Théorème 1.14 *(théorème des écarts complémentaires)*

Soient x et y des solutions réalisables respectivement du primal et du dual. Une condition nécessaire et suffisante pour que x et y soient solutions optimales est qu'elles vérifient les relations suivantes :

$$\begin{aligned} (c^t - y^t A)x &= 0 \\ y^t(Ax - b) &= 0 \end{aligned} \tag{1.18}$$

1.2.5 Solution duale associée à une base

Soit B une base du problème (PL). Les coûts réduits des variables de base étant nuls, on a $c_B^t - c_B^t A_B^{-1} A_B = 0$, qui donne $y_B A_B = c_B^t$. Ainsi, $y_B = c_B^t A_B^{-1}$ et donc

y_B est une solution réalisable de base du problème (D). Si de plus $\widehat{c}_N \leq 0$ alors, la base B réalise l'optimum si elle duale réalisable.

Ceci donne lieu à l'algorithme dual du simplexe qui considère une situation dans laquelle $\widehat{c}_N \leq 0$, avec une solution de base non réalisable ($b_i < 0$). L'algorithme consiste alors à choisir un pivot pour rendre les $b_i \geq 0$ tout en maintenant les $\widehat{c}_N \leq 0$.

1.3 Résolution d'un programme linéaire à variables entières

Comme il a été mentionné en début de ce chapitre, les variables du programme linéaire (PL) peuvent être discrètes. On obtient dans ce cas, le programme linéaire en nombres entiers suivant :

$$(PLNE) \begin{cases} \max z = c^t x \\ Ax = b \\ x \geq 0,\ x \textit{ vecteur entier} \end{cases} \tag{1.19}$$

avec A une $m \times n$-matrice à coefficients dans $\mathbb{Z}$ et b est vecteur de $\mathbb{Z}^m$. Les coordonnées du vecteur x sont astreintes à ne prendre que des valeurs entières positives ou nulles.

Nous continuons à noter S l'ensemble des contraintes de $(PLNE)$ sans la contrainte d'intégrité des variables :

$$S = \{x \mid Ax = b,\ x \geq 0\} \tag{1.20}$$

et D l'ensemble des solutions réalisables de $(PLNE)$. L'ensemble S n'est autre que le domaine des solutions réalisables du programme linéaire relaxé (PL) obtenu en relâchant les contraintes d'intégrité des variables ; $D \subset S$. La résolution de (PL) donne une borne supérieure pour la fonction objectif de $(PLNE)$.

L'approche polyèdrale pour la résolution du problème $(PLNE)$ consiste à déterminer un système d'inégalités linéaires décrivant l'enveloppe convexe de D, $con(D)$. Si

on arrive à générer ce système d'inégalités linéaires par un algorithme polynomial, le problème d'optimiser sur D se réduit au problème d'optimiser sur $con(D)$ qui est un problème facile de programmation linéaire. Compte tenu de la difficulté de décrire l'enveloppe convexe $conv(D)$, on cherche à trouver un ensemble intermédiaire entre S et $conv(D)$ sur lequel l'optimisation de la fonction objectif se fait en un temps polynomial, puis d'utiliser la récursivité pour obtenir des approximations plus serrées de $conv(D)$.

Le principe consiste donc à construire une suite de polyèdres dans $\mathbb{R}^n_+$,

$$\begin{aligned} S &= S_1 \supset S_2 \supset ...S_k \supset S_{k+1} \supset ...conv(D) \\ S_{k+1} &= S_k \cap \{x \mid \alpha_k x \leq \beta_k\} \end{aligned} \tag{1.21}$$

Une inéquation de la forme $\alpha_k x \leq \beta_k$ est dite **valide** pour $(PLNE)$ si elle est satisfaite par toute solution de D.

Une **coupe** est une contrainte valide qui n'est pas satisfaite par au moins une solution de S.

Généralement, les inégalités valides sont de deux types. Il y a celles portant sur des structures provenant de l'étude de problèmes combinatoires spécifiques, typiquement dérivées des techniques polyèdrales, et il y a celles qui sont générales, dérivées de techniques algébriques. Dans cette deuxième classe nous trouvons, par exemple, les coupes fractionnaires et les coupes mixtes de Gomory exposées dans les années soixante [55] et [56], les coupes de Chvátal-Gomory de Chvátal [29].

Les coupes mixtes sont appelées ainsi parce qu'elles peuvent être générer sur des problèmes où les variables ne sont pas toutes astreintes à être discrètes.

1.3.1 Coupes de Gomory

L'algorithme utilisant les coupes de Gomory procéde comme suit :

Soit $\overline{x}$ une solution optimale du problème relaxé (PL) trouvée par la méthode du simplexe. Si $\overline{x}$ est une solution entière, elle est optimale pour $(PLNE)$. Sinon,

choisir une variable x_j telle que la valeur $\overline{x}_j$ est fractionnaire et considérer la ligne correspondante du tableau du simplexe, par exemple la ligne i :

$$x_j + \sum_{k \in N} \widehat{a}_{ik} x_k = \overline{x}_j \tag{1.22}$$

où N est l'ensemble des indices des variables hors-base.

La contrainte

$$\sum_{k \in N} f(\widehat{a}_{ik}) x_k \geq f(\overline{x}_j) \tag{1.23}$$

est alors déduite de l'expression précédente, où $f(r) = r - \lfloor r \rfloor$ désigne la partie fractionnaire du nombre réel r.

Cette coupe, appelée coupe fractionnaire de Gomory, ou coupe fondamentale, peut être rajouter au tableau courant du simplexe.

Dans la pratique, les coupes fractionnaires de Gomory convergent trés lentement. D'autres coupes ont été introduites dont l'intention de les rendres plus performantes. En particulier, Gomory [55] lui même a donné les coupes suivantes :

Proposition 1.15 *Pour tout entier naturel t, l'inéquation*

$$\sum_{k \in N} f(t\widehat{a}_{ik}) x_k \geq f(t\overline{x}_j) \tag{1.24}$$

est une coupe. De plus, si $f(\widehat{a}_{ik}) < \frac{1}{2}$ et $\frac{1}{2} \leq t f(\widehat{a}_{ik}) < 1$, alors la coupe 1.24 est plus profonde que la coupe 1.23.

Proposition 1.16 *L'inéquation*

$$\sum_{k \in N} \min \left\{ f(\widehat{a}_{ik}), f(\overline{x}_j) \frac{1 - f(\widehat{a}_{ik})}{1 - f(\overline{x}_j)} \right\} x_k \geq f(\overline{x}_j) \tag{1.25}$$

est une coupe. De plus, elle est plus profonde que la coupe 1.23.
De même, pour tout entier naturel t, l'inéquation

$$\sum_{k \in N} \min \left\{ f(t\widehat{a}_{ik}), f(t\overline{x}_j) \frac{1 - f(t\widehat{a}_{ik})}{1 - f(t\overline{x}_j)} \right\} x_k \geq f(t\overline{x}_j) \tag{1.26}$$

est une coupe plus profonde que 1.24.

1.3.2 Coupes de Chvátal-Gomory

Chvátal considère un programme linéaire en nombres entiers

$$(P_1)\begin{cases} \max z = c^t x \\ Ax \leq b \\ x \geq 0,\ x \text{ vecteur entier} \end{cases} \tag{1.27}$$

avec des contraintes d'inégalités.

Proposition 1.17 *Pour tout vecteur $\lambda \in \mathbb{R}^m_+$, l'inégalité suivante est une coupe pour (P_1) :*

$$\lfloor \lambda^t A \rfloor\, x \leq \lfloor \lambda^t b \rfloor \tag{1.28}$$

Il a été démontré dans [81] que cette dernière coupe est équivalente à la coupe de Gomory 1.23. C'est pour cette raison qu'on l'appelle coupe de Chvátal-Gomory.

Proposition 1.18 *Si $f(\lambda^t b) < \frac{1}{2}$ et t est un entier naturel tel que $\frac{1}{2} \leq t f(\lambda^t b) < 1$, alors en remplaçant λ par $f(t\lambda)$ on obtient une coupe au moins aussi bonne (meilleure ou équivalente) que la coupe 1.28.*

1.3.3 Coupe disjonctives

Le concept de problème de programmation disjonctive a été introduit par Balas [12].

Soit Q_0 et Q_1 deux polyèdres de $\mathbb{R}^n$ non vides fermés et disjoints, explicitement définis par les inégalités

$$Q_i = \{x \in \mathbb{R}^n \mid A_i x \leq b_i\},\quad i = 0, 1 \tag{1.29}$$

où A_i est une $m_i \times n-$matrice réelle et $b_i \in \mathbb{R}^{m_i}$. Alors, $Q = conv(Q_0 \cup Q_1)$ est un polyèdre.

Coupes de "split"

Soit R un polyèdre de $\mathbb{R}^n$ défini par un ensemble d'inégalités : $R = \{x \in \mathbb{R}^n;\ Ax \leq b\}$, où A est une $m \times n-$matrice réelle et $b \in \mathbb{R}^m$.

Soit $c \in \mathbb{R}^n$ un vecteur, tel que $\max_{x \in R}(c^t x) < +\infty$ et $\widehat{x}$ une solution optimale du programme linéaire :

$$\begin{cases} \max c^t x \\ x \in R \end{cases} \tag{1.30}$$

On exige que $\widehat{x}$ soit un point extrême de R (ce qui est le cas si on calcule $\widehat{x}$ par l'algorithme du simplexe).

Soit $\pi \in \mathbb{R}^n$, $\pi_0 \in \mathbb{R}$, qui définissent une coupe de "split" : on suppose que $\pi_0 < \pi^t \widehat{x} < \pi_0 + 1$ et on pose :

$$\begin{aligned} Q_0 &= R \cap \{x \in \mathbb{R}^n \mid \pi^t x \leq \pi_0\} \\ Q_1 &= R \cap \{x \in \mathbb{R}^n \mid \pi^t x \geq \pi_0 + 1\} \end{aligned} \tag{1.31}$$

Dans le cas d'un programme linéaire en nombres entiers (ou mixte), on résout le programme linéaire associé (PL) pour trouver $\widehat{x}$, et on choisi la coordonnée i de $\widehat{x}$ telle que $\widehat{x}_i$ n'est pas entière alors qu'elle est soumise à une contrainte d'intégrité. On ajoute alors la coupe "split" :

$$x_i \leq \lfloor \widehat{x}_i \rfloor \ \text{ ou } x_i \geq \lceil \widehat{x}_i \rceil \tag{1.32}$$

Coupes de "lift and project"

Ce sont des coupes particulières des coupes de "split" pour les programmes linéaires à variables bivalentes ou mixtes [13], [33].

Soit le programme linéaire à variables bivalentes :

$$\begin{cases} \max Z = c^t x \\ Ax \leq b \\ x_j \in \{0, 1\}\,,\ j = 1, ..., n \end{cases} \tag{1.33}$$

- Résoudre le problème relaxé suivant :

$$\begin{cases} \max Z = c^t x \\ A'x \leq b' \end{cases} \tag{1.34}$$

avec $A'x \leq b'$ contenant toutes les contraintes de bornes. Soit $\widehat{x}$ une solution optimale.

- Considérer le problème disjonctif :

$$\begin{cases} A'x \leq b' \\ x_i = 0 \end{cases} \quad \text{ou} \quad \begin{cases} A'x \leq b' \\ x_i = 1 \end{cases} \tag{1.35}$$

- Décrire l'enveloppe convexe du programme disjonctif, définie par

$$S_i = conv(\{A'x \leq b',\ x_i = 0\} \cup \{A'x \leq b',\ x_i = 1\}) \tag{1.36}$$

- Construire une inégalité valide sur S_i qui élimine $\widehat{x}$, c'est-à-dire une coupe, de la façon suivante :

Une inégalité $\alpha x \leq \beta$ est valide pour S_i si et seulement si (α, β) satisfait le système :

$$\begin{cases} \alpha = u^1 A' + v^1 e_i, \qquad \alpha = u^2 A' + v^2 e_i \\ \beta \leq u^1 A' + v^1 0, \qquad \beta \leq u^2 A' + v^2 \\ u^1,\ u^2 \geq 0 \\ \alpha,\ \beta,\ v^1,\ v^2 \ \textit{réels} \end{cases} \tag{1.37}$$

Cette inégalité devient une coupe en supprimant la solution $\widehat{x}$. Le couple (α, β) est alors choisi comme solution du programme linéaire suivant :

$$\begin{cases} \max \beta - \alpha\widehat{x} \\ \alpha = u^1 A' + v^1 e_i, \qquad \alpha = u^2 A' + v^2 e_i \\ \beta \leq u^1 A' + v^1 0, \qquad \beta \leq u^2 A' + v^2 \\ u^1,\ u^2 \geq 0 \\ \alpha,\ \beta,\ v^1,\ v^2 \ \textit{réels} \end{cases} \tag{1.38}$$

On peut aussi résoudre les problèmes de $(PLNE)$ et mixtes en fractionnant le domaine en régions admissibles bornées par des frontières alignées sur des entiers : on sépare ainsi les domaines, et on évalue quelle région explorer en premier : on appelle cette méthode la séparation et évaluation, ou branch-and-bound en anglais.

Il faut noter qu'en général, les coupes ne sont jamais utilisées seules. Elles sont combinées avec des méthodes de type séparation et évaluation.

1.3.4 Méthode par séparation et évaluation

C'est une des méthodes les plus utilisées pour la résolution de $(PLNE)$ [72]. Elle combine de façon élégante la méthode primale-duale du simplexe et le principe de séparation et évaluation dans une arborescence de recherche. Elle repose sur les étapes suivantes :

Au nœud 0 de l'arborescence :
On pose $U = \underline{z}$ un minorant de la valeur optimale de la fonction objectif et on résout le problème relaxé (PL) avec la méthode primale du simplexe.
Séparation : Si la solution optimale, disons $\widehat{x}$, a ses coordonnées $\widehat{x}_j \in \mathbb{N}$, pour tout $j = 1, ..., n$, terminer. Cette solution est optimale pour $(PLNE)$. Sinon, choisir une variable x_j qui n'est pas entière. Partitionner l'ensemble des solutions en deux sous-ensembles en ajoutant l'une ou l'autre des contraintes :

$$x_j \leq \lfloor \widehat{x}_j \rfloor \quad \text{ou} \quad x_j \geq \lceil \widehat{x}_j \rceil$$

où $\lfloor a \rfloor$ indique la partie entière du nombre réel a, et $\lceil a \rceil = \lfloor a \rfloor + 1$.
Deux nouveaux nœuds sont alors créés. Notons que $\widehat{x}$ viole à la fois ces deux contraintes et que le coût optimal de chacun des deux sous-problèmes sera inférieur à celui de (PL).
Evaluation :.Résoudre le programme (PL) avec la nouvelle contrainte en utilisant la méthode duale du simplexe.
Test de stérilisation :
1. Si la nouvelle solution optimale x est telle que $z_{PL}(x) \leq U$.

2. Si le dual du simplexe implique qu'il n'y a pas de solutions réalisables.
3. Si la solution optimale x est réalisable pour le problème initial ($PLNE$), c'est-à-dire entière. Si de plus, $z_{PL} > U$, alors poser $U = z_{PL}$ et garder la solution x comme meilleure solution de la branche courante.

A un nœud k :
Faire : l'évaluation, le test de stérilisation et eventuellement, la séparation.

Plusieurs choix sont possibles pour le choix d'un nœud à évaluer : profondeur d'abord, meilleur d'abord et largeur d'abord. Le plus utilisé est profondeur d'abord pour trouver rapidement une solution réalisable et ne pas trop encombrer la mémoire du calculateur.

La méthode s'arrête lorsque tous les nœuds pendants de l'arborescence (nœuds de degré 1) sont sondés (le test de stérilisation est positif).

1.3.5 Méthode "branch and cut"

Cette méthode couple la méthode par séparation et évaluation avec des coupes pour résoudre des programmes linéaires en nombres entiers ou mixtes [81]. Le principe est de résoudre la relaxation continue du programme linéaire en nombres entiers à l'aide de l'algorithme du simplexe. Si la solution optimale du problème relaxé n'est pas entière, on construit une coupe (ou plusieurs) à partir du tableau optimal du simplexe qu'on rajoute à ce dernier et on optimise de nouveau le tableau du simplexe obtenu. On répéte ce procédé jusqu'à ce qu'une solution entière, soit trouvée, (solution optimale dans le cas d'un $PLNE$), ou un test d'arrêt est vérifié. A cette étape, si on n'a pas encore trouvé de solution entière, alors la partie séparation et évaluation de l'algorithme commence.

Chapitre 2

Programmation multi-objectif linéaire discrète

2.1 Introduction

L'introduction des problèmes de la programmation linéaire multi-objectif à variables entières ($MOILP$: Multi$O$bjective Integer Linear Programming) a permis d'élargir considérablement le champ d'application du monde réel de la programmation linéaire. En effet, très souvent, un seul critère ne décrit pas avec précision les objectifs fixés par les décideurs. Même une combinaison linéaire des objectifs peut ne pas générer une solution qui satisfait les besoins des decideurs. Ainsi, pour la plupart des problèmes pratiques, plusieurs objectifs contradictoires doivent être pris en compte simultanément ; l'amélioration de l'un induit une détérioration d'un autre. Dans de tels problèmes on ne s'intéresse pas à la recherche de solutions optimales comme pour un problème de la programmation linéaire en nombres entiers ($PLNE$), mais plutôt à trouver un ensemble de solutions de compromis qui pourraient satisfaire aux exigences fixées par les décideurs, et à partir duquel ils peuvent choisir une solution. Dans les problèmes multi-objectif on ne parle plus de solutions optimales, mais plutôt de solutions efficaces réalisant un bon compromis.

Un problème de la programmation linéaire multi-objectif en nombres entiers peut être formulé comme suit :

$$(P)\begin{cases} \max z^1 = c^1 x \\ \max z^2 = c^2 x \\ \vdots \\ \max z^k = c^k x \\ Ax \leq b \\ x \geq 0 \\ x \ \textit{vecteur entier} \end{cases} \tag{2.1}$$

avec c^i un vecteur ligne de $\mathbb{R}^n$ pour tout $i = 1, \ldots, k,\ A$ est une $m \times n-$matrice et b un $m-$vecteur à coefficients entiers.

Après l'introduction de la problématique multi-objectif à travers les définitions et concepts de base, nous passons en revue des méthodes exactes de résolution de problèmes de la programmation linéaire multi-objectif en nombres entiers. Compte tenu de la compléxité exponentielle de ses méthodes générales, des méthodes approchées sont développées pour donner des solutions à des problèmes particuliers d'optimisation combinatoire multi-objectif de grande taille avec plusieurs objectifs. Si ces dernières méthodes ont reçue un intérêt grandissant durant les vingt dernières années, il n'en est pas de même pour les méthodes exactes qui sont en infériorité numérique.

Dans la littérature, plusieurs approches de résolution du problème de la programmation multi-objectif sont considérées. Dans les méthodes d'optimisation à priori, on fixe à l'avance un compromis entre les objectifs. On résout alors un programme dont l'objectif est une aggrégation de tous les objectifs du problème initial. Cependant, la modélisation du compromis n'est pas toujours facile et si la solution trouvée n'est pas satisfaisante, on doit résoudre de nouveau le problème avec un autre compromis [**102**], [**106**].

Les méthodes interactives font intervenir le décideur dans le processus de recherche de solutions en répondant à différentes questions afin d'orienter la recherche. Cette

approche permet donc de bien prendre en compte les préférences du décideur, mais nécessite sa présence tout au long du processus de recherche **[8]**, **[42]**.

Les méthodes d'optimisation à posteriori cherchent soit l'ensemble de toutes les solutions de bon compromis par des méthodes exactes, soit un ensemble de bonnes solutions bien réparties. La solution appropriée est alors sélectionnée parmi les solutions trouvées **[30]**, **[108]**, **[42]**, **[83]**, **[104]**, **[78]**.

Philip **[85]** a définit le problème d'optimisation d'une fonction sur l'ensemble des solutions efficaces. Ce problème d'optimisation globale suppose connue une fonction d'utilité à optimiser sur l'ensemble des solutions de bon compromis, **[113]**. Le problème considérant toutes les fonctions ainsi que les contraintes linéaires et les variables entières, est traité dans **[2]**, **[65]** et **[22]**.

Nous nous placerons dans le cadre de méthodes où la modélisation des préférences n'est pas requise et où le procédé d'optimisation doit être puissant afin de fournir l'ensemble des solutions de bons compromis par des méthodes exactes.

2.2 Concepts de base

On note $D = \{x \in \mathbb{Z}^n \mid Ax = b, x \geq 0\}$ l'ensemble des solutions réalisables de (P) et on note S l'ensemble D sans la contrainte d'intégrité des variables.

A chaque solution réalisable x dans D, on associe son image $z(x) = (z^1(x), z^2(x), \cdots, z^k(x))$ dans $\mathbb{R}^k$ et on construit donc, l'ensemble :

$$Z_D = \left\{ z \in \mathbb{R}^k \mid z = Cx,\ x \in D \right\} \tag{2.2}$$

où C est la $k \times n-$matrice composée des vecteurs lignes c^i de $\mathbb{R}^n$, $i = 1, \ldots, k$.

L'ensemble S est un polyèdre convexe de $\mathbb{R}^n$ et le caractère linéaire des critères permet d'établir que Z_S est également un polyèdre de $\mathbb{R}^k$, dont les points extrêmes correspondent à ceux de S.

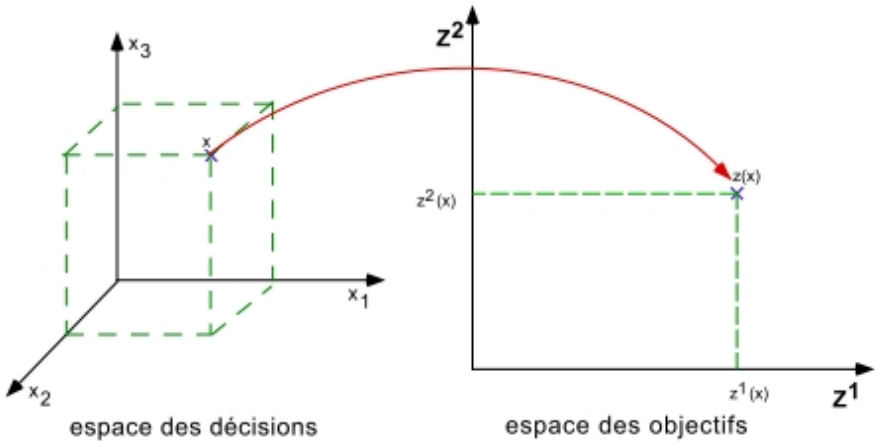

Figure 2.1 De l'espace des décisions à l'espace des critères

2.2.1 Relation de dominance

La relation de dominance pour un problème multi-objectif est définie dans l'espace des objectifs (critères) $\mathbb{R}^k$. Soient deux vecteurs critères z et w de Z_D, on dit que z domine w si pour tout indice $i \in \{1, \ldots, k\}$, $z_i \geq w_i$ avec au moins une inégalité stricte.

Cette relation définit dans Z_D une relation d'ordre partiel.

Un vecteur $z \in Z_D$ est dit faiblement non dominé s'il n'existe aucun autre vecteur $z' \in Z_D$ tel que $z'_i > z_i$, pour tout $i = 1, \ldots, k$.

2.2.2 Efficacité

La notion d'efficacité ou efficience est définie dans l'espace des décisions $\mathbb{Z}^n$. On dit qu'une solution $x \in D$ est efficace, ou Pareto optimale, s'il n'existe pas une autre solution $y \in D$ telle que le vecteur Cy domine le vecteur Cx.

Une solution x est dite faiblement efficace si le vecteur Cx est faiblement non dominé.

Il est clair qu'une solution efficace est faiblement efficace, mais l'inverse n'est pas vrai.

Résoudre le problème (P) revient à trouver, soit l'ensemble des solutions efficaces dans l'espace des décisions, soit l'ensemble des solutions non dominées dans l'espace des critères.

Théorème 2.1 *Caractérisation d'une solution efficace [15].*
Soit x^* une solution réalisable pour le problème (P). x^* est une solution efficace pour le problème (P) si et seulement si la valeur optimale de la fonction objectif φ est nulle dans le programme linéaire suivant :

$$(PE) \begin{cases} \max \varphi = \sum_{i=1}^{k} v_i \\ c^i x - v_i = c^i x^*, \ i = 1, \ldots, k \\ x \in D \\ v_i \geq 0, \ i = 1, \ldots, k \end{cases} \tag{2.3}$$

2.2.3 Point idéal

C'est le vecteur $I \in \mathbb{R}^k$ qui a pour coordonnées $\overline{z}^i = \max_{x \in D} \{z^i = c^i x\}$, $i = 1, \ldots, k$.

En général, $I \notin Z_D$ de par la nature des critères du problème (P) qui sont conflictuels.

2.2.4 Matrice des gains et point nadir

Soit x_i^* une solution optimale obtenue en optimisant le critère z^i sur D. La matrice carrée d'ordre r suivante est appelée matrice des gains associée au problème (P) :

$$\begin{pmatrix} \overline{z}^1 & z_{12} & \cdots & z_{1k} \\ z_{21} & \overline{z}^2 & \cdots & z_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ z_{r1} & z_{r2} & \cdots & \overline{z}^k \end{pmatrix} \tag{2.4}$$

avec $\overline{z}^i = \max_{x \in D} z^i = c^i x = c^i x_i^*,\ \forall i = 1, \ldots, k$, et $z_{ij} = c^i x_j^*,\ \forall i = 1, \ldots, k$, $\forall j = 1, \ldots, k$, avec $i \neq j$.

Il faut noter que cette matrice dépend de la solution optimale x_i^* et peut donc, ne pas être unique.

Le point nadir $N \in \mathbb{R}^k$ a pour coordonnées : $n_j = \min_{i=1,\ldots,k} \{z_{ij},\ j = 1, \ldots, k\}$.

Si la matrice des gains n'est pas unique alors, le point N aussi n'est pas unique.

2.2.5 Solutions supportées et non supportées

Considérons le problème (PR) relaxation continue du problème (P) :

$$(PR) \begin{cases} \max z^1 = c^1 x \\ \max z^2 = c^2 x \\ \vdots \\ \max z^k = c^k x \\ Ax \leq b \\ x \geq 0 \end{cases} \tag{2.5}$$

Théorème 2.2 *[52]. Etant donné le problème*

$$(P_\lambda) : Max\{\lambda^t C x | x \in S\} \tag{2.6}$$

avec $\lambda \in \Lambda = \left\{ \lambda \in \mathbb{R}^k \mid \sum_{i=1}^{k} \lambda_i = 1,\ \lambda_i \geq 0,\ i = 1, \ldots, k \right\}$,
alors x *est une solution efficace pour* (PR) *si et seulement s'il existe* $\lambda \in \Lambda$ *tel que* x *est une solution optimale du problème paramétrique* (P_λ).

D'après ce théorème, l'ensemble des solutions efficaces du problème (PR) est bien caractérisé par les solutions du problème $(P_\lambda),\ \lambda \in \Lambda$. Ces solutions se trouvent sur la frontière de S. Ce n'est pas le cas pour le problème (P) [17]. Ce dernier admet des solutions efficaces, appelées solutions non supportées, qui ne sont pas optimales pour (P_λ) et ce en raison de la non convexité du domaine des solutions réalisables D. Celles qui sont optimales pour (P_λ) sont appelées solutions efficaces supportées et correspondent aux points extrêmes de l'enveloppe convexe de la frontière efficace.

2.2.6 Résolution graphique [102]

C'est une généralisation de la résolution graphique d'un (PL). L'amélioration d'une fonction objectif se faisant perpendiculairement à son gradient et dans le sens de ce dernier, l'hyperplan engendré par cette dernière génère un demi-espace d'amélioration possible lorsqu'on le fait passer par un point $\overline{x}$. L'intersection des demi-espaces d'amélioration des fonctions objectifs considérées au point $\overline{x}$ induit un cône de sommet $\overline{x}$. Si l'ensemble obtenu par intersection du cône avec l'ensemble des solutions réalisables S (respectivement D) se réduit au seul point $\overline{x}$, alors $\overline{x}$ est une solution efficace pour (PR) (respectivement pour (P), s'il est à coordonnées entières).

Définition 2.3 *Soit $v \in V$, $V \subset \mathbb{R}^n$ et $V \neq \varnothing$. Alors, V est un cône si $\alpha v \in V$ pour tout scalaire $\alpha \geq 0$.*
L'origine $0 \in \mathbb{R}^n$ est contenu dans chaque cône.

Définition 2.4 *Soit un ensemble de vecteurs $\{v^1, ..., v^k\}$ de $\mathbb{R}^n$ et l'ensemble V tel que :*

$$V = \{v \in \mathbb{R}^n \mid v = \sum_{i=1}^{k} \alpha_i v^i,\ \alpha_i \geq 0\ \forall i\} \tag{2.7}$$

V est le cône convexe généré par l'ensemble $\{v^1, ..., v^k\}$.

Un générateur $v^i \in \{v^1, ..., v^k\}$ est non essentiel si $\{v^1, ..., v^k\}\backslash\{v^i\}$ peut générer V. En d'autres termes, c'est celui qui peut s'écrire comme combinaison linéaire à coefficients non négatifs des autres générateurs. Il est essentiel sinon.

Définition 2.5 *Soit $C \subset \mathbb{R}^n$ le cône convexe généré par les gradients des k fonctions objectifs. Le cône convexe $C^{\geq}$ donné par :*

$$C^{\geq} = \{y \in \mathbb{R}^n \mid Cy \geq 0 \text{ et } Cy \neq 0\} \cup \{0_{\mathbb{R}^n}\} \tag{2.8}$$

est le cône polaire semi-positif de C.

Définition 2.6 *Soit $\overline{x} \in D$ et $C^{\geq}$ le cône polaire semi-positif du cône généré par les gradients des k fonctions objectifs.*
L'ensemble dominant en $\overline{x}$ est donné par :

$$D_{\overline{x}} = \{\overline{x}\} \oplus C^{\geq} = \{x \in \mathbb{R}^n \mid x = \overline{x} + y,\ y \in C^{\geq}\} \tag{2.9}$$

L'ensemble dominant en $\overline{x}$ contient tous les points dont les vecteurs critères dominent le vecteur critère de $\overline{x}$. Géométriquement, il représente la translation du cône polaire $C^{\geq}$ au point $\overline{x} \in D$, à partir de l'origine.

Le théorème suivant montre l'importance de cet ensemble dans la détection des solutions efficaces :

Théorème 2.7 *Soit $D_{\overline{x}}$ l'ensemble dominant en $\overline{x} \in D$. Alors, $\overline{x}$ est efficace si et seulement si $D_{\overline{x}} \cap D = \{\overline{x}\}$.*

Preuve. Supposons que $D_{\overline{x}} \cap D \neq \{\overline{x}\}$, alors il existe $x \in D_{\overline{x}} \cap D,\ x \neq \overline{x}$. Comme $x \in D_{\overline{x}}$, alors $x = \overline{x} + y,\ y \in C^{\geq}$.
Ceci signifie que $Cy \geq 0$ et $Cy \neq 0$, c'est-à-dire $Cx \geq C\overline{x}$ et $Cx \neq C\overline{x}$ avec $x \in D$. Ainsi, $\overline{x}$ n'est pas solution efficace.
D'autre part, supposons que $D_{\overline{x}} \cap D = \{\overline{x}\}$. S'il existe x tel que Cx domine $C\overline{x}$ alors $x \in D_{\overline{x}}$, mais dans ce cas $x \notin D$. Ainsi il n'existe pas une solution réalisable dont le vecteur critère domine celui de $\overline{x}$, et par suite $\overline{x}$ est solution réalisable efficace. □

Corollaire 2.8 *Si $C^{\geq} = \{0_{\mathbb{R}^n}\}$, alors $\forall x \in D,\ x$ est solution efficace*

En effet ; $\forall \overline{x} \in D,\ D_{\overline{x}} \cap D = \{x \in \mathbb{R}^n \mid x = \overline{x} + 0\} \cap D = \{\overline{x}\}$.

Example 1

Considérons le programme multi-objectif linéaire discrèt suivant :

$$(P)\begin{cases} \max z^1 = x_1 \\ \max z^2 = -x_1 + 3x_2 \\ x_1 + 2x_2 \leq 7 \\ 0 \leq x_1 \leq 5 \\ 0 \leq x_2 \leq 3 \\ x_1,\ x_2\ \textit{entiers} \end{cases} \tag{2.10}$$

Le maximum du critère z^1 est égal 5. Il est atteint au point $x_1^* = (5,\ 0)^t$ sur l'ensemble des solutions réalisables :
$D = \{(x_1, x_2) \in \mathbb{Z}^2 \mid x_1 + 2x_2 \leq 7,\ 0 \leq x_1 \leq 5,\ 0 \leq x_2 \leq 3\}$.
Ce point n'est pas unique puisque le point $y_1^* = (5,\ 1)^t$ réalise aussi l'optimum de z^1. La solution $x_2^* = (0,\ 3)^t$ est optimale pour le critère z^2 sur D et elle est unique.

La détection graphique des solutions efficaces en utilisant l'ensemble dominé $D_{\overline{x}}$, permet de trouver l'ensemble solutions suivantes :
$Eff = \{(5,1),\ (4,1),\ (3,2),\ (2,2),\ (1,3),\ (0,3)\}$.

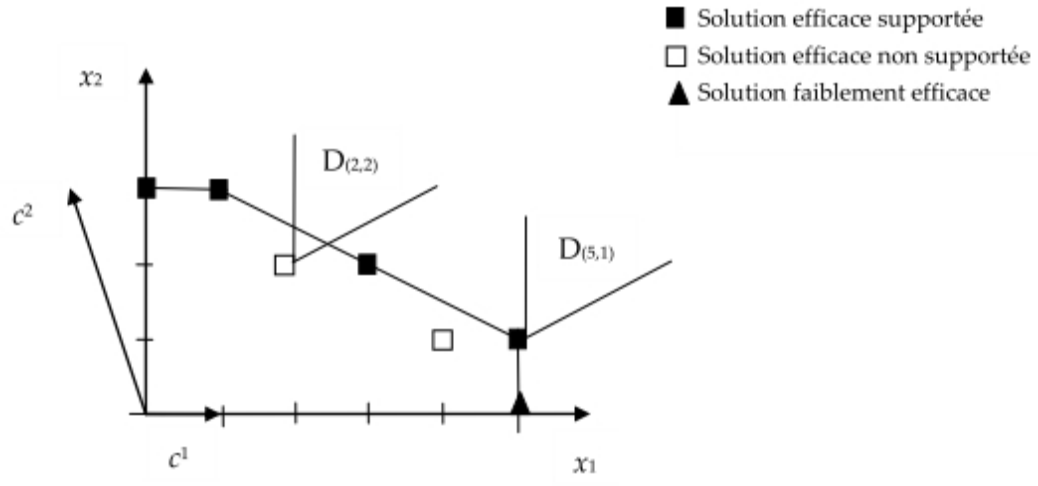

Figure 2.2 Représentation des solutions dans l'espace de décisions

La matrice des gains n'est pas unique dans ce cas, cependant, relativement à la solution x_1^*, elle s'écrit :

$$\begin{pmatrix} 5 & 0 \\ -5 & 9 \end{pmatrix}$$

Le point idéal est $I = (5, 9)^t$ et le point nadir est, dans ce cas, $N = (0, -5)$.

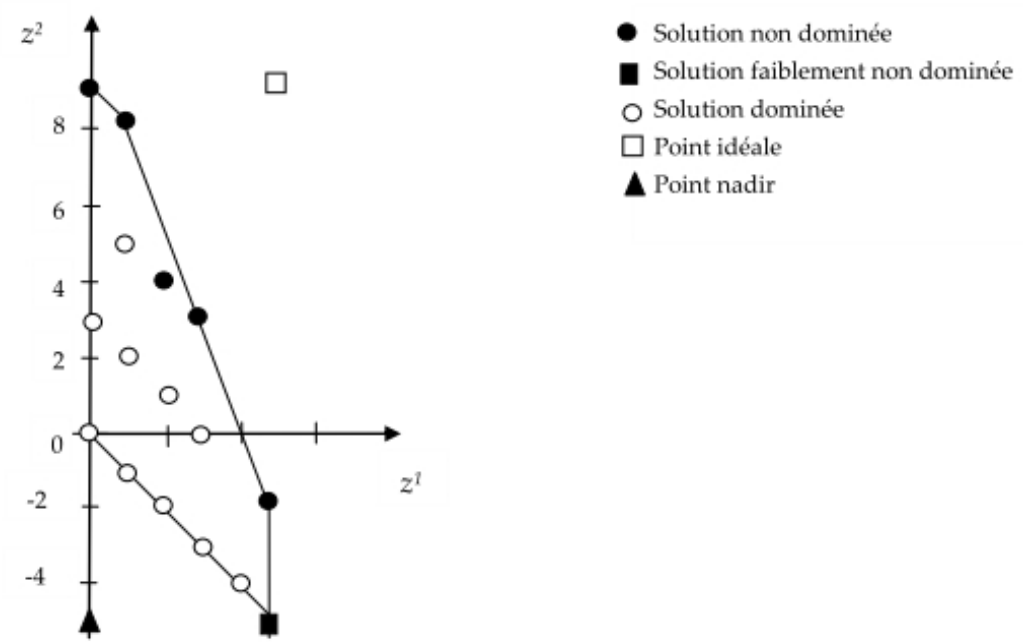

Figure 2.3 Représentation des solutions dans l'espace des critères

2.3 Quelques méthodes de résolution

Avec l'avènement de la programmation multi-objectif, il est tout à fait naturel que l'on s'intéresse aux problèmes d'optimisation en considérant plus d'une fonction à optimiser. Particulièrement pour les problèmes d'optimisation combinatoire, une nouvelle branche de la programmation multi-objectif est née et est connue sous le nom d'optimisation combinatoire multi-objectif, ou MultiObjective Combinatorial Optimisation (MOCO) en anglais. Les problèmes de programmation multi-objectifs en nombres entiers sont étroitement liés à des problèmes combinatoires, et comme

pour les problèmes d'optimisation combinatoire mono-objectif, deux types de méthodes de résolution sont développées ; des méthodes dédiées à des classes particulières de problèmes multi-objectif et des méthodes générales qui opèrent sur des programmes linéaires en nombres entiers ou mixtes dont la structure des données et des contraintes est quelconque et ne présente pas nécessairement des particularités. Il faut noter cependant que, si un problème d'optimisation combinatoire est facile à résoudre, il n'est pas de même pour sa version multi-objectif. Pour exemple, le problème d'affectation bi-objectif est NP-complet, alors que le même problème est connu pour être dans la classe P dans sa version mono-objectif.

Dans ce paragraphe, nous donnons un bref aperçu sur des méthodes générales pour la résolution d'un $MOILP$. Rappelons que la résolution d'un problème multi-objectif consiste à déterminer soit l'ensemble des solutions efficaces dans l'espace des décisions noté Eff, soit l'ensemble des solutions non dominées dans l'espace des critères noté SND. Dans la littérature, l'accent est mis sur la recherche de l'ensemble des solutions non dominées compte tenu de son cardinal qui est généralement moins important que celui des solutions efficaces ; plusieurs solutions efficaces pouvant donner lieu à un même vecteur non dominé.

Il arrive que deux solutions efficaces différentes dans l'espace de décisions ont exactement les mêmes valeurs pour tous les objectifs. Si on garde les deux solutions dans Eff, on parle alors de l'ensemble complet, sinon c'est juste un ensemble minimal.

2.3.1 Méthode de Klein et Hannan [68]

Klein et Hannan ont donné un algorithme d'enumération implicite qui consite à résoudre une séquence de programmes linéaires mono-objectif en variables entières de plus en plus contraignants. Les contraintes additionnelles permettent à la fois de supprimer les solutions efficaces déjà trouvées et d'autres solutions réalisables entières qui ne sont pas efficaces. La méthode résout $(q-1)^k$ sous-problèmes à l'itération j, où q indique le nombre de solutions efficaces jusqu'à l'itération $j-1$. Le nombre de sous problèmes résolus par la méthode croit exponentiellement en

fonction de q.

La méthode est décrite pour générer aussi bien l'ensemble de toutes les solutions efficaces qu'une partie seulement de cet ensemble.

Algorithme 2.9 (KH)

Etape 1. *Résoudre le problème* (P_1) *définit par*

$$(P_1) : \max\{z^s = c^s x \mid x \in D\} \tag{2.11}$$

Si la solution optimale de (P_1)*, soit* x^1*, est unique alors elle est efficace pour* (P)*. 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* $Eff(P_1)$ *des solutions efficaces de* (P_1) *générées à l'étape 1.*

Etape générale j. *A l'étape* j*, le problème* (P_j) *est résolu :*

$$(P_j) \begin{cases} \max z^s = c^s x \\ x \in D \\ \bigcap\limits_{l=1}^{q} \left(\bigcup\limits_{\substack{i=1 \\ i \neq s}}^{k} c^i x \geq c^i y^l + f_i \right) \end{cases} \tag{2.12}$$

Avec : $f_i \geq 1$ *entier,* y^l $(l = 1, \ldots, q)$ *les points efficaces obtenus jusqu'à l'étape* $j - 1$*.*

La procédure s'arrête à une étape n *pour laquelle le problème obtenu* (P_n) *est non réalisable.*

Si à chaque étape j, $f_i = 1$, $\forall i = 1, \ldots, k$, $i \neq s$, la procédure trouve l'ensemble de toutes les solutions efficaces du problème (P). Cependant, si $f_i > 1$ pour certaines valeurs de i, seulement un sous ensemble de solutions efficaces est généré.

Les auteurs montrent que si le problème (P_j) posséde un ensemble de solutions optimales X_j^*, alors le sous ensemble $Eff(P_j)$ des solutions efficaces dans X_j^*, est aussi efficace pour le problème (P). De plus, si la solution optimale de (P_j) est unique, alors elle est efficace pour (P).

2.3.2 Méthode de Gupta et Malhotra [59]

La première méthode proposée par Gupta et Malhotra est une variante de la méthode proposée par Klein et Hannan. Elle est décrite pour générer l'ensemble de toutes les solutions efficaces en réduisant, à chaque étape, le nombre de contraintes additionnelles de Klein et Hannan. Malheureusement, ce n'est pas toujours le cas ; la méthode peut s'arrêter avant terme, sans trouver toutes les solutions efficaces. Un contre exemple est donné par **[4]** et dans **[1]**

Algorithme 2.10 (GM)

Etape 1. *Résoudre le problème* (P_1) *définit par*

$$(P_1) : \max\{z^1 = c^1 x \mid x \in D\} \tag{2.13}$$

Déterminer toutes les solutions optimales de (P_1)*, et construire l'ensemble des solutions efficaces* $E(P_1) = \{y_1^1,\ y_1^2, \ldots,\ y_1^q\}$ *par élimination des solutions dominées.*

Etape générale (j+1). *Résoudre le problème* (P_{j+1}) *définit par :*

$$(P_{j+1}) \begin{cases} \max z^1 = c^1 x \\ x \in D \\ c^1 x \leq \max\{c^1 y - 1 \mid y \in E(P_j)\} = c^1 y^s - 1 \\ c^i x \geq c^i y^s + 1 \text{ pour au moins un } i,\ i = 2, \ldots, k \end{cases} \tag{2.14}$$

où $E(P_j)$ *représente l'ensemble de toutes les solutions potentiellement efficaces obtenues à l'étape* j*.*

S'il y a plus d'un élément y^s *donnant le maximum de* $\{c^1 y - 1 \mid y \in E(P_j)\}$ *alors, sélectionner* y^s *arbitrairement. Enregistrer toutes les solutions potentiellement efficaces obtenues à l'étape* $j+1$ *dans* $E(P_{j+1})$*.*

Test d'arrêt. *La procédure s'arrête à une étape* n *dans l'un des cas suivants : toutes les solutions de* (P_n) *ne sont pas efficaces pour* (P)*, ou bien* (P_n) *n'est pas réalisable.*

En fait, c'est le premier test d'arrêt qui est défaillant car on peut omettre des solutions efficaces si on s'arrête prématurément. En effet, il est possible que le problème

(P_{n+p}), $p \geq 1$, admette des solutions efficaces même lorsque le problème (P_n) n'en admet pas.

Une deuxième méthode est décrite par les mêmes auteurs dans **[59]**, mais là encore la méthode ne permet pas de trouver toutes les solutions efficaces. Pour plus de détails sur la méthode et la présentation d'un contre exemple, on pourra se référer aux travaux de Moulaï **[79]** , de Chaabane **[21]** et de Ouail **[82]**.

2.3.3 Méthode de Abbas et Moulaï [4]

Utilisant les coupes fractionnaires de Gomory, l'algorithme développé par les auteurs range les solutions réalisable du problème (P) dans l'ordre décroissant des valeurs de l'un des critères (ranking), sans omettre les solutions entières alternatives. Elle peut être vue comme une alternative à celle de Gupta et Malhotra où les auteurs ont proposé un autre test d'arrêt permettant à l'algorithme de trouver toutes les solutions efficaces.

On considère le problème :

$$(P_1) : \max\{z^1 = c^1 x \mid x \in D\}$$

dont le problème relaxé est :

$$(P_1R) : \max\{z^1 = c^1 x \mid x \in S\} \tag{2.15}$$

avec $S = \{x \in \mathbb{R}^n \mid Ax = b,\ x \geq 0\}$.

Pour les besoins de la description de la méthode, on définit les paramétres suivants pour $k \geq 1$:

$\cdot$ $S_k = \{x \in \mathbb{R}^{n_k} \mid A_k x = b_k,\ A_k \in \mathbb{R}^{m_k \times n_k},\ b \in \mathbb{R}^{m_k},\ x \geq 0\}$ comme étant la région courante tronquée de S obtenue par application de la coupe :

$$\sum_{j \in N_{k-1} \setminus \{j_{k-1}\}} x_j \geq 1 \tag{2.16}$$

et éventuellement des coupes successives de Gomory, avec j_{k-1} un indice hors base quelconque.

· x^k : la k *ieme* solution optimale entière du problème (P_1) obtenue sur S_k.

· B_k : est une base de S_k.

· N_k : ensemble des indices des variables hors base de x^k.

· $\widehat{A}_k = (A_{B_k})^{-1} A_k = (\widehat{a}_{f(i)j})\ i = 1, \ldots, m_k;\ j = 1, \ldots, n_k$

où $f(i) \in \{1, \ldots, n_k\}$ est l'indice de la variable de base associée à la ligne i, $i \in \{1, \ldots, m_k\}$ à l'étape k.

· $\widehat{C}_k = C_k - \pi_k A_k$, avec $\pi_k = (C_{B_k})(A_{B_k})^{-1}$.

· $\Gamma_k = \{j \in N_k \mid (\widehat{c}_k^1)_j < 0$ et $(\widehat{c}_k^i)_j > 0$ pour au moins un critère $i,\ i = 2, \ldots, r\}$.

· $\Omega k = \{j \in N_k \mid (\widehat{c}_k^1)_j = 0\}$.

· $\Psi k = \{j \in N_k \mid (\widehat{c}_k^1)_j < 0$ et $(\widehat{c}_k^i)_j < 0$ pour au moins un critère $i,\ i = 2, \ldots, r\}$.

Définition 2.11 *Une arête E^{j_k}, $j_k \in N_k$ incidente à x^k est définie comme étant l'ensemble :*

$$E^{j_k} = \left\{ X \in \mathbb{R}^{n_k} \mid \left\{ \begin{array}{l} x_i = x_i^k - \theta_{j_k} \widehat{a}_{f(i)j_k},\ i \in B_k \\ x_{j_k} = \theta_{j_k} \\ x_l = 0 \quad \forall l \in N_k \backslash \{j_k\} \end{array} \right. \right\} \tag{2.17}$$

où : $0 \leq \theta_{j_k} \leq \theta = min_{i \in B_k} \left\{ \frac{x_i^k}{\widehat{a}_{f(i)j_k}} \mid \widehat{a}_{f(i)j_k} > 0 \right\}$.

Les points entiers se trouvant sur l'arête E^{j_k} sont identifiés de telle sorte que θ_{j_k} soit entier et $\theta_{j_k} \times \widehat{a}_{f(i)j_k}$ entier $\forall i \in B_k$.

On note par nb_{j_k} le nombre de solutions entières se trouvant sur l'arête E^{j_k} y compris x^k.

Notons par $SND(P)$ l'ensemble des solutions potentiellement non dominées générées jusqu'à l'étape $k, k \geq 1$.

Algorithme 2.12 (AM)

Etape 1. *Résoudre le problème* (P_1) *et trouver la solution optimale entière* x^1 *sur* S_1*. Construire l'ensemble* Ω_1*.*

Etape 2. *Si* $\Omega_1 = \varnothing$*,* x^1 *est l'unique solution optimale sur* S_1*.*

Soit $(z_1^1, z_1^2, ..., z_1^r)$ *le vecteur critère correspondant, alors* $SND(P) := \{(z_1^1, z_1^2, ..., z_1^r)\}$*.*

Tronquer le point x^1 *par la coupe de Dantzig :*

$$\sum_{j \in N_k} x_j \geq 1 \tag{2.18}$$

et éventuellement, par application de la méthode duale du simplexe et des coupes successives de Gomory, on obtient une solution entière x^2 *dans la région tronquée* S_2*. Mettre à jour* $SND(P)$*.*

Si $\Omega_1 \neq \varnothing$*, choisir un indice quelconque* $j_1 \in \Omega_1$ *et calculer le nombre* θ *de l'opération pivot.*

(a) *Si* $\theta \geq 1$*, déterminer toutes les solutions entières alternatives à* x^1*,* y_1^q*,* $q = 2, \ldots, nb_{j_1}$*, le long de l'arête* E^{j_1} *et mettre à jour* $SND(P)$*.*

Comme les solutions alternatives ont la même valeur de z^1 *que celle de* x^1*, le premier point potentiellement non dominé est choisit comme le* $r-$*uplet ayant la plus grande valeur de* z^2*, sinon choisir celui qui a la plus grande valeur de* z^3 *et ainsi de suite jusqu'à l'obtention du premier* $r-$*uplet potentiellement non dominé.*

Tronquer l'arête E^{j_1} *par la coupe :* $\sum_{j \in N_1 \setminus \{j_1\}} x_j \geq 1$*.*

L'algorithme dual du simplexe et des coupes successives de Gomory éventuelles, permettent d'obtenir une solution entière x^2 *dans la région tronquée* S_2*. Mettre à jour* $SND(P)$*.*

(b) *Si pour tout* $j_1 \in \Omega_1$*, on a* $\theta < 1$*, alors choisir un indice quelconque* $j_1 \in \Omega_1$ *et appliquer la coupe :* $\sum_{j \in N_1 \setminus \{j_1\}} x_j \geq 1$*.*

De la même manière (appliquer la méthode duale du simplexe et des coupes de Gomory éventuelles), on obtient une solution entière x^2 *dans la région tronquée* S_2*. Mettre à jour* $SND(P)$*.*

Etape k**.** $(k \geq 3)$ *Choisir un indice* $j_{k-1} \in \Omega_{k-1}$ *et explorer l'arête correspondante*

à la recherche de solutions entières $y_{k-1}^q,\ q = 2, \ldots, nb_{j_{k-1}}$ *alternatives à* x^{k-1}.
Mettre à jour l'ensemble $SND(P)$.
L'arête $E^{j_{k-1}}$ *est tronquée par la coupe :* $\sum_{j \in N_{k-1} \setminus \{j_{k-1}\}} x_j \geq 1$.
Après application de la méthode duale du simplexe et éventuellement, des coupes successives de Gomory, la solution optimale entière obtenue sur la région S_k *est* x^k.
Test d'arrêt. *Le processus se termine quand l'opération pivot de la méthode duale du simplexe devient impossibile, indiquant que la région courante ne contient aucun point entier.*

2.3.4 Méthode de Abbas et Chaabane [1]

Cette méthode est une forme modifiée de la méthode de Gupta et Malhotra **[59]**, où le test d'arrêt a été modifié afin de produire toutes les solutions efficaces du problème (P). Cette méthode utilise les mêmes paramètres que ceux définis dans la méthode précédente.

Notons par $SND(P)$ l'ensemble des solutions potentiellement non dominées générées jusqu'à l'étape $k, k \geq 1$.

Algorithme 2.13 (AC)
Etape1. *Résoudre le problème* (P_1). *Soit* x^1 *la solution optimale et* $(z_1^1, z_1^2, ..., z_1^r)$ *le vecteur critère correspondant.*
Si $\Omega_1 = \varnothing$, *alors la solution optimale est unique.* $SND(P) := \{(z_1^1, z_1^2, ..., z_1^r)\}$ *et aller à l'étape 2.*
Si $\Omega_1 \neq \varnothing$, *alors la solution optimale peut ne pas être unique. Pour chaque* $j \in \Omega_1$, *calculer* θ.
(a) *Si pour tout* $j \in \Omega_1$, *on a* $\theta < 1$, *alors il n'y a pas de solutions alternatives à* x^1 *le long des arêtes* $E^j,\ j \in \Omega_1$.
$SND(P) := \{(z_1^1, z_1^2, ..., z_1^r)\}$ *et aller à l'étape 2.*
(b) *Sinon, tant qu'il existe au moins un* $j \in \Omega_1$, *tel que* $\theta \geq 1$ *faire :*
· *Explorer l'arête* E^j,

· *Evaluer en chacune des solutions entiières trouvées les r critères,*

· *Mettre à jour* $SND(P)$.

Choisir arbitrairement un $j \in \Omega_1$, *initialiser* k *à* 1 *et aller à l'étape (2.2).*

Etape 2. $k = 1$

Etape 2.1. *Construire l'ensemble* Ψ_k.

Si $\Psi_k = \varnothing$, *aller à l'étape (2.2) (la coupe devient une coupe de Dantzig :* $\sum_{j \in N} x_j \geq 1$. *Sinon, poser* $\psi = \Psi_k$ *et aller à (a).*

(a) *Choisir un indice* $j_k \in \psi$ *et calculer le nombre* θ.

· *Si* $\theta < 1$, *il n'y a aucune solution entière sur l'arête* E^{j_k}, $\psi := \psi \backslash \{j_k\}$. *Si* $\psi = \varnothing$, *choisir unj*$_k \in \Psi_k$ *et aller à l'étape (2.2), sinon aller à (a).*

· *Sinon, déterminer les solutions entières sur* E^{j_k}, *évaluer en chacune d'elles les r critères et mettre à jour l'ensemble SND(P). Aller à l'étape (2.2).*

Etape 2.2. *Utiliser la coupe* $\sum_{j \in N_k \backslash \{j_k\}} x_j \geq 1$ *pour réduire le domaine de recherche et par application des méthodes duale du simplexe et les coupes de Gomory si nécessaire, on obtient* x^{k+1} *comme étant la solution optimale du problème augmenté. Mettre à jour* $SND(P)$, $k := k + 1$ *et aller à l'étape (2.1).*

Test d'arrêt. *La procédure prend fin quand l'opération pivot est impossible, le problème est devenu non réalisable dans la nouvelle région tronquée et la liste finale* $SND(P)$.

Remarque 2.14 *Aussi bien dans l'algorithme AM que dans l'algorithme AC, on passe en revue toutes les solutions réalisables du problème (P). De plus, pour la recherche des solutions entières, ces deux algorithmes utilisent les coupes fractionnaires de Gomory qui, comme il a été souligné dans le chapitre précédent, convergent très lentement.*

2.3.5 Méthode de Sylva et Crema [104]

La méthode développée par Sylva et Crema est une variante de celle de Klein et Hannan [**68**]. Son principe repose sur la résolution d'une succession de programmes

linéaires en nombres entiers ($PLNE$) optimisant à chaque étape une combinaison positive des r critères du problème (P). Ce dernier est augmenté de nouvelles contraintes et variables bivalentes dans le but d'éliminer des solutions non efficaces et assurer la détection d'une nouvelle solution efficace. Le nombre de $PLNE$ à résoudre est égale au nombre de solutions non dominées de (P) plus un problème non réalisable. Les auteurs considèrent que les données du problème (P) sont entières, aussi bien la matrice des contraintes A, le vecteur second membre b que les coefficients c_j^i de tous les critères.

Algorithme 2.15 (SC)

Etape 1. *Après avoir fixer le vecteur poids $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_r)$ à des valeurs strictement positives, la première étape de l'algorithme consiste en la résolution du problème :* $(P_1) \quad \max \left\{ \sum_{i=1}^{r} \lambda_i c^i x \mid x \in D \right\}$

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

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

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

Après k étapes du processus :

Si le problème (P_k) est non réalisable, alors l'algorithme s'arrête.

Sinon, une nouvelle solution efficace x^k est trouvée et le nouveau problème (P_{k+1}) est définit à partir de (P_k) en éliminant toutes les solutions vérifiant $c^i x \leq c^i x^k$, $\forall i = 1, \ldots, r$. Ceci se traduit par le rajout des contraintes suivantes :

$$\begin{cases} c^i x \geq (c^i x^k + f_i) y_i^k - M_i (1 - y_i^k) \quad i = 1, \ldots, r. \\ \sum_{i=1}^{r} y_i^k = 1 \quad y_i^k \geq 0, \quad i = 1, \ldots, r \end{cases} \tag{2.19}$$

où $-M_i$ est un minorant pour toute valeur réalisable de la ième fonction objectif et $f_i \geq 1$, entier, représente la plus petite augmentation possible du ième critère.

Etape générale k. *Résoudre le problème* (P_k) :

$$(P_k)\begin{cases} \max \sum_{i=1}^{r} \lambda_i c^i x \\ x \in D \\ c^i x \geq (c^i x^j + f_i) y_i^j - M_i(1 - y_i^j),\ i = \overline{1,r};\ j = \overline{1,k-1} \\ \sum_{i=1}^{r} y_i^j = 1,\ j = \overline{1,k-1}, \\ y_i^j \in \{0,1\},\ i = \overline{1,r};\ j = \overline{1,k-1} \end{cases} \tag{2.20}$$

Pour $f_i = 1 \ \ \forall i = 1, \ldots, r$, les auteurs montrent que la méthode génère toutes les solutions non dominées. Lorsque $f_i > 1$, elle prend une valeur permettant d'atteindre la valeur minimale souhaitait par le décideur pour le *ième* critère. Dans ce cas, seul un sous ensemble de solutions non dominées est trouvé.

Il faut noter que cette méthode présente l'avantage de générer une solution efficace, et donc une solution non dominée correspondante, à chaque étape de résolution d'un programme linéaire en nombres entiers (P_k). Cependant, l'incovénient réside dans le nombre de contraintes et variables additionnelles au programme initial (P) pour obtenir le programme (P_{k+1}) qui est non seulement fonction du nombre de critères du problème (P), mais aussi du nombre de solutions efficaces. En effet, le programme (P_{k+1}) est augmenté de $k(r+1)$ contraintes et kr variables bivalentes par rapport à (P), où k désigne le nombre de solutions efficaces trouvées jusqu'à la fin de l'étape k.

Les auteurs proposent une autre méthode qui ne génère qu'une partie de l'ensemble des solutions non dominées **[105]**. L'approche consiste à trouver un ensemble de solutions non dominées bien dispersées, en se basant sur la maximisation de la distance norme infinie d'un ensemble de solutions connues. Ils affirment que leur approche fournit une représentation plus homogène de l'ensemble non dominée par rapport à leur méthode décrite dans **[104]** lorsque $f_i > 1$.

2.3.6 Méthode de Özlen et Azizoğlu [83]

Soit le problème à résoudre :

$$(MP)\begin{cases} \min f_1(x) \\ \min f_2(x) \\ \vdots \\ \min f_r(x) \\ x \in D \end{cases} \tag{2.21}$$

où $f : \mathbb{R}^n \to \mathbb{R}^r$ telle que $f(x) = (f_i(x))_{i=\overline{1,r}}$.

Cette méthode décrite pour générer l'ensemble des solutions non dominées de (MP), est une amélioration de la méthode connue sous le nom $\epsilon-$contrainte **[61]**, qui consiste à optimiser le problème sur l'un des objectifs en bornant tous les autres objectifs, ce qui revient pour l'objectif $i \in \{1, \ldots, r\}$ du problème (MP) à résoudre le problème suivant :

$$\begin{cases} \min f_i(x) \\ x \in D \\ f_p(x) \leq \epsilon_p, \ \forall p \in \{1, \ldots, r\} \setminus \{i\} \end{cases} \tag{2.22}$$

Alors que dans la méthode $\epsilon-$contrainte les valeurs ϵ_p sont diminuées d'une unité dans le cas de minimisation multi-objectif, la méthode décrite dans **[83]** diminue ces valeurs en tenant compte de la solution non dominée actuelle. En outre, la fonction objectif n'est plus l'un des critères, mais une combinaison pondérée et appropriée des critères de (MP) qui assure l'efficacité des solutions obtenues.

Dans cette méthode, on montre que le problème multi-objectif (MP) est équivalent (au sens des solutions efficaces) au problème suivant, obtenu en transformant le $r^{ème}$

critère en contrainte :

$$(Q_{r-1})\begin{cases} \min f_1(x) + \epsilon_r f_r(x) \\ \vdots \\ \min f_{r-1}(x) + \epsilon_r f_r(x) \\ f_r(x) \leq l_r \\ x \in D \end{cases} \tag{2.23}$$

où l_r est une borne inférieure de f_r. Ce problème est à son tour équivalent au problème obtenu en transformant le critère $r-1$ en contrainte et ainsi de suite jusqu'à arriver à la résolution du programme linéaire en nombres entiers mono-objectif suivant :

$$(Q_1)\begin{cases} \min f_1(x) + \sum_{i=2}^{r} w_i f_i(x) \\ f_i(x) \leq l_i, \ i = 2, \ldots, r \\ x \in D \end{cases} \tag{2.24}$$

où les poids w_i sont calculés en fonction des $\overline{f_i}$ et $\underline{f_i}$, bornes supérieure et inférieure de f_i respectivement, de sorte que la solution optimale de (Q_1) soit efiicace pour le problème (MP) et induit donc, une solution non dominée..

L'idée est que pour résoudre un problème bi-objectif, on le ramène à un problème mono-objectif. Pour résoudre un problème tri-objectif, on détermine d'abord les solutions non dominées d'un problème bi-objectif correspondant, et de façon générale, pour résoudre un problème avec r objectifs, on doit le ramener à un problème à $r-1$ objectifs en utilisant chaque fois la transformation $\epsilon-$contrainte.

Génération de l'ensemble des solutions non dominées du problème multi-objectif (MP) :

Algorithme 2.16 (ÖA)

Etape 0. $SND_r = \varnothing$, *(ensemble des solutions non dominées)*

Trouver les bornes supérieures $\overline{f_i}$, *les bornes inférieures* $\underline{f_i}$ *de tous les critères* i,

$i = 2, \ldots, r$.

Poser $w_r = \frac{1}{(\overline{f_2}-\underline{f_2}+1)(\overline{f_3}-\underline{f_3}+1)\cdots(\overline{f_r}-\underline{f_r}+1)}$, *et* $l_r = \overline{f_r}$,

Etape 1. *Résoudre le problème* (Q_{r-1}) *avec les* $r-1$ *premiers objectifs*

Si (Q_{r-1}) *est non réalisable, Stop.*

Etape 2. *Soit* SND_{r-1} *l'ensembles des solutions non dominées du problème* (Q_{r-1})

$SND_r := SND_r \cup SND_{r-1}$,

$l_r := \max\{f_r(x) \mid \ f(x) \in SND_{r-1}\} - 1$

Aller à l'étape 1

Notons que cette méthode résout initialement $2r$ programmes linéaires en nombres entiers pour le calcul de $\overline{f_i}$ et $\underline{f_i}$ de tous les critères i, $i = 2, \ldots, r$, et que le programme linéaire en nombres entiers (Q_1) est résolu autant de fois qu'il y a de solutions non dominées. Cependant, la dimension de (Q_1) ne varie pas ; le nombre de variables et de contraintes est le même durant tout le processus de résolution. Cette méthode dépend donc, du nombre de critères et de celui des solutions non dominées du problème (MP).

2.3.7 Méthode en deux phases [111]

La méthode deux-phases élaborée par Ulungu et Teghem pour la résolution d'un problème d'affectation bi-objectif, est inspirée des travaux de **[57]** . Elle se décompose en deux étapes : la première consiste à trouver toutes les solutions supportées du front Pareto en aggrégeant les critères, puis la deuxième phase cherche les solutions non supportées entre toute paire de solutions supportées adjacentes. Cette méthode travaille essentiellement dans l'espace des critères.

Première phase

L'objectif de la première phase est d'obtenir l'ensemble des solutions supportées. Ces solutions ont l'avantage d'être relativement faciles à trouver puisqu'elles optimisent une certaine combinaison linéaire des objectifs.

Ainsi, durant la première phase de la méthode, les deux solutions extrêmes (solutions optimisant chacune un des deux objectifs) sont recherchées. Puis, de façon récursive, dès que deux solutions supportées z et z' sont trouvées, la méthode recherche d'éventuelles autres solutions supportées entre z et $z\prime$, à l'aide de combinaisons linéaires bien choisies des objectifs. A la fin de la première phase l'ensemble des solutions supportées est donc trouvé.

Deuxième phase

La deuxième phase consiste alors en la recherche des solutions non supportées appartenant au front Pareto. Ces solutions ne peuvent être obtenues par combinaisons d'objectifs. Ulungu et Teghem proposent alors d'utiliser les solutions supportées trouvées pour réduire l'espace de recherche en argumentant que les solutions Pareto non supportées restantes sont forcément dans les triangles rectangles basés sur deux solutions supportées consécutives. Ainsi, une recherche est exécutée entre chaque couple de solutions supportées adjacentes. La méthode de recherche au sein de ces triangles dépend du problème étudié. A la fin de la deuxième phase, toutes les solutions non dominées sont trouvées en utilisant soit un ordre lexicographique, soit une méthode de "ranking".

Notons que cette méthode a été adaptée à beaucoup de problèmes d'optimisation combinatoire multi-objectif tels que, sac à dos, ordonnacement, recouvrement,...etc, et a été récemment généralisée à plus de deux objectifs par Przybylski, Gandibleux et Ehrgott [**86**].

2.4 Conclusion

Ce chapitre est consacré à l'introduction de la problématique multi-objectif en général, et à la programmation linéaire multi-objectif en nombres entiers, en particulier. Les définitions et les concepts de base sont présentés ainsi que les principaux résultats nécessaires à la résolution d'un problème $MOILP$. Nous avons décrit

l'approche graphique de résolution d'un programme linéaire multi-objectif car nous jugeons qu'elle est très instructive de par la manipulation des outils et résultats de la programmation mathématique auxquels elle fait appel. Une liste non exhaustive de méthodes exactes de résolution de $MOILP$ est donnée. Ces méthodes sont passées en revue et les avantages et les inconvénients de chacune sont mentionnées.

Il existe très peu de méthodes dédiées à la résolution de $MOILP$, ce qui a motivé notre recherche dans cette direction. Dans les deux chapitres suivants, nous présentons notre contribution dans ce domaine par le développement de deux méthodes, l'une dans l'espace des decisions et l'autre dans l'espace des critères.

Chapitre 3

Génération des solution entières efficaces pour le problème multi-objectif linéaire

3.1 Introduction

Dans une première méthode dédiée au problème $MOILP$ à la recherche de solutions efficaces [26], différents types de coupes (Gomory, Chvátal-Gomory,...) ont été testés pour la génération de solutions entières. Combien même une amélioration a été apporté dans la non redondance des solutions entières trouvées (décomposition en sous-ensembles réalisables disjoints) par Ait Mehdi [7] dans son travail de mémoire de magister, le temps nécessaire à l'exécution de la méthode reste généralement prohibitif. Pour palier à ces inconvénients, une nouvelle méthode est mise au point basée sur le principe du branch & bound couplé à une nouvelle coupe efficace [27].

3.2 Définitions et notations

Le problème à résoudre dans l'espace de décisions est :

$$(P)\begin{cases} \max z^1 = c^1 x \\ \max z^2 = c^2 x \\ \vdots \\ \max z^k = c^k x \\ x \in S \\ x \ vecteur\ entier \end{cases} \tag{3.1}$$

où $S = \{x \in \mathbb{R}^n \mid Ax \leq b,\ x \geq 0\}$ est le domaine des solutions réalisables du problème relaxé de (P), $A \in \mathbb{Z}^{m+n}$, $b \in \mathbb{Z}^m$ et $c^i \in \mathbb{R}^n$ pour tout $i = 1, \ldots, k$. On suppose que le domaine S est non vide et compact.

Soit x_l^* une solution entière du problème (P), on associe à x_l^* les paramètres suivants :

- B_l : ensemble des indices des variables de base,
- N_l : ensemble des indices des variables hors base,
- H_l : ensemble des directions possibles d'amélioration des critères,

$$H_l = \left\{j \in N_l \mid \exists i \in \{1, \ldots, k\}\,;\ \widehat{c}_j^i > 0\right\} \cup \left\{j \in N_l \mid \widehat{c}_j^i = 0\ \forall i \in \{1, \ldots, k\}\right\} \tag{3.2}$$

avec $\widehat{c}_j^i$ la j*ème* coordonnée du coût réduit du i*ème* critère,

- la relation suivante qui définie une coupe,

$$\sum_{j \in H_l} x_j \geq 1 \tag{3.3}$$

- les ensembles suivants,

$$\begin{aligned} S_0 &= S \\ S_{l+1} &= \left\{x \in S_l \mid \sum_{j \in H_l} x_j \geq 1\right\} \end{aligned} \tag{3.4}$$

$$T_{l+1} = \left\{ x \in S_l \mid \sum_{j \in N_l \setminus H_l} x_j \geq 1 \right\} \tag{3.5}$$

On définie le programme paramétrique

$$(P_l) \begin{cases} \max z = \sum_{i=1}^{k} \lambda_i c^i x \\ x \in S_l \end{cases} \tag{3.6}$$

avec $\lambda_i \geq 0$ pour tout $i = 1, \ldots, k$, dont les valeurs sont fixées à priori,

Une coupe est dite $efficace$ pour le problème (P), si son adjonction au domaine S supprime au moins une solution réalisable continue de S, mais ne supprime pas de solutions réalisables entières efficaces de S.

Notons que la définition de coupe efficace pour $MOILP$, généralise la définition d'une coupe pour un problème $PLNE$.

3.3 Principe de la méthode

L'idée repose sur le balayage du domaine des solutions réalisables S afin de détecter des solutions entières potentiellement efficaces, tout en se donnant la possibilité d'éliminer des parties du domaine S ne contenant pas de solutions entières efficaces, lorsque cela est possible, par adjonction des coupes $\sum_{j \in H_l} x_j \geq 1$. Le plus mauvais des cas correspond à $H_l = N_l$ et notre coupe est réduite à la coupe bien connue de Dantzig.

La méthode du simplexe est utilisée pour résoudre le problème (P_0) et le tableau utilisé est augmenté de k nouvelles lignes pour permettre aux critères d'être évalués en même temps que le critère z de (P_0). La méthode est basée sur le principe du branch & bound pour la recherche de solutions entières. A chaque nœud l de l'arborescence, un programme (P_l) est résolu. S'il n'est pas réalisable, le nœud l est sondé. Si la solution optimale $\overline{x}_l$ de (P_l) n'est pas entière, le principe de sépara-

tion est déclanché : on crée deux nouveaux nœuds par adjonction des contraintes additionnelles $x_j \leq \lfloor \overline{x}_{lj} \rfloor$ et $x_j \geq \lceil \overline{x}_{lj} \rceil$ respectivement, avec $\overline{x}_{lj}$ fractionnaire. Le principe de la recherche en profondeur d'abord est retenu pour l'obtention de solutions réalisables entières. Dès qu'une telle solution x_l^* est trouvée, le vecteur des critères en x_l^*, $(z^1(x_l^*),\ z^2(x_l^*), \ldots, z^k(x_l^*))$, est comparé à tous les vecteurs critères des solutions potentiellement efficaces déjà trouvées et l'ensemble Eff des solutions efficaces est mis à jour. Si l'ensemble H_l est vide indiquant qu'aucun critère ne peut être amélioré, le nœud l est sondé. Sinon, la coupe efficace $\sum_{j \in H_l} x_j \geq 1$ est rajoutée et l'optimisation reprend à la recherche d'autres solutions entières.

Algorithme 3.1 (*Ensemble Efficace Complet (EEC)*)

Etape 1. *(Initialisation)*

$S_0 := S,\ l := 0$ *et* $Eff := \emptyset$ *; (ensemble des solutions efficaes de* (P)*)*

Résoudre le programme linéaire (P_0) *au nœud* 0. *Soit* $\overline{x}_0$ *la solution optimale obtenue.*

Si $\overline{x}_0$ *n'est pas entière, aller à l'étape 2a. Sinon, aller à l'étape 2b.*

Etape 2. *(Etape générale)*

Tant qu'il existe un nœud non sondé dans l'arborescence faire :

- *Choisir le nœud* l *le plus récemment créé non encore sondé et résoudre le programme linéaire correspondant* (P_l).
- *Si* (P_l) *est non réalisable, alors le nœud* l *est sondé.*
- *Sinon, soit* $\overline{x}_l$ *une solution optimale de* (P_l).
- *Si* $\overline{x}_l$ *n'est pas entière, aller à l'étape 2a.*
- *Sinon, poser* x_l^* *la solution entière trouvée et aller à l'étape 2b.*

Etape 2a. *Soit* $\overline{x}_{lj}$ *une coordonnée fractionnaire de* $\overline{x}_l$.

Séparer le nœud l *en deux nouveaux nœuds : ajouter la contrainte* $x_j \leq \lfloor \overline{x}_{lj} \rfloor$ *au premier nœud, la contrainte* $x_j \geq \lceil \overline{x}_{lj} \rceil$ *au second nœud et aller à l'étape 2.*

Etape 2b. *Si* Cx_l^* *n'est pas dominé par* Cy *pour toute solution* $y \in Eff$, *alors* $Eff := Eff \cup \{x_l^*\}$.

S'il existe $y \in Eff$ *telle que* Cy *est dominé par* Cx_l^*, *alors* $Eff := Eff \setminus \{y\} \cup \{x_l^*\}$.

Déterminer les ensembles B_l, N_l *et* H_l.

Si $H_l = \emptyset$, alors le nœud correspondant est sondé, aller à l'étape 2.

Sinon :

- *Rajouter la contrainte $\sum_{j \in H_l} x_j \geq 1$ pour obtenir l'ensemble S_{l+1},*
- *Résoudre le programme linéaire obtenu (P_{l+1}) par la méthode duale du simplexe et poser $\overline{x}_{l+1}$ la solution optimale trouvée,*
- *Si $\overline{x}_{l+1}$ est entière, poser x^*_{l+1} la solution entière trouvée et aller à l'étape 2b,*
- *Sinon, aller à l'étape 2a.*

3.4 Exemple illustratif

Considérons l'exemple suivant dans $\mathbb{R}^2$:

$$(P)\begin{cases} \max z^1 = 3x_1 - x_2 \\ \max z^2 = -x_1 + x_2 \\ 4x_1 + 3x_2 \leq 20 \\ x_1 - x_2 \leq 3 \\ x_2 \leq 4 \\ x_1 \geq 0,\ x_2 \geq 0,\ \textit{entiers} \end{cases} \tag{3.7}$$

Le programme (P_0) est résolu avec l'objectif z^1 sans contrainte d'intégrité des variables. La solution optimale $(29/7, 8/7)$ est obtenue dans le tableau du simplexe suivant :

B	b	x_3	x_4
x_2	$8/7$	$1/7$	$-4/7$
x_1	$29/7$	$1/7$	$3/7$
x_5	$20/7$	$-1/7$	$4/7$
$-z^1$	$-79/7$	$-2/7$	$-13/7$
$-z^2$	3	0	1

Tableau 3.1

La séparation est déclanchée avec la création de deux nœuds. Le nœud 1 est sondé car en rajoutant la contrainte $x_1 \geq 5$, le problème est non réalisable. Au nœud 2, la contrainte $x_1 \leq 4$ est rajoutée au tableau 3.1 et la solution entière optimale $(4, 1)$ est obtenue dans le tableau 3.2 suivant, après avoir utiliser le dual du simplexe :

B	b	x_4	x_6
x_2	1	-1	1
x_1	4	0	1
x_5	3	1	-1
x_3	1	3	-7
$-z^1$	-11	-1	-2
$-z^2$	3	1	0

Tableau 3.2

$Eff := \{(4, 1)\}$, $N_2 = \{4, 6\}$ et $H_2 = \{4\}$.
La coupe efficace $x_4 \geq 1$ est alors rajoutée et la solution optimale $(26/7, 12/7)$ est donnée dans le tableau 3.3 :

B	b	x_3	x_7
x_2	12/7	1/7	$-4/7$
x_1	26/7	1/7	3/7
x_5	16/7	$-1/7$	4/7
x_6	2/7	$-1/7$	$-3/7$
x_4	1	0	-1
$-z^1$	$-66/7$	$-2/7$	$-13/7$
$-z^2$	2	0	1

Tableau 3.3

Séparation en deux nouveaux nœuds car la solution n'est pas entière. La contrainte $x_2 \leq 1$ est rajoutée au tableau 3.3 au nœud 3 et le nœud 4 est créé après adjonction de la contrainte $x_2 \geq 2$ au tableau 3.3. La solution optimale $(3, 1)$ est obtenue au nœud 3 :

B	b	x_7	x_8
x_2	1	0	7
x_1	3	1	7
x_5	3	0	-1
x_6	1	-1	-1
x_4	1	-1	0
x_3	5	-4	-7
Z^1	-8	-3	-2
Z^2	2	1	0

Tableau 3.4

$Eff := \{(4, 1), (3, 1)\}$, $N_3 = \{7, 8\}$ and $H_3 = \{7\}$,
On rajoute la coupe efficace $x_7 \geq 1$ et la solution entière optimale $(2, 1)$ est obtenue dans le tableau 3.5 :

B	b	x_8	x_9
x_2	1	7	0
x_1	2	7	1
x_5	3	-1	0
x_6	2	-1	-1
x_4	2	0	-1
x_3	9	-7	-4
x_7	1	0	-1
Z^1	-5	-2	-3
Z^2	1	0	1

Tableau 3.5

$Eff := \{(4,1),(3,1),(2,1)\}$, $N_4 = \{8,9\}$ et $H_4 = \{9\}$,

On rajoute la coupe efficace $x_9 \geq 1, \ldots$etc.

L'exploration en profodeur à partir du nœud 3 donne le dernier tableau 3.6 suivan :

B	b	x_8	x_{12}
x_2	1	7	0
x_1	-1	7	1
x_5	3	-1	0
x_6	5	-1	-1
x_4	5	0	-1
x_3	21	-7	-4
x_7	4	0	-1
x_9	3	0	-1
x_{10}	2	0	-1
x_{11}	1	0	-1
$-z^1$	4	-2	-3
$-z^2$	-2	0	1

Tableau 3.6

et le nœud correspondant est sondé car le dual n'est pas réalisable.

$Eff := \{(4,1),(3,1),(2,1)\}$.

Au nœud 4, la contrainte $x_2 \geq 2$ est rajoutée au tableau 3.3 et la solution optimale, non entière, $(7/2, 2)$ est obtenue. Les nœuds 5 et 6 sont créés. Le nœud 5 est sondé compte tenu de la contrainte $x_1 \geq 4$.

Au nœud 6, la contrainte $x_1 \leq 3$ est rajoutée et la solution entière optimale $(3,2)$ est obtenue.

$Eff := \{(4,1),(3,1),(3,2)\}$, $N_6 = \{8,9\}$ et $H_6 = N_6$,

La contrainte $x_8 \geq 1$ est rajoutée.

En procédent de cette manière, dans le tableau 3.7 on obtient la solution entière optimale $(2,4)$ au dernier nœud 8 :

B	b	x_{11}	x_{12}
x_2	4	0	-1
x_1	2	1	0
x_5	0	0	1
x_6	2	-1	0
x_4	5	-1	-1
x_7	4	-1	-1
x_9	1	-1	0
x_8	2	0	-1
x_3	0	-4	3
x_{10}	1	0	-1
$-z^1$	-2	-3	-1
$-z^2$	-2	1	1

Tableau 3.7

$Eff := \{(4,1),(3,1),(3,2),(2,4)\}$, $N_9 = \{11,12\}$ et $H_9 = N_9$,

On rajoute la contrainte $x_{11} + x_{12} \geq 1$ et on obtient le dernier tableau 3.8 :

B	b	x_3	x_{13}
x_2	5	0	-1
x_1	5/4	1/4	3/4
x_5	-1	0	1
x_6	11/4	$-1/4$	$-3/4$
x_4	27/4	$-1/4$	$-7/4$
x_7	23/4	$-1/4$	$-7/4$
x_9	7/4	$-1/4$	$-3/4$
x_8	3	0	-1
x_{11}	3/4	$-1/4$	$-3/4$
x_{10}	2	0	-1
x_{12}	1	0	-1
Z^1	5/4	$-3/4$	$-13/4$
Z^2	$-15/4$	1/4	7/4

Tableau 3.8

Le dual n'est pas réalisable, donc le nœud 8 est sondé.

Comme tous les nœuds de l'arborescence sont sondé alors, l'algorithme s'arrête et l'ensemble efficace complet est trouvé :

$Eff := \{(4,1),(3,1),(3,2)(2,2),(2,3),(2,4),(1,4),(0,4)\}$.

3.5 Principaux résultats

Dans cette section, les justifications des étapes décrites dans la méthode ci-dessu sont établies. Les résultats suivants prouvent qu'à chaque étape l de la méthode, aucune solution efficace entière se trouvant dans l'ensemble S_l ne peut être supprimée quand nous considérons l'ensemble $S_{l+1} \subset S_l$. Considérons x_l^* la solution entière trouvée à l'étape l et notons $F' = \{x \in F \mid x \text{ est entier et } x \neq x_l^*\}$.

Lemme 3.2 $S'_l = S'_{l+1} \cup T'_{l+1}$

Preuve. Soit $x \in S'_l$, alors x est dans le domaine fermé généré par la coupe de Dantzig $\sum_{j \in N_l} x_j \geq 1$. Comme les ensembles H_l et $N_l \backslash H_l$ définissent une partition de l'ensemble N_l, la coupe de Dantzig peut alors s'écrire comme suit : $\sum_{j \in H_l} x_j + \sum_{j \in N_l \backslash H_l} x_j \geq 1$.
Si la solution x vérifie l'inégalité $\sum_{j \in H_l} x_j \geq 1$, alors $x \in S_{l+1}$. Sinon, x vérifie l'inégalité $\sum_{j \in N_l \backslash H_l} x_j \geq 1$ car x est une solution entière, et ainsi $x \in T'_{l+1}$. Par conséquent, $x \in S'_{l+1} \cup T'_{l+1}$ et donc, $S'_l \subset S'_{l+1} \cup T'_{l+1}$.
D'autre part, il est clair que $S'_{l+1} \cup T'_{l+1} \subseteq S'_l$ et nous pouvons donc conclure que $S'_l = S'_{l+1} \cup T'_{l+1}$. □

Il faut noter que les ensembles S'_{l+1} et T'_{l+1} ne sont pas nécessairement disjoints. Cependant, le théorème suivant montre que toutes les solutions efficaces de (P_l) sont dans le domaine S'_{l+1}.

Théorème 3.3 *Soit $x \in S'_l$ une solution efficace, alors $x \in S'_{l+1}$.*

Preuve. Soit $x \in S'_l$ et supposons que $x \notin S'_{l+1}$, alors $x \in T'_{l+1}$ d'aprés le lemme précédent. Ainsi les coordonnées x_j de x vérifient l'inégalité suivante : $\sum_{j \in N_l \backslash H_l} x_j \geq 1$
Ceci est équivalent à la condition suivante : $x_j = 0$ pour tout $j \in H_l$ et il existe $j \in N_l \backslash H_l$ tel que $x_j \geq 1$.
En utilisant le tableau du simplexe en x^*_l, toute ligne correspondant à un critère i, $i \in \{1, ..., k\}$, s'écrit alors :

$$\begin{aligned} c^i x &= c^i x^*_l + \sum_{j \in N_l} \widehat{c}^i_j x_j, \\ &= c^i x^*_l + \sum_{j \in H_l} \widehat{c}^i_j x_j + \sum_{j \in N_l \backslash H_l} \widehat{c}^i_j x_j, \\ &= c^i x^*_l + \sum_{j \in N_l \backslash H_l} \widehat{c}^i_j x_j \end{aligned}$$

Ainsi, pour tout critère $i \in \{1, ..., k\}$, $c^i x \leq c^i x^*_l$ avec au moins une inégalité stricte car s'il existe $j \in N_l \backslash H_l$ tel que $\widehat{c}^i_j = 0$ pour tout critère i, l'indice j serait dans H_l et on obtient tout de suite un contradiction. On conclu que x n'est pas une solution

efficace car son vecteur critère est dominé par celui de x_l^* et par conséquent, toutes les solutions efficaces du domaine S'_l se trouvent dans le domaine S'_{l+1}. □

Corollaire 3.4 *La contrainte* $\sum_{j \in H_l} x_j \geq 1$ *définie une coupe efficace pour le programme* (P_l) *en rajoutant la contrainte d'intégrité sur les variables.*

Preuve. Il est clair que cette contrainte est valide d'après le théorème précédent, puisque toutes les solutions entières efficaces se trouvant dans le domaine courant S_l vérifient la contrainte. De plus, la solution entière x_l^* étant une solution de base optimale de (P_l), ou bien d'un programme linéaire obtenu à partir de (P_l) par adjonction de contraintes de séparation de la méthode branch & bound, ses variables hors base sont nulles, ce qui signifie que $\sum_{j \in H_l} x_{lj}^* = 0$. Nous pouvons donc conclure que la contrainte $\sum_{j \in H_l} x_j \geq 1$ est une coupe efficace. □

Théorème 3.5 *L'algorithme EEC génère toutes les solutions efficaces de (P) et converge en un nombre fini d'itérations.*

Preuve. L'ensemble des solutions réalisables S étant compact, il contient un nombre fini de solutions entières. A chaque étape l, l'algorithme détermine une solution entière x_l^* lorsqu'elle existe. En prenant en considération le lemme et le théorème précédents, on élimine au moins la solution x_l^* quand la coupe efficace est rajoutée. D'autre part, lorsque l'ensemble H_l est vide, la solution x_l^* représente un point idéal relativement au domaine courant S_l et signifiant donc, que ce dernier ne contient pas de solutions efficaces. □

3.6 Expériences numériques

L'algorithme *EEC* a été mis en œuvre sous le langage de programmation Matlab 7.0, en utilisant un PC Pentium 4, processeur 1,60 GHz , 512 Mo de RAM. Il a été testé avec m contraintes générées aléatoirement, $m \in \{5, 10\}$ et k fonctions objectifs , $k \in \{4, 10\}$. Les coefficients sont des entiers non corrélés répartis uniformément

dans l'intervalle $[20, 90]$ pour les contraintes et $[30, 100]$ pour les fonctions objectifs. Pour chaque contrainte j, la valeur du second membre b_j est fixée à la valeur maximale entre $\alpha\%$ de la somme des coefficients (partie entière), $\alpha \in \{17, 20, 25\}$, et la valeur maximale des coefficients de cette contrainte. Des problèmes avec n variables, $n \in \{20, 25, 30\}$, sont considérés. Les variables sont bornées et prennent les valeurs possibles 0, 1 et 2. Pour chaque instance (n, m, k, α), une séquence de 20 problèmes est résolu et l'ensemble complet des solutions efficaces a été généré pour chaque problème. Dans la dernière colonne de chaque tableau, CE/C indique le rapport entre le nombre de coupes efficaces rajoutées et le nombre total des coupes utilisées.

(n, m, k, α)	$Sol.\ eff$		$CPU(s)$		CE		CE/C
	Moy	Max	Moy	Max	Moy	Max	
$(20, 5, 4, 25)$	99.7	216	281.6	342.2	241.6	340	96%
$(20, 10, 4, 25)$	60.1	228	265.8	306.9	157	195	98.9%
$(20, 10, 10, 25)$	763.6	1889	311.73	378.45	147.3	182	99%
$(25, 5, 4, 20)$	76.5	152	1280.1	1609.2	665.5	807	99.8%
$(25, 10, 4, 20)$	48.8	140	1082.5	1179.4	365.3	434	99.7%
$(25, 10, 10, 20)$	1277.4	3749	1354.6	1853.5	378.7	438	98%
$(30, 5, 4, 17)$	119.9	300	2224.4	2553	905.4	1019	99.8%
$(30, 10, 4, 17)$	36.4	47	2167.7	2351.4	574.2	624	99.8%
$(20, 20, 4, 25)$	24.35	48	46.92	67.7	382.6	588	100%
$(25, 25, 4, 20)$	35.8	142	184.29	255.4	771.1	1242	100%
$(30, 30, 4, 17)$	33.2	119	587.3	870.6	963.4	2062	100%

Tableau 3.9. Résultats de l'expérimentation

Dans la première expérimentation, les problèmes considérés sont des $MOILP$ généraux et les résultats cumulés sont présentés dans la première partie du tableau 1. Évidemment, dans ce cas, les résultats montrent que le temps CPU augmente rapidement avec la taille des données, le procédé étant exact, mais le nombre de critères

ne fait pas augmenté le temps CPU de façon significative. En revanche, il convient de noter que le rapport CE/C tend vers 100%, ce qui prouve que presque toutes les coupes ajoutées sont des coupes efficaces. Cela signifie que dans la plupart des cas, l'ensemble construit H_l est différent de l'ensemble N_l des indices des variables hors base. Notons également que la méthode devient plus rapide lorsque le cardinal de l'ensemble H_l est petit, car le domaine à supprimer de l'ensemble des solutions réalisables est plus grand. C'est notamment le cas lorsque la matrice des contraintes est triangulaire. Dans la deuxième partie du tableau 1, nous avons rapporté les résultats de problèmes $MOILP$ traités avec des matrices de contraintes triangulaires. Dans ce cas, le temps CPU a diminué, mais en même temps, le nombre de coupes efficaces a augmenté. Cela prouve que les domaines non considérés contiennent de nombreuses solutions entières non efficaces que la méthode ne devra pas générer.

La méthode décrite dans [104] a également été programmée pour la recherche de l'ensemble efficace complet. Nous donnons ci-après les résultats obtenus :

n	m	$\lvert Eff\rvert$	*Algo.EFC*		*Algo.SC*	
			itèr	$CPU(s)$	*itèr*	$CPU(s)$
5	5	21.2	577	0.34	51569	813.68
10	5	25.2	1314.6	0.59	142111.6	3058.37
15	5	24.8	2720.8	1.27	187036	2830.17
10	10	20.6	1018.8	0.64	66245.6	907.89
15	10	16.8	2086	0.82	51976	291.94

Tableau 3.10. Résultats de l'étude comparative donnés on moyenne pour 4 objectifs.

où *itèr* représente le nombre d'itérations de la méthode du simplexe et $|Eff|$ désigne le cardinal de l'ensemble efficace complet en moyenne.

Comme il a été prévu pour la méthode de Sylva & Crema, la taille des problèmes $PLNE$ à résoudre est étroitement liée au nombre de critères et le temps CPU augmente très vite avec la taille des données par rapport à notre méthode. Cependant,

un sous-ensemble de solutions efficaces peut être obtenu dès que le calcul est interrompu, contrairement à notre méthode qui ne donne l'ensemble efficient qu'à la fin de l'algorithme EEC.

3.7 Conclusion

Dans cette étude, une nouvelle méthode exacte combinant le principe bien connu de branch & bound en programmation linéaire discrète avec une nouvelle coupe efficace est décrite pour générer toutes les solutions entières efficace d'un problème $MOILP$. Elle peut être considérée comme une méthode générale consacrée aux problèmes $MOILP$, et peut résoudre aussi bien des problèmes dont les variables de décision sont entières qu'en $0-1$. L'étude comparative prouve que notre méthode est plus rapide que celle proposée par Sylva et Crema dont les temps CPU augmentent rapidement d'une itération à l'autre, en raison des contraintes et variables supplémentaires. La méthode n'a été testée que sur des problèmes de taille moyenne, en raison de la difficulté du problème à résoudre. Toutefois, l'arborescence de l'algorithme proposé peut être parallélisée afin de permettre la résolution des problèmes de grande taille. D'autre part, concernant l'évaluation au niveau d'un nœud de l'arborescence, nous avons pu constater qu'il peut arriver que l'ensemble H_l ne soit pas vide et pourtant le domaine courant ne contient plus de solutions efficaces. Afin d'élaguer plus tôt des branches inutiles de l'arborescence et accélérer la convergence de l'algorithme EEC, d'autres voies de recherche de tests plus performants doivent être explorées.

Chapitre 4

Génération des solutions entières non dominées du problème multi-objectif linéaire

4.1 Introduction

Au lieu de résoudre $MOILP$ dans l'espace des décisions pour trouver l'ensemble efficace complet, on peut s'intéresser à la résolution de $MOILP$ dans l'espace des objectifs ou critères pour la recherche des solutions non dominées. L'avantage est qu'à chaque solution non dominée, on associe une seule solution efficace, par contre plusieurs solutions efficaces peuvent donner lieu à une même solution non dominée.

La méthode [3] est décrite pour résoudre $MOILP$ dans l'espace des critères. Le principe de cette méthode est aussi basé sur le branch & bound couplé à des coupes efficaces dans l'espace des critères ; c'est-à-dire des contraintes faisant intervenir les critères.

4.2 Concepts de base

On considère le programme multi-objectif général suivant :

$$\begin{cases} \max(z^i = f_i(x))\ i = 1, \ldots, k \\ x \in X \end{cases} \tag{4.1}$$

où $X \subseteq \mathbb{R}^n$ est l'ensemble des solutions réalisables et $f_i,\ i = 1, ..., k$, sont des fonctions réelles.

Le problème [**?**] est appelé programme linéaire multi-objectif $(MOLP)$ si :

$f_i(x) = c^i x,\ \forall i = 1, ..., k$, et $X = \{x \in \mathbb{R}^n \mid Ax = b, x \geq 0\}$, avec la $k \times n-$matrice $C = (c^i_j)\ i = 1, \ldots, k;\ j = 1, \ldots, n$, la $m \times n-$matrice A et le $m-$vecteur b réels.

De plus, si les variables sont entières, on obtient un programme linéaire multi-objectif à variables entières $(MOILP)$:

$$(P) \begin{cases} \max Cx \\ x \in X \\ x\ \textit{vecteur entier} \end{cases} \tag{4.2}$$

et nous considèrons que toutes les composantes de la matrice A ainsi que le vecteur b sont entiers, et l'ensemble des solutions réalisables dans l'espace des décisions X est non vide et compact.

L'ensemble des vecteurs critères réalisables Y du problème $MOLP$ est défini comme suit :

$$Y = \left\{z \in \mathbb{R}^k \mid z^i = c^i x,\ x \in X\right\} = f(X) \tag{4.3}$$

Comme X, l'ensemble Y est aussi non vide et compact.

Pour générer en même temps, aussi bien les solutions supportées que celles non supportées pour le problème (P), une méthode basée sur le branch & bound et utilisant des coupes efficaces dans l'espace des critères est développée.

Soit $x^l \in X$ une solution entière du problème (P), on définie les ensembles suivants en x^l et relativement à chaque critère $i,\ i = 1, \ldots, k$, :

$$H^i_l = \left\{j \in N_l \mid \widehat{c}^i_j > 0\right\} \tag{4.4}$$

où N_l est l'ensemble des indices des variables hors-base et $\widehat{c}_j^i$ est la $jème$ composante du coût réduit du vecteur critère i.

$$K_l = \left\{ i \in \{1, ..., k\} \mid H_l^i \neq \emptyset \right\} \tag{4.5}$$

indique l'ensemble des critères pouvant être améliorés à partir de x^l.

Une contrainte est une coupe efficace pour le problème (P) dans l'espace des critères, si son adjonction n'élimine pas de solutions entières non dominées de l'ensemble Y.

A partir du tableau du simplexe au point x^l, nous avons la relation suivante :

$$c^i x = c^i x^l + \sum_{j \in N_l} \widehat{c}_j^i x_j, \ \forall i \in \{1, ..., k\} \tag{4.6}$$

et sous l'hypothèse que la matrice C est à coefficients entiers, on en déduit la contrainte suivante pour chaque critère $i \in K_l$:

$$c^i x \geq c^i x^l + \sum_{j \in N_l \setminus H_l} \left\lfloor \widehat{c}_j^i \right\rfloor x_j + \max \left\{ 1, \ \left\lfloor \widehat{c}_{j_0}^i \right\rfloor \right\} \tag{4.7}$$

où $\widehat{c}_{j_0}^i = \min_{j \in H_i} \left\{ \widehat{c}_j^i \right\}$ pour $i \in K_l$. Nous montrons que cette contrainte définie une coupe efficace dans l'espace des critères pour le problème (P). Pour ce faire, nous définissons les ensembles suivants :

$$\begin{aligned} X_0 &= X \\ X_{l+1} &= \bigcup_{i \in K_l} \{x \in X_l \mid x \text{ vérifie } 4.7\} \end{aligned} \tag{4.8}$$

Pour détecter les solutions non dominées dans l'ensemble Y, il suffit de trouver les solutions entières situées dans l'ensemble des solutions réalisables X et utiliser la relation entre les ensembles X et Y .

Fondamentalement, l'idée est d'être capable de générer l'ensemble minimum efficace

dans l'ensemble X. En effet, deux solutions différentes entières efficaces peuvent produire le même vecteur non dominé. Pour ce faire, nous définissons un programme linéaire (P_l), à chaque étape l correspondant à la recherche d'une solution entière nouvelle, comme suit :

$$(P_l) \begin{cases} \max \sum_{i=1}^{k} c^i x \\ x \in X_l \end{cases} \tag{4.9}$$

4.3 Description de la méthode

L'approche proposée génère toutes les solutions entières non dominées sans passer en revu toutes les solutions possibles de Y. C'est une méthode branch and bound qui fait appel aux coupes efficaces de type 4.7 pour passer d'un vecteur entier à un autre dans Y. On fait appel à la méthode du simplexe pour résoudre le programme linéaire (P_l) à l'étape l de la méthode. Les vecteurs critères sont alors adjoints au tableau du simplexe et évoluent de façon dynamique dans ce dernier de la même façon que le critère z de (P_l). Si la solution optimale obtenue n'est pas entière, ce qui correspond à un nœud de type 1 de l'arbre, seul un processus de branchement est effectué pour détecter une solution entière. Rappelons que contrairement aux autres méthodes existantes, la méthode n'a pas besoin de rechercher une solution entière optimale, mais seulement une solution entière voisine à la solution optimale courante qui n'est pas entière. Quand une solution entière est obtenue, le nœud est de type 2, le vecteur critère correspondant est comparé à ceux déjà trouvés et l'ensemble des solutions potentiellement non dominées est mis à jour. En vue de rechercher une autre solution entière, les directions d'amélioration des critères sont utilisés pour construire des coupes efficaces permettant d'éviter l'exploration de domaines ne contenant que des solutions entières dominées.

Un nœud l de l'arborescence est sondé, si le programme correspondant (P_l) n'est pas réalisable ou bien $K_l = \varnothing$.
Sinon, soit x^l la première solution entière obtenue après résolution du programme (P_l) et l'application du processus de branchement si nécessaire. À cette étape et

pour chaque critère i, $i \in K_l$, un nouveau programme linéaire (P_k), $k > l$, est résolu après l'ajout de la coupe efficace 4.7. Cette dernière coupe assure une amélioration pour le critère i ce qui permet de pouvoir repartir à la recherche d'autres solutions non dominées.

Algorithme 4.1 (Solutions Non Dominées (SND))

Etape 1. *(Initialisation)*

$X_0 := X$, $l := 0$ et $SND := \emptyset$ (ensemble des solutions non dominées pour le problème (P)),

Résoudre le programme linéaire (P_0) au nœud 0 de l'arborescence et poser x la solution optimale trouvée,

Si x n'est pas une solution entière, aller à l'étape 2-1. Sinon, à l'étape 2-2.

Etape 2. *(étape générale)*

Tant que l'arborescence contient des nœuds non sondé, faire :

- *Choisir le nœud l le plus récemment créé qui n'est pas encore sondé et résoudre le programme linéaire correspondant (P_l),*
- *Si le programme (P_l) n'admet pas de solutions réalisables, alors le nœud l est sondé,*
- *Sinon, soit x la solution optimale trouvée,*
- *Si x n'est pas entière, aller à l'étape 2-1,*
- *Sinon, aller à l'étape 2-2,*

Etape 2-1. (Séparation)

Choisir une coordonée j de x dont la valeur α_j n'est pas entière et séparer le nœud l en deux nouveaux nœuds : ajouter la contrainte $x_j \leq \lfloor \alpha_j \rfloor$ au premier nœud, la contrainte $x_j \geq \lfloor \alpha_j \rfloor + 1$ au second nœud et aller à l'étape 2,

Etape 2-2. (Evaluation)

Soit x^l une solution entière.

Si Cx^l n'est pas dominée par Z, pour toute solution $Z \in SND$, alors $SND := SND \cup \{Cx^l\}$,

S'il existe une solution $Z \in SND$ telle que Z est dominée par Cx_l, alors $SND := SND \backslash \{Z\} \cup \{Cx^l\}$,

Déterminer l'ensemble N_l des indices des variables hors-base,

Pour chaque vecteur critère i, déterminer $H_l^i = \left\{ j \in N_l \mid \widehat{c}_j^i > 0 \right\}$,
Déduire l'ensemble $K_l = \{ i \in \{1, ..., k\} \mid H_l^i \neq \emptyset \}$,
Si $K_l = \varnothing$, *alors le nœud* l *est sondé, aller à l'étape 2,*
Sinon, pour chaque indice $i \in K_l$, *rajouter la contrainte 4.7 pour obtenir* $|K_l|$ *nouveaux programmes linéaires* $(P_k), k > l$, *aller à l'étape 2*

4.4 Résultats fondamentaux

Dans cette section, les justifications des étapes décrites dans la méthode ci-dessus sont établies. On considère un nœud de type 2 dans l'arborescence et on note x^l la solution entière courante, nous devons alors chercher une autre solution entière lorsqu'elle existe. Pour ce faire, on rajoute une coupe pour supprimer la solution entière $z(x^l)$ de l'ensemble Y, sans supprimer des solutions entières contenues dans l'ensemble des solutions réalisables courant X_l correspondant aux vecteurs non dominées dans l'ensemble Y. Le résultat suivant prouve qu'en ajoutant la contrainte 4.7 à un nœud de type 2, aucune solution non dominée n'est supprimée dans l'ensemble Y.

Théorème 4.2 *Soit x une solution entière de l'ensemble X_l telle que $Cx \neq Cx^l$. Si le vecteur Cx est non dominé, alors x se trouve dans l'ensemble X_{l+1}.*

Preuve. Soit x une solution entière se trouvant dans l'ensemble X_l, alors à partir du tableau optimal courant on peut écrire ce qui suit :

$$c^i x = c^i x^l + \sum_{j \in N_l} \widehat{c}_j^i x_j, \ \forall i \in \{1, ..., k\}$$

1) Si $i \in \{1, ..., k\} \backslash K_l$, alors $\widehat{c}_j^i \leq 0 \ \forall j \in N_l$. Ainsi,

$$\sum_{j \in N_l} \widehat{c}_j^i x_j \leq 0$$

et par suite

$$c^i x \leq c^i x^l.$$

2) Si $i \in K_l$, alors :

$$c^i x = c^i x^l + \sum_{j \in N_l \setminus H_l^i} \widehat{c}_j^i x_j + \sum_{j \in H_l^i} \widehat{c}_j^i x_j$$

Supposons que $x^l \notin X_{l+1}$ alors :

$$x^l \notin \left\{ x \in X_l \mid c^i x \geq c^i x^l + \sum_{j \in N_l \setminus H_l^i} \left\lfloor \widehat{c}_j^i \right\rfloor x_j + \max\left\{1,\ \left\lfloor \widehat{c}_{j_0}^i \right\rfloor\right\} \right\} \ \forall i \in K_l,$$

Comme $x \in X_l$, alors l'inégalité suivante est vérifiée :

$$c^i x < c^i x^l + \sum_{j \in N_l \setminus H_l^i} \left\lfloor \widehat{c}_j^i \right\rfloor x_j + \max\left\{1,\ \left\lfloor \widehat{c}_{j_0}^i \right\rfloor\right\} \ \forall i \in K_l \tag{4.10}$$

D'autre part,

$$c^i x \geq c^i x^l + \sum_{j \in N_l \setminus H_l^i} \left\lfloor \widehat{c}_j^i \right\rfloor x_j + \sum_{j \in H_l^i} \left\lfloor \widehat{c}_j^i \right\rfloor x_j \forall i \in K_l$$

En utilisant les deux dernières inégalités, on obtient :

$$\sum_{j \in H_l^i} \left\lfloor \widehat{c}_j^i \right\rfloor x_j < \max\left\{1,\ \left\lfloor \widehat{c}_{j_0}^i \right\rfloor\right\} \quad \forall i \in K_l$$

1er cas :

Supposons que pour un critère $i \in K_l$, on a :

$$\max\left\{1,\ \left\lfloor \widehat{c}_{j_0}^i \right\rfloor\right\} = \left\lfloor \widehat{c}_{j_0}^i \right\rfloor$$

alors :

$$\sum_{j \in H_l^i} \frac{\left\lfloor \widehat{c}_j^i \right\rfloor}{\left\lfloor \widehat{c}_{j_0}^i \right\rfloor} x_j < 1$$

et compte tenu de la définition de $\left\lfloor \widehat{c}_{j_0}^i \right\rfloor$, on a $\frac{\left\lfloor \widehat{c}_j^i \right\rfloor}{\left\lfloor \widehat{c}_{j_0}^i \right\rfloor} \geq 1,\ \forall j \in H_l^i$, ce qui montre que $x_j = 0\ \forall j \in H_l^i$.

Par conséquent,

$$c^i x = c^i x^l + \sum_{j \in N_l \setminus H_l^i} \widehat{c}_j^i x_j$$

et ainsi,

$$c^i x \leq c^i x^l.$$

2ème cas :

Supposons que pour un critère $i \in K_l$, on a : $\max\left\{1,\ \left\lfloor \widehat{c}^i_{j_0} \right\rfloor\right\} = 1$

En utilisant l'inégalité 4.10, on peut écrire : $c^i x < c^i x^l + \sum_{j \in N_l \setminus H^i_l} \left\lfloor \widehat{c}^i_j \right\rfloor x_j + 1,$

ce qui implique que : $c^i x \leq c^i x^l$.

On peut donc conclure que $Cx \leq Cx^l$

En tenant compte du fait que $Cx \neq Cx^l$, on peut dire que le vecteur Cx est dominé par le vecteur Cx^l. □

Corollaire 4.3 *La contrainte 4.7 est une coupe efficace dans le domaine X_l pour tout l.*

Preuve. Il est clair que la contrainte 4.7 est une contrainte valide puisque toutes les solutions entières efficaces dans le domaine X_l pour tout l, satisfont cette contrainte d'après la proposition précédente. D'autre part, en remplaçant x par x^l dans 4.7, on abouti à une absurdité. Donc, la solution entière courante x^l ne vérifie pas la contrainte 4.7. En conclusion, nous pouvons affirmé que la contrainte 4.7 est une coupe efficace. □

Remarque 4.4 *Relativement à l'ensemble K_l, chaque fois qu'une solution entière est obtenue, ce qui correspond à un nœud de type 2 dans l'arborescence, nous devons considérer autant de problèmes de programmation linéaire qu'il y a de critères à améliorer. En résolvant chacun d'eux après avoir ajouté seulement la coupe efficace correspondante 4.7, on n'exclut pas qu'un vecteur non-dominé soit trouvé plusieurs fois. Pour éviter la génération des vecteurs déjà trouvés, l'ensemble X_{l+1} est remplacé par d'autres ensembles construits de façon à éviter les redondances comme indiqué dans le théorème suivant.*

Théorème 4.5 *Soit $K_l = \{i_1, i_2, ..., i_t\} \subseteq \{1, 2, ..., k\}$. On définie les ensembles suivants :*

$$A_s = \left\{ x \in X_l \mid c^{i_s} x \geq c^{i_s} x^l + \sum_{j \in N_l \setminus H_l^{i_s}} \left\lfloor \widehat{c}_j^{i_s} \right\rfloor x_j + \max\left\{1,\ \left\lfloor \widehat{c}_{j_0}^{i_s} \right\rfloor\right\} \ \textit{pour } s = \overline{1,t} \right\}$$

B_1=A_1,

$$B_s = \left\{ x \in X_l | \left\{ \begin{array}{l} c^{i_s} x \geq c^{i_s} x^l + \sum\limits_{j \in N_l \setminus H_l^{i_s}} \left\lfloor \widehat{c}_j^{i_s} \right\rfloor x_j + \max\left\{1,\ \left\lfloor \widehat{c}_{j_0}^{i_s} \right\rfloor\right\} \\ c^{i_p} x \leq c^{i_p} x^l + \sum\limits_{j \in N_l \setminus H_l^{i_p}} \left\lfloor \widehat{c}_j^{i_p} \right\rfloor x_j + \max\left\{1,\ \left\lfloor \widehat{c}_{j_0}^{i_p} \right\rfloor\right\} - 1,\ p = \overline{1, s-1} \end{array} \right. \right\}$$

On a alors l'égalité suivante :

$$\bigcup_{s=1}^{t} A_s = \bigcup_{s=1}^{t} B_s \tag{4.11}$$

De plus,

$$B_r \cap B_s = \emptyset,\ \forall r,\ s \in \{1, ..., t\},\ r \neq s. \tag{4.12}$$

Preuve. Remarquons déjà que

$$B_s = A_s \cap \left(\overline{A_1} \cap \overline{A_2} \cap \ldots \overline{A_{s-1}}\right) pour s = \overline{2,t}. \tag{4.13}$$

On montre d'abord par récurrence que la propriété $P(t)$ suivante est vraie pour tout $t \geq 1$:

$$P(t) : \bigcup_{s=1}^{t} A_s = \bigcup_{s=1}^{t} B_s \tag{4.14}$$

Par hypothèse $A_1 = B_1$. De plus $B_2 = A_2 \cap \overline{A_1}$, alors $B_1 \cup B_2 = A_1 \cup (A_2 \cap \overline{A_1}) = A_1 \cup A_2$.

Ainsi, la propriété $P(2)$ est vraie.

Supposons que la propriété soit vraie jusqu'à l'ordre t et montrons qu'elle reste vraie à l'ordre $t+1$.

On a :

$$\bigcup_{s=1}^{t+1} B_s = \bigcup_{s=1}^{t} B_s \cup B_{t+1} = \bigcup_{s=1}^{t} A_s \cup \left[A_{t+1} \bigcap_{s=1}^{t} \overline{A_s} \right] \tag{4.15}$$

par hypothèse de récurrence.

Donc,

$$\bigcup_{s=1}^{t+1} B_s = (\bigcup_{s=1}^{t+1} A_s) \cap \left[(\bigcup_{s=1}^{t} A_s) \cup \overline{(\bigcup_{s=1}^{t} A_s)} \right] = \bigcup_{s=1}^{t+1} A_s, \tag{4.16}$$

ce qui prouve que la propriété $P(t+1)$ est vraie.

D'après le théorème de récurrence, la propriété $P(t)$ est vraie $\forall t \geq 1$

Montrons maintenant que $B_r \cap B_s = \emptyset,\ \forall r,\ s \in \{1, ..., t\},\ r \neq s$.

On a :

$$B_r \cap B_s = \left[A_r \cap \left(\overline{A_1} \cap \overline{A_2} \cap \ldots \cap \overline{A_{r-1}}\right)\right] \cap \left[A_s \cap \left(\overline{A_1} \cap \overline{A_2} \cap \ldots \cap \overline{A_{s-1}}\right)\right] \tag{4.17}$$

Si $r < s$, alors $r \leq s-1$ et ona :

$$B_r \cap B_s = A_r \cap A_s \cap \overline{A_1} \cap \overline{A_2} \cap \ldots \cap \overline{A_{r-1}} \cap \overline{A_r} \cap \ldots \cap \overline{A_{s-1}} \tag{4.18}$$

comme la séquence $1, 2, \ldots r-1$ est incluse dans la séquence $1, 2, \ldots s-1$,alors l'ensembe $\overline{A_r}$ figure nécessairement parmi les ensembles $\overline{A_1},\ \overline{A_2}, \ldots, \overline{A_{s-1}}$ et dans l'intersection de tous les ensembles décrivant $B_r \cap B_s$, on aura l'intersection $A_r \cap \overline{A_r}$ qui est vide. Ainsi, $B_r \cap B_s = \emptyset$.

Comme r et s jouent un rôle symétrique, le résultat reste vrai pour le cas $s < r$.

En conclusion, $B_r \cap B_s = \emptyset,\ \forall r,\ s \in \{1, ..., t\},\ r \neq s$, ce qui termine la deuxième partie du théorème. □

On peut voir clairement que l'ensemble X_{l+1} est simplement la réunion des ensembles A_s précédemment définis et peut être remplacé par la réunion des ensembles B_s.

Proposition 4.6 *L'algorithme SND génère toutes les solutions non dominées du problème (P) en un nombre fini d'itérations.*

Preuve. L'ensemble des solutions réalisables de (P) étant contenu dans X qui est compact, il contient un nombre fini de solutions entières. L'algorithme détermine un point entier x^l à l'étape l, $l \geq 1$, lorsqu'il existe. Lorsque l'ensemble X_{l+1} est

considéré, on a déjà éliminé le point x^l de l'ensemble X_l et, éventuellement d'autres points entiers dont les vecteurs critères sont dominés, sans supprimer des points entiers dont les vecteurs critères sont non dominés, d'après la proposition et son corollaire énoncés plus haut. Si x^l n'est pas supprimé, c'est que l'ensemble K_l est vide auquel cas le nœud correspondant est sondé. □

4.5 Exemple numérique

On considère le problème $MOILP$ suivant :

$$(P)\begin{cases} \max f_1(x) = x_1 + 3x_2 \\ \max f_2(x) = -x_2 \\ 2x_1 + 3x_2 \leq 5 \\ 2x_1 + x_2 \leq 4 \\ x_1 \geq 0,\ x_2 \geq 0 \ \textit{entiers} \end{cases}$$

L'ensemble des solutions réalisables du problème relaxé est :

$$X = \left\{ x \in \mathbb{R}^2 \mid 2x_1 + 3x_2 \leq 5,\ 2x_1 + x_2 \leq 4,\ x_1 \geq 0,\ x_2 \geq 0 \right\}$$

Le programme linéaire (P_0) est :

$$(P_0)\begin{cases} \max f(x) = f_1(x) + f_2(x) = x_1 + 2x_2 \\ (x_1, x_2) \in X \end{cases}$$

La résolution de (P_0) donne :

	b	x_1	x_2	x_3	x_4
x_2	$\frac{5}{3}$	$\frac{2}{3}$	1	$\frac{1}{3}$	0
x_4	$\frac{7}{3}$	$\frac{4}{3}$	0	$\frac{-1}{3}$	1
$-f$	$\frac{-10}{3}$	$-\frac{1}{3}$	0	$-\frac{2}{3}$	0
f_1	-5	-1	0	-1	0
f_2	$\frac{5}{3}$	$\frac{2}{3}$	0	$\frac{1}{3}$	0

Tableau 4.1

La solution optimale est non entière $(0, \frac{5}{3})$. La séparation se fait donc par rapport à la variable x_2 et on obtient les deux sous problèmes :

$$(P_1)\begin{cases}(P_0)\\ x_2 \leq 1\end{cases} \qquad (P_2)\begin{cases}(P_0)\\ x_2 \geq 2\end{cases}$$

Le problème (P_2) étant non réalisable, le nœud correspondant est sondé.

La résolution de (P_1) donne :

	b	x_1	x_2	x_3	x_4	x_5
x_2	1	0	1	0	0	1
x_4	1	0	0	-1	1	2
x_1	1	1	0	$\frac{1}{2}$	0	$\frac{-3}{2}$
$-f$	-3	0	0	$\frac{-1}{2}$	0	$\frac{-1}{2}$
$-f_1$	-4	0	0	$\frac{-1}{2}$	0	$\frac{-3}{2}$
$-f_2$	1	0	0	0	0	1

Tableau 4.2

La solution optimale entière est $x^1 = (1, 1)$ dont le vecteur critère est :
$(f_1(x^1), f_2(x^1)) = (4, -1)$.

Ainsi, $SND := \{(4, -1)\}$, $N_1 = \{3, 5\}$, $H_1^1 = \varnothing$, $H_1^2 = \{5\}$, $K_1 = \{2\}$,
$N_1 \backslash H_1^2 = \{3\}$ et $\min_{j \in H_1^2} \{\widehat{c}_j^2\} = \widehat{c}_5^2 = 1$.

Comme $|K_1| = 1$, alors un seul problème est créé, (P_3), en rajoutant la coupe efficace :$f_2(x) \geq f_2(x^1) + \lfloor \widehat{c}_3^2 \rfloor x3+\max\{1, \lfloor \widehat{c}_5^2 \rfloor\}$, dont l'expression est donnée par :
$-x_2 \geq -1 + \lfloor 0 \rfloor x_3 + 1$

La résolution de (P_3) donne :

	b	x_1	x_2	x_3	x_4	x_5	x_6
x_2	0	0	1	0	0	0	1
x_3	1	0	0	1	1	0	-2
x_1	2	1	0	0	$\frac{1}{2}$	0	$\frac{-1}{2}$
x_5	1	0	0	0	0	1	-1
$-f$	-2	0	0	0	$\frac{-1}{2}$	0	$\frac{-3}{2}$
$-f_1$	-2	0	0	0	$\frac{-1}{2}$	0	$\frac{-5}{2}$
$-f_2$	0	0	0	0	0	0	1

Tableau 4.3

La solution optimale entière est $(2, 0)$ et son vecteur critère est aussi $(2, 0)$.

$SND := \{(4, -1), (2, 0)\}$ puisque $(2, 0)$ n'est pas dominé par $(4, -1)$

$N_3 = \{4, 6\}$, $H_3^1 = \emptyset$, $H_3^2 = \{6\}$, $K_3 = \{2\}$, $N_3 \backslash H_3^2 = \{4\}$ et $\min_{j \in H_3^2} \left\{\widehat{c}_j^2\right\} = \widehat{c}_6^2 = 1$.

Un nouveau problème (P_4) est créé à partir de (P_3) après rajout de la coupe efficace : $-x_2 \geq 0 + 0.x_4 + 1$.

	b	x_1	x_2	x_3	x_4	x_5	x_6	x_7
x_2	-1	0	1	0	0	0	0	1
x_3	3	0	0	1	1	0	0	-2
x_1	$\frac{5}{2}$	1	0	0	$\frac{1}{2}$	0	0	$\frac{-1}{2}$
x_5	2	0	0	0	0	1	0	-1
x_6	1	0	0	0	0	0	1	-1
$-f$	$\frac{-1}{2}$	0	0	0	$\frac{-1}{2}$	0	0	$\frac{-3}{2}$
$-f_1$	$\frac{1}{2}$	0	0	0	$\frac{-1}{2}$	0	0	$\frac{-5}{2}$
$-f_2$	-1	0	0	0	0	0	0	1

Tableau 4.4

Le problème (P_4) est non réalisable et par conséquent, le nœud correspondant est sondé.

Tous les nœuds de l'arborescence étant sondé, l'algorithme se termine et l'ensemble des solutions non dominées du problème (P) est $SND = \{(4, -1), (2, 0)\}$.

4.6 Expérimentation et résultats

Les deux méthodes décrites dans [104] et [83] ainsi que la méthode présentée dans la section 3, notées 1, 2 et 3 respectivement, ont été mises en œuvre dans un environnement Matlab 7.0. Pour la méthode 3, toutes les procédures ont été programmées, y compris les méthodes du simplexe et du dual du simplexe, alors que pour les deux autres, leurs programmes utilisent le programme "lp-solve" pour résoudre la séquence des programmes $PLNE$. Les trois méthodes ont été testées sur des problèmes $MOILP$ générés aléatoirement avec deux, trois et quatre objectifs. Pour exécuter les tests, nous avons utilisé un ordinateur équipé d'un processeur Intel Pentium 3,6 GHz et 1 Go de mémoire. Les coefficients des fonctions objectifs et des contraintes sont des entiers non corrélés uniformément répartis entre les valeurs -10 et 99. Pour chaque contrainte, la valeur du second membre b a été fixée à $\delta\%$ de la somme de ses coefficients où $\delta \in \{50, 33, 25, 20\}$. Pour chaque instance (n, m, k), une série de 20 problèmes est résolue et tout l'ensemble non-dominé SND a été généré pour tous ces problèmes. Le tableau suivant récapitule les résultats obtenus en moyenne.

Nous pouvons voir que la méthode 2 [83] est plus rapide que les autres méthodes pour des problèmes à deux et tois objectifs. Toutefois, pour des problèmes avec quatres objectifs générant un grand nombre de solutions non dominées, il semble que la méthode 3 [3] est plus rapide que les deux autres méthodes. A partir de ces résultats, on peut clairement noter l'influence du nombre de solutions non dominées dans la performance des méthodes de Özlen et Azizoğlu [83] et de Sylva et Crema [104] ; plus ce nombre est élevé et plus le temps d'exécution et l'espace mémoire dont

ils ont besoin augmentent.

instance			méthode 1	méthode 2	méthode 3	$\|SND\|$
n	m	k				
15	5	2	2.96	0.23	17.95	12.30
20	5	2	5.37	0.43	97.49	13
25	5	2	10.39	0.71	411.34	13.90
15	5	3	114.51	1.53	21.11	19.70
20	5	3	3264.79	5.59	107.81	26.10
25	5	3	1318.61	6.13	403.83	23.90
15	10	3	90.61	3.94	14.34	21.80
20	10	3	1446.81	7.64	62.37	23.40
25	10	3	1040.03	8.46	213.38	19.40
15	5	4	1153.86	5.59	18.25	18.10
20	5	4	×	917.65	132.40	127
25	5	4	×	1468.65	431.93	133.20
30	10	4	×	2227.20	644.38	96

Tableau 4.5 Résultats des 3 méthodes avec temps CPU en secondes.

Afin de mieux illustrer l'application du calcul, nous considérons le problème d'emballage connu sous le vocable anglais de "set packing problem ", (SPP). C'est une variante du problème de sac-à-dos multidimensionnel dans lequel les poids sont binaires et les capacités sont toutes égales à 1.

Le problème SPP peut être décrit comme suit : Etant donné un ensemble fini d'objets valués, $I = \{1, \ldots, n\}$ et une collection de sous-ensembles de I, $\{T_j\}$, $j \in J = \{1, ..., m\}$, une solution est un sous-ensemble $R \subseteq I$, tel que $|T_j \cap R| \leq 1$, $\forall j \in J$. L'objectif est de maximiser la valeur totale de la solution obtenue. Le modèle

mathématique s'écrit :.

$$(SPP)\begin{cases} \max z = \sum_{i \in I} c_i x_i \\ \sum_{i \in I} t_{ij} x_i \leq 1 \quad \forall j \in J \\ x_i \in \{0,1\} \quad \forall i \in I \end{cases}$$

où les t_{ij} sont donnés, $t_{ij} \in \{0,1\}\ \forall i \in I, \forall j \in J$.

Dans un graphe $G = (V, E)$ dont les sommets sont munis de poids, le problème est celui du stable de poids maximum. Bien qu'il existe des articles sur les approches à mettre en œuvre pour les problèmes de recouvrement et de partition, le problème SPP est relativement moins analysé [84], [32].

Selon Garey et Johnson [50], le problème est fortement NP-difficile même dans le cas mono-objectif.

Le problème multi-objectif de "set packing" est un problème appartenant la classe des problèmes d'optimisation combinatoire multi-objectif (MOCO : Multiple Objective Combinatorial Optimisation), similaire aux problèmes bien connus de recouverment et partitionnement avec des objectifs multiples, mais n'a pas reçu beaucoup d'attention que les deux derniers problèmes. Dans l'article de Delorme et *al.* [38], l'ensemble efficient du problème "set packing" bi-objectif a été calculé en utilisant une procédure dichotomique pour toutes les instances considérées dans le site [75] de MCDM. Connaissant la complexité de ce problème, il est rapporté par les auteurs que sa résolution exacte n'a pas été possible dans un temps de calcul raisonnable : les solutions ont été obtenues après plusieurs jours de calcul en utilisant un solveur $CPLEX$. Seuls les méta-heuristiques ont été conçues pour l'approximation des solutions à ce problème.

Nous avons réalisé des tests sur certaines instances de problèmes bi-objectif disponibles sur le site [75] . Seuls les cas qui requièrent jusqu'à deux heures de temps CPU en moyenne ont été rapportés dans le tableau suivant :.

instance	m	*CPU (s)*	*Itèr.simplexe*	*nœuds*	$\|SND\|$
2mis100300C	40	325.28	23454	3486	34
	50	1700.51	113479	17126	48
2mis100300E	90	0.17	110	2	5
	97	909.82	41818	3800	6
2mis100300F	130	0.75	186	6	3
	140	1225.14	35723	3094	4
2spp100300A	30	24.93	5159	798	22
	50	1890.17	170176	25546	13
2spp100300B	50	22	4102	838	15
	60	650.29	52633	8778	24
	65	1400.34	90187	13824	25
2spp100300C	20	1.13	172	22	6
	30	115.25	6395	1054	42
	40	4539.06	231414	33460	50
2spp100300E	91	0.39	90	1	4
	100	1508.18	83025	7378	2
2spp100300F	80	0.29	81	1	5
	91	7574.95	284336	26888	3

Tableau 4.6 Résultats pour le problème "set packing"

4.7 Conclusion

Une méthode exacte combinant le principe branch & bound avec des coupes efficaces est décrite pour générer toutes les solutions non dominées, sans résoudre aucun problème PLNE le long des étapes de l'algorithme SND proposé. Dans le tableau du simplexe augmentée, chaque critère est exprimé seulement avec les variables hors base, ce qui facilite l'adjonction des coupes efficaces. La méthode présentée peut

être classée comme une méthode générale pour les problèmes $MOILP$ du moment qu'aucune hypothèse n'est faite sur la matrice des contraintes et aucune restriction n'est faite sur le nombre de critères.

Contrairement aux méthodes de Sylva & Crema [104] et de Özlen & Azizoğlu [83], le nombre de critères n'influence pas la méthode décrite et les problèmes avec des centaines de solutions non dominées peuvent être résolus en un temps d'exécution relativement raisonnable. Comme pour la méthode décrite dans la section précédente, l'évaluation au niveau d'un nœud doit être plus restrictive pour éviter l'exploration de domaines qui ne contiennent pas de solutions non dominées. Notons que la version actuelle du programme de traitement des données est focalisée sur la résolution du problème $MOILP$, mais les procédures utilisées ne sont pas optimisées, c'est pour cette raison que seuls les problèmes de petite et moyenne taille ont été résolus. Le programme pourrait être optimisé en utilisant les procédures prédéfinies de Matlab Optimization Toolbox. D'autre part, l'arborescence de l'algorithme proposé peut être parallélisée afin de permettre la résolution des problèmes de grande taille.

Chapitre 5

Programmation Fractionnaire

5.1 Introduction

La programmation fractionnaire a connu un essor considérable ces trente dernières années. Elle a été largement passé en revue par beaucoup d'auteurs [80], [93] et il existe des ouvrages entièrement dédiés à ce sujet [35], [46], [63], [99]. Une bibliographie de 491 travaux présentés par Stancu-Minasian dans **[101]**, attire l'attention sur la quantité de travail qui a été fait dans le domaine ces dernières années. Cette bibliographie réalisée sur la programmation fractionnaire est la suite de cinq bibliographies précédentes de l'auteur **[100].** Schaible [93] a édité un examen complet du travail dans la programmation fractionnaire décrivant certains de ses développements principaux. Le livre de Stancu-Minasian [99] contient l'état de l'art de la théorie et la pratique de la programmation fractionnaire, permettant au lecteur de devenir rapidement au courant de ce qui a été fait dans le domaine.

L'intérêt porté à ce sujet tient à la variété de problèmes d'optimisation rencontrés en ingénierie et en économie pour lesquels on considère l'optimisation d'un rapport de deux fonctions. Les programmes linéaires ou non linéaires fractionnaires, en continu, en nombres entiers ou en variables 0-1 apparaissent dans plusieurs domaines tels que les bases de données, l'optimisation combinatoire, la programmation stochastique et

l'économie [99]. De nombreuses applications des programmes fractionnaires ont été décrites dans la littérature [45], [48], [62], [87], [94], [95].

Les problèmes mathématiques d'optimisation avec une fonction objectif qui est un rapport de deux fonctions linéaires ont beaucoup d'applications : dans les domaines des finances (planification d'une entreprise, gestion de feuille de solde bancaire), de transport maritime, de ressources en eau, de santé, et ainsi de suite. En effet, dans de telles situations, il est souvent question d'optimiser un rapport de deux fonctions telles que dette/capitaux propres, rendement/employé, coût effectif/coût standard, bénéfice/coût, inventaire/ventes, risque des actifs/capital, étudiant/coût, docteur/patient, sujet à des contraintes. En outre, si les contraintes sont linéaires, nous obtenons un problème de programmation linéaire fractionnaire .

Un des premiers programmes fractionnaires (cependant non appelés ainsi) est un modèle d'équilibre pour une économie en expansion, présenté par Von Neumann en 1937 [112]. Le modèle détermine le taux de croissance d'une économie comme maximum du plus petit de plusieurs rapports sortie-entrée. Cependant, une étude systématique de la programmation fractionnaire a commencé beaucoup plus tard. En 1962, Charnes et Cooper ont publiés leur papier classique dans lequel ils prouvent qu'un programme linéaire fractionnaire peut être réduit à un programme linéaire en utilisant un changement de variable non-linéaire approprié [24]. Séparément, Martos prouva en 1964 [73] que des programmes linéaires fractionnaires peuvent être résolus avec une procédure de parcours de sommets adjacents du domaine des solutions réalisables juste comme des programmes linéaires avec la méthode du simplexe. Il a identifié que les propriétés généralisées de convexité des rapports linéaires permet une telle prolongation de la technique de programmation linéaire. Depuis, différentes approches ont été proposées dans la littérature pour résoudre les deux problèmes de programmation linéaire fractionnaire en variables réelles (PLF) et en variables discrètes ($PLFE$). Celles-ci peuvent être divisées en des travaux qui ont développés des méthodes de résolution générales (par exemple, [19], [24], [58], [74], [99], [97] et [107]) et ceux qui se sont concentrés sur des applications (voir par exemple, [9], [46] et [99]).

Si l'étude des programmes fractionnaires avec seulement un rapport dans la fonction objectif a en grande partie dominé la littérature dans ce domaine jusqu'à environ 1980, il n'en demeure pas moins que des programmes fractionnaires avec plusieurs rapports ont été souvent étudiés dans le contexte plus large de la programmation convexe généralisée [11]. Les rapports de fonctions convexes aussi bien que des composés de tels rapports ne sont pas convexes en général, même dans le cas des rapports linéaires. Des programmes fractionnaires de ce type sont considérés dans [16], **[45]** et **[48].**

5.2 Les différents types de programmes fractionnaires

Les programmes fractionnaires linéaires ou non linéaires, en variables continues ou discrètes, apparaîssent souvent dans des modèles mathématiques liés à des problèmes de l'économie. Cependant, ils ont été aussi utilisés en physique, en programmation stochastique, dans les bases de données, en analyse numérique et en optimisation combinatoire ([99] et [47]). Nous rapportons dans ce qui suit, les modéles généraux de programmes fractionnaires de la litérature.

Soient f et g deux fonctions définies de $\mathbb{R}^n$ dans $\mathbb{R}$ et D un domaine non vide et borné de $\mathbb{R}^n$. Supposons que $g(x) > 0$ pour tout $x \in D$ et considérons le programme non linéaire suivant :

$$(PF) \begin{cases} \min\limits_{x \in D} Z = \dfrac{f(x)}{g(x)} \end{cases} \tag{5.1}$$

Le problème (PF) est appelé *programme fractionnaire*. En général, le domaine D des solutions réalisables est donné par :

$$D = \{x \in C \mid h_i(x) \leq 0,\ i = 1, \ldots, l\} \tag{5.2}$$

avec $C \subseteq \mathbb{R}^n$ et $h_i : \mathbb{R}^n \longrightarrow \mathbb{R}$, $1 \leq i \leq l$, un ensemble de fonctions continues à valeurs réelles.

Lorsque f, g et h_i, $1 \leq i \leq l$, sont des fonctions affines et $C = \mathbb{R}^n_+$ décrit l'orthant non négatif de $\mathbb{R}^n$, alors le problème (PF) d'optimisation est appelé *programme linéaire fractionnaire (PLF)*, ou *programme hyperbolique*, et prend alors la forme suivante :

$$(PLF) \begin{cases} \min Z = \dfrac{c^t x + \alpha}{d^t x + \beta} \\ Ax \leq b \\ x \geq 0 \end{cases} \tag{5.3}$$

avec $D = \{x \in \mathbb{R}^n \ / \ Ax \leq b, \ x \geq 0\}$, c et x sont des vecteurs de $\mathbb{R}^n$, d un vecteur non nul de $\mathbb{R}^n$, α et β des scalaires, A une $m \times n$-matrice réelle et b un vecteur de $\mathbb{R}^m$. Il est clair que si d est un vecteur nul avec $\beta \neq 0$ alors, (PLF) n'est autre qu'un problème classique de la programmation linéaire.

Si de plus, les variables sont astreintes à ne prendre que des valeurs entières, le programme (PLF) devient un *programme linéaire fractionnaire discrèt*, qui fait l'objet de nos préoccupations dans la suite de ce travail.

Nous obtenons un *programme quadratique fractionnaire* (PQF) si les fonctions f et g sont quadratiques, les fonctions h_i, $1 \leq i \leq l$, sont affines et $C = \mathbb{R}^n_+$ dans le problème (PF), dont la forme générale s'écrit comme suit :

$$(PQF) \begin{cases} \min Z = \dfrac{x^t H x + c^t x + \alpha}{x^t G x + d^t x + \beta} \\ Ax \leq b \\ x \geq 0 \end{cases} \tag{5.4}$$

avec H et G deux matrices symétriques définies positives, c d et x sont des vecteurs de $\mathbb{R}^n$, α et β des scalaires, A une mxn-matrice réelle et b un vecteur de $\mathbb{R}^m$

Le problème (PF) de minimisation est appelé *programme fractionnaire convexe* si C est un ensemble convexe, f et h_i, $1 \leq i \leq l$, sont des fonctions convexes et g est une fonction concave positive sur le domaine D. De même, le problème (PF) de maximisation est appelé *programme fractionnaire concave* si f est concave et g convexe sur D convexe.

D'autres problèmes considèrent plus d'un rapport dans la fonction objectif. Dans

le programme fractionnaire généralisé connu sous le nom de *programme fractionnaire min-max*, on considère le problème d'optimisation non linéaire qui consiste à minimiser le maximum d'un ensemble fini de fonctions fractionnaires :

$$\begin{cases} \min\limits_{x \in D} Z = \max\limits_{1 \leq k \leq r} \dfrac{f_k(x)}{g_k(x)} \end{cases} \tag{5.5}$$

avec $g_k(x) > 0$ pour tout k, $1 \leq k \leq r$, et pour tout $x \in D$.

Comme précédemment, les différentes fonctions peuvent être particulières et suivant le cas, on définira le programme fractionnaire généralisé de type linéaire, quadratique ou convexe. Ce type de programmes fractionnaires a été utilisé en économie dans des applications diverses liées aux finances et la gestion de façon générale.

Le programme d'optimisation non linéaire dont l'objectif est une somme de ratios s'écrit :

$$\begin{cases} \min\limits_{x \in D} Z = \sum\limits_{k=1}^{r} \dfrac{f_k(x)}{g_k(x)} \end{cases} \tag{5.6}$$

avec $g_k(x) > 0$ pour tout k, $1 \leq k \leq r$, et pour tout $x \in D$. Pour ce problème aussi, des spécifités sur les fonctions définissent des classes particulières de programmes fractionnaires.

Enfin, la prise en charge de plusieurs fonctions objectifs fractionnaires et conflictuelles, définie un autre type de programmes fractionaires qui s'écrivent :

$$\begin{cases} \min\limits_{x \in D} Z = \left(\dfrac{f_1(x)}{g_1(x)}, \dfrac{f_2(x)}{g_2(x)}, \ldots, \dfrac{f_r(x)}{g_r(x)} \right) \end{cases} \tag{5.7}$$

avec la condition que $g_k(x) > 0$ pour tout k, $1 \leq k \leq r$, et pour tout $x \in D$.

5.3 Applications de la programmation fractionnaire

Les programmes fractionnaires apparaissent souvent dans les problèmes de modélisation de processus économiques dans lesquels des activités utilisent certaines

ressources dans des proportions variés, tandis que l'objectif est d'optimiser un certain indicateur se ramenant au rapport le plus favorable bénéfice sur investissement, compte tenu de contraintes imposées par la disponibilité des biens. Ils peuvent aussi modéliser d'autres phénomènes dans des branches variées de l'activité humaine telles que la Physique [43], la théorie de l'information [76], [6]. Parmi les exemples liés à l'économie, nous citons les problèmes suivants : la planification financière d'une entreprise (dette/capitaux propres), le planning de production (inventaire/ventes, rendement/employé), le secteur de la santé et la plannification dans un hopital (coût/patient, infirmière/patient).

Data Envelopment Analysis (DEA) est un concept relativement nouveau pour l'évaluation de la performance d'un ensemble d'entités appelées unités de prise de décision (DMU, decision making units), qui convertit des entrées dans plusieurs sorties. Charnes, Cooper et Rhodes [25] ont introduit un programme fractionnaire comme modèle pour évaluer l'efficacité de différentes unités de prise de décision (DMU, decision making units). Une mesure possible de l'efficience d'une organisation est celle de l'efficience budgétaire ou technique, analysée selon un point de vue économique. Sous cette optique, il convient alors de comptabiliser les coûts qui sont nécessaires à une compagnie pour obtenir certains résultats, certains outputs. La méthode DEA permet de comparer cette efficience entre différentes unités de prise de décisions. Etant donnée une collection d'unités de décision, l'efficacité de chaque unité de prise de décision est obtenue à partir de la maximisation d'un rapport sorties/entrées pondérées. Les poids variables sont alors l'efficacité de chaque membre relatif à ceux des autres. La mesure de l'efficience est donc relative. Les DMU les plus efficientes obtiennent un score de 100 %, l'inefficience des autres étant mesurée par la distance entre elles et les meilleures.

Pour les problèmes de maximisation de la productivité, les travaux de Gilmore et Gomory [54] portent sur un problème classique de découpe dans l'industrie du papier pour lequel compte tenu des conditions données, il est plus approprié de réduire au minimum le rapport des chutes sur la quantité de la matière première utilisée plutôt que réduire au minimum juste les chutes. Ce problème de découpe est

formulé comme un programme linéaire fractionnaire. Dans une étude de cas, Hoskins et Blom [64] emploient la programmation fractionnaire pour optimiser l'affectation de personnel d'entrepôt. L'objectif est de réduire au minimum le rapport du coût de la main-d'oeuvre par rapport aux volumes entrant et sortant de l'entrepôt.

Dans certains problèmes d'allocation de ressource, on s'intéresse à la maximisation du bénéfice sur l'investissement. Dans ce cas, le rapport profit/capital ou profit/revenue doit être maximisé. Certains problèmes de sélection de portefeuilles donnent lieu à un programme fractionnaire concave dans lequel on maximise le rapport du bénéfice et du risque prévus. Les processus de décision de Markov peuvent également mener à la maximisation du rapport de la moyenne et l'écart type. Une application de la programmation fractionnaire dans la théorie de sélection de portefeuilles est donnée dans [71]. Les auteurs arguent du fait que le rapport de deux variances donne des modèles sophistiqués de prévisions avec une puissance prédictive significative.

Dans les problèmes de cheminements dans les graphes valués, on peut avoir à déterminé un circuit qui réduit au minimum le rapport coût-temps ou maximise le rapport profit-temps. Ces modèles sont des programmes fractionnaires combinatoires [37], [87].

Il existe aussi des applications de la programmation fractionnaire dans des domaines autres que ceux liés à l'économie. Dans la théorie de l'information, la capacité d'un canal de transmission peut être définie comme étant le taux maximal de transmission sur toutes les probabilités. C'est un programme fractionnaire non quadratique concave. Un exemple d'un programme fractionnaire dans la physique est donné par Falk [44]. Il maximise le rapport signal/bruit d'un filtre optique qui est un programme fractionnaire quadratique concave.

5.4 Rappels sur des éléments de la programmation mathématique

5.4.1 Définitions

Dans $\mathbb{R}^n$, la notation $\|\|$ désignera la norme euclidienne :

$$\|x\| = \left(\sum_{j=1}^{n} x_j^2 \right)^{\frac{1}{2}} \tag{5.8}$$

Définition 5.1 *Soit $S \subseteq \mathbb{R}^n$, $S \neq \varnothing$. L'ensemble $V(x_0) = \{x \in S; \|x - x_0\| < \varepsilon\}$ est appelé voisinage de $x_0 \in S$, où $\varepsilon > 0$.*

Définition 5.2 *On dit que x appartient à l'intérieur de S, noté $int(S)$, s'il existe $\varepsilon > 0$ tel que $V(x_0) \subset S$.*

Définition 5.3 *On dit que x appartient à l'adhérence de S, noté $\overline{S}$, si pour tout $\varepsilon > 0$, $V(x_0) \cap S \neq \varnothing$.*

Définition 5.4 *S est dit ensemble ouvert si $S = int(S)$. Il est dit fermé si $S = \overline{S}$.*

Définition 5.5 *Soit $(S, \|.\|)$ un espace vectoriel normé. S est dit borné si $\forall x \in S$, $\exists M > 0$ tel que $\|x\| \leq M$.*

Remarque 5.6 *Un ensemble borné n'est pas nécessairement fermé et inversement. Par exemple, $\mathbb{R}$ est fermé mais pas borné et l'intervalle $[0, 1[$ est borné mais pas fermé.*

Proposition 5.7 *$S \subset \mathbb{R}^n$ est compact si et seulement s'il est fermé et borné.*

5.4.2 Notions sur la convéxité

L'importance de la convexité en optimisation peut se résumer en la phrase de Rockafellar [**88**] : "*La vraie ligne de démarcation en optimisation n'est pas entre linéaire et non linéaire mais entre convexe et non convexe*"

Ensembles convexes

Définition 5.8 *Un ensemble S est dit convexe si :*

$$\forall x, y \in S,\ \forall \lambda \in [0,1]\ ona\ \lambda x + (1-\lambda)y \in S. \tag{5.9}$$

Cette définition peut s'interpréter en disant que le segment reliant x et y doit être dans S. Elle se généralise de la façon suivante : on dira qu'un vecteur y est une combinaison linéaire convexe des points $\{x_1, ..., x_p\}$ s'il existe des coefficients réels λ_i, $i = 1...p$, tels que :

$$y = \sum_{i=1}^{p} \lambda_i x_i,\ \text{avec}\ \lambda_i \geq 0, \forall i = 1...p,\ \text{et}\ \sum_{i=1}^{p} \lambda_i = 1 \tag{5.10}$$

On peut citer quelques cas particuliers : $\mathbb{R}^n$ tout entier est un ensemble convexe, de même qu'un singleton $\{a\}$.

Définition 5.9 *On appelle enveloppe convexe d'un ensemble $S \subset \mathbb{R}^n$, l'ensemble des points de $\mathbb{R}^n$ qui s'écrivent comme combinaisons convexes des points de S. On le note :*

$$conv(S) = \left\{ x \in \mathbb{R}^n \mid x = \sum_{i=1}^{p} \lambda_i x_i,\ x_i \in S, \lambda_i \geq 0, i = 1...p, et\ \sum_{i=1}^{p} \lambda_i = 1 \right\} \tag{5.11}$$

C'est le plus petit ensemble convexe contenant S.

Nous donnons dans ce qui suit quelques propriétés sur les ensembles convexes :
1− $S \subseteq \mathbb{R}^n$ est convexe si et seulement si tout point combinaison convexe de points de S est dans S.
2− S est convexe si et seulement si $S = conv(S)$.
3− Soit une famille $\{S_i\}$ $i = 1...k$, d'ensembles convexes et $S = \cap_{i=1}^{k} S_i$. Alors S est convexe.

L'ensemble $\{x \in \mathbb{R}^n \mid a^t x = b\}$ représente un hyperplan (une variété linéaire affine) de $\mathbb{R}^n$ et l'ensemble $\{x \in \mathbb{R}^n \mid a^t x \leq b\}$ représente un demi-espace fermé de $\mathbb{R}^n$ dont l'hyperplan correspondant constitue la frontière.

Définition 5.10 *Un polyhèdre convexe P est l'intersection d'un nombre fini de demi-espaces fermés et/ou d'hyperplans.*
Un polyhèdre convexe P est dit borné, s'il existe une valeur β finie et positive telle que

$$|x_j| \leq \beta \quad \forall j = 1 \ldots n, \quad \forall x \in P \tag{5.12}$$

Un polyhèdre convexe, borné et non vide est appelé polytope.

Définition 5.11 *On dit que x est un point extrême (ou sommet) de P,*

$$si\ x = \lambda y + (1-\lambda)z,\ pour\ y, z \in P,\ 0 < \lambda < 1,\ alors\ x = y = z. \tag{5.13}$$

En d'autres termes, x n'est à l'intérieur d'aucun segment de P.

Fonctions convexes

Définition 5.12 *On dit qu'une fonction $f : S \to \mathbb{R}$, $S \subseteq \mathbb{R}^n$, définie sur l'ensemble S convexe et non vide, est convexe si :*

$$\forall (x, y) \in S^2, \forall \lambda \in [0,1]\ on\ a :\ f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y) \tag{5.14}$$

f est dite strictement convexe sur S si :

$$\forall x, y \in S,\ x \neq y,\ \forall \lambda \in]0,1[,\ f(\lambda x + (1-\lambda)y) < \lambda f(x) + (1-\lambda) f(y). \tag{5.15}$$

Lorsque $n = 1$ cette définition s'interprète bien géométriquement : le graphe de la fonction f sur l'intervalle $]x,\ y[$ se trouve toujours en dessous du segment reliant les points $(x, f(x))$ et $(y, f(y))$.

La fonction f est dite *concave* (*strictement concave*) sur S, si $-f$ est convexe (*strictement convexe*) sur S.

Définition 5.13 *L'épigraphe de f, noté épi(f), est le sous-ensemble de $\mathbb{R}^{n+1}$ défini par :*

$$épi(f) = \{(x, \alpha) \in S \times \mathbb{R} \mid\ f(x) \leq \alpha\}. \tag{5.16}$$

Géométriquement, pour une fonction convexe f, tout point se trouvant sur le segment de droite d'extrémités $(x, f(x))$, $(y, f(y)$ est entièrement contenu dans l'épigraphe de f.

Corollaire 5.14 *Soit $f : S \to \mathbb{R}$, $S \subseteq \mathbb{R}^n$, S convexe et non vide. On définit pour $(x, y) \in S^2$, la fonction $\varphi : [0, 1] \to \mathbb{R}$ par $\varphi(t) = f(tx + (1-t)y)$. φ est convexe sur $[0, 1]$, $\forall (x, y) \in S^2$ si et seulement f est convexe sur S.*

Preuve. Si φ est convexe sur $[0, 1]$, on a en particulier :

$$\varphi(\lambda.1 + (1-\lambda).0) \leq \lambda\varphi(1) + (1-\lambda)\varphi(0), \ \forall \lambda \in [0, 1] \tag{5.17}$$

ce qui donne exactement :

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y). \tag{5.18}$$

et donc, f est convexe sur S.

Réciproquement, soit t, z et λ dans $[0, 1]$, on a alors, $\lambda t + (1-\lambda)z \in [0, 1]$, et

$$\begin{aligned} \varphi(\lambda t + (1-\lambda)z) &= f((\lambda t + (1-\lambda)z)x + (1 - (\lambda t + (1-\lambda)z))y) \\ &= f((\lambda tx + (1-\lambda t)y + (1-\lambda)zx - (1-\lambda)zy) + \lambda y - \lambda y \\ &= f(\lambda(tx + (1-t)y) + (1-\lambda)(zx + (1-z)y)) \\ &\leq \lambda f(tx + (1-t)y) + (1-\lambda)f(zx + (1-z)y) \\ &= \lambda\varphi(t) + (1-\lambda)\varphi(z). \end{aligned} \tag{5.19}$$

ce qui signifie que φ est convexe sur $[0, 1]$. □

Proposition 5.15 *f est convexe sur S si et seulement si épi(f) est un ensemble convexe.*

Preuve. $\Rightarrow\rfloor$ Soit (x, α), $(y, \beta) \in$ *épi*(f) et $\lambda \in [0, 1]$, alors $f(x) \leq \alpha$ et $f(y) \leq \beta$. D'autre part,

$$\begin{aligned} f(\lambda x + (1-\lambda)y) &\leq \lambda f(x) + (1-\lambda)f(y) \\ &\leq \lambda\alpha + (1-\lambda)\beta \end{aligned} \tag{5.20}$$

ce qui signifie que le couple $(\lambda(x,\alpha)+(1-\lambda)(y,\beta) \in épi(f)$.

$\Leftarrow\rfloor$ Soit $x, y \in S$ et $\lambda \in [0,1]$, on a,

$$\begin{aligned}(\lambda(x,f(x)),(1-\lambda)(y,f(y))) &\in épi(f) \\ &\Rightarrow (\lambda x+(1-\lambda)y, \lambda f(x)+(1-\lambda)f(y)) \in épi(f) \\ &\Rightarrow f(\lambda x+(1-\lambda)y) \leq \lambda f(x)+(1-\lambda)f(y)\end{aligned} \tag{5.21}$$

$\Rightarrow f$ est convexe sur S. □

Proposition 5.16 *f est continue en tout point de l'intérieur de S.*

Preuve. voir Bazarra [14]. □

Proposition 5.17 *Toute combinaison linéaire à coefficients positifs de fonctions convexes est une fonction convexe.*

Preuve. Soit $f_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, ..., p$, p fonctions convexes, $c_i \geq 0,\ i = 1, ..., p$, et f la fonction définie par : $f(x) = \sum_{i=1}^{p} c_i f_i(x),\ x \in \mathbb{R}^n$.
Soit $x, y \in \mathbb{R}^n$ et $\lambda \in [0,1]$.

$$\begin{aligned}f(\lambda x+(1-\lambda)y) &= \sum_{i=1}^{p} c_i f_i(\lambda x+(1-\lambda)y) \\ &\leq \sum_{i=1}^{p} c_i\left[\lambda f_i(x)+(1-\lambda)f_i(y)\right] \\ &\leq \lambda\sum_{i=1}^{p} c_i f_i(x)+(1-\lambda)\sum_{i=1}^{p} c_i f_i(y) \\ &= \lambda f(x)+(1-\lambda)f(y)\end{aligned} \tag{5.22}$$

Donc, f est convexe. □

5.4.3 Fonctions convexes généralisées

Soit $f : S \to \mathbb{R}$, où $S \subset \mathbb{R}^n$ est convexe.

Définition 5.18 *On dit f est quasiconvexe sur S, si*

$$\forall x_1, x_2 \in S,\ \forall \lambda \in [0,1],\ f(\lambda x_1+(1-\lambda)x_2) \leq \max\{f(x_1), f(x_2)\}. \tag{5.23}$$

Définition 5.19 *f est dite quasiconcave sur S, si*

$$\forall x_1, x_2 \in S,\ \forall \lambda \in [0,1]\,,\ f(\lambda x_1 + (1-\lambda)x_2) \geq \max\{f(x_1), f(x_2)\}\,. \tag{5.24}$$

Définition 5.20 *Soit f différentiable sur S. f est dite pseudoconvexe sur S, si*

$$\forall x_1, x_2 \in S,\ (x_2 - x_1)^t \nabla f(x_1) \geq 0 \Rightarrow f(x_2) \geq f(x_1). \tag{5.25}$$

Définition 5.21 *Considérons f différentiable sur S. f est dite pseudoconcave sur S, si*

$$\forall x_1, x_2 \in S,\ (x_2 - x_1)^t \nabla f(x_1) \leq 0 \Rightarrow f(x_2) \leq f(x_1). \tag{5.26}$$

Proposition 5.22 *Si f est convexe alors, f est quasiconvexe. Si de plus, f est différentiable alors, f est pseudoconvexe.*

Proposition 5.23 *Si f est concave alors, f est quasiconcave. Si de plus, f est différentiable alors, f est pseudoconcave.*

Remarque 5.24 *Les réciproques des deux propositions sont fausses en général.*

5.4.4 Conditions d'optimalité en optimisation non linéaire

Conditions nécessaires d'optimalité

Théorème 5.25 *Si $f : S \subseteq \mathbb{R}^n \to \mathbb{R}$ est une fonction convexe et différentiable sur S convexe alors, tout minimum local de f est un minimum global.*

Preuve. Soit x^* un minimum locale. alors il existe un voisinage de x^*, $V(x^*)$, tel que $f(x^*) \leq f(x)$, pour tout $x \in S \cap V(x^*)$.
Supposons que x^* ne soit pas minimum global, alors il existe $\overline{x} \in S$ tel que $f(\overline{x}) < f(x^*)$. Etant donné que f est convexe sur S, on a alors : pour toute valeur de λ, $0 \leq \lambda \leq 1$,

$$f(\lambda\overline{x} + (1-\lambda)x^*) \leq \lambda f(\overline{x}) + (1-\lambda)f(x^*) < \lambda f(x^*) + (1-\lambda)f(x^*) = f(x^*). \tag{5.27}$$

Donc, $f(\lambda\overline{x} + (1-\lambda)x^*) < f(x^*)$.
Pour une valeur suffisamment petite mais non nulle de λ, $\lambda\overline{x}+(1-\lambda)x^* \in S \cap V(x^*)$.
Ainsi, l'inégalité précédente contredit le fait que x^* est une solution optimale locale.
□

Théorème 5.26 *Soit $f : S \subseteq \mathbb{R}^n \to \mathbb{R}$ une fonction convexe et différentiable sur S convexe et $x^* \in S$ tel que $\nabla f(x^*) = 0$. Alors, x^* est un minimum global sur S.*

Preuve. Soit $x \in S$, f étant convexe et différentiable sur S alors $f(x) \geq f(x^*) + \nabla f(x^*)^t(x - x^*)$.
Comme $\nabla f(x^*) = 0$, alors $f(x) \geq f(x^*)$, donc, x^* est un minimum global pour f.
□

Théorème de Kuhn-Tucker

Considérons Le problème (P) définit comme suit :

$$(P) \begin{cases} \min f(x) \\ x \in S \end{cases} \tag{5.28}$$

où S est l'ensemble des solutions réalisables donné par :

$$S = \{x \in \mathbb{R}^n \mid g_i(x) \leq 0,\ i = 1 \ldots m\} \tag{5.29}$$

On suppose que f et $g_i,\ i = 1 \ldots m$, sont différentiables sur un domaine de $\mathbb{R}^n$. Les contraintes d'égalités de la forme $h(x) = 0$ sont remplacées par les deux contraintes $h(x) \leq 0$ et $-h(x) \leq 0$.

On définit les ensembles suivants :
$I(x) = \{i \in \{1, \ldots, m\} \mid g_i(x) = 0\}$: ensemble des indices des contraintes actives en x,
$Z(x) = \{z \in \mathbb{R}^n \mid \exists\overline{\alpha} > 0,\ (x + \alpha z) \in S,\ 0 < \alpha < \overline{\alpha}\}$: ensembles des directions admissibles en x,

$\overline{Z(x)}$: fermeture de $Z(x)$,
$Y(x) = \{z \in \mathbb{R}^n,\ z \neq 0 \mid z^t \nabla g_i(x) \leq 0,\ \forall i \in I(x)\}$.

On a alors le résultat suivant :

Lemme 5.27 *Soit $x \in S$, alors $\overline{Z(x)} \subset Y(x)$.*

Définition 5.28 *On dit que S satisfait l'hypothèse de qualification des contraintes en $x \in S$ si $\overline{Z(x)} = Y(x)$.*

Remarque 5.29 *Faire l'hypothèse de qualification des contraintes en $x \in S$, revient donc à supposer que z est une direction admissible si et seulement si z vérifie $z^t \nabla g_i(x) \leq 0,\ \forall i \in I(x)$.*

Lemme 5.30 *Pour que $\overline{Z(x)} = Y(x)$ en tout point $x \in S$, il suffit que l'une des deux conditions suivantes soit vérifiée :*
1- toutes les fonctions g_i sont linéaires,
2- toutes les fonctions g_i sont pseudoconvexes et il existe $\overline{x} \in S$ tel que $g_i(\overline{x}) < 0,\ \forall i = 1 \dots m$ (cette condition est connue sous le nom de condition de Slater)

Lemme 5.31 *Pour que $\overline{Z(x_0)} = Y(x_0)$ $x_0 \in S$, il suffit que les gradients $\nabla g_i(x_0),\ i \in I(x_0)$, soient linéairement indépendants.*

Le théorème de Kuhn-Tucker s'énonce alors comme suit :

Théorème 5.32 *Soit $x^* \in S$*
1- Supposons que $\overline{Z(x^*)} = Y(x^*)$. Si x^* est un minimum pour le problème (P') alors, il existe un unique vecteur $\lambda \in \mathbb{R}^m$ vérifiant le système suivant de Kuhn-Tucker :

$$(K.T) \begin{cases} \nabla f(x^*) + \sum_{i=1}^{m} \lambda_i \nabla g_i(x^*) = 0 \\ \lambda_i g_i(x^*) = 0,\ i = 1 \dots m \\ \lambda_i \geq 0,\ i = 1 \dots m \end{cases} \tag{5.30}$$

2- Si f est pseudoconvexe et $g_i,\ i = 1 \dots m$, quasiconvexes, une condition nécessaire et suffisante pour que x^* soit solution optimale pour le problème (P) 5.28 est qu'il existe $\lambda \in \mathbb{R}^m$ vérifiant les conditions du système $(K.T)$.5.30

5.5 Quelques résultats sur la programmation fractionnaire

On considère le programme linéaire fractionnaire (PFL) qui, rappelons le, s'écrit sous la forme suivante :

$$(PLF)\begin{cases}\min\limits_{x\in D} f(x) = \dfrac{c^t x+\alpha}{d^t x+\beta}\end{cases} \tag{5.31}$$

avec $D = \{x \in \mathbb{R}^n \mid Ax \leq b,\ x \geq 0\}$, c et x sont des vecteurs de $\mathbb{R}^n$, d un vecteur non nul de $\mathbb{R}^n$, α et β des scalaires, A une $m \times n$-matrice réelle et b un vecteur de $\mathbb{R}^m$. On suppose que D est un ensemble compact et que $d^t x + \beta > 0$ sur D.

Lemme 5.33 *f est à la fois pseudoconvexe et pseudoconcave sur D.*

Preuve. On montre d'abord que f est pseudoconvexe sur D.
Soit x, $y \in D$ vérifiant l'inégalité $(y-x)^t \nabla f(x) \geq 0$, montrons que $f(y) \geq f(x)$.
On a : $\nabla f(x) = \frac{(d^t x+\beta)c-(c^t x+\alpha)d}{(d^t x+\beta)^2}$.
Comme $(y-x)^t \nabla f(x) \geq 0$ et que $d^t x+\beta > 0$, on a : $(y-x)^t[(d^t x+\beta)c-(c^t x+\alpha)d] \geq 0$.
D'autre part ; $(y-x)^t[(d^t x+\beta)c-(c^t x+\alpha)d] = d^t x c^t y + \beta c^t y - d^t x c^t x - \beta c^t x$
$-c^t x d^t y - \alpha d^t y + c^t x d^t x + \alpha d^t x = d^t x(c^t y+\alpha) + \beta(c^t y+\alpha)$
$-c^t x(d^t y+\beta) - \alpha(d^t y+\beta) = (c^t y+\alpha)(d^t x+\beta) - (d^t y+\beta)(c^t x+\alpha)$.
Ainsi, $(c^t y+\alpha)(d^t x+\beta) \geq (d^t y+\beta)(c^t x+\alpha)$.
En divisant les deux termes de l'inégalité par le produit $(d^t x+\beta)(d^t y+\beta)$, on obtient : $\frac{(c^t y+\alpha)}{(d^t y+\beta)} \geq \frac{(c^t x+\alpha)}{(d^t x+\beta)}$,
ce qui signifie que $f(y) \geq f(x)$ et donc, f est pseudoconvexe.
De la même façon, on montre que si $(y-x)^t \nabla f(x) \leq 0$ alors $f(y) \leq f(x)$, et donc, f est aussi pseudoconcave. □

Théorème 5.34 *Si le problème (PLF) admet des solutions optimales, alors au moins une solution optimale x^* est un point extrême de D*

Preuve. Supposons que (PLF) admet des solutions optimales. Soit $x_1, x_2, ..., x_k$ les k points extrêmes de D et supposons que $x^* \neq x_i, \forall i = 1, ..., k$. Alors, $f(x^*) < \min_{1 \leq i \leq k} f(x_i) = \alpha$. Considérons l'ensemble $D_\alpha = \{x \in D \mid f(x) \geq \alpha\}$. On a alors, $x_i \in D_\alpha, \forall i = 1, ..., k$, et D_α est convexe compte tenu du fait que f est quasiconvexe D'autre part, $x^* = \sum_{i=1}^{k} \lambda_i x_i$, avec $\sum_{i=1}^{k} \lambda_i = 1, \lambda_i \geq 0, i = 1, ..., k$; et on a alors $x^* \in D_\alpha$, c-à-d, $f(x^*) \geq \alpha$, contradiction. Ainsi $f(x^*) = f(x_i)$ pour un certain point extrême x_i. □

Trois grandes stratégies de résolution d'un programme fractionnaire émergent dans la littérature : la résolution directe, la résolution d'un problème équivalent à objectif simplifié et la résolution par paramétrisation.

5.5.1 Résolution du programme quasiconvexe

Les programmes fractionnaires linéaires partagent quelques propriétés importantes avec les programmes linéaires :

(i) f étant quasiconvexe sur D, tout minimum local est un minimum global,

(ii) si une solution x vérifie les conditions suivantes d'optimalité de Kuhn-Tucker, elle est optimale pour (PLF) :

$$(KT) \begin{cases} \nabla f^t(x) + v^t A - u^t = 0 \\ u^t x = 0 \\ u \geq 0, \ v \ \textit{réel} \end{cases} \tag{5.32}$$

Ceci est dû au fait que f est pseudoconvexe et les contraintes étant linéaires, elles sont quasiconconvexes aussi. (Les conditions (KT) sont nécessaires et suffisantes pour une solution optimale de (PLF)).

(iii) toute solution optimale est atteinte en un point extrême du polyèdre convexe D du programme linéaire fractionnaire (théorème précédent).

L'extention de l'algorithme du simplexe au cas des programmes linéaires fractionnaires est dûe à Martos [73] et Swarup [103].

5.5.2 Linéarisation

Charnes et Cooper [24] ont montré qu'à l'aide de la transformation suivante, un programme linéaire fractionnaire peut être réduit à un programme linéaire :

$$\begin{cases} z = \frac{1}{d^t x + \beta} \\ y = z.x \end{cases} \tag{5.33}$$

ce qui permet de réduire (PLF) au programme linéaire suivant :

$$(PL) \begin{cases} \min c^t y + \alpha z \\ Ay - bz \leq 0 \\ d^t y + \beta z = 1 \\ y \geq 0, \\ z \geq 0 \end{cases} \tag{5.34}$$

où $y \in \mathbb{R}^n$ et z est un scalaire.

On montre que :
1− Si (y, z) est une solution réalisable pour (PL) alors, $z > 0$.
2− Si (y^*, z^*) est une solution optimale pour (PL) alors, $x^* = \frac{y^*}{z^*}$ est solution optimale pour (PLF).

5.5.3 Résolution d'un programme paramétré

Il y a une classe riche des algorithmes basés sur le problème auxiliaire suivant avec le paramétré réel λ :

$$(PLF_\lambda) : z(\lambda) = \min_{x \in D} c^t x + \alpha - \lambda(d^t x + \beta) \tag{5.35}$$

Remarquons que cette nouvelle fonction à minimiser est linéaire alors que celle du programme (PLF) ne l'est pas.

Soit x^* une solution optimale de (PLF) et $\lambda^* = \frac{c^t x^* + \alpha}{d^t x^* + \beta}$.
Dinkelbach [39] a montré que :

$$z(\lambda) = 0 \Longleftrightarrow \lambda = \lambda^* \tag{5.36}$$

et qu'une solution optimale de (PLF_{λ^*}) est aussi solution optimale de de (PLF). Ainsi, résoudre (PLF) est équivalent à trouver λ avec $z(\lambda) = 0$. Pour ce faire, différentes méthodes basées aussi bien sur la programmation linéiare que non linéaire ont été développées [58], [97], [101].

Chapitre 6

Solutions entières efficaces du problème multi-objectif linéaire fractionnaire

6.1 Introduction

Le problème de la programmation multi-objectif linéaire fractionnaire ($MOLFP$: Multiple Objective Linear Fractional Programming) est l'un des modèles les plus populaires utilisés dans la prise de décision à critères multiples. De nombreuses études et applications ont été rapportées dans la littérature dans des centaines de livres, monographies, articles. Pour un aperçu de ces études et des applications, voir, par exemple [5], [18], [20], [23], [34], [69], [77], [90], [99], [100], [101], [102].

Contrairement à la programmation linéaire multi-objectif ($MOLP$), Steuer [102] montre que l'ensemble des solutions efficaces d'un problème $MOLFP$ n'est pas nécessairement fermé ; certains des points intérieurs de l'ensemble des solutions réalisables peuvent être efficaces alors que d'autres ne le sont pas et les sommets efficaces ne sont pas tous reliés par des arêtes efficaces. Il devient difficile de générer l'ensemble des solutions efficaces. Ainsi, Kornbluth et Steuer [69] proposent un algorithme pour

le problème $MOLFP$ qui génère l'ensemble des solutions faiblement efficaces au moyen d'un algorithme basé sur la méthode du simplexe. Une nouvelle technique pour optimiser la somme pondérée des fonctions objectifs linéaires fractionnaires est proposée par Costa [34]. Cette technique ne génère qu'une seule solution non dominé pour le problème $MOLFP$ associée à un vecteur de poids donné. A chaque étape de la méthode de dichotomie, le domaine non dominé est divisé en deux sous-domaines et chacun d'eux est analysé afin d'écarter celui ne contenant pas la solution non dominée.

Dans le présent travail, nous nous somme penchés sur le problème $MOLFP$ à variables discrètes ($MOILFP$: Multiple Objective Integer Linear Fractional Pragramming) et l'étude porte sur l'extention des résultats trouvés pour le problème $MOILP$ au problème $MOILFP$. Une nouvelle technique pour générer l'ensemble efficient du problème $MOILFP$ dans l'espace des décisions est alors présentée et la convergence de l'algorithme proposé et prouvée.

Nous tenons à souligner que le problème $MOILFP$ n'a pas reçu autant d'attention que le problème $MOILP$, ce qui a justifié notre intérêt à étudier ce problème. Dans la littérature on ne trouve que très peu de méthodes dédiées au problème $MOILFP$. La méthode de Abbas & Moulaï décrite dans [5] dans laquelle les auteurs proposent un généralisation de leur méthode dans le cas d'un $MOILP$. Un calcul considérable est nécessaire pour obtenir une solution entière optimale d'un problème linéaire fractionnaire en nombres entiers ($PLFNE$) dans la première étape, et la méthode doit passer en revu tous les points entiers de l'ensemble des solutions réalisables. Dans [60], Gupta et Malhotra proposent une méthode qui généralise la méthode décrite par les mêmes auteurs dans [59] au cas linéaire fractionnaire, cependant, la même erreur est reproduite sur le critère d'arrêt. Dans son mémoire de magister, Ouaïl [82] a donné un exemple où l'algorithme décrit dans [60] se termine sans donner tout l'ensemble des solutions efficaces, et a proposé une nouvelle méthode généralisant la méthode de Abbas et Chaabane [1] au cas linéaire fractionnaire. Saad et Hughes [90] ont décrit une méthode dans le cas particulier ne considérant que deux critères.

La méthode que nous proposons dans [28] utilise la méthode de Cambini et Martein [19] pour l'obtention d'une solution optimale pour le problème relaxé de (PLF), puis une solution entière est trouvée par l'application du processus de séparation de la méthode branch & bound. En outre, de la même manière que pour le problème $MOILP$, une coupe efficace est construite en prenant en compte l'évolution des critères de $MOILFP$. De cette manière, nous sommes en mesure d'éliminer non seulement des solutions non entières du domaine des solutions réalisables, mais aussi des solutions entières qui ne sont pas efficaces. Ainsi, notre méthode permet d'éviter d'énumérer toutes les solutions entières possible du problème.

Les notations et définitions utilisées dans ce travail sont présentées dans la section 2. Dans la section 3, l'algorithme de génération des solutions efficaces pour le problème $MOILFP$ est développé. Les principaux résultats théoriques sont présentés dans la section 4. Un exemple illustratif est donné dans la section 5 et des tests de calcul des solutions efficaces sont indiqués dans la section 6. La section 7 est réservée à la conclusion et des perspectives.

6.2 Formulation et concepts de base

Le but de ce travail est de mettre au point une méthode exacte pour résoudre le problème $MOILFP$:

$$(P)\begin{cases} \max f_1(x) = \dfrac{c^1x + \alpha^1}{d^1x + \beta^1} \\ \max f_2(x) = \dfrac{c^2x + \alpha^2}{d^2x + \beta^2} \\ \vdots \\ \max f_k(x) = \dfrac{c^kx + \alpha^k}{d^kx + \beta^k} \\ x \in S \\ x \text{ vecteur entier} \end{cases} \tag{6.1}$$

où $k \geq 2$; c^i, d^i sont des $1 \times n-$vectors ; α^i, β^i sont des scalaires pour tout $i \in \{1, 2, ..., k\}$; $S = \{x \in \mathbb{R}^n \mid Ax \leq b, x \geq 0\}$; A est une $m \times n-$matrices réelle et

$b \in \mathbb{R}^m$. Dans toute la suite de cette étude, nous supposons que l'ensemble S est non vide, compact dans $\mathbb{R}^n$ et $d^i x + \beta^i > 0$ sur S pour tout $i,\ i \in \{1, 2, ..., k\}$. On note $f = (f_i)_{i=1,...,k}$

Un point $x \in \mathbb{R}^n$ est appelé *solution efficace*, ou *solution Pareto-optimale*, pour le problème $MOILFP$, lorsque $x \in S \cap \mathbb{Z}^n$ et il n'existe pas un autre point $y \in S \cap \mathbb{Z}^n$ tel que $f(y) \geq f(x)$ et $f(y) \neq f(x)$. Autrement, x est non efficace et le vecteur $f(y)$ *domine* le vecteur $f(x)$.

L'approche adoptée dans ce travail pour générer toutes les solutions efficaces du problème (P) est basée sur la résolution du programme linéaire fractionnaire suivant, à chaque étape l :

$$(P_l) \begin{cases} \max f_1(x) = \dfrac{c^1 x + \alpha^1}{d^1 x + \beta^1} \\ x \in S_l \end{cases} \tag{6.2}$$

avec $S_0 = S$ et sans la contrainte d'intégrité des variables. Notons qu'à la place de f_1, on peut de façon similaire considérer le problème (P_l) avec une autre fonction objectif f_i pour n'importe quelle valeur de $i,\ i \in \{2, \ldots, r\}$. D'autre part, avant de définir la suite des ensembles S_l, nous avons besoin de définir les paramètres qui suivent.

Soit x_l^* la solution entière obtenue après résolution de (P_l) par la méthode de Cambini et Martein [19] basée sur la méthode du simplexe et déclanchement du processus de séparation éventuellement. On note B_l l'ensemble des indices des variables de base et N_l l'ensemble des indices des variables hors base de x_l^*. Soit $\overline{\gamma_j^i}$ la $j^{ème}$ composante du vecteur gradient réduit $\overline{\gamma^i}$ défini pour chaque critère i, $i \in \{1, 2, ..., k\}$ par :

$$\overline{\gamma^i} = \overline{\beta^i}\overline{c^i} - \overline{\alpha^i}\overline{d^i} \tag{6.3}$$

où $\overline{c^i}$, $\overline{d^i}$, $\overline{\alpha^i}$ et $\overline{\beta^i}$ sont les valeurs mises à jour de c^i, d^i, α^i et β^i respectivement. Notons que le gradient de f_i et le gradient réduit correspondant $\overline{\gamma^i}$ pour tout i, $i \in$

$\{1, 2, ..., k\}$, sont de même signe. Ainsi, le calcul de $\overline{\gamma^i}$ est suffisant pour déterminer la direction de croissance pour chaque critère i.

Afin de donner l'expression mathématique de l'ensemble S_{l+1}, nous définissons l'ensemble suivant en x_l^* :

$$H_l = \left\{ j \in N_l \mid \exists i \in \{1, 2, ..., k\} \; ; \; \overline{\gamma_j^i} > 0 \right\} \cup \left\{ j \in N_l \mid \overline{\gamma_j^i} = 0, \, \forall i \in \{1, 2, ..., k\} \right\} \tag{6.4}$$

Enfin, l'ensemble S_{l+1} est définit comme suit :

$$S_{l+1} = \left\{ x \in S_l \mid \sum_{j \in H_l} x_j \geq 1 \right\} \tag{6.5}$$

Dans la section suivante, l'approche de résolution du programme (P) est présentée.

6.3 Approche de résolution

Dans cette section, une méthode exacte basée sur le processus de branchement et utilisant une coupe efficace est décrite dans le but de générer toutes les solutions entières efficaces du problème (P). Tout d'abord, la méthode est présentée en détail, suit alors l'algorithme de résolution de (P). Nous finissons la section avec les résultats théoriques qui prouvent la finitude et la convergence de l'algorithme.

6.3.1 Description de la méthode

Soit x une solution optimale de (P_l). Si x est entière, $f(x)$ est comparé à l'ensemble des solutions potentiellement efficaces déjà trouvés et l'ensemble des solutions efficaces est actualisé. La direction de croissance de chaque critère est déterminé à l'aide de son gradient. La méthode utilise ces informations pour construire une coupe en mesure de supprimer des solutions entières qui ne sont pas efficaces pour le problème (P) et détermine une nouvelle solution réalisable entière si elle existe. Dans le cas

où cette solution, qui est optimale dans l'ensemble courant des solutions réalisables, n'est pas entière, deux nouveaux programmes linéaires fractionnaires sont créés en utilisant le processus de branchement. Chacun d'eux sera résolu comme le problème (P_l). Le processus s'arrête lorsque tous les domaines créés ont été exploré. Un domaine sera considéré comme un domaine exploré, si le sommet correspondant dans l'arborescence est sondé. Un sommet est sondé dans les deux situations suivantes : aucun critère ne peut être amélioré, auquel cas H_l est vide, ou bien le problème courant n'a pas de solutions réalisables.

Initialement, l'algorithme de Cambini et Martein [19] résout le programme linéaire fractionnaire en variables continues suivant :

$$(P_0)\begin{cases} \max f_1(x) = \dfrac{c^1 x + \alpha^1}{d^1 x + \beta^1} \\ x \in S_0 \end{cases} \tag{6.6}$$

Ceci est basé sur le concept de solution niveau optimale. Un point réalisable $\overline{x}$ est une solution niveau optimale pour (P_0), si $\overline{x}$ est une solution optimale pour le programme linéaire suivant :

$$(P_{\overline{x}})\begin{cases} \max(c^1 x + \alpha^1) \\ d^1 x = d^1 \overline{x} \\ x \in S_0 \end{cases} \tag{6.7}$$

Si de plus $\overline{x}$ est point extrême de S_0, $\overline{x}$ est dit solution niveau optimale de base. Il est connu qu'une solution optimale pour le programme linéaire fractionnaire (P_0) est une solution niveau optimale de base. Selon cette étude, l'algorithme génère une séquence finie de solutions niveau optimale de base, la première, disons x^0, est une solution optimale pour le programme linéaire :

$$\begin{cases} \min(d^1 x + \beta^1) \\ x \in S_0 \end{cases} \tag{6.8}$$

Si x^0 est unique, alors elle est aussi une solution niveau optimale de base pour le programme (P_0), sinon, résoudre le programme linéaire (P_{x^0}) pour obtenir une solution niveau optimale de base.

La solution du programme (P_0), obtenue en un nombre finie de solutions niveau optimale, est optimale si et seulement si $\overline{\gamma_j^1} \leq 0$ pour tout $j \in J_0^1$, où

$$J_0^1 = \left\{ j \in N_0 \mid \overline{d_j^1} > 0 \right\} \tag{6.9}$$

Sinon, il existe un indice $j \in J_0^1$ pour lequel $\overline{\gamma_j^1} > 0$. La variable hors base x_r, $r \in J_0^1$, qui doit rentrer dans la base est indiquée par l'indice r tel que :

$$\frac{\overline{c_r^1}}{\overline{d_r^1}} = \max \left\{ \frac{\overline{c_j^1}}{\overline{d_j^1}},\ j \in J_0^1 \right\} \tag{6.10}$$

Le format originel des fonctions objectifs fractionnaires et la structure d'origine des contraintes sont maintenues et les itérations sont effectuées dans un tableau du simplexe augmentée qui comprend $m + 3k$ lignes supplémentaires. Les m premières lignes correspondent aux contraintes d'origine, les lignes $m + 3(i-1) + 1$ et $m + 3(i-1) + 2$ correspondent au numérateur et dénominateur de la fonction objectif fractionnaire f_i, $i \in \{1, 2, ..., k\}$ du programme (P) respectivement, et la ligne $m+3i$ correspond au vecteur $\overline{\gamma_l^i}$ à l'étape l.

A chaque étape de l'algorithme, toutes les lignes évoluent suivant l'opération de pivotage lorsque la variable hors base x_r, $r \in J_0^1$, rentre dans la base, à l'exception des lignes $m + 3i$, pour $i \in \{1, 2, ..., k\}$, qui sont modifiées suivant la formule (6.3) de $\overline{\gamma_l^i}$

Chaque programme (P_l) correspond au nœud l dans une arborescence structurée. Un nœud l de l'arborescence est sondé, si le programme correspondant ((P_l) n'est pas réalisable ou $H_l = \varnothing$ (domaine exploré).

Si la solution optimale $\widetilde{x}_l$ du programme (P_l) n'est pas entière, soit $\widetilde{x}_{lj}$ une composante de $\widetilde{x}_l$ égale à un nombre fractionnaire. Le nœud l de l'arborescence est alors séparée en deux nœuds qui lui sont imposées par les contraintes supplémentaires $x_j \leq \lfloor \alpha_j \rfloor$ et $x_j \geq \lfloor \alpha_j \rfloor + 1$. Dans chaque nœud, le programme linéaire fractionnaire obtenu doit être résolu, jusqu'à ce qu'à l'obtention d'une solution entière, lorsqu'elle existe. En présence d'une solution entière, la coupe efficace

$$\sum_{j \in H_l} x_j \geq 1 \tag{6.11}$$

est rajoutée au programme et le nouveau programme est résolu en utilisant la méthode duale du simplexe. Le procédé se termine lorsque tous les nœuds créés sont sondés.

6.3.2 Algorithme

L'algorithme de génération de toutes les solutions entières efficace du programme (P) est présenté dans les étapes suivantes. Les nœuds de l'arborescence sont traités selon le principe de retours en arrière vers le nœud le plus récemment créé (backtracking).

Algorithme 6.1 (solutions efficaces pour MOILFP (EMOILFP))

Etape 0. *Initialisation*

$l = 0$;

Créer le premier nœud avec le programme (P_0) *;*

$Eff = \varnothing$ *; (ensemble des solutions entières efficaces du programme* (P)*)*

Etape 1. *Etape Générale*

Tant qu'il existe un nœud non encore sondé, faire :

Choisir le nœud de plus grand numéro l *non encore sondé ;*

Résoudre le programme linéaire fractionnaire correspondant (P_l) *par la méthode duale du simplexe et la méthode de Cambini et Martein. (Initialement, pour la résolution du programme* (P_0)*, seule la méthode Cambini et Martein est utilisée) ;*

Si le programme (P_l) *n'a pas de solutions réalisables, alors le nœud correspondant est sondé.*

Sinon, soit $\widetilde{x}_l$ *la solution optimale obtenue. Si* $\widetilde{x}_l$ *n'est pas entière, aller à l'étape 1a, sinon aller à l'étape 1b.*

Etape 1a. *Séparation*

Choisir une coordonnée non entière $\widetilde{x}_{lj}$ *de* $\widetilde{x}_l$ *et séparer le nœud* l *actuel en deux nœuds* k, $k \geq l+1$, *et* h, $h \geq l+1$, $h \neq k$ *;*

Dans le tableau courant du simplexe, la contrainte $x_j \leq \lfloor \widetilde{x}_{lj} \rfloor$ *est rajoutée et un nouveau domaine est considéré au nœud* k *et de façon similaire, la contrainte* $x_j \geq$

$\lfloor \widetilde{x}_{lj} \rfloor + 1$ *est rajoutée pour obtenir un autre domaine au nœud h (chaque programme créé doit être résolu en utilisant le même processus jusqu'à ce qu'une solution entière réalisable soit trouvée). Aller à l'étape 1.*

Etape 1b. *Mise à jour de l'ensemble* Eff

Si le vecteur $f(\widetilde{x}_l)$ *n'est pas dominé par le vecteur* $f(x)$ *pour toute solution* $x \in Eff$, *alors* $Eff := Eff \cup \{\widetilde{x}_l\}$. *S'il existe* $x \in Eff$ *telle que* $f(\widetilde{x}_l)$ *domine* $f(x)$, *alors* $Eff := Eff \setminus \{x\} \cup \{\widetilde{x}_l\}$.

Déterminer les ensembles N_l *et* H_l;

Si $H_l = \varnothing$, *alors le nœud correspondant est sondé. Aller à l'étape 1*

Sinon, rajouter la coupe efficace $\sum_{j \in H_l} x_j \geq 1$ *au programme* (P_l). *Aller à l'étape 1.*

6.4 Résultats théoriques

Les théorèmes suivants montrent que l'algorithme EMOILFP génère toutes les solutions entières efficaces du programme (P) en un nombre fini d'étapes.

Théorème 6.2 *Supposons que* $H_l \neq \varnothing$ *au point entier courant* x_l^*. *Si* x *est une solution entière efficace dans le domaine* $S_l \setminus \{x_l^*\}$, *alors* $x \in S_{l+1}$.

Preuve. Soit x une solution entière se trouvant dans le domaine $S_l \setminus \{x_l^*\}$ telle que $x \notin S_{l+1}$, alors $\sum_{j \in H_l} x_j = 0$, ce qui implique que $x_j = 0$ pour tout $j \in H_l$.
A partir du tableau du simplexe correspondant à la solution optimale x_l^*, les critères sont évalués comme suit

$$f_i(x) = \frac{\sum_{j \in N_l} \overline{c_j^i} x_j + \overline{\alpha^i}}{\sum_{j \in N_l} \overline{d_j^i} x_j + \overline{\beta^i}} \ \forall i \in \{1, ..., k\}, o\grave{u} \ \frac{\overline{\alpha^i}}{\overline{\beta^i}} = f_i(x_l^*). \tag{6.12}$$

Ainsi, on peut écrire :

$$f_i(x) = \frac{\sum_{j \in N_l \setminus H_l} \overline{c_j^i} x_j + \overline{\alpha^i}}{\sum_{j \in N_l \setminus H_l} \overline{d_j^i} x_j + \overline{\beta^i}}, \ \forall i \in \{1, ..., k\} \tag{6.13}$$

D'autre part, $\overline{\gamma_j^i} = \overline{\beta^i}\overline{c_j^i} - \overline{\alpha^i}\overline{d_j^i} \leq 0$, pour tout $j \in N_l \backslash H_l$ et $\overline{\gamma_j^i} = \overline{\beta^i}\overline{c_j^i} - \overline{\alpha^i}\overline{d_j^i} < 0$ pour au moins un critère, implique que $\overline{c_j^i} \leq \dfrac{\overline{\alpha^i}\overline{d_j^i}}{\overline{\beta^i}}$ pour tout $j \in N_l \backslash H_l$, car $\overline{\beta^i} = d^i x_l^* + \beta^i > 0$ pour tous les critères $i \in \{1, ..., k\}$. Les variables de décision étant non négatives, on obtient $\overline{c_j^i} x_j \leq \dfrac{\overline{\alpha^i}\overline{d_j^i}}{\overline{\beta^i}} x_j$ pour tout $j \in N_l \backslash H_l$ et par suite,

$$\sum_{j \in N_l \backslash H_l} \overline{c_j^i} x_j \leq \sum_{j \in N_l \backslash H_l} \frac{\overline{\alpha^i}\overline{d_j^i}}{\overline{\beta^i}} x_j \Rightarrow \sum_{j \in N_l \backslash H_l} \overline{c_j^i} x_j + \overline{\alpha^i} \leq \sum_{j \in N_l \backslash H_l} \frac{\overline{\alpha^i}\overline{d_j^i}}{\overline{\beta^i}} x_j + \overline{\alpha^i} \tag{6.14}$$

Pour tout critère f_i, $i \in \{1, ..., k\}$, l'inégalité suivante est obtenue :

$$\begin{aligned}
f_i(x) &= \frac{\sum\limits_{j \in N_l \backslash H_l} \overline{c_j^i} x_j + \overline{\alpha^i}}{\sum\limits_{j \in N_l \backslash H_l} \overline{d_j^i} x_j + \overline{\beta^i}} \qquad (6.15) \\
&\Rightarrow f_i(x) \leq \frac{\sum\limits_{j \in N_l \backslash H_l} \dfrac{\overline{\alpha^i}\overline{d_j^i}}{\overline{\beta^i}} x_j + \overline{\alpha^i}}{\sum\limits_{j \in N_l \backslash H_l} \overline{d_j^i} x_j + \overline{\beta^i}} \\
&\Rightarrow f_i(x) \leq \frac{\dfrac{\overline{\alpha^i}}{\overline{\beta^i}} \left(\sum\limits_{j \in N_l \backslash H_l} \overline{d_j^i} x_j + \overline{\beta^i} \right)}{\sum\limits_{j \in N_l \backslash H_l} \overline{d_j^i} x_j + \overline{\beta^i}} \\
&\Rightarrow f_i(x) \leq \frac{\overline{\alpha^i}}{\overline{\beta^i}} \\
&\Rightarrow f_i(x) \leq f_i(x_l^*)
\end{aligned}$$

Par conséquent, $f_i(x) \leq f_i(x_l^*)$ pour tout $i \in \{1, ..., k\}$ et $f_i(x) < f_i(x_l^*)$ pour au moins un indice i. Donc, $f(x_l^*)$ domine $f(x)$ et la solution x n'est pas efficace. □

Corollaire 6.3 *Supposons que $H_l \neq \varnothing$ au point entier courant x_l^*. Alors, la contrainte 6.11 est une coupe efficace.*

Preuve. D'après le théorème précédent, aucune solution efficace n'est supprimée lorsque la contrainte 6.11 est rajoutée. On peut alors dire que c'est une contrainte valide efficace. De plus, x_l^* ne vérifie pas cette contrainte puisque $x_j = 0$, pour tout $j \in N_l$. On conclut que la contrainte 6.11 est une coupe efficace. □

Proposition 6.4 *Si $H_l = \varnothing$ au point entier courant x_l^*, alors $S_l \setminus \{x_l^*\}$ est un domaine exploré.*

Preuve. $H_l = \varnothing$ signifie que x_l^* est une solution entière optimale pour tous les critères (x_l^* constitue un point idéal localement dans le domaine S_l) et donc $S_l \setminus \{x_l^*\}$ ne contient pas de solutions efficaces. □

Théorème 6.5 *L'algorithme EMOILFP converge en un nombre fini d'itérations en générant l'ensemble des solutions efficaces du programme (P).*

Preuve. L'ensemble S des solutions réalisables de (P) étant compact, il contient un nombre fini de solutions entières. Chaque fois qu'une solution entière optimale x_l^* est trouvée, la coupe efficace 6.11 est rajoutée. Ainsi, compte tenu du théorème et du corollaire précédents, au moins la solution x_l^* est éliminée, mais aucune solution efficace n'est supprimée. □

6.5 Exemple illustratif

Le programme (P) suivant est donné comme exemple de problème de programmation multi-objectif linéaire fractionnaire en nombres entiers $(MOILFP)$ dans Kornbluth et Steuer [69] :

$$(P)\begin{cases} \max f_1(x) = \dfrac{x_1 - 4}{-x_2 + 3} \\ \max f_2(x) = \dfrac{-x_1 + 4}{x_2 + 1} \\ \max f_3(x) = -x_1 + x_2 \\ -x_1 + 4x_2 \leq 0 \\ 2x_1 - x_2 \leq 8 \\ x_1 \geq 0,\ x_2 \geq 0,\ \textit{entiers} \end{cases} \tag{6.16}$$

Initialement le programme (P_0) est résolu et la solution optimale est donnée dans le tableau du simplexe suivant :

B	b	x_3	x_4
x_2	$8/7$	$2/7$	$1/7$
x_1	$32/7$	$1/7$	$4/7$
c^1	$-4/7$	$-1/7$	$-4/7$
d^1	$-13/7$	$2/7$	$1/7$
γ^1		$-3/7$	$-8/7$
c^2	$4/7$	$1/7$	$4/7$
d^2	$-15/7$	$-2/7$	$-1/7$
γ^2		$1/7$	$8/7$
c^3	$24/7$	$-1/7$	$3/7$

Tableau 6.1

Comme $\gamma_j^1 \leq 0$ pour tout $j \in J_0^1$, $J_0^1 = N_0 = \{3,\ 4\}$, alors la solution obtenue $(32/7, 8/7)$ est optimale pour le programme (P_0), mais elle n'est pas entière. Le processus de séparation commence :

Nœud 1 : $x_1 \geq 5 \Longleftrightarrow -1/7x_3 - 4/7x_4 \geq 3/7$. ceci est impossible et le nœud 1 est sondé.

Nœud 2 : $x_1 \leq 4 \Longleftrightarrow -1/7x_3 - 4/7x_4 \leq -4/7$.

Cette contrainte est rajoutée et la méthode duale du simplexe est appliquée. La solution entière optimale du programme (P_2) est obtenue dans le tableau suivant :

B	b	x_3	x_5
x_2	1	1/4	1/4
x_1	4	0	1
x_4	1	1/4	−7/4
c^1	0	0	−1
d^1	−2	1/4	1/4
γ^1		0	−2
c^2	0	0	1
d^2	−2	−1/4	−1/4
γ^2		0	2
c^3	3	−1/4	3/4

Tableau 6.2

$\gamma_j^1 \leq 0$ pour toutl $j \in J_2^1$, $J_2^1 = N_2 = \{3,\ 5\}$, alors la solution courante $(4, 1)$ est optimale. $Eff := \{(4, 1)\}$, $H_2 = \{5\}$ et $S_3 = \{x \in S_2 \mid x_5 \geq 1\}$.La coupe efficace $x_5 \geq 1$ est rajoutée et après pivotage, on obtient :

B	b	x_2	x_6
x_3	3	4	1
x_1	3	0	1
x_4	2	−1	−2
x_5	1	0	−1
c^1	1	0	−1
d^1	−3	−1	0
γ^1		−1	−3
c^2	−1	0	1
d^2	−1	1	0
γ^2		−1	1
c^3	3	1	1

Tableau 6.3

$J_3^1 = \varnothing$ alors $(3,0)$ est une solution entière optimale et $Eff := \{(4,1),(3,0)\}$.

En procédant de cette manière, on obtient :

B	b	x_1	x_{10}
x_6	3	1	0
x_3	0	1	4/3
x_4	8	1	-1
x_5	4	1	0
x_7	2	0	-1
x_9	1	3	8
x_8	0	1	1
x_2	0	-1	-1
c^1	4	1	0
d^1	-3	-1	-1
γ^1		-1	-4
c^2	-4	-1	0
d^2	-1	1	1
γ^2		-5	-4
c^3	0	0	1

Tableau 6.4

$J_4^1 = \varnothing$ alors $(0,0)$ est une solution entière optimale, $Eff := \{(4,1),(3,0),(0,0)\}$, $N_4 = \{1,10\}$, $H_4 = \{10\}$ et $S_5 = \{x \in S_4 \mid x_{10} \geq 1\}$. La coupe efficace $x_{10} \geq 1$ est rajoutée et on obtient :

B	b	x_{11}	x_{10}
x_6	2	1	-1
x_3	-1	1	$1/3$
x_4	7	1	-2
x_5	3	1	-1
x_7	2	0	-1
x_9	-2	3	5
x_8	-1	1	0
x_2	1	-1	0
x_1	1	-1	1
c^1	3	1	-1
d^1	-2	-1	0
γ^1		-1	-4
c^2	-3	-1	1
d^2	-2	1	0
γ^2		-5	-4
c^3	0	0	1

Tableau 6.5

Le dual est non réalisable, alors le nœud correspondant est sondé.

L'algorithme $EMOILFP$ s'arrête puisque tous les nœuds sont sondés et l'ensemble de toutes les solutions entières efficaces du programme (P) est $Eff = \{(4,1), (3,0), (2,0), (1,0), (0,0)\}$.

6.6 Expérimentation et résultats

Le programme informatique de l'algorithme $EMOILFP$ est codé en MATLAB 7.0 et exécuté sur un DELL Pentium 4, 3.40 GHz 4, 1 Go de RAM. Le logiciel utilisé

est testé sur des instances générés aléatoirement. Nous montrons les résultats de l'expérience de calcul dans le tableau suivant :.

(n, m, α)		$(15, 10, 33)$	$(20, 10, 25)$	$(25, 5, 17)$	$(25, 10, 17)$
Nb Solutions.	Moy	99.50	204.50	200.45	98.00
$Efficaces$	Max	215	324	397	228
	Min	4	23	67	17
$CPU(second)$	Moy	38.52	185.96	400.48	306.74
	Max	54.39	337.08	673.17	367.11
	Min	23.094	143.56	193.87	177.09
Nb Itérations	Moy	75837.7	126488.50	521763.45	255726.55
du simplexe	Max	101207	365750	719105	306665
	Min	55851	2717	369414	145324
$Coupes\ Efficaces$	Moy	885.3	2341.50	2607.15	979.00
(EC)	Max	1152	2593	3909	1268
	Min	451	679	839	440
$(EC)/(Coupes)$	Moy	0.91	0.97	0.84	0.81

Tableau 6.6

La méthode est testée avec les valeurs de $m = 5$ et 10 contraintes avec $k = 4$ objectives et n variables, $n \in \{15, 20, 25\}$ générées aléatoirement. Les coefficients sont des entiers non corrélés uniformément distribués dans l'intervalle $[1, 100]$ pour les contraintes et dans $[1, 80]$ pour les objectifs et ceci pour les deux premiers types de dinstances traitées. Pour les deux derniers types de instances testées, les valeurs entières de la matrice des contraintes varient dans l'intervalle $[1, 50]$ et pour ceux des critères dans $[1, 30]$. La valeur du second membre b est défini à $\alpha\%$ de la somme des coefficients (partie entière) de chaque contrainte, où $\alpha \in \{17, 25, 33\}$. Avec chaque instance (n, m, α), une série de 20 problèmes est résolu et tout l'ensemble des solutions efficaces a été généré pour toutes ces instances.

La méthode étant exacte, il était prévu que le nombre d'itérations du simplexe soit très grand compte tenu du fait que, pour ce type de problèmes, le nombre de solutions efficaces augmente rapidement avec la taille des données. En outre, nous tenons à souligner que le rapport CE/Coupes tend vers la valeur un, montrant que le nombre de coupes efficaces introduit dans la méthode avoisine le nombre total des coupes et indiquant que ce type de coupes efficaces a un impact positif sur la recherche de l'ensemble des solutions efficaces.

6.7 Conclusion

Dans cette étude, une méthode exacte pour générer l'ensemble des solutions efficaces d'un problème $MOILFP$ est présenté. La méthode ne nécessite aucune optimisation non linéaire. Un programme linéaire fractionnaire est résolu en utilisant l'algorithme de Cambini et Martein [19], puis en utilisant le concept bien connu de branch and bound, des solutions entières sont générées. Les coupes efficaces proposées exploitent tous les critères dans un tableau du simplexe, et seules les parties du domaine des solutions réalisables contenant des solutions efficaces sont explorées. En outre, la coupe efficace est facilement construite et rajouter au tableau courant d'une solution entière. La méthode est dédiée à la résolution de problèmes $MOILFP$ généraux puisqu'aucune restriction n'est faite sur le nombre de critères, ou bien sur le type de matrices des contraintes du problème. Signalons toutefois que plus le second membre b est grand, plus le nombre de solutions réalisables entières augmente ce qui ralenti de façon significative la convergence de l'algorithme. De plus, comme pour le problème $MOILP$, l'étape d'évaluation du principe du branch & bound n'est pas encore bien exploitée. Afin de palier à cet inconvénient et rendre l'algorithme plus performant, l'arborescence doit être exploitée pour la construction d'un algorithme parallèle. Notons qu'en l'absence de méthodes dédiées au problème étudié, aucune étude comparative n'est faite.

Conclusion et Perspectives

Dans ce présent travail, nous avons passé en revu dans les premiers chapitres, les concepts fondamentaux de la programmation mono-objectif ainsi que les développements récents dans la littérature, aussi bien dans le cas linéaire que fractionnaire linéaire en nombres entiers. Ces éléments représentent les outils de base nécessaires au développement des notions et concepts de la programmation multi-objectif qui a fait l'objet du noyau central de cette thèse. C'est ainsi que nous nous sommes focalisé sur la mise en œuvre de méthodes exactes de résolution des problèmes de la programmation multi-objectif et nous avons déployées, dans une première étape, deux nouvelles méthodes générales de résolution d'un programme multi-objectif linéaire en nombres entiers ; la première méthode génère l'ensemble des solutions efficaces dans l'espace des décisions alors que la deuxième engendre toutes les solutions non dominées dans l'espace des objectifs. Ceci a été rendu possible grâce au couplage du principe du branch & bound avec des coupes efficaces que nous avons conçus spécialement pour chacune des deux méthodes. Des études comparatives avec d'autres méthodes récentes de la littérature ont révélées un meilleur comportement des deux méthodes pour des problèmes considérant plus de trois objectifs et produisant un nombre important de solutions efficaces ou non dominées.

Nous avons aussi abordés l'étude du problème de la programmation fractionnaire linéaire en nombres entiers $MOILFP$.et nous avons réussi à généraliser la méthode donnant l'ensemble efficace complet au problème $MOILP$. Une expérience est établie mettant en exergue les résultats obtenus sur des instances générées aléatoirement et un exemple illustratif est développé montrant le déroulement de l'algorithme.

Dans ce cas, aucune comparaison n'est faite en l'absence dans la littérature, de méthodes générales dédiées au problème.

Il faut noter cependant, qu'on ne prétend pas résoudre des problèmes de grande taille, compte tenu que les méthodes sont exactes et peuvent nécessiter un espace mémoire important du calculateur utilisé et beaucoup de temps de calcul.

Parmi les travaux qui peuvent présenter des perspectives et qu'on souhaite aborder pour l'avenir on y trouve :

- Le réglage des paramètres et une meilleure adaptation de la méthode branch & bound à même d'assurer une plus grande performance des méthodes décrites,

- La parallélisation des méthodes,

- L'adaptation à des problèmes MOCO particuliers,

- La généralisation au problème multi-objectif quadratique en variables entières,

- La recherche de solutions non dominées pour les problèmes fractionnaires et quadratiques multi-objectif en variables entières,

- La généralisation aux problèmes étudiés mixtes.

Bibliographie

[1] Abbas M. and Chaabane D. (2002) An algorithm for solving multiple objective integer linear programming problem, *RAIRO Operations Research* 36, pp. 351-364.

[2] Abbas M. and Chaabane D. (2006) Optimizing a linear function over an integer efficient set, *European Journal of Operational Research*, 174, 1140–1161.

[3] Abbas M., Chergui M.E-A. and Aït Mehdi M. (2010) Efficient Cuts for Generating the Non dominated Vectors for Multiple Objective Integer Linear Programming, *Special Issue on Polyhedra & Combinatorial Optimization, IJMOR* (à paraître).

[4] Abbas M., Moulaï M. (1999) Solving multiple objective integer linear programming, *Ricerca Operativa* 29/89, 15–38.

[5] Abbas M. and Moulaï M. (2002) Integer Linear Fractional Programming with Multiple Objective, *Ricerca Operativa Journal of the Italian Operations Research Society* 103,104, 15-38.

[6] Aggrawal S.P., Sharma I.C. (1970) Maximization of the transmission rate of a discrete, constant channel, *Unternhmensforschung*, 14, 152-155.

[7] Ait Mehdi M. (2008) Contribution algorithmique en programmation multi-objectif discrète, *mémoire de Magister en Mathématiques*, option recherche opérationnelle, USTHB, Algérie.

[8] Alves M., Clímaco J. (2007) A Review of Interactive Methods for Multiobjective Integer and Mixed-Integer Programming, *EJOR* 180, 99–115.

[9] Arora S.R., Swarup K., Puri M.C. (1977) The set covering problem with linear

fractional functional, *Indian Journal of Pure and Applied Mathematics*, Vol. 8, p. 578-588.

[10] Avriel M. (1976) Nonlinear Programming : Analysis and Methods, *Prentice-Hall*, Englewwod Cliffs, NJ.

[11] Avriel M., Diewert W.E., Schaible S. and Zang I. (1988) Generalized Concavity, *Plenum Press*, New York.

[12] Balas E. (1979) Disjunctive programming. In P. L. Hammer, E. L. Johnson, and B. H. Korte, editors, *Discrete Optimization II*, 5, pp 3–51, Amsterdam. *Annals of Discrete Mathematics*, North-Holland.

[13] Balas E., Ceria S. and Cornuéjols G. (1993) A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Mathematical Programming*, 58 :295–324.

[14] Bazaraa M.S., Sherali H.D. and Shetty C.M. (2006) Nonlinear Programming. Theory and Algorithms, Third Edition, *Willey & Sons*

[15] Benson H. P. (1978) Existence of efficient solutions for vector maximization problems, *Journal of Optimization Theory and Appl.*, 26, 569–580.

[16] Billionnet A., Djebali K. (2006) Résolution d'un problème combinatoire fractionnaire par la programmation linéaire mixte, *RAIRO - Operations Research*, 40, 2, p. 97-111.

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

[18] Caballero R. and Hernández M. (2004) The controlled estimation method in the multiobjective linear fractional problem, *Computers & Operations Research*, 31(11), 1821–1832.

[19] Cambini, A. and Martein, L. (1992) Equivalence in Linear Fractional Programming, *Optimization* 23, 41-51.

[20] Cambini A., Martein L. and Stancu-Minasian I.M. (1999) A survey of bicriteria fractional problems, *Advanced modeling and optimization*, *An electronic International Journal* 1, 1, 9-46.

[21] Chaabane D. (2005) Contribution à l'optimisation multicritère en variables discrètes, *thèse de doctorat d'Etat en mathématiques*, option recherche opérationnelle, USTHB, Algérie.

[22] Chaabane D., Pirlot M. A (2010) A Method for Optimizing over the Integer Efficient Set, *Journal of Industrial and Management Optimization*, Volume 6 , Number 4 , pp.811–823.

[23] Chakraborty M. and Gupta S. (2002) Fuzzy mathematical programming for multi objective linear fractional programming problem, *Fuzzy Sets and Systems* 125, 335- 342.

[24] Charnes A. and Cooper, W.W. (1962) Programming with linear fractional functionals, *Naval Research Logistics Quarterly.* 9, 181-186.

[25] Charnes A., Cooper W.W. and Rhodes E. (1978) Measuring the efficiency of decision making units, *EJOR* 2, 429-444.

[26] Chergui M.E-A., Abbas M. et Moulaï M. (2005) Génération des solutions efficaces d'un programme linéaire multi-objectif en nombres entiers, *Proceedings du Colloque sur l'Optimisation et les Systèmes d'Informations* (COSI), Algérie.

[27] Chergui M.E-A., Moulaï M. and Ouaïl F.Z. (2008) Solving the Multiple Objective Integer Linear Programming Problem, *Modelling, Computation and Optimization in Information Systems and Management Sciences Communications in Computer and Information Science*, Volume 14, Part 1, 69-76.

[28] Chergui M.E-A. and Moulaï M. (2008) An Exact Method for a Discrete Multiobjective Linear Fractional Optimization, *Journal of Applied Mathematics and Decision Sciences*, Vol. 2008, Article ID 760191, 12 pages.

[29] Chvátal V. (1973) Edmonds polytopes and a hierarchy of combinatorial problems, *Discrete Math.* 4, 305–337.

[30] Clímaco J., Ferreira C.and Captivo M.(1997) Multicriteria integer programming : An overview of the different algorithmic approaches, *Multicriteria Analysis* 2, 248–258.

[31] Cook S. A. (1971) The complexity of theorem proving procedures. In *stoc71*, pages 151–158.

[32] Cornuéjols G. (2001) Combinatorial optimization : packing and covering, *Regional Conference Series In Applied Mathematics. Society for Industrial and Applied Mathematics*, Philadelphia, PA, USA, Vol. 74.

[33] Cornuéjols G. (2008) Valid Inequalities for Mixed Integer Linear Programs, *Mathematical Programming* B 112, 3-44.

[34] Costa J.P. (2007) Computing non-dominated solutions in MOLFP, *European Journal of Operational Research* 181, 1464–1475.

[35] Craven B.D. (1988) Fractional programming : *Sigma Series in Applied Mathematics 4*, Heldermann Verlag.

[36] Dantzig G.B. (1963) Linear programming and extention, *Princeton University Press.*

[37] Dantzig G.B., Blattner W. and Rao M.R. (1966) Finding a cycle in a graph with minimum cost to time ratio with application to a ship routing problem, *Theory of Graphs International Symposium*, Dunod, Paris and Gordon and Breach, New York, 77-8.

[38] Delorme X., Gandibleux X. and Degoutin F. (2010) Evolutionary, constructive and hybrid procedures for the bi-objective set packing problem, *European Journal of Operational Researc*h, Vol. 204, 206–217.

[39] Dinkelbach W. (1967) On nonlinear fractional programming, *Management Science* 13 , 492-498.

[40] Ecker J.G., Kouada I.A. (1978) Finding All Efficient Extreme Points for Multiple Objective Linear Programs. Mathematical Programming 14 (2), 249-261.

[41] Ehrgott M. and Gandibleux X. (2000) An Annotated Bibliography of Multiobjective Combinatorial Optimization, *Report in Wirtschaftsmathematik,* N0 62, Fachbereich Mathematik - Universitat Kaiserslautern.

[42] Ehrgott M., Figueira J. and Gandibleux X. (editors) (2006) Multiple Objective Discrete and Combinatorial Optimization, *Annals of Operations Research*, Vol. 147.

[43] Ermolev L.G. (1972) A problem of fractional nonlinear programming, *Cybernetics*, 8,2, 224-227.

[44] Falk J.E. (1969) Maximization of signal-to-noise ratio in an optical filter. *SIAM Journal of Applied Mathematics*, 7 :582–592.

[45] Falk J.E., Paloscay S.W.(1992) Optimising the sum of linear fractional functions, *Advances in Global Optimisation*, Kluwer Academic Publishers, p. 221-258.

[46] Frenck, J.B.G., Schaible S. (2001) Fractional programming. *In Encyclopedia of Optimization*, Floudas CA, Pardalos PM (eds.), Kluwer Academic Publishers, 162-172.

[47] Frenck J.B.G., Schaible S. (2004) Fractional Programming, *ERASMUS Research Institute of Management*, 55 pages.

[48] Freund R.W., Jarre F. (2001) Solving the sum-of-ratios problem by an interior-point method, *Journal of Global Optimization*, Vol. 19, p. 83-102.

[49] Gal T. (1977) A General Method for Determining the Set of All Efficient Solutions to a Linear Vector Maximum Problem, *European Journal of Operational Research* 1 (5), 307-322.

[50] Garey, M. R. and Johnson, D. S. (1979) Computers and intractability : a guide to the theory of NP-Completeness, *V.H. Freeman and Company.*

[51] Geoffrion A.M. (1967) Solving Bi-Criterion Mathematical Programs, *Operations Research*, 15 :1, 39-54.

[52] Geoffrion A.M. (1968) Proper efficiency and the theory of vector maximization, *Journal of Mathematical Analysis and Applications,* 22, pp. 618-630.

[53] Geoffrion A.M., Dyer J.S. and Feinberg A. An Interactive Approach for Multi-Criterion Optimization, with an Application to the Operation of an Academic Department, *Management Science*, 19 :4, 357-368.

[54] Gilmore P.C. and Gomory.R.E. (1963) A linear programming approach to the cutting stock problem-part II *Operations Research*, 11 :863–888.

[55] Gomory R.E. (1958) Outline of an algorithm for integer solutions to linear programs, *Bull. Amer. Math. Soc.* 64, 275–278.

[56] Gomory R.E. (1960) An algorithm for the mixed-integer problem, *Report RM-2597*, Rand Corporation.

[57] Gonzales J.J, Reeves G.R. and Franz L.S. (1985) An Interactive Procedure for Solving Multiple Objective Integer Linear Programming Problems, i Haimes Y. and Chankong (eds), Decision Making with Multiple Objectives, Springer-Verlag, Berlin, p.250-260.

[58] Granot D. and Granot F. (1977) On integer and mixed integer fractional programming problems, *Annals of Discrete Mathematics* 1, 221-231.

[59] Gupta R. and Malhotra R. (1992) Multi-criteria integer linear programming problem, *Cahiers de CERO* 34, pp. 51-68.

[60] Gupta R. and Malhotra R. (1995) Multi-criteria integer linear fractional programming problem, *Optimization* 35, pp. 373-389.

[61] Haimes Y., Lasdon L. and Wismer D. (1971) On a bicriterion formulation of the problems of integrated system identification and system optimization, *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 1, pp.296–297.

[62] Hansen P., Poggi De Aragao M. V., Ribeiro C.C.(1991) Hyperbolic 0-1 programming and information retrieval, *Mathematical Programming*, Vol. 52, p. 255-263.

[63] Horst R., Pardalos P.M. and Nguyen V. T. (2000) Introduction to Global Optimization, *Series : Nonconvex Optimization and Its Applications*, Vol. 48 2nd ed.

[64] Hoskins J.A and Blom R. (1984) Optimal allocation of warehouse personnel : a case study using fractional programming. *FOCUS* (U.K), 3(2) :13–2

[65] Jorge M. J. (2009) An algorithm for optimizing a linear function over an integer efficient set, *European Journal of Operational Research*, 195, 98–103.

[66] Karmarkar N. (1984) A new polynomial-time algorithm for linear programming, *Combinatorica*, 4 :373–395.

[67] Khachiyan L. (1979) A polynomial algorithm in linear programming, *Soviet Mathematics Doklady*, 20 :191–194.

[68] Klein D., Hannan E. (1982) An algorithm for multiple objective integer linear programming problem, *EJOR* 9, 378–385.

[69] Kornbluth J.S.H. and Steuer R.E. (1981) Multiple Objective Linear Fractional Programming, *Management Science* 27 : 1024-1039.

[70] Korhonen P., Wallenius J. and Zionts S. (1984) Solving Discrete Multiple Criteria Problem Using Convex Cones, *Management Science* 30(11), 1336–1345.

[71] Lo A. and MacKinlay C. (1997) Maximizing predictability in the stock and bond markets. *Macroeconomic Dynamics*, 1(1) :102–134.

[72] Luenberger D. G. and Ye Y. (2008) Linear and Nonlinear Programming, Third Edition, *Springer*.

[73] Martos B. (1964) Hyperbolic programming, *Logistics Quarterly*, 11 :135–155 ; originally published in *Math. Institute of Hungarian Academy of Sciences* 5, 383-406, (1960).

[74] Martos B (1975) Nonlinear programming, Theory and Methods, *North-Holland*, Amsterdam.

[75] MCDMlib. Mcdm numerical instances library. http ://www.terry.uga.edu/mcdm/.

[76] Meisler B. and Oettli, W. (1967) On the capacity of a discrete, constant channel, *Inform. and Control* 11, 341-351.

[77] Metev, B. and Gueorguieva, D. (2000) A simple method for obtaining weakly efficient points in multiobjective linear fractional programming problems, *European Journal of Operational Research*, 126(2), 386–390.

[78] Mezmaz M., Melab N., Talbi E-G. (2007) Combining metaheuristics and exact methods for solving exactly multi-objective problems on the Grid, *Journal*

of Mathematical Modelling and Algorithms, Vol.6, No.3, pp.393-409.

[79] Moulaï M. (2002) Optimisation multicritère fractionnaire linéaire en nombres entiers, *thèse de doctorat d'Etat en mathématiques*, option recherche opérationnelle, USTHB, Algérie.

[80] Nagih A. and Plateau G. (1999) Problèmes fractionnaires : Tour d'horizon sur les applications et méthodes de résolution, *RAIRO Oper. Res.* 33 383-419.

[81] Nemhauser G.L., Wolsey L.A. (1988) Integer and combinatorial optimization, *Wiley*, Chichester.

[82] Ouaïl F.Z. (2008) Optimisation vectorielle discrète, *mémoire de Magister en Mathématiques*, option recherche opérationnelle, USTHB, Algérie.

[83] Özlen M. and Azizoğlu, M. (2009) Multi-objective integer programming : A general approach for generating all non-dominated solutions, *European Journal of Operational Research*, Vol. 199, pp.25–35.

[84] Padberg M. W. (1973) On the facial structure of set packing polyhedra, *Mathematical Programming*, Vol. 5, pp.199–215.

[85] Philip J. (1972) Algorithms for the vector maximization problem, *Mathematical Programming*, 2, 207–229.

[86] Przybylski A., Gandibleux X. and Ehrgott M. (2007) A Two Phase Method for Multiobjective Integer Programming and its Application to the Assignment Problem with Three Objectives. *Technical Report LINA*.

[87] Radzik T. (1998) Fractional combinatorial optimization, *Handbook of Combinatorial Optimization*, Edited by Z.-Z. Du and P. Paradaloas, Kluwer Academic Publishers, p. 429-478.

[88] Rockafellar R.T.(1993) Lagrange multipliers and optimality, *SIAM Rev.*, 35, p. 183-238.

[89] Roy B. (1971) Problems and methods with multiple objective functions, *Mathematical Programming*, Vol. 1, No. 2, p. 239-266.

[90] Saad O.M. and Hughes J.B. (1998) Bicriterion integer linear fractional programs with parameters in the objective functions, *J. Inform. Optim. Sci.*

19, 1, 97-108.

[91] Sakarovitch M. (1984) Optimisation combinatoire, Graphes et programmation linéaire, *Hermann.*

[92] Sakarovitch M. (1984) Optimisation combinatoire, programmation discrète, *Hermann.*

[93] Schaible S.(1981) Fractional programming : applications and algorithms, *European Journal of Operational Research* 7, 111-120.

[94] Schaible S. (1983) Fractional programming, *Zeitschrift für Operations Research*, 27, 39-54.

[95] Schaible S. (1995) Fractional programming, in R. Horst and P.M. Pardalos (eds.), *Handbook of Global Optimization*, Kluwer Academic Publishers, 495-608, 1995.

[96] Schrijver A. (1986) Theory of Linear and Integer Programming. *Wiley and Sons.*

[97] Seshan, C.R. and Tikekar, V.G. (1980) Algorithms for integer fractional programming, *Journal of the Indian Institute of Science Section B-Physical and Chemical Series* 62, 9-16.

[98] Simonard M. (1972) Programmation linéaire, 2ème édition, tomes 1 et 2, *Dunod.*

[99] Stancu-Minasian I.M. (1997) Fractional Programming : Theory, Methods and Applications, *Kluwer Academic Publishers.*

[100] Stancu-Minasian I.M. (1999) A fifth bibliography of fractional programming, *Optimization* 45, 1-4, 343-367.

[101] Stancu-Minasian I.M. (2006) A sixth bibliography of fractional programming, *Optimization* 55, 4, 405–428.

[102] Steuer R.E. (1986) Multiple Criteria Optimization : Theory, Computation and Applications, *Wiley*, New York .

[103] Swarup K. (1965) Linear fractional functionl programminig, *Operations Research* 13, N° 6, 1029-1036.

[104] Sylva J., Crema A. (2004) A method for finding the set of non-dominated vectors for multiple objective integer linear programs, *EJOR* 158(1), 46–55.

[105] Sylva J., Crema A. (2007) A method for finding well-dispersed subsets of non-dominated vectors for multiple objective mixed integer linear programs. *European Journal of Operational Research* 180, 1011–1027.

[106] Talbi E-G. (2009) Metaheuristics : from design to implementation, *Wiley*, USA.

[107] Tantawy S.F. (2007) A New Method for Solving Linear Fractional Programming Problems, *Australian Journal of Basic and Applied Sciences*, 1(2) : 105-108.

[108] Teghem J., Kunsch P. (1986) A survey of techniques to determine the efficient solutions to multi-objective integer linear programming, *Asia Pacific Journal of Oper. Res.* 3, 95–108.

[109] Teghem J. (1998) Programmation lineaire, *Ellipses.*

[110] Ulungu E.L., Teghem J. (1994) Multi-objective Combinatorial Optimization Problem : A Survey, *Journal of Multi-Criteria Decision Analysis* 3, 83–104.

[111] Ulungu E.L., Teghem J. (1995) The two phases method : An efficient procedure to solve bi-objective combinatorial optimization problems. *Foundation of computing and decision science*, 20 :149–156.

[112] Von Neumann, J. (1937) ¨Uber ein ¨okonomisches Gleichungssystem und eine Verallgemeinerung des Brouwerschen Fixpuntsatzes. In K. Menger, editor, *Ergebnisse eines mathematischen Kolloquiums (8)*, Leipzig und Wien, pages 73–83.

[113] Yamamoto Y. (2002) Optimization over the efficient set : Overview, *Journal of Global Optimization* 22 285–317.

[114] Zionts S. and Wallenius J. (1983) An interactive multiple objective linear programming method for a class of underlying nonlinear value functions, *Management Science* 29, 519–529.