N° d'ordre : 74/2020-C/MT

RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ DES SCIENCES ET DE LA TECHNOLOGIE HOUARI BOUMEDIÈNE

Faculté de Mathématiques

THÈSE

Présentée pour l'obtention du grade de Docteur

En : **Mathématiques**

Spécialité : **Recherche Opérationnelle et Management**

Par : CHERFAOUI Yasmine

Titre :

Optimisation d'un Programme "Linéaire plus Linéaire Fractionnaire" Discret à Objectif Multiple

Soutenue publiquement, le 03/12/2020, devant le jury composé de :

Mr.	AÏDER MÉZIANE	Professeur	à l'USTHB (Alger)	Président
Mr.	MOULAÏ MUSTAPHA	Professeur	à l'USTHB (Alger)	Directeur de thèse
Mr.	AÏDENE MOHAMED	Professeur	à l'UMMTO (Tizi-Ouzou)	Examinateur
Mr.	BENTERKI DJAMEL	Professeur	à l'Univ. Ferhat Abbas (Setif)	Examinateur
Mr.	CHAABANE DJAMAL	Professeur	à l'USTHB (Alger)	Examinateur

« La nuit n'est jamais complète.
Il y a toujours, puisque je le dis,
Puisque je l'affirme,
Au bout du chagrin
Une fenêtre ouverte,
Une fenêtre éclairée,
Il y a toujours un rêve qui veille,
Désir à combler,
Faim à satisfaire,
Un cœur généreux,
Une main tendue, une main ouverte,
Des yeux attentifs,
Une vie, la vie à se partager. »
La nuit n'est jamais complète de Paul Eluard

Résumé

Les problèmes posés dans la réalité relèvent souvent du domaine de l'optimisation mathématique où la fonction-objectif est non-linéaire, n'est pas convexe et ne possède pas de propriété de convexité généralisée. La difficulté de ces problèmes réside dans le fait que l'enveloppe convexe du domaine ne contient pas nécessairement un optimum et qu'un optimum local n'est pas toujours global. Cette difficulté croit particulièrement pour les problèmes multiobjectifs puisqu'ils n'admettent pas d'optimum unique mais une multitude de solutions dites efficientes ou PARETO optimales.

Dans cette thèse, nous avons proposé une méthode d'exploration implicite combinant le principe de la recherche arborescente à une méthode de coupe pour générer toutes les solutions efficientes des problèmes d'optimisation d'un programme "linéaire plus linéaire fractionnaire" discret à objectif multiple. C'est un problème d'optimisation multiobjectif qui possède beaucoup d'applications pratiques mais dont la base théorique reste assez insuffisante puisque la fonction linéaire plus linéaire fractionnaire n'est ni convexe ni concave et ne possède pas de propriété de convexité généralisée.

Par ailleurs, dans ce travail on propose aussi de remédier à la problématique causée par la grande cardinalité de l'ensemble des solutions efficientes en résolvant un type inédit de problème, il s'agit du problème d'optimisation multiobjectif sur l'ensemble des solutions efficientes. Ainsi, pour un programme linéaire discret on propose une méthode qui permet de générer une partie de l'ensemble des solutions efficientes dont la cardinalité et la répartition suivent les préférences de plusieurs décideurs.

Mots-clés : Programmation Multiobjectif, Programmation Fractionnaire, Programmation en Nombre Entier, Séparation et coupe.

Abstract

Problems posed in reality often belong within the domain of mathematical optimization where the objective-function is non-linear, is not convex and does not have a generalized convexity property. The difficulty of these problems lies in the fact that the convex envelope of the domain does not necessarily contain an optimum and that a local optimum is not always a global optimum. This difficulty increases particularly for multiobjective problems since they do not admit a single optimum but a multitude of so-called efficient or PARETO optimal solutions.

In this thesis, we proposed an implicit exploration method combining the principle of tree searching with a cutting method to generate all the efficient solutions to the optimization problems of a discrete "linear plus linear fractional" multiobjective program. It is a multiobjective optimization problem which has many practical applications but whose theoretical basis remains rather insufficient since the linear plus linear fractional function is neither convex nor concave and does not have a generalized convexity property.

Moreover, in this work we also propose to remedy to the problematic caused by the large cardinality of the set of efficient solutions by solving a new type of problem, the problem of multiobjective optimization over the efficient set. Thus, for a discrete linear

program, we propose a method that allows to generate a part of the set of efficient solutions whose cardinality and distribution follows the preferences of several decision-makers.

Keywords : Multiobjective Programming, Fractional Programming, Integer Programming, Branch and cut.

ملخص

غَالبًا مَا تنتمي المسَائِل المطروحة في الوَاقع إِلَى مجَال التحسين الريَاضي حيث تكون الوظيفة الموضوعية غير خطية و ليست محدبة و ليس لديهَا خَاصية تحديبية عَامة. تكمن صعوبة هذه المسَائِل في حقيقة ان المغلف المحدب لِلّمجَال لَا يحتوي بَالضرورة علَى الحل الأَمثل وَأن الحلول المثلَى المحلية ليست دَائمًا عَامة. تزدَاد هذه الصعوبة بشكل خَاص بَالنسبة لّمسَائِل متعدِّدة الأَهدَاف نظرًا لِأَنهَا لَا تملك حل أمثل وَاحد ولكن العديد من الحلول تدعَى الحلول الفعَالة او حلول بَارِيتو مثَالية.

في هذه الأَطروحة اقترحنَا طريقة استكشَاف ضمنية تجمع بين مبدَأ البحث التفرعي مع طريقة القطع لِتَحصِيل جميع الحلول الفعَالة لمسَائِل تحسين برنَامج خطي زَائد كسري خطي ذَات الأَعدَاد الصحيحة المتعدّد الأَهدَاف. إِنهَا مسَألة تحسين متعدّدة الأَهدَاف لهَا العديد من التطبيقَات العملية ولكن أَسَاسهَا النظري لَا يزَال غير كَاف لِأَن الدَالة الكسرية الخطية زَائد الخطية ليست محدبة و لَا مقعرة و لَا تملك خَاصية تحديبية عَامة.

علَاوة علَى ذلك نقترح ايضَا في هذَا العمل معَالجة المسَألة التي تسبّبهَا الكَارديِنَالية الكبيرة لمجموعة الحلول الفعَالة عن طريق حل نوع جديد من المسَائِل أَلَا وهي مسَألة تحسين متعدِّد الأَهدَاف علَى المجموعة الحلول الفعَالة. وبِالتَالي بِالنسبة لِبرنَامج الخطي ذَت الأَعدَاد الصحيحة نقترح طريقة تسمح بتوليد جزء من مجموعة الحلول الفعَالة يتبع أصل وتوزيع تفضيلَات العديد من صَانعي القرَار.

كلمَات مفتَاحية : برمجة متعدّدة الأَهدَاف، برمجة جزئِية، تفرع و قطع.

Remerciements et dédicaces

Ma profonde gratitude va au Professeur M. Moulaï le directeur de cette thèse, pour l'aide qu'il a fournie et tout ces judicieux conseils et ces encouragements, pour sa patience, sa disponibilité et surtout pour toutes les connaissances qu'il a voulu transmettre.

Je tiens aussi à remercie le Professeur M. Aïder pour m'avoir honoré d'accepter de présider le jury. Je remercie aussi les autres membres du jury :

— Professeur M. Aidene, enseignant chercheur à l'université de Tizi-Ouzou,

— Professeur D. Benterki, enseignant chercheur à la faculté des sciences de l'université de Sétif,

— Professeur D. Chaabane, enseignant chercheur à la faculté des mathématiques de l'USTHB,

pour leur présence, pour leur lecture attentive de ma thèse ainsi que pour les remarques qu'ils m'adresseront lors de cette soutenance afin d'améliorer mon travail.

Je remercie mes collègues Amel, Fatma, Wassila et Yassine pour leurs aides, leurs soutiens et leur solidarité.

Je remercie aussi Tante Fadila et Oncle Djilali pour leurs soutiens.

Je tiens aussi à remercier toutes celles et tous ceux qui ont contribué à mon parcours universitaire.

Je remercie en dernier lieu mes frères Réda, El-Hadi et Youssef et leurs femmes respectives Hassina, Badia et Sihem.

Je dédie ce travail à mes adorables nièces et neveux Farah, Anfal, Hiba, Abdelkrim et Abdelghafour.

Á la mémoire de mes parents

Malika et Abdelkrim

Liste des abréviations

s.c.	**s**ous **c**ontrainte.
PL	**P**rogramme **L**inéaire.
PLNE	**P**rogramme **L**inéaire en **N**ombre **E**ntiers.
MOILP	**M**ulti**O**bjective **I**nteger **L**inear **P**rogram. *Programme linéaire multiobjectif en nombre entier.*
MOILFP	**M**ulti**O**bjective **I**nteger **L**inear **F**ractionnal **P**rogram. *Programme linéaire fractionnaire multiobjectif en nombre entier.*
MOILLFP	**M**ulti**O**bjective **I**nteger **L**inear plus **L**inear **F**ractionnal **P**rogram. *Programme linéaire plus linéaire fractionnaire multiobjectif en nombre entier.*
B&B	**B**ranch and **B**ound. *Procédure par séparation et évaluation.*
B&C	**B**ranch and **C**ut. *Procédure par séparation et troncature.*
Max.	Maximum (temp).
Min.	Minimum (temp).

Liste des symboles

$c = (c_j)_{j=1,\dots,n}$	Vecteur ligne à n composantes sauf mention contraire.
cx	Produit scalaire du vecteur ligne c par le vecteur colonne x.
$x = (x_j)_{j=1,\dots,n}$	Vecteur colonne de variables de décision à n composantes $\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$.
c^T	Vecteur Transposé de c
$A = (a_i^T)_{i=1,\dots,m}$	La matrice de m lignes dont chaque ligne i est le vecteur a_i^T.
$\|x\|$	Norme du vecteur x.
$\|x\|_\infty$	Norme infinie du vecteur x.
$\lfloor x_j \rfloor$	Partie entière inférieure du scalaire x_j.
$\lceil x_j \rceil$	Partie entière supérieure du scalaire x_j.
$\mathbb{R}$	L'espace des nombres réels.
$\mathbb{Z}$	L'espace des nombres entiers.
$\|E\|$	La cardinalité de l'ensemble E.
$\cap$	L'intersection

Table des figures

Table des matières

Introduction générale

Contexte général de la thèse

L'optimisation est aussi ancienne que l'humanité sur Terre et tout le monde y est confronté tous les jours. L'optimisation consiste en le choix d'une alternative parmi d'autre dont la réalisation est la plus bénéfique. Dans de nombreux contextes décisionnels, un décideur fonde son choix sur plusieurs objectifs. Par exemple, lors de l'achat d'un produit quelconque, on cherche à minimiser le prix et à maximiser la qualité, lorsque on voyage dans un véhicule motorisé on cherche toujours à prendre le chemin le plus court et le plus rapide pour minimiser la consommation de carburant et la durée du voyage. Ce type de problèmes à objectif multiple conduit au domaine de la recherche de l'optimisation multiobjectif ou bien multicritère, vectorielle, etc...

Pour un problème d'optimisation multiobjectif non trivial, il n'existe pas de solution unique qui optimise simultanément chaque objectif, les objectifs étant antagonistes l'amélioration d'un objectif se fait au détriment d'un autre. Par exemple, un plan de production pourrait être d'un coût élevé et à faible risque, et un autre à coût faible mais à haut risque. F. Y Edgeworth est le premier à avoir examiné la prise de décision économique multiobjective et a envisagé qu'une alternative est candidate éligible s'il n'existait aucune autre alternative ou solution réalisable pour laquelle l'amélioration d'un objectif ne conduirait pas à une détérioration d'au moins un des autre restants [41]. Ces solutions sont qualifiées de points **efficients** ou efficaces ou bien solutions **"Pareto optimales"**.

L'ensemble de toutes les solutions efficientes est appelé l'ensemble de Pareto ou **ensemble efficient** et son image dans l'espace critère est appelé **front de Pareto**.

Très souvent les problèmes posés dans la réalité relèvent aussi du domaine de l'optimisation mathématique où la fonction-objectif est non-linéaire. Parmi les programmes non-linéaires, les programmes fractionnaires sont ceux dont les fonctions-objectifs sont

composées de fractions. Depuis la première problématique portant sur un modèle d'équilibre pour une économie en expansion introduit par J. Von Neumann [119], beaucoup de problèmes pratiques ont été modélisés à l'aide de programmes fractionnaires. Ainsi, plusieurs bibliographies assez exhaustives ([109], [97], [94], [6], etc...) ont répertorié de nombreuses applications de la programmation fractionnaire et des milliers de contributions à ce domaine. L'objet de cette thèse porte sur un problème fractionnaire multiobjectif. Puisque chaque solution efficiente d'un problème multiobjectif pourrait être intéressante pour un décideur, la question est de savoir comment pouvons-nous résoudre un problème fractionnaire à objectifs multiples ?

Une première approche de réponse consiste à agréger les différentes fonctions-objectifs en une fonction-objectif globale, puis à considérer le problème comme un problème d'optimisation à objectif unique. De nombreuses façons d'agréger les différentes fonctions-objectifs ont été étudiées et développées. Néanmoins, cette approche n'est pertinente que s'il est possible de définir une agrégation qui représente les préférences du décideur et s'il est alors possible d'optimiser la fonction agrégée. Le problème agrégé est généralement un programme dont la fonction-objectif est une somme de fractions. Hors selon l'article [99], l'un des programmes fractionnaires les plus difficiles à résoudre est le problème de somme de fractions, il s'avère être un problème NP-difficile malgré sa structure spéciale. En conséquence, outre le fait qu'il n'est pas facile de normaliser les préférences d'un décideur, cette approche conduit à l'un des programmes fractionnaires les plus difficiles à résoudre.

Une seconde approche consiste à prendre en compte chaque objectif et générer l'ensemble efficient ou le front de Pareto. Cela fournit une représentation des compromis entre les objectifs et peut aider le décideur à identifier des solutions d'intérêt. Cependant, la perspective multiobjective n'amoindrit pas les difficultés liées à chaque objectif pris individuellement. Bien au contraire, les problèmes multiobjectifs sont considérés comme étant plus difficiles que les problème mono-objective vu que plusieurs solutions sont Pareto optimales, aussi il convient de noter que pour les problèmes multiobjectif en variables continues l'ensemble efficient et le front de Pareto sont infinis (dans la pratique il vaut mieux modéliser via des variables à valeurs numériques discrètes ainsi l'ensemble efficient est dénombrable).

Motivation de la thèse

Les problèmes de programmation linéaire plus linéaire fractionnaire se distinguent des autres problèmes de la programmation fractionnaire par le fait qu'ils ont beaucoup d'applications pratiques surtout dans les domaines économique et financier, ils se posent notamment lorsqu'on doit optimiser le compromis entre un termes absolus (tel que le coût réel, le profit, la production,...) et un terme relatif (tel que le coût nominal, la profitabilité, la productivité, ...). Ce consensus de compromis apparaît dans de nombreux problèmes tels que les problèmes de sélection de portefeuilles, les problèmes de transport maritime, les problèmes d'optimisation des capitaux, les problèmes de développement des fonds, etc...

Toutefois peu de papiers proposent des procédures de résolution de ces problèmes car ils sont NP-difficile, principalement à cause du fait qu'ils ne sont pas convexes. Ils sont quasi-convexe lorsque le terme linéaire de la fonction-objectif est positif sur tout le domaine réalisable ou quasi-concave s'il est négatif [93], ils peuvent aussi être pseudo-convexe dans d'autre cas [74]. Actuellement, aucun article ne propose de procédure qui permet de trouver une seule solution efficiente d'un programme linéaire plus linéaire fractionnaire à objectif multiple. Toutes les méthodes actuelles étant des méthodes approchées, il est très motivent d'élaborer une technique capable d'énumérer exhaustivement toutes les solutions efficientes sans pour autant énumérer intégralement toutes les solutions entières du domaine.

Contribution de la thèse

L'objectif principale de cette thèse est d'apporter une contribution à un domaine assez peu explorer à cause de sa difficulté, ainsi le but fixé à travers ce travail est de trouver une méthode exacte d'optimisation d'un programme linéaire plus linéaire fractionnaire discret à objectif multiple. Par méthode exacte, on entend une méthode capable de générer l'ensemble de toutes les solutions efficientes. Ainsi, cette thèse comporte deux contributions à la programmation multiobjectif :

— La contribution principale de cette thèse est une méthode de résolution exacte de problème d'**optimisation linéaire plus linéaire fractionnaire discret à objectif multiple**. La littérature actuelle ne propose pas une méthode comparable

bien que le problème ait un réel intérêt pratique.

— La seconde contribution porte sur la définition d'un nouveau problème de la programmation multiobjectif permettant de générer une partie de l'ensemble efficient suivant les fonctions de préférences des décideurs. Puisque, la cardinalité de l'ensemble efficient est souvent très grande, il devient alors difficile pour le décideur de choisir la solution du meilleur compromis représentant ces préférences, aussi le temps de calcul de l'énumération de tous les membres de l'ensemble efficient augmente d'un manière exponentielle. Pour remédier à ces problèmes on propose la résolution d'un nouveau type de problème appelé l'"**optimisation multiobjectif sur l'ensemble efficient**" pour un programme multiobjectif linéaire discret.

L'organisation de la thèse

Cette thèse est organisée en deux parties :

— La première partie est réservée pour les préliminaires, elle comporte trois chapitres le premier sur l'optimisation, le deuxième sur l'optimisation fractionnaire et le troisième sur l'optimisation multiobjectif. Dans cette partie, serra exposé brièvement des concepts fondamentaux de l'optimisation, de l'optimisation fractionnaire et l'optimisation multiobjectif, qui sont essentiels à la compréhension du reste de cette thèse.

— La deuxième partie présente les deux contributions de cette thèse en deux chapitres. Le quatrième chapitre traite de l'optimisation multiobjectif sur l'ensemble efficient et le dernier de l'optimisation linéaires plus linéaires fractionnaires discrets à objectifs multiple.

Cette thèse se termine par une conclusion générale et quelques perspectives de recherche.

Première partie :

Préliminaires

Chapitre 1

L'optimisation

Sommaire

1.1 Introduction et revue historique

La recherche de l'optimalité est un principe fondamental qui s'établit dans les phénomènes naturels et régit les comportements biologiques et les activités sociales. L'optimisation a eu plusieurs périodes dans l'histoire, l'une de ses premières périodes a commencé en 1646 lorsque P. De Fermat a proposé, dans son article [46], une approche générale pour calculer les maxima / minima locaux d'une fonction différentiable, c'est-à-dire fixer la dérivée de la fonction à zéro.

La période de l'optimisation la plus importante est celle où G.B. Dantzig avait proposé pour la première fois la méthode du simplexe pour résoudre les problèmes de programmation linéaire en 1947, et déclara dans [32] :

> *"What seems to characterize the pre-1947 era was lack of any interests in trying to optimize".*

Ce qui semblait caractériser l'ère d'avant 1947, c'était l'absence de tout intérêt à essayer d'optimiser.

En raison du manque d'intérêt pour l'optimisation, de nombreux travaux importants apparus avant 1947 ont été ignorés[38]. La découverte de la programmation linéaire a ouvert une nouvelle ère pour l'optimisation.

DANTZIG a également mentionné dans ce même papier comment il avait fait sa découverte :

"My own contribution grew out of my World War II experience in the Pentagon. During the war period (1941-1945), I had become an expert on programming planning methods using desk calculators. In 1946 I was mathematical advisor to the US Air Force Comptroller in the Pentagon. I had just received my PhD (for research I had done mostly before the war) and was looking for an academic position that would pay better than a low offer I had received from Berkeley. In order to entice me to not take another job, my Pentagon colleagues, D. Hitchcock and M. Wood, challenged me to see what I could do to mechanize the planning process.

I was asked to find a way to more rapidly compute a time-staged development, training and logistical supply program. In those days mechanizing planning meant using analog devices or punch-card equipment. There were no electronic computers."

Ma propre contribution découle de mon expérience de la Seconde Guerre Mondiale au Pentagone. Pendant la période de guerre (1941-1945), j'étais devenu un expert des méthodes de planification-programmation à l'aide de calculatrices du bureau. En 1946, j'étais conseiller en mathématiques auprès du contrôleur de la US Air Force au Pentagone. Je venais juste d'obtenir mon PhD (pour des recherches que j'avais effectuées principalement avant la guerre) et je cherchais un poste universitaire qui rapporterait mieux qu'une mauvaise offre que j'avais reçu de Berkeley. Afin de m'inciter à ne pas occuper un autre emploi, mes collègues du Pentagone, D. HITCHCOCK *et* M. WOOD *m'ont mis au défi de voir ce que je pouvais faire pour mécaniser le processus de planification.*

On m'a demandé de trouver un moyen de concevoir rapidement une procédure itération-temps de développement, formation et d'approvisionnement logistique . À cette époque, la planification consistait à utiliser des appareils

analogiques ou des cartes à perforer. Il n'y avait pas d'ordinateurs électroniques.

En 1951, A.W. TUCKER et son élève H.W. KUHN ont publié les conditions de KUHN – TUCKER [69], ouvrant ainsi la première porte vers la programmation non-linéaire. Cependant, A. TAKAYAMA a fait le commentaire suivant :

"Linear programming aroused interest in constraints in the form of inequalities and in the theory of linear inequalities and convex sets. The Kuhn- Tucker study appeared in the middle of this interest with a full recognition of such developments.

However, the theory of nonlinear programming when constraints are all in the form of equalities has been known for a long time - in fact, since Euler and Lagrange. The inequality constraints were treated in a fairly satisfactory manner already in 1939 by Karush. Karush's work is apparently under the influence of a similar work in the calculus of variations by Valentine. Unfortunately, Karush's work has been largely ignored."

La programmation linéaire a suscité l'intérêt pour les contraintes sous la forme d'inégalités et pour la théorie des inégalités linéaires et des ensembles convexes. L'étude de Kuhn - Tucker est apparue au centre de cet intérêt et a une reconnaissance totale de ces développements.

Cependant, la théorie de la programmation non-linéaire lorsque toutes les contraintes sont sous forme d'égalités est en fait connue depuis fort longtemps, depuis EULER *et* LAGRANGE*. Les contraintes d'inégalité ont été traitées de manière assez satisfaisante dès 1939 par* KARUSH*. Le travail de* KARUSH *est apparemment fait sous l'influence d'un travail similaire sur le calcul des variations de* VALENTINE*. Malheureusement, le travail de* KARUSH *a été largement ignoré .*

En 1955, L.R. FORD et D.R. FULKERSON ont lancé l'étude sur les Flots dans les réseaux [47]. Ceci est considéré comme un point de départ pour l'optimisation combinatoire [60].

En 1958, R.E. GOMORY a publié un article sur la méthode de coupe pour la programmation en nombres entiers [54]. Ceci est considéré comme le point initial de la programmation en nombres entiers et une direction importante de l'optimisation combinatoire.

En 1961, ce qui est considéré par la littérature comme le premier papier sur l'optimisation fractionnaire est publié par A. Charnes et W.W. Cooper [25].

Aujourd'hui, l'optimisation a été intégrée dans presque tous les domaines et en particulier l'optimisation fractionnaire qui compte plusieurs applications pratiques. De nouvelles branches d'optimisation sont apparues comme l'optimisation globale, l'optimisation multiobjectif, l'optimisation non-différentielle, la programmation géométrique, optimisation à grande échelle, etc...

1.2 Généralités et concepts de base

Tous les problèmes d'optimisation ayant un objectif explicite peuvent généralement être exprimés sous forme d'un programme d'optimisation ou sous la forme générique suivante :

$$\begin{cases} minimiser/maximiser \quad f(x) \\ s.c. \\ x \in S \end{cases} \tag{1.1}$$

où $S = \{x \in \Omega | g_j(x) = 0,\ j = 1, \ldots, m.\ h_j(x) \geq 0,\ j = 1, \ldots, p\}$, $f(x), g_j(x)$ et $h_j(x)$ sont des fonctions scalaires du $n-$vecteur colonne réel x et $\Omega \subset \mathbb{R}^n$ est l'intersection des ensembles de définition de ces fonctions.

Ici, les composantes x_k de $x = (x_k)_{k=1,\ldots,n}^T$ sont appelées variables de conception ou plus souvent **variables de décision**. Elles peuvent être continues, discrètes ou mixtes.

La fonction $f(x)$ est appelée **fonction-objectif** ou fonction de coût. Les $g_j(x)$ sont les m contraintes en termes d'égalités, et les $h_j(x)$ sont les p contraintes en termes d'inégalités, il y a donc un total de $m + p$ contraintes.

L'espace couvert par les variables de décision s'appelle l'espace de recherche, tandis que l'espace formé par les valeurs de la fonction-objectif s'appelle l'espace des solutions.

Selon la nature des variables de décisions, de la fonction-objectif et des contraintes, on peut classer les différents problèmes d'optimisation.

Si la fonction objectif $f(x)$ est linéaire et les contraintes $g_j(x)$ et $h_j(x)$ sont toutes linéaires, cela devient un problème de programmation linéaire. En outre, si $f(x)$, $g_j(x)$ ou $h_j(x)$ n'est pas linéaire ça devient un problème de programmation non-linéaire.

Un problème de programmation linéaire dont les valeurs des variables de décision

sont en nombres entiers, est appelé programme linéaire en nombres entiers ou programme linéaire discret.

Ainsi un **programme fractionnaire** est un programme dont la fonction-objectif comporte une fraction de deux fonctions [48].

Il convient de souligner que le terme *programmation* désigne ici planification, cela n'a rien à voir avec la programmation informatique.

Dans tout ce qui suit on va considérer une fonction-objectif à "maximiser".

La programmation mathématique comprend de nombreux concepts, les concepts les plus fondamentaux sont ceux de la réalisabilité et de l'optimalité :

Définition 1.2.1. Solutions réalisables/admissible, Ensemble réalisable/admissible
Un point x qui satisfait toutes les contraintes est appelé un point réalisable ou admissible, dans ce cas $x \in S$.
L'ensemble de tous les points réalisables S s'appelle le domaine ou l'ensemble réalisable.

Définition 1.2.2. Maximum/Minimum local, Maximum/Minimum Global Un point x_* est appelé maximum local si $f(x)$ est défini dans un $\delta - voisinage$ [1] $V(x_*)$ et satisfait $f(x_*) \geq f(x)$ pour tout $x \in V(x_*)$.
Un point x^* est appelé maximum global (ou maximum) si pour toute solution réalisable $x \in S$ on a $f(x^*) \geq f(x)$.
Les minima peuvent être définis de la même manière en remplaçant le $\geq$ par $\leq$.

Dans certain cas, un optimum local coïncide forcement avec un optimum global c'est le cas lorsque le problème est **convexe**, les définitions suivantes sont tirées du livre [90] :

Définition 1.2.3. Ensemble convexe

Un sous ensemble S de $\mathbb{R}^n$ est dit convexe si et seulement si,

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

Définition 1.2.4. Combinaison convexe, enveloppe convexe

Soient $x^1, \ldots, x^r$, r points dans $\mathbb{R}^n$, une combinaison convexe des point $x^1, \ldots, x^r$ est

1. Dans un espace topologique, un voisinage d'un point est une partie de l'espace qui contient un ouvert qui comprend ce point

le point x de $\mathbb{R}^n$ tel que :

$$x = \sum_{i=1}^{r} \lambda_i x^i,\ \lambda_i \geq 0,\ \sum_{i=1}^{r} \lambda_i = 1$$

Étant donné $S \subseteq \mathbb{R}^n$, $Co(S)$ dénote l'enveloppe convexe de S, c'est-à-dire l'ensemble des points de $\mathbb{R}^n$ qui sont une combinaison convexe des points de S.

C'est le plus petit ensemble convexe qui contient S.

Définition 1.2.5. Fonction convexe, fonction concave

1. La fonction $f : S \to \mathbb{R}$ est dite convexe sur l'ensemble S convexe non-vide de $\mathbb{R}^n$ si et seulement si :

$$\forall x, y \in S, \forall \lambda \in [0,1] : f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y)$$

2. La fonction $f : S \to \mathbb{R}$ est dite concave sur l'ensemble S convexe non-vide de $\mathbb{R}^n$ si et seulement si :

$$\forall x, y \in S, \forall \lambda \in [0,1] : f(\lambda x + (1-\lambda)y) \geq \lambda f(x) + (1-\lambda) f(y)$$

Donc une fonction f est concave si et seulement si $(-f)$ est convexe.

Théorème 1.2.1. *1. Soit $f : S \to \mathbb{R}$ une fonction **convexe** sur l'ensemble S convexe non-vide de $\mathbb{R}^n$ alors tout minimum local de f est un miminum global de f.*

*2. Soit $f : S \to \mathbb{R}$ une fonction **concave** sur l'ensemble S convexe non-vide de $\mathbb{R}^n$ alors tout maximum local de f est un maximum global de f.*

Ainsi, on peut définir un problème convexe.

Définition 1.2.6. Problème d'optimisation convexe

On dit d'un problème d'optimisation qu'il est convexe s'il consiste à minimiser une fonction convexe (respectivement maximiser une fonction concave) sur un domaine convexe.

La convexité a des implications conséquentes sur les méthodes de résolutions, mais c'est une hypothèse assez forte d'où l'importance de d'autres concepts tels que la quasi-convexité, notamment dans l'optimisation non-linéaire.

Définition 1.2.7. Fonction quasi-convexe, Fonction quasi-concave

1. La fonction $f : S \to \mathbb{R}$ est dite quasi-convexe sur l'ensemble S convexe non-vide de $\mathbb{R}^n$ si et seulement si :

$$\forall x, y \in S, \forall \lambda \in [0,1] : f(\lambda x + (1-\lambda)y) \leq \max\{f(x), f(y)\}$$

2. La fonction $f : S \to \mathbb{R}$ est dite quasi-concave sur l'ensemble S convexe non-vide de $\mathbb{R}^n$ si et seulement si :

$$\forall x, y \in S, \forall \lambda \in [0,1] : f(\lambda x + (1-\lambda)y) \geq \min\{f(x), f(y)\}$$

Complexité algorithmique

L'*efficacité* d'un algorithme est souvent mesurée par la **complexité algorithmique** ou la complexité de calcul. Dans la littérature, cette complexité est également appelée complexité de Kolmogorov [118]. La complexité est donc un moyen de mesurer d'une façon générique et en fonction de la taille du problème le nombre d'étapes ou le temps de résolution d'un algorithme.

Un problème facile à résoudre est un problème dont la solution peut être obtenue par des algorithmes avec un temps de résolution (ou un nombre d'étapes) qui est fonction polynomiale de la taille du problème. Les algorithmes de résolution à temps polynomial sont considérés comme *efficaces*.

Un problème est dit **Polynomial (P)** si le nombre d'itération nécessaires pour trouver sa solution est délimitée par un polynôme par rapport à sa taille et il existe au moins un algorithme efficace pour le résoudre.

Par contre, un problème *difficile* nécessite un temps de résolution qui est une fonction exponentielle de la taille du problème, et donc les algorithmes à temps exponentiel sont considérés comme inefficaces.

Définition 1.2.8. Problème NP, Problème NP-complet,Problème NP-difficile

— Un problème appartient à la **classe non déterministe polynomial (NP)** si intuitivement on peut vérifier avec une « complexité polynomiale » si une alternative candidate est bien solution du dit problème mais il n'y a pas de règle avec une « complexité polynomiale » connue pour faire une telle supposition (non-déterministe).

— Un problème de la classe NP est dit **NP-complet** si tous les problèmes NP se ramènent à celui-ci via une réduction polynomiale.
— Un problème est **NP-difficile** s'il est au moins aussi difficile que n'importe quel problème de la classe NP.

La plupart des problèmes pratiques sont NP-difficile, et donc on ne connaît pas d'algorithme efficace pour les résoudre.

1.3 Programmation linéaire et algorithme du simplexe

La programmation linéaire est un domaine d'optimisation largement utilisé et très populaire. Le problème de la programmation linéaire a d'abord été démontré comme étant résoluble en temps polynomial par L.G. Khachiyan en 1979 [63] mais une percée théorique et pratique plus importante dans le domaine est venue en 1984 quand N. Karmarkar a introduit une nouvelle méthode de point intérieur pour résoudre des problèmes de programmation linéaire [62]. Toute fois, l'algorithme du simplexe introduit par G.B. Dantzig [33] est le plus populaire algorithme de résolution d'un programme linéaire.

Dans la pratique, l'algorithme du simplexe est très efficace et garantit de trouver l'optimum global si certaines précautions sont prises contre le bouclage de l'algorithme. C'est un algorithme facilement implémentable qui résout efficacement les problèmes aléatoires ainsi que les problèmes pratiques. Cependant, cet algorithme n'est pas de complexité **P** contrairement à l'algorithme de Karmarkar ou Khachiyan.

La compréhension des solutions algorithmiques utilisées en programmation linéaire nécessite l'introduction de notions de polyèdres, polytopes et de solutions de base [61] :

Définition 1.3.1. Hyperplan, Demi-plans fermés

Soit $\alpha \in \mathbb{R}^n$ non-nul et $\beta \in \mathbb{R}$, l'ensemble défini par :

$$H = \{x \in \mathbb{R}^n | \alpha x = \beta\}$$

est un hyperplan de $\mathbb{R}^n$.

On définit à partir de H les demi-plans fermés positif et négatif, donnés par :

$$H^- = \{x \in \mathbb{R}^n | \alpha x \leq \beta\}$$

$$H^+ = \{x \in \mathbb{R}^n | \alpha x \geq \beta\}$$

qui sont des espaces convexes.

Définition 1.3.2. Polyèdre, Ensemble borné

Un polyèdre $\mathcal{P} \subset \mathbb{R}^n$ est un ensemble décrit par

$$\mathcal{P} = \{x \in \mathbb{R}^n | Ax \leq b\}$$

ou $b \in \mathbb{R}^m$ et A est une $m \times n$ matrice.

Un ensemble $\mathcal{S} \subset \mathbb{R}^n$ est borné s'il existe une constante $\omega \in \mathbb{R}$ tel que $\|x\| \leq \omega$ pour tout $x \in \mathcal{S}$.

Un polyèdre est l'intersection d'un nombre fini de demi-plans.

Définition 1.3.3. Polytope

Un polytope est un polyèdre borné.

Définition 1.3.4. Sommet /Point extrême

Soit $\mathcal{P}$ est un polyèdre, $v \in \mathcal{P}$ est un point extrême ou un sommet de $\mathcal{P}$ si v ne peut être écrit comme la combinaison convexe stricte de deux points $x,\ y \in \mathcal{P}$ quelconque.

Considérant un polytope $\mathcal{P}$ définit par les contraintes d'inégalités :

$$\mathcal{P} = \{x \in \mathbb{R}^n | Ax \leq b\} \tag{1.2}$$

L'écriture standard du polyèdre $\mathcal{P}$ se fait en remplaçant les inégalités par des égalités en introduisant des variables d'écarts pour chaque ligne :

$$a_j x + e_j = b_j,\ e_j \in \mathbb{R}^+$$

et les composantes non-positive du vecteur x par deux autres composantes x'_k, x''_k dans $\mathbb{R}^+$ telles que $x_k = x'_k - x''_k$.

Donc sans perte de généralité chaque polyèdre peut être écrit sous la forme standard :

$$\mathcal{X} = \{x \in \mathbb{R}^n | Ax = b,\ x \geq 0\} \tag{1.3}$$

ou $A = (a_j^T)_{j=1,\ldots,m}$, $a_j = (a_j^k)_{k=1,\ldots,n}$ et $b = (b_j)_{j=1,\ldots,m}$.

Un Programme Linéaire (PL) est formulé sous la forme standard par :

$$(PL) \begin{cases} \max cx \\ s.c. \\ x \in \mathcal{X} \end{cases} \tag{1.4}$$

ou et $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ et A est une $m \times n-$matrice de rang m, x est le $n-$vecteur de décision.

L'ensemble réalisable $\mathcal{X}$ d'un PL est un polyèdre.

Définition 1.3.5. Base, solution de base, solution de base réalisable

Soit A une $m \times n-$matrice de plein rang [2]. Une base est une $m \times m-$matrice $\mathcal{B}$ extraite de A dont le déterminant est non nul, c'est une matrice carrée régulière de rang m dont les colonnes $j_1, j_2, \ldots, j_m$ tirées de A, sont linéairement indépendantes.

Soient les indices de colonnes $j_1 < j_2 < ... < j_m$ d'une base $\mathcal{B}$, la solution de base x est le vecteur unique satisfaisant $Ax = b$ sous la condition que toutes ses composantes $x_j = 0$ pour tout $j \notin \{j_1, j_2, \ldots, j_m\}$.

Si en plus d'être une solution de base x vérifie $x_j \geq 0,\ j = 1, \ldots, n$ alors x est dite de base réalisable

On peut donc adopté le partitionnement suivant :

$$A = (\mathcal{B}, \mathcal{N})$$

$$x = \begin{pmatrix} x_{\mathcal{B}} \\ x_{\mathcal{N}} \end{pmatrix}$$

$$c = (c_{\mathcal{B}}, c_{\mathcal{N}})$$

Donc si $x = (x_{\mathcal{B}}, 0)$ et $Ax = b$, x est dite solution de base. Les composantes de x associées aux colonnes de B s'appellent des **variables de base**. Si l'une des variables de base d'une solution de base est zéro, cette solution est appelée solution de base **dégénérée**.

2. Un matrice de plein rang est une matrice dont toutes les lignes sont linéairement indépendantes les une des autre.

Définition 1.3.6. Solutions adjacentes, Arête

Pour un polyèdre $\mathcal{X}$ donné, deux solutions de base distinctes sont dites adjacentes si elles ont en communs $m-1$ colonnes. Si deux solutions adjacentes sont également réalisables alors le segment liant ces deux sommets est une arête de l'ensemble réalisable.

Le théorème suivant identifie l'importance particulière des solutions de base réalisables.

Théorème 1.3.1. *Supposons que $\mathcal{X}$ n'est pas vide. Alors un point réalisable x est un sommet de $\mathcal{X}$ si et seulement si x est une solution réalisable de base.*

L'algorithme du simplexe consiste en l'exploration du domaine en passant d'une solution de base à une autre solution de base adjacente toute en améliorant le coût réduit définit ci-dessous :

Définition 1.3.7. Coût réduit

Soit x une solution de base, $\mathcal{B}$ la matrice de base associée et $c_{\mathcal{B}}$ les coefficients du vecteur de la fonction-objectif associé aux variables de base. Pour j donné, le coût réduit $\overline{c}_j$ de la variable x_j est défini par :

$$\overline{c}_j = c_j - c_{\mathcal{B}}\mathcal{B}^{-1}a^j$$

Théorème 1.3.2 (Le théorème fondamental de la programmation linéaire)**.** *Étant donné le problème de programmation linéaire (1.4), les affirmations suivantes sont vraies :*

1. *Si $\mathcal{X}$ n'est pas vide, il existe une solution de base réalisable.*
2. *Si (1.4) a une solution optimale, il existe une solution de base réalisable optimale.*

Par conséquent, un problème de programmation linéaire peut être résolu en cherchant parmi ses solutions de base réalisables (sommets de $\mathcal{X}$). La méthode du simplexe peut être décrite par les deux phases suivantes.

I Calcul d'un sommet initial x^0 de $\mathcal{X}$.

II Partant du sommet x^0, une suite de sommets $x^0, \ldots, x^s, \ldots, x^r$ est calculée de telle sorte que x_{s+1} soit adjacent à x_s, et telle que $cx^{s+1} > cx^s$. Le procédé se termine si le coût correspondant à une solution x^r ne peut pas augmenté.

Le passage d'un sommet à un autre adjacent est appelé **pivotage**. Le nombre entier r est le nombre de pivotage dans la méthode du simplexe.

1.4 Programmation linéaire en nombre entier

La partie de la programmation linéaire vue jusqu'ici est axée sur les problèmes de programmation continue, en ce sens que les variables de décision sont autorisées à être fractionnaires. C'est parfois une hypothèse réaliste, cependant dans la réalité les solutions fractionnaires ne sont pas pratique et nous devons considérer le Programme Linéaire en Nombre Entier (PLNE) :

$$(PLNE)\begin{cases} \max cx \\ s.c. \\ x \in \mathcal{X} \cap \mathbb{Z}^n \end{cases} \tag{1.5}$$

La contrainte qui force les variables à prendre des valeurs entières est appelée **contrainte d'intégrité**. Lorsqu'on ignore cette contrainte, on parle d'un problème **relaxé**, et l'on a alors affaire à un PL.

Bien que que la méthode de simplexe est efficace pour résoudre les PL, il n'existe pas de technique "efficace" pour résoudre des PLNE puisque le problème relaxé ne donne pas toujours une solution entière et son arrondissent au plus proche nombre entier peut ne pas être réalisable et n'est pas forcement un optimum. Au lieu de techniques efficaces pour résoudre un PLNE, un certain nombres de procédures ont été développées dont la performance est fortement dépendante du problème. Les méthodes les plus connues à ce jour sont :

1. les procédures par séparation et d'évaluation ou énumérations implicites [14],
2. et les méthodes des plans sécants ou de coupe ou bien de troncature [53].

En outre, des procédures composites de ces deux méthodes et plusieurs méthodes approchées ont été proposées par la littérature.

1.4.1 Procédure par séparation et évaluation

Les procédures par séparation et évaluation (Branch and Bound **B&B**) sont essentiellement une stratégie de «*diviser pour régner*». L'idée est de partitionner la région réalisable en sous-divisions plus gérables, puis, si nécessaire, de partitionner davantage les sous-divisions. En général, il existe un certain nombre de façons de diviser la région réalisable et par conséquent, un certain nombre d'algorithmes de **B&B**.

La représentation de l'exécution de la méthode de **B&B** se fait à travers une arborescence. Pour appliquer la méthode de **B&B** en générale, on doit être en possession :

1. d'un principe de séparation qui du fait de la structure du domaine $\mathcal{X}$ permet d'isoler des sous-domaines de plus en plus « réduits » en vue d'aboutir à des problèmes simples qu'on sait résoudre.
2. de règles d'évaluation et de sélection qui en fonction d'un ou de plusieurs critères servent après chaque séparation à éliminer certains des sous-domaines de $\mathcal{X}$ obtenus et à sélectionner parmi ceux qui restent celui sur lequel le principe de séparation doit à nouveau être appliqué.

Donc la relation de filiation de l'arbre lie un sous-domaine à ceux auxquels il donne naissance par séparation et pour développer cet arbre pour un PLNE il faut disposer des trois règles qui spécifient la procédure **B&B** :

Règle d'évaluation c'est un algorithme pour calculer une limite supérieure (pour un problème de maximisation) sur la solution optimale d'un sous-domaine.

Règle de séparation qui sépare les sous-domaines représentés par des nœuds en deux nœuds dont les domaines sont mutuellement exclusifs.

Règle de sélection de noœud ou stratégie de branchement c'est l'ordre dans lequel sont traité les nœuds ou les sous-problème associés. Les stratégies classiques consistent soit à parcourir l'arborescence des sous-problèmes en largeur en traitant le nœud le plus anciennement créé (breadth-first search), en profondeur en traitant le nœu d le plus récemment créé (deep-first search) ou en recherchant le sous-problème non traité de meilleur évaluation (best-first search).

En plus de la **règle de stérilisation** qui est commune à toutes les procédures B&B, ainsi un nœud n'a plus de descendance (feuille) si :

1. Le sous-problème associé est irréalisable.
2. L'évaluation du nœud est inférieure à la meilleure évaluation exacte actuelle (le coût de la meilleure solution réalisable trouvée jusqu'à présent).
3. L'évaluation du nœud est exacte.

L'algorithme 1 résume une procédure générale de B&B de résolution de PLNE.

Algorithme 1 : Algorithme de B&B pour résoudre un PLNE

1. **(Initialisation)** : Poser $l = 0$, $\mathcal{X}_l = \mathcal{X}$, $Z_{PLNE} = -\infty$, et l'ensemble des solutions optimales $\mathbb{L} = \emptyset$, aller en **(3)**.
2. **(Sélection du nœud)** Sélectionner un nœud l pas encore sondé selon une règle de sélection préétablie et aller en **(4)**, s'il n y a aucun nœud non-sondé aller en **(6)**.
3. **(Évaluation)** Résoudre le PL :

$$(P_l) \begin{cases} \max cx \\ s.c. \\ x \in \mathcal{X}_l \end{cases}$$

   Si (P_l) n'a pas de solution aller en **(4)**, sinon soit x_l^* sa solution optimale.
   Si $cx_l^* < Z_{PLNE}$ aller en **(4)**, sinon :
   — si x_l^* entière,
      — si $cx_l^* > Z_{PLNE}$ poser $Z_{PLNE} = c^T x_l^*$ et $\mathbb{L} = \{x^*\}$ et aller en **(4)**,
      — si $cx_l^* = Z_{PLNE}$ poser $\mathbb{L} = \mathbb{L} \cup \{x^*\}$ et aller en **(4)**,
   — sinon aller en **(5)**.
4. **(Stérilisation)** Sondé le nœud et aller en **(2)**.
5. **(Séparation)** Soit x_l^* la solution non-entière associée au nœud l, choisir $j \in \{1, 2, \ldots, n\}$ un indice tel que α_j la $j-$ème composante de x_l^* est fractionnaire et crée les deux nœud fils de l : $l' \geq l+1$ où $\mathcal{X}_{l'} = \mathcal{X}_l \cap \{x \leq \lfloor \alpha_j \rfloor\}$ et $l'' > l+1$ où $\mathcal{X}_{l''} = \mathcal{X}_l \cap \{x \geq \lceil \alpha_j \rceil\}$. Aller en **(2)**.
6. **(Arrêts)** Si $\mathbb{L} = \emptyset$ alors il n y a pas de solution optimale de (1.5), sinon $\mathbb{L}$ contient toutes les solutions optimales de (1.5) qui ont un coût optimal de Z_{PLNE}

1.4.2 Méthode des plans sécants, de troncature ou de coupe

Les méthodes des plans sécants, de troncature ou de coupe fonctionnent en résolvant une séquence de PLs qui sont des relaxations de PLNE. Les relaxations sont progressivement améliorées pour donner de meilleures approximations au PLNE. Ces méthodes ne divisent pas la région réalisable en sous-domaines, comme dans les approches **B&B**, mais utilisent plutôt un programme linéaire unique qui s'affine en ajoutant de nouvelles contraintes. Les nouvelles contraintes réduisent successivement la région réalisable jusqu'à ce qu'une solution entière optimale soit trouvée à terme.

La procédure générale d'une méthode de plans sécants est résumée par l'algorithme

2.

Algorithme 2 : Algorithme général des techniques de plans sécants

1. Résoudre une relaxation de PLNE en continu. Si la solution est entière, arrêter c'est la solution optimale. Sinon aller en **(2)**.
2. Ajouter une contrainte permettant au moins d'élaguer la solution courante et aller en **(3)**.
3. Résoudre le problème courant si sa solution est entière arrêter c'est la solution optimale. Sinon aller en **(2)**.

Le concept d'ajout de contraintes d'inégalité linéaire au problème de programmation linéaire a été introduit pour la première fois par Dantzig, Fulkerson et Johnson [34] mais c'est Gomory [54] qui a fournit le premier algorithme de plans sécants générant systématiquement des troncatures pouvant être appliquées aux PLNE.

Définition 1.4.1. inégalité valide, coupe valide

Étant donné le PLNE (1.5), on dit que l'inéquation $\alpha x \leq \beta$ est valide si elle est satisfaite par toute solution réalisable de (1.5).

Une coupe est une inégalité valide qui n'est pas satisfaite pour tout point du domaine du problème (1.4) le relaxé de (1.5).

1.4.2.1 Coupe de Dantzig

La coupe de Dantzig [34] est une méthode intuitive d'élaguer les solutions non-entières du domaine, elle consiste à rajouter une contrainte qui oblige la somme des variables non-basiques à être supérieure à 1, ainsi lors de la prochaine résolution la solution optimale changera de base et la solution non-entière précédente sera élaguée. La coupe de Dantzig s'écrit comme suit :

$$\sum_{j \in \mathcal{N}} x_j \geq 1 \tag{1.6}$$

où $\mathcal{N}$ désigne l'ensemble des indices hors-base. Cependant, comme le montre Gomory [53], l'utilisation de la coupe de Dantzig ne garantit pas la convergence de la procédure, c'est-à-dire qu'il n'est pas nécessaire que dans un nombre fini d'étapes la solution entière optimale soit atteinte.

1.4.2.2 Coupe fractionnaire de Gomory

Les coupes fractionnaires de Gomory sont générées en appliquant un arrondi à l'entier le plus proche sur une ligne pivot, à une variable de base dont la valeur est fractionnaire. La méthode du simplexe de résolution d'un PL produit un ensemble d'équations de la forme :

$$x_i + \sum_{j \in \mathcal{N}} \bar{a}_i^j x_j = \bar{b}_i, \ i = 1, \ldots, m$$

où x_i est une variable de base, réécrivant cette équation de façon à ce que les parties entières soient à gauche et les parties fractionnaires à droite.

$$x_i + \sum_{j \in \mathcal{N}} \lfloor \bar{a}_i^j \rfloor x_j - \lfloor \bar{b}_i \rfloor = \bar{b}_i - \lfloor \bar{b}_i \rfloor - \sum_{j \in \mathcal{N}} \left(\bar{a}_i^j - \lfloor \bar{a}_i^j \rfloor \right) x_j.$$

Pour tout point entier de la région réalisable, le côté droit de cette équation est inférieur à 1 et le côté gauche est un entier. Donc,

$$\bar{b}_i - \lfloor \bar{b}_i \rfloor - \sum_{j \in \mathcal{N}} \left(\bar{a}_i^j - \lfloor \bar{a}_i^j \rfloor \right) x_j \leq 0 \tag{1.7}$$

est vrai pour n'importe quel point entier dans la région réalisable (du PLNE).

De plus, les variables non-basiques sont égales à 0 et si la variable de base x_i n'est pas un entier pour la solution de base x,

$$\bar{b}_i - \lfloor \bar{b}_i \rfloor - \sum_{j \in \mathcal{N}} \left(\bar{a}_i^j - \lfloor \bar{a}_i^j \rfloor \right) x_j = \bar{b}_i - \lfloor \bar{b}_i \rfloor > 0.$$

Ainsi, l'inégalité 1.7 ci-dessus exclut la solution de base réalisable fractionnaire et est donc une coupe valide.

En introduisant une nouvelle variable d'écart x_k pour cette inégalité, une nouvelle contrainte est ajoutée au programme linéaire, à savoir :

$$x_k + \sum_{j \in \mathcal{N}} \left(\lfloor \bar{a}_{i,j} \rfloor - \bar{a}_{i,j} \right) x_j = \lfloor \bar{b}_i \rfloor - \bar{b}_i, \ x_k \geq 0.$$

1.4.3 Technique composite : le Branch and Cut

La coupe fractionnaire de GOMORY est rapide, mais peu fiable, la procédure de **B&B** est fiable mais lente. La procédure Branch and Cut (**B&C**) combine les avantages de ces deux méthodes.

Les méthodes de **B&C** pour résoudre les PLNE proposent une stratégie d'alternance de techniques de coupe et de règle de séparation ainsi l'algorithme 3 résume en générale une procédure de **B&C** de résolution d'un PLNE :

Algorithme 3 : Algorithme de **B&C** de résolution d'un PLNE

1. **(Initialisation)** : Poser $l = 0$, $\mathcal{X}_l = \mathcal{X}$, $Z_{PLNE} = -\infty$, et l'ensemble des solutions optimales $\mathbb{L} = \emptyset$
2. **(Sélection du nœud)** Sélectionner un nœud l pas encore sondé selon une règle de sélection préétablie et aller en **(3)** s'il n y a aucun nœud non-sondé aller en **(6)**.
3. **(Évaluation)** Résoudre le PL :

$$(P_l) \begin{cases} \max cx \\ s.c. \\ x \in \mathcal{X}_l \end{cases}$$

   Si (P_l) n'a pas de solution aller en **(4)** sinon soit x_l^* sa solution optimale :
   Si $cx^* < Z_{PLNE}$ aller en **(4)**, sinon :
   — si x_l^* entière,
      — si $cx_l^* > Z_{PLNE}$ poser $Z_{PLNE} = cx_l^*$, $\mathbb{L} = \{x^*\}$ et aller à **(4)**,
      — si $cx_l^* = Z_{PLNE}$ poser $\mathbb{L} = \mathbb{L} \cup \{x^*\}$ et aller à **(4)**,
   — sinon aller en **(5)** ou en **(6)**.
4. **(Stérilisation)** Sondé le nœud et aller en **(2)**.
5. **(La coupe)** Ajouté une coupe valide et aller en **(3)**.
6. **(Séparation)** Soit x_l^* la solution non-entière associée au nœud l, choisir $j \in \{1, 2, \ldots, n\}$ un indice tel que α_j la composante de x_l^* est fractionnaire et crée les deux nœud fils de l : $l' = l + 1$ où $\mathcal{X}_{l'} = \mathcal{X}_l \cap \{x \leq \lfloor \alpha_j \rfloor\}$ et $l'' = l + 2$ où $\mathcal{X}_{l''} = \mathcal{X}_l \cap \{x \leq \lceil \alpha_j \rceil\}$. Aller en **(2)**.
7. **(Arrêts)** Si $\mathbb{L} = \emptyset$ alors il n y a pas de solution optimale au problème (1.5), sinon $\mathbb{L}$ contient toutes les solutions optimales de (1.5) qui ont un coût de Z_{PLNE}

1.5 Conclusion

Ce chapitre a servi d'introduction générale aux concepts de base de l'optimisation nécéssaire à la compréhension des contributions de cette thèse. Les notions d'algorithme de simplexe, d'exploration implicite et de **B&C** sont des concepts fondamentaux qui sont utilisés dans les algorithmes de résolution de PLNE et de programme en nombre entier en général.

Chapitre 2

Optimisation Fractionnaire

Sommaire

2.1 Introduction

Dans plusieurs modèles de la programmation non-linéaire la fonction-objectif est sous forme d'une ou plusieurs fractions de fonctions linéaires ou non-linéaires. Ces problèmes d'optimisation sont plus communément appelés des programmes fractionnaires. L'intérêt de la programmation fractionnaire réside dans le fait que divers problèmes d'optimisation de l'ingénierie et de l'économie considèrent la minimisation ou la maximisation d'un rapport entre deux fonctions qui décrivent des propriétés physiques et / ou économiques comme par exemple le rapport du coût sur le temps, ou bien du coût sur le volume [122]. De ce fait la programmation fractionnaire possède plusieurs applications pratiques dans

l'industrie [7, 18], la théorie de l'information [3], le traitement optique de l'information [45], l'analyse numérique [89], la programmation stochastique [13], les algorithmes de décomposition des grands systèmes linéaires [10], les problèmes de décision des chaînes de Markov [36], les problèmes de transport et de transport maritime [4, 16], les problèmes de la théorie des jeux [100], etc...

L'optimisation fractionnaire a reçu une attention particulière au cours des trois dernières décennies. Cette attention est particulièrement visible dans les travaux de I. M. Stancu-Minasian [105], [104], [108], [107], [103], [106] et le livre du même auteur [110] qui sont des recueils de bibliographie sur la programmation fractionnaire.

Les programmes fractionnaires ont été classé par J. B. G. Frenk et S. Schaible [49] en deux classes selon la forme de la fonction-objectif, ainsi ils distinguent les programmes monofractionnaires et les programmes multifractionnaires. Les programmes multifractionnaires sont aussi divisibles en deux catégories, les programmes fractionnaires min-max et les programmes de somme de fractions.

2.2 Programmes monofractionnaires

2.2.1 Formulation et applications

Étant donnée f, g des fonctions réelles définies de $\mathbb{R}^n$ dans $\mathbb{R}$, avec g ne s'annulant pas sur le sous-ensemble S de $\mathbb{R}^n$, un problème de programmation monofractionnaire (P_F) est formulé d'une manière générique par :

$$(P_F)\begin{cases} \max \dfrac{f(x)}{g(x)} \\ s.c. \\ x \in S \end{cases}$$

Si S est un polyèdre de $\mathbb{R}^n$ et f, g sont des fonctions affines ou linéaires alors le problème d'optimisation (P_F) est appelé programme linéaire fractionnaire. (P_F) est appelé programme quadratique fractionnaire si pour un polyèdre S les fonctions f et g sont quadratiques.

Les programmes monofractionnaires ont plusieurs applications pratiques, ils sont particulièrement utiles pour la modélisation des problèmes économiques tels que le problème

de réduction des stocks [8], d'affectation du personnel [56], d'allocation de ressources [80], de sélection de portefeuille et de planning financier [97], etc... Ils sont aussi utilisés pour modéliser des problèmes non-économiques tels que la planification des soins de santé et des hôpitaux [110] et les processus physiques [45]. Beaucoup de recueils bibliographiques sont dédiés des applications des programmes monofractionnaires [30][95][109]...

2.2.2 Résultats théoriques et algorithmes

Les programmes fractionnaires ont souvent été étudiés dans le contexte de la convexité généralisée [1], puisqu'en général les rapports des fonctions convexes et concaves ainsi que les composites de ces rapports ne sont pas convexes même dans le cas de rapports linéaires. Mais ils sont souvent convexes-généralisés [5]. En conséquence, la plupart des algorithmes proposent de résoudre des problèmes de programmation linéaire fractionnaire ou plus généralement de programmation fractionnaire concave [2]. Il y a cinq stratégies différentes de résolution [48] :

Résolution directe Lorsque la fonction-objectif possède certaines propriétés liées à la convexité ou à la convexité généralisée le problème peut être résolu par une procédure du simplexe.

Résolution par un problème équivalant Une transformation a été suggérée par A. Charnes et W. W. Cooper [25] pour les programmes linéaires fractionnaires mais est tout aussi valable pour les programmes fractionnaires concaves [48]. Cette transformation consiste au changement de variables suivant :

$$y = \frac{x}{g(x)},\ t = \frac{1}{g(x)}$$

Résolution par un programme paramétrique L'une des stratégies les plus populaire pour la résolution des programmes linéaires et non-linéaires fractionnaires est l'approche paramétrique décrite par W. Dinkelbach dans [37]. Dans le cas de la programmation linéaire fractionnaire cette méthode permet de réduire la résolution du problème à la résolution d'une suite de problèmes de programmation

1. Pseudo-convexité/concavité, quasi-convexité/concavité, etc...
2. Le programme (P_F) est concave si le numérateur f est concave sur S et g est convexe sur S et l'ensemble S est convexe.

linéaire. Considérant le problème (P_F) et la fonction :

$$F(\lambda) = \max_{x \in S} \{f(x) - \lambda(g(x))\} \ \ \lambda \in \mathbb{R}$$

La méthode de W. Dinkelbach est résumé par l'algorithme :

Algorithme 4 : Dinkelbach

Etape 0 Soit $x^{(0)} \in S$, calculer $\lambda^{(0)} = \dfrac{f(x^{(0)})}{g(x^{(0)})}$ et poser $k := 1$, aller à l'étape 1 ;

Etape 1 Déterminer $x^{(k)} := argmax_{x \in S} \left\{ f(x) - \lambda^{(k)} g(x) \right\}$, aller à l'étape 2 ;

Etape 2 Si $F(\lambda^{(k)}) = 0$ alors $x^* = x^{(k)}$ est une solution optimale, stop. Sinon aller à l'étape 3 ;

Étape 3 Soit $\lambda^{(k+1)} = \dfrac{f(x^{(k)})}{g(x^{(k)})}$; Poser $k := k + 1$; Aller a l'étape 1 ;

Résolution par des algorithmes du point intérieur En plus des quatre autres stratégies classiques ci-dessus, des stratégie du point intérieur on été utilisé pour résoudre des programmes fractionnaires non-linéaires [51, 50].

2.2.3 Cas particulier : Programme linéaire fractionnaire

En 1962, A. Charnes et W. W. Cooper [25] ont publié leur article classique dans lequel ils montrent qu'un programme linéaire fractionnaire peut être réduit à un programme linéaire équivalant en utilisant une transformation non-linéaire de variable.

Séparément, en 1964 le mathématicien hongrois B. Martos [75] a considéré le même problème qu'il nomma programme hyperbolique et a montré que ce programme peut être résolu avec une procédure de simplexe. Ceci est rendu possible grâce à la pseudo-linéarité de la fonction linéaire fractionnaire.

Un programme linéaire fractionnaire a la formulation générique suivante :

$$(P_{LF}) \begin{cases} \max Q(x) = \dfrac{cx + \alpha}{dx + \beta} \\ s.c. \\ x \in \mathcal{X} \end{cases}$$

ou

$$\mathcal{X} = \{x \in \mathbb{R}^n : Ax \leq b, x \geq 0\}$$

Avec $A \in \mathbb{R}^n \times \mathbb{R}^m$ supposée de plein rang, c et d des n-vecteurs, b un m-vecteur, α et β des scalaires et x le n-vecteur de décision.

Les principales stratégies de résolution des programmes linéaires fractionnaires se distinguent en deux :

La résolution directe Les programmes linéaires fractionnaires étant pseudo-linéaires il est possible d'appliquer des procédures du simplexe pour les résoudre comme cela a été fait dans [75] et [19].

La résolution d'un programme équivalant La méthode la plus populaire est celle de A. Charnes et W.W. Cooper [25]. Dans l'article [115], S.F. Tantawy propose une nouvelle approche pour transformer un problème linéaire fractionnaire en un problème linéaire.

Considérant le problème (P_{LF}) en supposant que $\beta \neq 0$, on a :

$$\begin{aligned} Q(x) &= \frac{cx+\alpha}{dx+\beta} = \frac{cx + \frac{\alpha}{\beta}dx - \frac{\alpha}{\beta}dx + \alpha}{dx+\beta} \\ &= \frac{\left(c - \frac{\alpha}{\beta}d\right)x + \frac{\alpha}{\beta}(dx+\beta)}{dx+\beta} \\ &= \left(c - \frac{\alpha}{\beta}d\right)\frac{x}{dx+\beta} + \frac{\alpha}{\beta} \end{aligned}$$

Et puisque $dx + \beta > 0$ on a :

$$\begin{aligned} & Ax \leq b && \times\frac{\beta}{\beta} \\ \Leftrightarrow\ & \frac{1}{\beta}(\beta Ax - \beta b) \leq 0 && \times\frac{1}{dx+\beta} \\ \Leftrightarrow\ & \frac{1}{\beta}\left(\frac{\beta Ax - \beta b}{dx+\beta}\right) \leq 0 \\ \Leftrightarrow\ & \frac{1}{\beta}\left(\frac{(\beta A + bd)x - bdx - \beta b}{dx+\beta}\right) \leq 0 \\ \Leftrightarrow\ & \frac{1}{\beta}\left(\frac{(\beta A + bd)x - b(dx+\beta)}{dx+\beta}\right) \leq 0 \end{aligned}$$

$$\Leftrightarrow \quad \frac{1}{\beta}\left(\frac{(\beta A + bd)x}{dx+\beta} - b\right) \leq 0$$
$$\Leftrightarrow \quad \left(A + b\frac{d}{\beta}\right)\frac{x}{dx+\beta} \leq \frac{b}{\beta}$$

Alors on a :

$$(P_eq)\begin{cases} \max Q(y) = py + \dfrac{\alpha}{\beta} \\ s.c. \\ A'y \leq b' \\ y \geq 0 \end{cases}$$

avec $y = \dfrac{x}{dx+\beta}$, $A' = \left(A + b\dfrac{d}{\beta}\right)$, $b' = \dfrac{b}{\beta}$ et $p = \left(c - \dfrac{\alpha}{\beta}d\right)$.
Le programme (P_{eq}) est un programme linéaire équivalant de (P_{LF}).

2.3 Programmes multifractionnaires

2.3.1 Formulation et applications

Il y a deux types de programmes fractionnaires à fraction multiple : les programmes de somme de fractions et les programmes fractionnaires min-max.

$$(P_{F_S})\begin{cases} \max \sum\limits_{i=1}^{r} \dfrac{f_i(x)}{g_i(x)} \\ s.c. \\ x \in S \end{cases} \quad et \quad (P_{min-max})\begin{cases} \max \left\{ \min\limits_{1\leq i\leq r} \dfrac{f_i(x)}{g_i(x)} \right\} \\ s.c. \\ x \in S \end{cases}$$

Le type de programme $(P_{min-max})$ apparaît en économie lorsque la croissance d'une économie en expansion est à déterminée [112] ou dans les problèmes de planning financier [52] ou bien dans l'ingénierie [55].

Les modèles du type (P_{F_S}) se posent naturellement lorsque plusieurs taux doivent être optimisés simultanément et un compromis qui optimise une somme pondérée de ces taux est recherché. Des applications de ce modèle sont étudiés dans [96, 98].

2.3.2 Résultats théoriques et algorithmes

Pour ce qui est de la résolution de ($P_{min-max}$), une extension de l'algorithme de Dinkelbach a été introduite dans le document [31], d'autres méthodes ont été décrites dans [31, 55].

La somme de fractions est un problème plus difficile car même si toutes les fractions sont des rapports de fonction concave/convexe ceci n'implique pas que cette somme possède des propriétés de convexité ou de convexité généralisée. Ainsi, en général les problèmes de somme de fractions ne sont pas convexes/concaves ni quasi-convexes/quasi-concaves. Dans [51], R. W. Freund et F. Jarre ont montré que ce problème est NP-difficile. De ce fait, les algorithmes qui ont abordé ce problème se limitent à moins de trois fractions [66][20][70].

2.3.3 Cas particulier : Programme linéaire plus linéaire fractionnaire

Considérant le programme :

$$(PLLF)\begin{cases} \max hx + \dfrac{cx+\alpha}{dx+\beta} \\ s.c. \\ x \in \mathcal{X} \end{cases} \tag{2.1}$$

Ce programme dit linéaire plus linéaire fractionnaire est un cas spéciale du programme de somme de deux fractions ou le dénominateur de l'un des deux fonctions est une constante. Il a une considération particulière car il peut modéliser beaucoup de cas pratique où un terme relatif plus un terme absolu sont à optimiser tel que le profit et la profitabilité [117] et de ce fait beaucoup de modèles en économie s'optimisent par la fonction linéaire plus linéaire fractionnaire [67][79][64]...

S. Schaible a étudié ce problème en terme de quasi-convexité et de quasi-concavité, il démontra que la fonction-objectif est quasi-convexe lorsque le terme linéaire de la somme est positif sur le domaine et quasi-concave s'il est négatif. De ce fait peu de méthodes de résolution sont proposées pour résoudre ce programme.

Considérant le partitionnement suivant :

$$A = (\mathcal{B}, \mathcal{N})$$

$$c = (c_{\mathcal{B}}, c_{\mathcal{N}})$$

$$d = (d_{\mathcal{B}}, d_{\mathcal{N}})$$

$$h = (h_{\mathcal{B}}, h_{\mathcal{N}})$$

$$x = \begin{pmatrix} x_{\mathcal{B}} \\ x_{\mathcal{N}} \end{pmatrix}$$

Dans la base $\mathcal{B}$ on a :

$$\bar{h} = h + h_{\mathcal{B}}\mathcal{B}^{-1}A,\ \bar{c} = c + c_{\mathcal{B}}\mathcal{B}^{-1}A,\ \bar{d} = d + d_{\mathcal{B}}\mathcal{B}^{-1}A$$

$$\bar{h}_0 = hx,\ \bar{d}_0 = dx + \beta,\ \bar{c}_0 = cx + \alpha.$$

et on note :

$$\gamma = \bar{\beta}^2\bar{h}_{\mathcal{N}} + \bar{\beta}\bar{c}_{\mathcal{N}} - \bar{\alpha}\bar{d}_{\mathcal{N}}$$

2.3.3.1 Méthode de résolution de A. Cambini *et al.*

Dans l'article [20], A. Cambini et al. ont considéré un programme de somme de deux fractions et l'ont réduit au programme $(PLLF)$ en utilisant la transformation de Charnes-Cooper [25], ils ont proposé ensuite une méthode de résolution itérative. Cette méthode est une procédure du simplexe basée sur la notion de solution niveau-optimale. Considérant le problème $(PLLF)$, l'idée de l'algorithme est de trouver un point maximum local en générant une séquence finie de solutions niveau-optimale.

Théorème 2.3.1. *Soit x^0 une solution de $\mathcal{X}$ alors x^0 est un optimum local du programme (2.1) si $\gamma > 0$.*

Pour la démonstration, il suffit de calculer la dérivée de la fonction $\bar{h}_{\mathcal{N}}x_{\mathcal{N}} + \dfrac{\bar{c}_{\mathcal{N}}x_{\mathcal{N}} + \bar{\alpha}}{\bar{d}_{\mathcal{N}}x_{\mathcal{N}} + \bar{\beta}}$.

Une solution réalisable $\hat{x}$ est dite solution niveau-optimale pour le problème $(PLLF)$

si $\hat{x}$ est une solution optimale pour le programme linéaire suivant :

$$(P_\xi)\begin{cases} \max \xi hx + cx + \alpha \\ s.c. \\ x \in \mathcal{X} \\ dx + \beta = \xi \end{cases}$$

Si de plus $\hat{x}$ est un point extrême de $\mathcal{X}$, alors $\hat{x}$ est dit solution-niveau optimale de base.

2.3.3.2 Méthode de résolution de A. Fakhri & M. Ghatee

Dans l'article [44], A. Fakhri et M. Ghatee tentent de minimiser la somme d'une fonction linéaire et d'une fonction fractionnaire linéaire sur un ensemble convexe fermé défini par certaines contraintes quadratiques linéaires et coniques. Ils présentent quelques conditions nécessaires et suffisantes pour la pseudo-convexité du problème et imposent ces conditions pour la résolution.

Proposition 2.3.2. *La fonction objectif du le problème (2.1) est pseudo-convexe sur $\mathcal{X}$ si et seulement si une des conditions suivantes est satisfaite :*

1. *il existe $a \geq 0$ tel que $h = ad$;*
2. *il existe $k \in \mathbb{R}$ tel que $c = kd$ et $\alpha - k\beta \geq 0$.*

Pour chacune de ces conditions et sous certaines hypothèses une reformulation en programmation conique de second ordre du problème est énoncée et une nouvelle procédure de solution applicable est proposée.

2.4 Programmes fractionnaires multiobjectifs

Le problème de maximisation simultané de plusieurs fractions conduit à un programme fractionnaire multiobjectif $(PFMO)$ qui est formulé comme suit :

$$(PFMO)\begin{cases} \max Q_i(x) = \dfrac{f_i(x)}{g_i(x)}, \quad i \in \{1, \ldots, r\} \\ s.c. \\ x \in S \end{cases}$$

Où $g_i(x) > 0 \ \forall x \in S, \ \forall i \in \{1, \ldots, r\}$.

Dans [15], les auteurs ont classifié les problèmes de programmation fractionnaire multiobjectif en neuf catégories que voici :

1. Programmation fractionnaire multiobjectif général ;
2. Programmation linéaire fractionnaire multiobjectif ;
3. Programmation linéaire fractionnaire stochastique multiobjectifs ;
4. Programmation linéaire fractionnaire multiobjectifs bi-niveaux ;
5. Programmation linéaire fractionnaire multiobjectifs avec paramètres flous ;
6. **Programmation linéaire plus linéaire fractionnaire multiobjectif** ;
7. Programmation linéaire fractionnaire multiobjectif mixte ;
8. Programmation linéaire fractionnaire min-max multiobjectif ;
9. Programmation linéaire fractionnaire robuste min-max multiobjectif.

En général, les problèmes de PFMO sont utilisés pour modéliser des rapports tirés des problèmes du monde réel tels que les stocks/ventes, les bénéfices/coûts, les dettes/investissement et autres. Ainsi, les PFMO se posent notamment lors de la planification financière ou dans les problèmes de minimisation de risque et dans plein d'autres domaines [68] [97][112][111] ...

Les articles de la littérature ont traité les PFMO de différentes perspectives, ainsi on trouve ceux qui abordent la dualité des programmes fractionnaires multiobjectifs, généralement pour des fractions concaves ou linéaires [9][109]... ; ou bien ceux qui proposent des méthodes approchées de résolution souvent pour les problèmes linéaires fractionnaires flous [57][58][83]...

En contre partie, on trouve les méthodes exactes telles que celle de M. Abbas et M. Moulaï [2] ou la méthode de M.E-A Chergui et M. Moulaï [27] qui sont des procédures de **B&C** utilisant un même schéma d'exploration mais différentes méthodes de coupes.

2.5 Conclusion

La principale difficulté lors de la résolution d'un programme fractionnaire c'est la non-convexité du problème. Le problèmes de programmation linéaire fractionnaire étant

pseudo-linéaire la littérature dispose de plusieurs approches de résolution exactes, même dans le cas multiobjectif et/ou discret. A contrario, les problème de programmation linéaires plus linéaires fractionnaires ne possèdent aucune propriété de convexité dans le cas générale, ainsi toutes les approches de la littératures sont des méthodes approchées ou des méthodes spécifiques à certain cas de pseudo-convexité.

Chapitre 3

Optimisation Multiobjectif

Sommaire

3.1 Point historique

En 1881, le professeur en économie F.Y. Edgeworth fut le premier à définir un optimum pour la prise de décision économique multicritères [41], il le fit pour le problème

multiutilé :

> *"It is required to find a point (x,y) such that in whatever direction we take an infinitely small step, P and π do not increase together but that, while one increases, the other decreases."*
>
> *Il est nécessaire de trouver un point (x,y) tel que dans quelle que soit la direction dans laquelle nous faisons un pas infiniment petit, P et π n'augmentent pas ensemble mais quant l'un augmente, l'autre diminue.*

Pareillement, V. Pareto était un ingénieur civil contemporain d'Edgeworth qui fut le créateur de la théorie de l'optimum de Pareto [84] :

> *"The optimum allocation of the resources of a society is not attained so long as it is possible to make at least one individual better off in his own estimation while keeping others as well off as before in their own estimation."*
>
> *L'allocation optimale des ressources d'une société n'est pas atteinte tant qu'il est possible d'améliorer la situation d'au moins un individu dans sa propre estimation tout en maintenant les autres aussi bien qu'avant dans leur propre estimation.*

De ce fait F.Y. Edgeworth et V. Pareto sont crédités pour avoir introduit pour la première fois le concept de non-infériorité dans un contexte économique. La traduction des travaux de V. Pareto en 1971 a stimulé le développement de méthodes multiobjectives en mathématiques appliquées et en ingénierie. La croissance de ce domaine s'est manifestée particulièrement après les travaux pionniers de W. Stadler [102], et R. Steuer [113].

3.2 Concepts fondamentaux

La formulation générique d'un problème multiobjectif est :

$$(POM)\begin{cases} maximiser/minimiser \quad F(x) = (f_i(x))_{i=1,\dots,r} \\ s.c. \\ x \in \mathcal{S} \end{cases} \tag{3.1}$$

où $x \in \mathbb{R}^n$ est le $n-$vecteur des variables de décision, $F = (f_i)_{i=1,\dots,r}$ est le vecteur des fonctions-objectifs et r le nombre d'objectifs ($r \geq 2$), $\mathcal{S} \subseteq \mathbb{R}^n$ représente l'ensemble des solutions réalisables et

$\mathcal{S} = \{x \in \Omega | h_j(x) \geq 0, j \in \{1, \ldots, p\}.\ g_k(x) = 0, k \in \{1, \ldots, m\}\}$ (Ω dans ce cas est l'intersection de tout les ensembles de définition des fonctions $f_i,\ i \in \{1, \ldots, r\},\ h_j,\ j \in \{1, \ldots, p\}$ et $g_k,\ k \in \{1, \ldots, m\}$).

Chacune des fonctions $f_i,\ i \in \{1, \ldots, r\}$ est soit à minimiser ou à maximiser. Sans perte de généralité on supposera que toutes les objectifs sont à maximiser.

L'ensemble $\Omega \subseteq \mathbb{R}^n$ est appelé **espace de décision** et l'image de cet ensemble par le vecteur des fonction-objectifs F est appelé **espace des critères** ou **espace des objectifs**.

$\mathcal{S}$ étant l'ensemble réalisable, l'image de $\mathcal{S}$ par le vecteur des fonction-objectifs F dans l'ensemble critère est appelé l'**ensemble réalisable dans l'espace des critères** et est noté par $\mathcal{Y}$ donc $\mathcal{Y} = \{y \in \mathbb{R}^r | y = F(x), x \in \mathcal{S}\}$.

Plusieurs noms ont été donnés à ce type de problème : optimisation vectorielle, optimisation multicritère, optimisation multi-attributs etc...

3.2.1 Dominance et optimalité de Pareto

La principale différence entre l'optimisation multiobjective et mono-objective vient de la définition d'optimalité. Pour l'optimisation multiobjective, il s'agit de l'optimalité de Pareto qui est définie par la dominance de Pareto [84] :

Définition 3.2.1 (**Dominance**)**.** Un vecteur $u \in \mathcal{Y}$ domine un autre vecteur $v \in \mathcal{Y}$ si est seulement si :

$$\forall i \in \{1, \ldots, r\},\ u_i \geq v_i \text{ et } \exists i \in \{1, \ldots, r\},\ u_i > v_i$$

et on écrit $u \geq v$.

Un vecteur u est ainsi dit **non-dominé** si et seulement si :

$$\nexists v \in \mathcal{Y} \text{ tel que } v \geq u$$

Pour toute paire de vecteurs critère $u \in \mathcal{Y}$ et $v \in \mathcal{Y}$, un et un seul des cas suivants peut se produire :

— soit u domine v donc on peut écrire $u \geq v$;

— ou v domine u et on peut écrire $v \geq u$;

— ou bien les deux vecteurs sont équivalents au sens de la dominance, si et seulement

si :

$$\exists i_1 \in \{1, \ldots, r\} \text{ et } \exists i_2 \in \{1, \ldots, r\} \text{ tel que } u_{i_1} > v_{i_1} \text{ et } u_{i_2} < v_{i_2}$$

Donc, on peut introduire la notion d'optimalité de PARETO :

Définition 3.2.2. Optimalité de Pareto, Efficience

Une solution réalisable $x^* \in \mathcal{S}$ est **Pareto optimale** ou **efficiente** si et seulement s'il n'existe pas de solution $x \in \mathcal{S}$ telle que $F(x)$ domine $F(x^*)$.

En d'autres termes, cette définition dit que x^* est PARETO optimale s'il n'existe pas de vecteur x réalisable qui augmente un critère sans causer une diminution simultanée d'au moins un critère. Par la suite, si $F(x)$ domine $F(y)$ on dira par abus de langage que x domine y.

Si x^* est efficiente, $F(x^*)$ est dite non-dominée, car il n'existe pas une solution dans $\mathcal{Y}$ qui domine le vecteur $F(x^*)$.

3.2.2 Ensemble efficient, Front de Pareto

Définition 3.2.3. Ensemble efficient

L'ensemble efficient $\mathcal{S}_E$, est définit comme suit :

$$\mathcal{S}_E = \{x \in \mathcal{S} | \nexists x' \in \mathcal{S},\ F(x') \geq F(x)\}$$

C'est l'ensemble de toutes les solutions efficientes de $\mathcal{S}$.

Définition 3.2.4. Front de Pareto

Étant donné un ensemble efficient $\mathcal{S}_E$, le front de PARETO note $\mathcal{Y}_N$ est l'ensemble défini par :

$$\mathcal{Y}_N = \{u \in \mathcal{Y} | u = F(x) \text{ et } x \in \mathcal{S}_E\}$$

C'est l'ensemble de tout les vecteurs non-dominés de $\mathcal{Y}$.

Définition 3.2.5. Efficience faible/ Non-dominance faible

Une solution $x^* \in \mathcal{S}$ est dite faiblement efficient si et seulement s'il n'y a pas d'autre solution $x^* \in \mathcal{S}$ telle que $f_i(x) > f_i(x^*)$ pour tous les $i \in \{1, \ldots, r\}$.

$\mathcal{S}_{WE}$ désigne l'ensemble des solutions faiblement efficientes.

Un point de l'espace critère $F(x)$ est dit faiblement non-dominé si et seulement si $x \in \mathcal{S}_{WE}$.

3.2.3 Les points particuliers

Certain points particuliers ont été définis afin de donner un ordre des valeurs de l'ensemble des solutions non-dominées. Ces points donnent une indication de la fourchette des valeurs que les points non-dominés peuvent atteindre.

Définition 3.2.6 (**Point idéal**)**.** Les coordonnées du point idéal (y^I) correspondent aux meilleurs valeurs de chaque objectif. Les coordonnées de ce point s'obtiennent en optimisant chaque fonction-objectif séparément :

$$y^I = \left(y_i | y = \max_{x \in \mathcal{S}} f_i(x) \right)_{i \in \{1,\dots,r\}}$$

Définition 3.2.7 (**Point anti-idéal**)**.** Le point anti-idéal y^A est le point qui a comme valeur pour chaque objectif la valeur minimale de l'objectif considérée si le problème est un problème de maximisation :

$$y^A = \left(y_i | y = \min_{x \in \mathcal{S}} f_i(x) \right)_{i \in \{1,\dots,r\}}$$

Définition 3.2.8 (**Point nadir**)**.** Le point nadir y^N est le vecteur dont les coordonnées correspondent aux pires valeurs de chaque objectif des points du front de Pareto :

$$y^N = \left(y_i | y = \min_{x \in \mathcal{S}_E} f_i(x) \right)_{i \in \{1,\dots,r\}}$$

ou bien

$$y^N = \left(y_i | \min_{y \in \mathcal{Y}_N} y_i \right)_{i \in \{1,\dots,r\}}$$

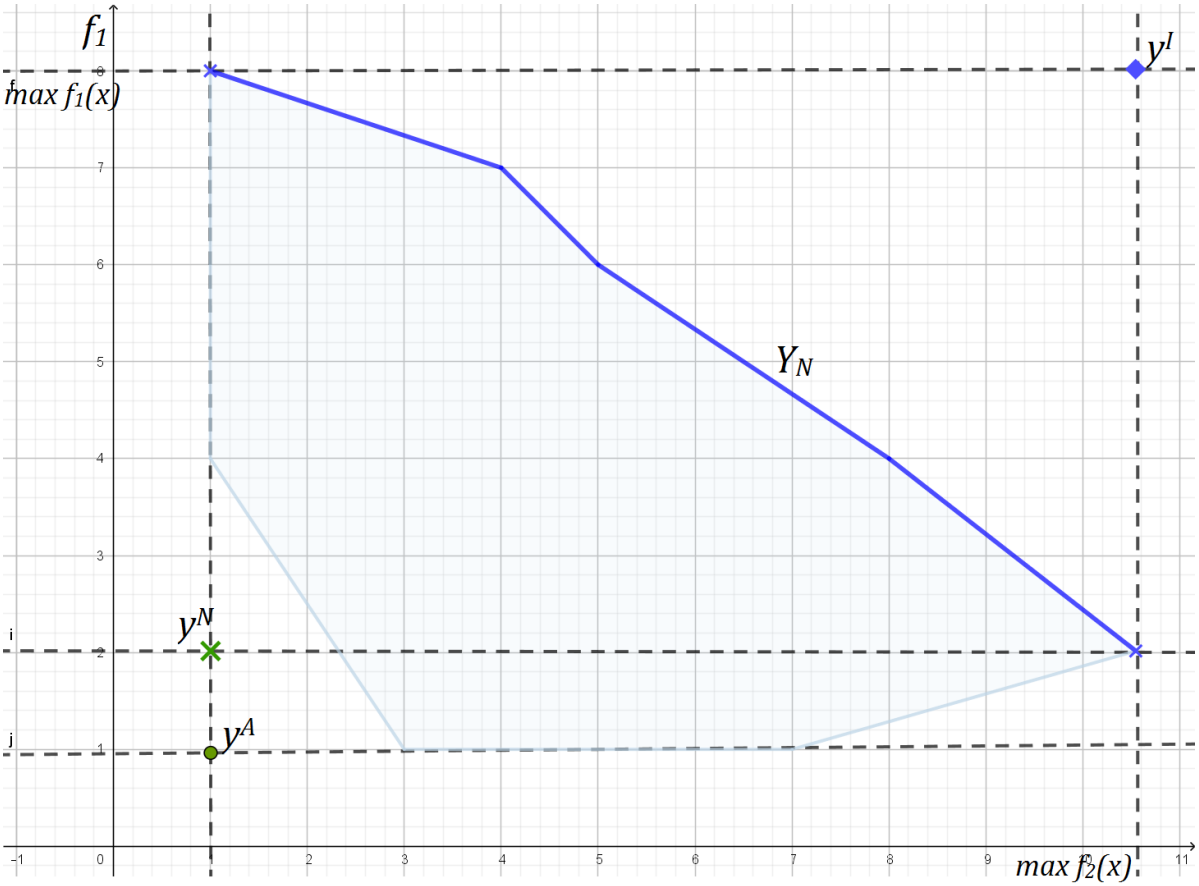


FIGURE 3.1 – Exemple d'ensemble critère d'un problème bi-objectif

Définition 3.2.9 (**Matrice des gains**). La Matrice des gains est une matrice carrée de dimension r construite comme suit :

$f_1(x^1)$	$f_2(x^1)$	$\dots$	$f_i(x^1)$	$\dots$	$f_r(x^1)$
$f_1(x^2)$	$f_2(x^2)$	$\dots$	$f_k(x^2)$	$\dots$	$f_r(x^2)$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$
$f_1(x^i)$	$f_2(x^i)$	$\dots$	$f_i(x^i)$	$\dots$	$f_r(x^i)$
$\vdots$	$\vdots$	$\ddots$	$\dots$	$\vdots$	$\vdots$
$f_1(x^r)$	$f_2(x^r)$	$\dots$	$f_i(x^r)$	$\dots$	$f_r(x^r)$

TABLE 3.1 – Table des gains

où $f_i(x^i) = \max\limits_{x \in \mathcal{S}} f_i(x),\ i \in \{1, \dots, r\}$, donc les éléments de la verticale de la **table des gains** sont égaux au coordonnés du vecteur idéal y^I.

3.3 Classification des méthodes multiobjectives

Dans la littérature [43], on peut trouver deux types de classification des méthodes de résolution des problèmes multiobjectifs. Le premier adopte un point de vue décideur alors que le second plus théorique et conceptuel trie les méthodes selon leurs façon de traiter les objectifs.

3.3.1 Classification de point de vue décideur

Cette classification se base sur l'usage de la méthode. Le décideur devant choisir un compromis, décide aussi de moment de son intervention dans le processus d'aide à la décision et on peut donc répartir les méthodes de résolution de problèmes multiobjectif en trois groupes, en fonction du moment où intervient le décideur :

3.3.1.1 Les méthodes a priori

Dans ce cas, le compromis que désire faire le décideur entre les objectifs est défini avant l'exécution de la méthode. Ainsi une seule exécution permettra d'obtenir la solution recherchée. C'est une approche rapide mais il faut cependant prendre en compte le temps de modélisation du compromis et la possibilité pour le décideur de ne pas être satisfait par la solution trouvée et de relancer la recherche avec un autre compromis.

3.3.1.2 Les méthodes progressives ou interactives

Dans ce cas, le décideur intervient dans le processus de recherche de solutions en répondant à différentes questions afin d'orienter la recherche. Le décideur modifie ainsi interactivement le compromis entre ses préférences et les résultats.

Cette approche permet de bien prendre en compte les préférences du décideur, mais nécessite sa présence tout au long du processus de recherche.

3.3.1.3 Les méthodes a posteriori

Dans ce cas, on cherche à fournir le décideur avec un ensemble de bonne solutions bien réparties, il pourra ensuite choisir la solution qui lui convient le mieux.

Ainsi, il n'est plus nécessaire de connaître et de modéliser les préférences du décideur (ce qui peut s'avérer être très difficile), mais il faut en contre-partie fournir un ensemble

de solutions efficientes bien réparties ou même l'ensemble efficient en entier, ce qui peut également être difficile et requiert un temps de calcul important.

3.3.2 Classification conceptuelle

Ce type de classement s'articule sur les notions d'agrégation et d'optimalité de PARETO, il englobe trois types de méthodes :

3.3.2.1 Les méthodes scalaires

Les méthodes agrégées ou scalaires ou encore techniques de scalarisation transforment le problème multiobjectif en un problème monocritère. Quatre principales techniques de scalarisation sont beaucoup utilisées :

La méthode de la somme pondéré Cette méthode consiste en l'optimisation d'une somme pondérée des r fonctions-objectifs en leur attribuant des coefficients de pondération [29]. Ces coefficients représentent l'importance relative que le décideur attribue à chaque objectif.

La méthode $\epsilon-$ contraintes Cette méthode consiste à sélectionner l'une des r fonctions-objectifs et à l'optimiser en considérant les autres $r-1$ objectifs comme des contraintes en spécifiant les bornes inférieures (de réservation) que le décideur est prêt à accepter [73]. Le programme à résoudre s'écrit comme suit :

$$(L_\epsilon)\begin{cases} \max f_1(x) \\ s.c. \\ x \in \mathcal{S} \\ f_i(x) \geq \epsilon_i,\ i = 2, \ldots, r \end{cases} \tag{3.2}$$

où les $\epsilon_2, \epsilon_3, \ldots, \epsilon_r$ sont des niveaux de satisfaction qui représentent les exigences minimales de décideurs pour chaque objectifs respectivement.

La méthode de la programmation par but La programmation par but est une méthode couramment utilisée en programmation mathématique lorsqu'il est impossible de respecter certaines contraintes. CHARNES et COOPER [24] ont présenté une méthode pour utiliser la programmation par but dans le cadre multiobjectif. Le décideur fixe un but f^* à atteindre, les valeurs des composantes du but f^* sont ensuite ajoutées au problème comme des contraintes supplémentaires.

La nouvelle fonction-objectif est modifiée de façon à minimiser la somme des écarts entre les résultats et les buts à atteindre :

$$\begin{cases} \min \sum_{i=1}^{r} |f_i(x) - f_i^*| \\ s.c. \\ x \in \mathcal{S} \\ f_i(x) \geq f_i^*,\ i = 1, \ldots, r \end{cases} \tag{3.3}$$

La méthode du but à atteindre Cette méthode est initialisée avec un vecteur critère $f^* \in \mathcal{Y}$. Ensuite, une direction de recherche est choisit en fournissant, en quelque sorte des coefficients de pondération comme pour la méthode de la somme pondérée $w_i \in \mathbb{R},\ i \in \{1, \ldots, r\}$, on minimise ensuite un coefficient scalaire λ qui représente l'écart par rapport à l'objectif initial f^* que l'on s'était fixé [85]. Le problème à résoudre devient :

$$\begin{cases} \min \lambda \\ s.c. \\ x \in \mathcal{S} \\ f_i(x) - w_i \cdot \lambda \leq f_i^*,\ i = 1, \ldots, r \end{cases}$$

3.3.2.2 Les méthodes fondées sur l'optimalité de Pareto

Ces méthodes s'appuient sur la notion de dominance au sens de Pareto (tel que définit précédemment) qui privilégie une recherche qui satisfait au mieux tout les objectifs.

*3.3.2.3 Les méthode non-*Pareto *et non-agrégés*

Certaines méthodes n'utilisent aucun des deux concepts précédents. Là ou ces deux concepts traitent les objectifs simultanément, les méthodes dites non-agrégées et non-Pareto procèdent par une recherche qui traite séparément les objectifs.

3.4 Problèmes multiobjectifs en nombres entiers

Les programmes multiobjectifs en nombres entiers appartiennent à une classe importante de problèmes qui se posent dans la prise de décision multiobjectif lorsque certaines

ou toutes les variables du décisions du modèle sont discrètes.

Lorsque des contraintes d'intégrités apparaissent, un changement survient sur le front de PARETO qui devient discret, ayant un front de PARETO discret les programmes multiobjectifs en nombres entiers présentent la particularité de la distinction entre les solutions non-dominées **supportées** et **non-supportées**.

Définition 3.4.1 (**Solution non-dominée supportée/non-supportée**)**.** Une solution non-dominée $y \in \mathcal{Y}$ est non-supportée si elle est dominée par une solution (non réalisable) appartenant à l'enveloppe convexe du domaine réalisable $Co(\mathcal{Y})$. Les autres solutions non-dominées sont supportées.

Les solutions **non-dominées supportées** sont celles situées sur l'*enveloppe convexe* du l'ensemble réalisable critère, toutes les autres solutions non-dominées sont **non-supportées** (voir l'exemple présenté sur figure 3.2).

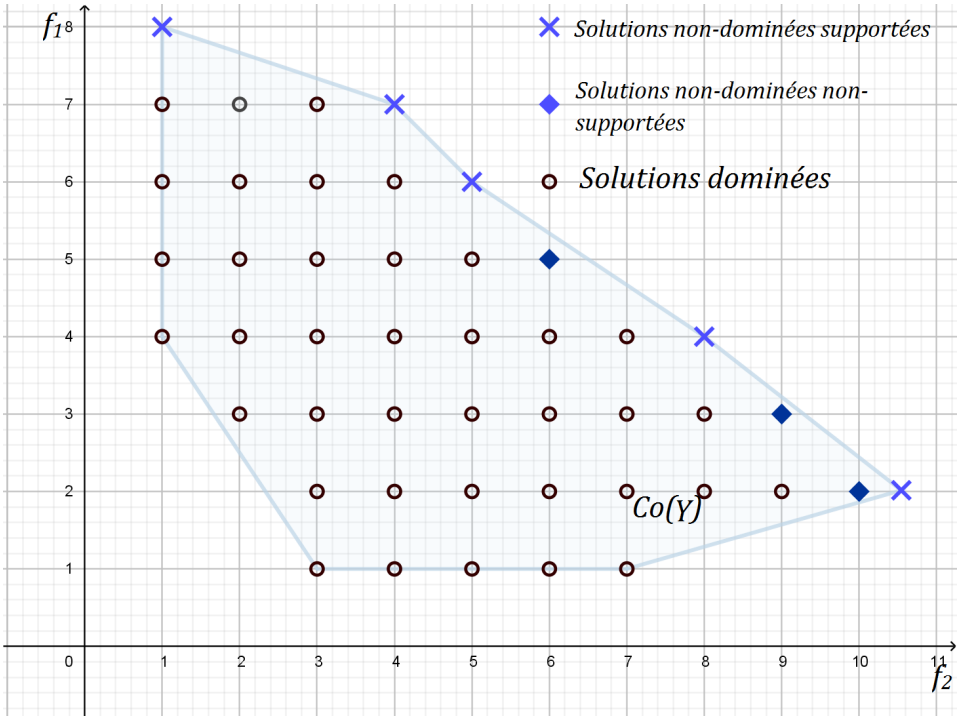


FIGURE 3.2 – Solutions supportées non-supportées d'un problème biobjectif

Au cours des dernières décennies, plusieurs méthodes ont été développées pour résoudre les MOILPs, ces méthodes peuvent être divisées en algorithmes de recherche dans l'espace de décision (c'est-à-dire des méthodes de recherche dans l'espace de solu-

tions réalisables) et en algorithmes de recherche dans l'espace critère (c'est-à-dire des méthodes de recherche dans l'espace des valeurs des fonction-objectifs).

3.4.1 Programmes multiobjectifs linéaires en nombres entiers

Un programme linéaire en nombres entiers (MultiObjective Integer Linear Program **MOILP**) peut être formulé par :

$$(MOILP)\begin{cases} \max Cx \\ s.c. \\ x \in \mathcal{X} \cap \mathbb{Z}^n \end{cases} \tag{3.4}$$

où $C = (c^i)_{i=1,\dots,r}^T$ est une $r \times n-$ matrice de réels, $r \geq 2$, $\mathcal{X} = \{x \in \mathbb{R}^n | Ax \leq b,\ x \geq 0\}$ où A est une $m \times n-$ matrice de contraintes et $b \in \mathbb{R}^m$. On note $\mathcal{X}_E$ l'ensemble efficient de l'espace de décision de $MOILP$ et $\mathcal{Y}_N$ l'espace critère du $MOILP$.

Les premiers travaux sur le MOILP ont commencé avec les travaux de G. R. Bitran à la fin des années 70 [16] qui a développé des algorithmes pour résoudre des programmes linéaires multiobjectifs avec des variables binaires basées sur des schémas énumératifs. Les articles [28, 116, 123] fournissent des bibliographies exhaustives sur les techniques permettant de résoudre les MOILPs apparues dans les années 80 à 90.

D. Klein et E. Hannan [65] ont proposé une méthode séquentielle pour générer tous les solutions efficientes d'un MOILP alors que L. G. Chalmet et al. [23] ont étudié les MOILP dans le cas de deux objectifs en utilisant dans la résolution la méthode de scalarisation de la somme pondérée. Cette méthode ne fournit que les solutions efficientes supportées, les auteurs incluent une contrainte supplémentaire qui assure également l'accès aux solutions efficientes non-supportées.

M. Ozlen et M. Azizoglu [82] ont présenté une méthode récursive exacte de recherche dans l'espace critère, leur approche est basée sur l'identification de plages d'efficiences des objectives, qui sont identifiés en résolvant des MOILP plus simples avec moins d'objectifs. Leur méthode est présentée pour trois objectifs et ensuite généralisée pour un nombre indéfini d'objectifs.

A. Przybylski et al. [88] ont présenté une généralisation de la méthode des deux phases pour résoudre le MOILP avec plusieurs objectifs. La première phase vise à calculer toutes les solutions efficientes supportées et la deuxième phase à calculer les points efficients non-supportés. Des résultats expérimentaux de la méthode sont donnés pour

le problème d'affectation à trois objectifs.

Plusieurs approches se sont inspiré des techniques de scalarisations et les ont améliorées dans le but de générer les solutions efficientes supportées et non-supportées :

J. Sylva et A. Crema [114] ont utilisé une méthode séquentielle utilisant la technique de la somme pondérée inspirée de celle de Klein et Hannan [65] pour retrouver toutes les solutions efficientes, leur méthode a été ensuite reprise et améliorer par B. Lokman et M. Köksalan [72].

La technique de scalarisations la plus utilisée est la méthode de scalarisation $\epsilon-$contrainte, un courant de méthodes ont été dédié à cette approche. M. Laumanns et al. [71] ont mis en œuvre un schéma adaptatif pour faire varier les ϵ et ont appelé leur méthode $\epsilon-$contrainte adaptative (adaptative $\epsilon-$constraint).

Dans [42], M. Ehrgott et S. Ruzika ont proposé une $\epsilon-$contrainte méthode améliorée. G. Mavrotas [76] a proposé la méthode AUGMECON qui permet de générer toutes les solutions efficientes, ensuite G. Mavrotas et K. Florios [78] ont amélioré cette méthode et présenté un nouvel algorithme AUGMECON2. Une variante de la méthode AUGMECON, appelée méthode SAUGMECON a été développée par W. Zhang et M. Reimann [121].

Quelque auteurs ont utilisé les techniques d'exploration implicite. G. Mavrotas et D. Diakoulaki [77] ont mis au point un algorithme de B&B pour trouver les points extrêmes non-dominés pour un programme linéaire à objective multiple ayant des variables binaires et des variables à valeur réelle. Dans le papier [27], M. E-A. Chergui et al. ont élaboré une coupe qui permet de retirer les points dominées, cette coupe a été utilisée dans un schéma de B&C pour générer toutes les solutions efficientes.

3.4.2 Génération de l'ensemble efficient

Soit à résoudre :

$$(P)\begin{cases} \max f_i(x),\ i=1,\ldots,r \\ s.c. \\ x \in \mathcal{X} \cap \mathbb{Z}^n \end{cases} \tag{3.5}$$

Présentant deux approches de la littérature pour générer tout l'ensemble efficient des programmes multiobjectif en nombres entiers.

3.4.2.1 Méthodes AUGMECON, AUGMECON2 et SAUGMECON

Les méthodes AUGMECON, AUGMECON2 et SAUGMECON sont inspirées de la méthode $\epsilon-$contrainte qui consiste en l'optimisation d'un seul objectif sous les contraintes supplémentaires que les autres objectifs atteignent un niveau de tolérance fixé au préalable par le décideur.

Des solutions peuvent être obtenues par des variations paramétriques des niveaux de satisfaction $\epsilon_2, \epsilon_3, \ldots, \epsilon_r$. Il a été démontré que si la solution du programme (3.2) de la méthode $\epsilon-$contrainte est unique, elle est efficiente [73]. Le désavantage de cette approche est qu'il est nécessaire de présélectionner l'objectif à optimiser et les valeurs des $\epsilon_i,\ i \in \{2, \ldots, r\}$, ce qui peut s'avérer être problématique car pour beaucoup de valeurs il n'y aura pas de solution réalisable. Un deuxième désavantage est celui de la multiplicité des solutions du programme (3.2) de la méthode $\epsilon-$contrainte. Dans ce cas la solution obtenue après résolution est seulement faiblement efficiente. Pour contourner ce dernier désavantage de la méthode $\epsilon-$contrainte, la méthode AUGMECON a été proposée par Mavrotas [76] où l'efficience des solutions obtenues est garantie.

La méthode AUGMECON transforme les contraintes d'inégalité sur les objectifs en contraintes d'égalité en introduisant des variables d'écarts non négatives, puis augmente la fonction-objectif par la somme pondérée de ces variables d'écarts. Avec cette méthode, le programme (L_ϵ) est reformulé comme suit :

$$(P_\epsilon)\begin{cases} \max f_1(x) + \delta\,(e_2/s_2 + e_3/s_3 + \ldots + e_r/s_r) \\ s.c. \\ x \in \mathcal{X} \cap \mathbb{Z}^n \\ f_i(x) - e_i = \epsilon_i,\ i = 2, \ldots, r \end{cases} \tag{3.6}$$

où les paramètres $s_2, s_3, \ldots, s_r$ sont les étendues pré-calculés des fonctions-objectifs respectives, $e_2, e_3, \ldots, e_r$ sont les variables d'écart des contraintes respectives et $\delta \in [10^{-6}, 10^{-3}]$.

Proposition 3.4.1. *Le programme ci-dessus (3.6) ne produit que des solutions efficientes (évite la génération de solutions faiblement efficientes).*

Démonstration. Supposons que le problème (3.2) a un optima alternative et que l'un d'eux (noté x^0) domine la solution optimale (noté x^*) obtenue de la résolution du problème (3.6). Cela signifie que le vecteur $(f_1(x^*), \epsilon_2 + e_2^*, \epsilon_3 + e_3^*, \ldots, \epsilon_r + e_r^*)$ est dominé

par le vecteur $(f_1(x^0), \epsilon_2 + e_2^0, \epsilon_3 + e_3^0, \ldots, \epsilon_r + e_r^0)$ ou autrement dit :

$$\begin{aligned} \epsilon_2 + e_2^* &\leq \epsilon_2 + e_2^0 \\ \epsilon_3 + e_3^* &\leq \epsilon_3 + e_3^0 \\ &\vdots \\ \epsilon_r + e_r^* &\leq \epsilon_r + e_r^0 \end{aligned}$$

avec au moins une inégalité stricte, notant que $f_1(x^*) = f_1(x^0)$ car ce sont des maximums alternatifs de (3.2). En faisant la somme de ces inégalités et en se basant sur le fait qu'il y a au moins une inégalité stricte, on conclue que :

$$\sum_{i=2}^{r} e_i^* < \sum_{i=2}^{r} e_i^0$$

Mais cela contredit l'hypothèse initiale selon laquelle la solution optimale de (3.6) maximise la somme de e_i. Il n'y a donc pas de solution x^0 qui domine la solution x^* obtenue ou, en d'autres termes, la solution x^* obtenue du problème (3.6) est efficientes.

□

AUGMECON2 est une version améliorée d'AUGMECON. AUGMECON2 modifie légèrement la fonction objectif comme suit :

$$(P'_\epsilon) \begin{cases} \max f_1(x) + \delta\Big(e_2/s_2 + 10^{-1} \times e_3/s_3 + \ldots + 10^{-(r-2)} \times e_r/s_r\Big) \\ s.c. \\ x \in \mathcal{X} \cap \mathbb{Z}^n \\ f_i(x) - e_i = \epsilon_i, \ , \ i = 2, \ldots, r \end{cases} \tag{3.7}$$

La méthode de génération de l'ensemble efficient est la suivante :

— A partir de la table des gains, on obtient l'étendu de chacune des $(r-1)$ fonctions-objectifs qui sont dans le bloc des contraintes ;

— ensuite l'étendu de la i-ème fonction-objectif est divisé en q_i intervalles égaux (Dans le cas ou le but est de générer tout l'ensemble efficient $q_i = s_i - 1$) ;

— après, les valeurs des ϵ_i sont variées en utilisant les $(q_i + 1)$ points équidistants intermédiaires (Pour générer tout l'ensemble efficient la variation est par unité c'est-à-dire $étape_i = 1$ dans l'organigramme suivant).

Tout cela fait donc un total de $(q_2+1)(q_3+1)\ldots(q_r+1)$ résolutions de programme (P_ϵ) ou (P'_ϵ). L'organigramme de cet algorithme est illustré à la Figure 3.3.

La méthode SAUGMECON améliore AUGMECON en combinant les caractéristiques de la méthode traditionnelle de $\epsilon-$contraintes et celles de la méthode AUGMECON. Comme dans la méthode traditionnelle de $\epsilon-$contraintes, Le méthode SAUGMECON utilise les inégalités sur les objectifs ajoutés au autre contraintes.

D'autre part, à l'instar de la méthode AUGMECON, cette méthode incorpore la somme des valeurs pondérées dans la fonction-objectif. Avec la méthode SAUGMECON.

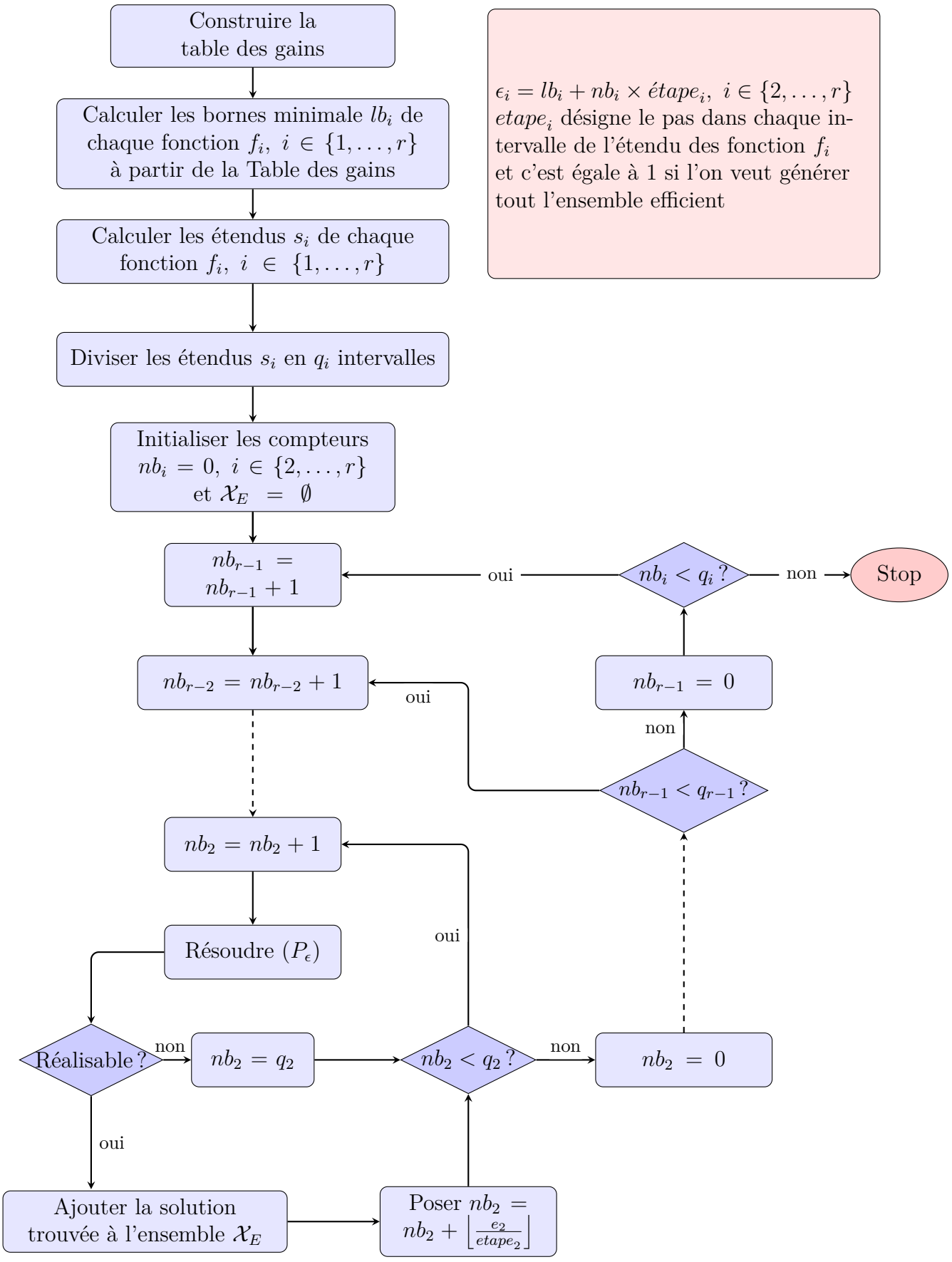


FIGURE 3.3 – Organigramme de la méthode AUGMECON et AUGMECON2

le programme ci-dessous peut être réécrit comme suit :

$$(SP_\epsilon)\begin{cases} \max f_1(x) + \delta\left(f_2(x)/s_2 + f_3(x)/s_3 + \ldots + f_r(x)/s_r\right) \\ s.c. \\ x \in \mathcal{X} \cap \mathbb{Z}^n \\ f_i(x) \geq \epsilon_i,\ i = 2, \ldots, r \end{cases} \tag{3.8}$$

3.4.2.2 Méthode de Chergui *et* Moulaï

Dans l'article [26], M. E-A CHERGUI et M. MOULAÏ présentent une méthode exacte pour retrouver toutes les solutions efficientes d'un programme linéaire fractionnaire multiobjectif en nombres entiers (MultiObjective Integer Linear Fractional Program **MOILFP**). Soit à résoudre :

$$(MOILFP)\begin{cases} \max \dfrac{c^i x + \alpha^i}{d^i x + \beta^i},\ i = 1, \ldots, r \\ x \in \mathcal{X} \cap \mathbb{Z}^n \end{cases} \tag{3.9}$$

La méthode décrite dans ce papier, est une procédure de B&C qui à chaque nœud l de l'arbre résout un programme linéaire fractionnaire (PLF), donc soit le PLF (3.10) associé au nœud l :

$$(P_l)\begin{cases} \max \dfrac{c^1 x + \alpha^1}{d^1 x + \beta^1} \\ x \in \mathcal{X}_l \end{cases} \tag{3.10}$$

avec $\mathcal{X}_l \subset \mathcal{X}$. A chaque nœud l où une solution x_l^* est obtenue, les notations suivantes sont utilisées :

$\mathcal{B}_l$ et $\mathcal{N}_l$ sont respectivement les ensembles des indices des variables de base et variables hors-base de la solution x_l^*, $\gamma^i = \bar{c}^i\bar{\beta}^i - \bar{\alpha}^i\bar{d}^i$ désigne le vecteur gradient réduit du critère i et $\bar{c},\ \bar{d},\ \bar{\alpha}$ et $\bar{\beta}$ désignent les valeur actualisées dans la base $\mathcal{B}_l$ de respectivement $c,\ d, \alpha$ et β.

On définit les ensembles $\mathcal{H}_l$ et $\mathcal{X}_{l+1}$ par :

$$\mathcal{H}_l = \left\{j \in \mathcal{N}_l | \exists i \in \{1, \ldots, r\}\ \gamma_j^i > 0\right\} \cup \left\{j \in \mathcal{N}_l | \gamma_j^i = 0, \forall i \in \{1, \ldots, r\}\right\}$$

$$\mathcal{X}_{l+1} = \left\{ x \in \mathcal{X}_l / \sum_{j \in \mathcal{H}_l} x_j \geq 1 \right\}$$

Toutes les opérations sont identifiées dans des nœuds et des branches dans une structure arborescente. A chaque nœud, on résout le programme linéaire (P_l).

Algorithme 5 : Méthode de CHERGUI et MOULAÏ pour la résolution d'un MOILFP

Étape 1. (Initialisation) $\mathcal{X}_0 = \mathcal{X}$, $l = 0$ et $\mathcal{X}_E = \emptyset$;

Résoudre le programme linéaire (P_0) au nœud 0 et soit x_l^* sa solution optimale.

Si x_l^* n'est pas entière, aller à l'**étape 2a**, sinon aller à l'**étape 2b**.

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

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

Étape 2a. Choisir l'une des coordonnées χ_j de x_l^* avec χ_j fractionnaire, séparer le nœud l actuel en deux nœuds : ajouter la contrainte $x_j \leq \lfloor \chi_j \rfloor$ au premier nœud, et la contrainte $x_j \geq \lfloor \chi_j \rfloor + 1$ au deuxième nœud et aller à l'**étape 2**.

Étape 2b. Si le vecteur critère de la solution x_l^* n'est pas dominé par le vecteur d'une autre solution de $\mathcal{X}_E$ alors $\mathcal{X}_E = \mathcal{X}_E \cup \{x_l^*\}$. S'il existe une solution telle que le vecteur critère de cette solution est dominé par celui de x_l^*, alors cette solution est retiré de $\mathcal{X}_E$.

Déterminer les ensembles $\mathcal{B}_l, \mathcal{N}_l$ et $\mathcal{H}_l$, Si $\mathcal{H}_l = \emptyset$, le nœud l correspondant est stérile, aller à l'**étape 2**. Sinon, ajouter la contrainte $\sum_{j \in \mathcal{H}_l} x_j \geq 1$ pour obtenir l'ensemble $\mathcal{X}_{l+1}$, résoudre le programme correspondant, soit x_l^* la solution optimale obtenue si x_l^* est une solution entière, aller à l'**étape 2b**. Sinon, aller à l'**étape 2a**.

Le nœud l de l'arbre est stérilisé si le programme correspondant (P_l) n'est pas réalisable ou si $\mathcal{H}_l = \emptyset$.

Si la solution optimale x_l^* du programme (P_l) n'est pas entière, le nœud l est séparé en deux nœuds avec les contraintes supplémentaires : $x_j \leq \lfloor \chi_j \rfloor$ et $x_j \geq \lfloor \chi_j \rfloor + 1$,

respectivement ou χ_j est égale à une composante de x_l^* qui est fractionnaire.

Dans le cas correspondant à une solution entière, les directions de croissance des critères sont utilisées à travers la coupe (3.11), afin d'éviter d'explorer des régions non-efficientes de l'espace des solutions réalisables. Ainsi seule, la partie du domaine des solutions réalisables dans laquelle au moins l'un des objectifs du problème, peut être amélioré est traitée. Ceci est rendu possible par l'ajout de la contrainte valide suivante :

$$\sum_{j \in \mathcal{H}_l} x_j \geq 1. \tag{3.11}$$

Cette méthode est résumée par l'algorithme 5 ci-dessus.

3.5 Conclusion

Ce chapitre avait pour objectif d'introduire dans un premier lieu les concepts de base de l'optimisation multiobjectif qui vont être utilisés dans cette thèse, notamment l'efficience et la dominance.

Ainsi après avoir revu les principales définitions de l'optimisation multiobjectif, deux méthodes de génération de l'ensemble efficient ont été présentées pour des problèmes de programmation multiobjectif en nombres entiers. L'accent a été mis sur des méthodes capable de générer tout l'ensemble efficient car la contribution principale de cette thèse est une méthode exacte qui génère aussi tout l'ensemble efficient. L'avantage de ces méthodes dites exactes est que ce sont des méthodes "a posteriori", qui ne demande pas l'intervention du décideur. Toutefois, dans plusieurs situations la génération de tout l'ensemble efficient n'est pas recommandée, cet ensemble ayant la plus part du temps une cardinalité très grande, sa génération entraine un temps de calcul volumineux et la confusion du décideur lors du choix d'une solution.

Deuxième partie :

Contribution

Chapitre 4

Optimisation linéaire sur l'ensemble efficient

Sommaire

4.1 Introduction

Au cours des dernières décennies, plusieurs méthodes ont été développées pour résoudre les MOILP dans le sens de la génération de la globalité de l'ensemble des solutions efficientes. Toutefois une énumération explicite de l'ensemble efficient d'un MOILP n'est pas recommandée pour des raisons pratiques, même en cas de problèmes de taille moyenne. Outre le temps de calcul très volumineux qu'impliquent de telles méthodes, la taille généralement très considérable de l'ensemble efficient tend à saturer le décideur, au point où le choix de la solution préférée devient une mission presque impossible.

Afin d'éviter de telles situations, une approche judicieuse consiste en l'utilisation d'une fonction linéaire de préférence comme un moyen de mesurer d'importance ou de discrimination parmi les nombreuses solutions efficientes. L'"*optimisation sur l'ensemble efficient*" est un axe intéressant pour la résolution des problèmes d'optimisation multiobjectifs.

4.2 Optimisation d'une fonction linéaire sur l'ensemble efficient

Le problème d'optimisation d'une fonction linéaire sur l'ensemble efficient est formulé sans perte de généralité comme suit :

$$(MOILP_E) \begin{cases} \max \phi(x) = dx \\ s.c. \\ x \in \mathcal{X}_E \end{cases} \tag{4.1}$$

ou d est un $n-$vecteur réel et $\mathcal{X}_E$ est l'ensemble efficient du MOILP (3.4).

4.2.1 Revue de littérature

De nombreux chercheurs se sont penchés sur le problème de l'optimisation d'une fonction linéaire sur l'ensemble efficient d'un programme linéaire multiobjectif (en variables continue) [120, 11, 22, 21, 35, 92, 40] à la suite des travaux pionniers de J. Philip [86], qui fut donc le premier à avoir formulé le problème de l'optimisation sur l'ensemble efficient. H. P. Benson quant-à lui fut le premier à avoir suggéré une application réelle de ce problème [12]. Cependant, seuls quelques papiers ont été dédiés au problème (4.1) avec des variables en nombre entier, ceci étant en partie dû à la non-convexité du domaine ce qui rend plus difficile l'obtention de solutions efficientes.

M. Abbas et D. Chaabane sont les premiers à avoir donné une méthode de résolution du problème (4.1) [1], ils ont proposé une procédure itérative qui à chaque itération ajoute deux types de coupes pour garantir une amélioration de la valeur de la fonction linéaire à optimiser.

J. M. Jorge [59] a développé ensuite une méthode de recherche dans l'espace critère qui s'inspire de la méthode de J. Sylva et A. Crema [114]. D. Chaabane et al.

[21] ont présenté une nouvelle version de l'algorithme de JORGE qui lui diffère par la résolution du programme de pondération de TCHEBYCHEV augmenté.

F. Z. OUAÏL et al. [81] ont présenté une méthode de recherche dans l'espace de décision, leur méthode consiste en une procédure de B&C qui à chaque solution entière trouvée dans l'arborescence applique une coupe qui oriente la recherche vers les solutions efficientes.

La plus récente méthode proposée à ce jour est celle de N. BOLAND et al. [17], l'algorithme qu'ils ont proposé repose sur une nouvelle méthode d'énumération des points non-dominés d'un MOILP, qui est le résultat de l'utilisation d'un nouveau schéma de décomposition de l'espace de critère.

4.2.2 Méthodes de résolution

Les articles qui proposent de résoudre le problème ($MOILP_E$) présents dans la littérature peuvent être divisés en deux catégories :

1. **Les méthodes utilisant des coupes ;**
2. **Les méthodes de corner-constraints.**

4.2.2.1 Les méthodes utilisant des coupes

Méthode de Abbas et Chaabane M. ABBAS et D. CHAABANE [1] ont proposé une méthode pour résoudre le problème ($MOILP_E$) qui utilise des coupes. Leur méthode est décrite par l'algorithme 6 ci-après.

En résumé leur méthode propose initialement de résoudre le programme :

$$\begin{cases} \max \phi(x) = dx \\ s.c. \\ x \in \mathcal{X} \cap \mathbb{Z}^n \end{cases} \tag{4.2}$$

Pour voir si la résolution est trivial. Si ce n'est pas le cas, à chaque itération cette procédure résout un programme linéaire en variables continues puis utilise la coupe de DANTZIG pour éliminer les solutions fractionnaires ou une coupe améliorant la valeur de ϕ pour retrouver la solution optimale.

Algorithme 6 : Algorithme de ABBAS et CHAABANE d'optimisation sur l'ensemble efficient

Résultat : Une solution x_{opt} optimale de $(MOILP_E)$ avec une valeur optimale de $\phi(x_{opt}) = d_{opt}$

Étape 1 Résoudre le programme (4.2)

Si (4.2) est irréalisable stop, $(MOILP_E)$ est irréalisable. Sinon, si sa solution optimale est efficiente, stop c'est la solution optimale de $(MOILP_E)$. Sinon aller en (2).

Étape 2 Poser $d_{opt} = -\infty$. Résoudre :

$$(P^1)\begin{cases} \max c^1 x \\ s.c. \\ x \in \mathcal{X} \cap \mathbb{Z}^n \end{cases} \tag{4.3}$$

Soit $\mathcal{B}_1$ et $\mathcal{N}_1$ les ensembles d'indice de base et hors-base de la solution x^1 de (P^1) respectivement.

1. Si $\mathcal{J}_1 = \{j \in \mathcal{N}_1 | \bar{c}_j^1 = 0\} = \emptyset$, alors la solution optimale x^1, est unique et efficace. Poser $d_{opt} = dx^1$, $x_{opt} = x^1$ et aller à l'étape 3.
2. Si $\mathcal{J}_1 \neq \emptyset$, alors la solution optimale x^1 du problème (P^1) peut ne pas être unique, tester l'efficience de x^1 ; si elle n'est pas efficiente, passer à l'étape 3. Sinon, poser $d_{opt} = dx^1$, $x_{opt} = x^1$ et aller également à l'étape 3.

Étape 3 Poser $l = 1$ et effectuer les sous-étapes suivantes :

(3.1) Construire l'ensemble $\Gamma_l = \{j \in \mathcal{N}_l | \bar{c}_j^1 \geq 0 \text{ et } \bar{d}_j \leq 0\}$:

— Si $\Gamma_l = \emptyset$, alors aller à l'étape 3.2 et ajouter la coupe de Dantzig $\sum_{j \in \mathcal{N}_l} x_j \geq 1$.

— Sinon, poser $\gamma = \Gamma_l$ et aller en (a)

(a) Si $\gamma = \emptyset$ alors soit $j_l \in \Gamma_l$, aller à sous-étape 3.2. Sinon, sélectionner $j_l \in \gamma$ et calculer $\theta_{j_l}^0$ la partie entière de $\min_{s \in \mathcal{B}_l}\{\frac{x_l}{\bar{a}_s^{j_l}} | \bar{a}_s^{j_l} > 0\}$.

— Si $\theta_{j_l}^0 = 0$, alors il n'y a pas de solution possible entière sur l'arête actuelle, mettre $\gamma = \gamma \setminus \{j_l\}$ et aller en (a).

— Sinon, si $\theta_{j_l}^0 \geq 1$ aller en (b).

(b) Si x_l est efficiente et $dx_l \geq d_{opt}$, calculer la valeur du paramètre β_l défini par : $\beta_l = \bar{d}_{j_l} - \sum_{s \in \mathcal{B}_l} \bar{d}_s \times \bar{a}_s^{j_l}$

Si cette valeur n'est pas égale à zéro, passer à l'étape (c), sinon mettre $\gamma = \gamma \setminus \{j_l\}$ et passer à (a).

Si x_l n'est pas efficiente ou $dx_l < d_{opt}$, passer à l'étape (c).

(c) Soit x_0 la solution efficiente dont le vecteur critère domine x_l, si $dx_0 > d_{opt}$, posser $d_{opt} = dx_0$, $x_{opt} = x_0$ et passe à la sous-étape 3.2. Sinon mettre $\gamma = \gamma \setminus \{j_l\}$ et passer à (a).

(3.2) Poser $l = l + 1$ et définir le nouveau domaine $\mathcal{X}_l$ comme sous-ensemble de $\mathcal{X}_{l-1}$ obtenu en appliquant la coupe $dx \geq dx_{opt}$ pour trouver une nouvelle solution optimale x_l et aller à la sous-étape 3.1.

(3.3) Poser $l = l + 1$ et définir le nouveau domaine $\mathcal{X}_l$ comme sous-ensemble de $\mathcal{X}_{l-1}$ obtenu en appliquant la coupe de Dantzig pour trouver une nouvelle solution optimale x_l et aller à la sous-étape 3.1.

La procédure se termine soit à la première étape lorsque la solution obtenue est efficiente, ou bien à l'impossibilité d'opérations de pivotage, indiquant que la région actuelle ne contient pas d'autre solution meilleure que la solution actuelle.

Méthode de Ouaïl et al. La méthode Ouaïl et al. [81] est une méthode de **B&C** qui comme la méthode de Abbas et Chaabane utilise deux types de coupes :

— le premier type de coupe améliore la valeur de ϕ,
— et l'autre type permet non-seulement d'éliminer les solutions fractionnaires comme le fait la coupe de Dantzig mais en plus permet aussi d'élaguer des solutions dominées. La méthode est résumé par l'algorithme 7.

Notant que la méthode de Ouaïl et al. est une procédure de B&C alors que la méthode de Abbas et Chaabane est une méthode de coupe. De ce faite, cette méthode est une méthode d'exploration implicite du domaine, à chaque nœud un programme linéaire est résolu. Si la solution obtenu est fractionnaire le schéma est classique, si par contre la solution obtenue est entière la coupe efficiente est éventuellement appliquée.

Cette méthode utilise la même coupe que de la méthode de Chergui et al. [27], et de ce faite la justification de la coupe efficiente est la même. La méthode est résumée par l'algorithme 7 ci-après.

4.2.2.2 Les méthodes de corner-constraints

J. M. Jorge a proposé une méthode itérative de résolution de ($MOILP_E$) [59] qui définit une séquence de problèmes entiers mono-objectif de plus en plus contraints, éliminant successivement les points dominés.

Algorithme 7 : Algorithme de OUÏAL et al. pour l'optimisation sur l'ensemble efficient

Résultat : Une solution x_{opt} optimale de $(MOILP_E)$ avec une valeur optimale d_{opt}

Étape 1 (initialisation) Poser $l = 0$ et la valeur optimale de la fonction-objectif $d_{opt} = -\infty$

Étape 2 (étape principale) Tant qu'il y a un nœud l non-saturé dans l'arbre, résoudre (P_l) :

$$(P_l) \begin{cases} \max \phi(x) = dx \\ s.c.\ x \in \mathcal{X}_l \end{cases} \tag{4.4}$$

en utilisant la méthode du simplexe ou la méthode dual du simplexe. Passer à l'étape 3.1.

Etape 3 (tests) 3.1 Test de faisabilité Si (P_l) est infaisable, alors stop, le nœud l est saturé. Sinon, soit x_l la solution trouvée, si $dx_{opt} \geq dx_l$, le nœud l est saturé. Sinon, passer à l'étape 3.2 ;

3.2 Test d'intégrité Si x_l est entier passer à l'étape 3.3, sinon passer à l'étape 4 ;

3.3 Test d'efficience Si x_l est efficiente alors ajouter la coupe $dx \geq dx_l$ et mettre à jour $x_{opt} = x_l$ et $d_{opt} = dx_l$; passer à l'étape 5.

Étape 4 (branchement) Choisir une coordonnée x_j de x_l telle que $x_j = \chi_j$ fractionnaire. Ensuite, diviser le programme (P_l) en deux sous-programmes, en ajoutant les contraintes $xj \leq \lfloor \chi_j \rfloor$ pour obtenir $(P_{l'})$, $x_j \geq \lceil \chi_j \rceil$ et pour obtenir $(P_{l''})$ tel que $l' > l+1, l'' > l+1$ et $l' \neq l''$, passer à l'étape 2.

Étape 5 (coupe efficiente) Construire l'ensemble $\mathcal{H}_l = \left\{ j \in \mathcal{N}_l | \exists i \in \{1, 2, \ldots, r\}; \bar{c}_j^i > 0 \right\} \cap \left\{ j \in \mathcal{N}_l | \bar{c}_j^i = 0,\ \forall i \in \{1, 2, \ldots, r\} \right\}$ Si $\mathcal{H}_l = \emptyset$ le nœud l est saturé ;
Sinon, construire le sous-programme $(P_{l'})$ dont le domaine est $\mathcal{X}_{l'} = \mathcal{X}_l \cap \{ \sum_{j \in \mathcal{H}_l} x_j \geq 1 \}$ (ajouter la coupe efficiente), passer à l'étape 2.

L'algorithme proposé par JORGE procède à chaque itération l à la résolution d'un programme linéaire entier noté (P_l) qui impose de nouvelles contraintes sur l'ensemble réalisable du problème résolu à l'itération précédente (P_{l-1}). Ce type de contraintes est connu dans la littérature sous le nom de "***corner constraints***" [65]. Ces contraintes permettent d'écarter toutes les solutions efficientes générées précédemment, mais également toutes autre solutions réalisables dont le vecteurs critère est dominé (ou identique) par le vecteur critère de l'une des solutions générées précédemment. L'algorithme 8 résume la méthode de JORGE.

Algorithme 8 : Algorithme de JORGE pour trouver un maximum sur l'ensemble efficient

Résultat : Une solution x^* optimale de $(MOILP_E)$

Étape 0 Initialisation :

— Soit $d^l = -\infty$, $d^u = +\infty$ et $l = 1$.

— Résoudre le programme :

$$(P)\begin{cases} \max \phi(x) = dx \\ s.c.\ x \in \mathcal{X} \cap \mathbb{Z}^n \end{cases} \tag{4.5}$$

— Si (P) est irréalisable stop, $(MOILP_E)$ est irréalisable . Sinon soit x^1 la solution optimale de (P), aller en (1).

Étape 1 Si $x^l \in \mathcal{X}_E$, stop. x^l est la solution optimale de $(MOILP_E)$. Sinon, $d^u = dx^l$ aller en (2).

Étape 2 Trouver $\hat{x}^l \in \mathcal{X}_E$ telle que $C\hat{x}^l$ domine Cx^l et soit $\bar{x}^l$ la solution optimale de programme :

$$(T_l)\begin{cases} \max \phi(x) = dx \\ s.c. \\ Cx = C\hat{x}^l \\ x \in \mathcal{X} \cap \mathbb{Z}^n \end{cases} \tag{4.6}$$

Si $\phi(\hat{x}^l) > d^l$ poser $d^l = \phi(\hat{x}^l)$ et $x^* = x^l$ comme solution optimale temporaire.
Si $\phi(\hat{x}^l) > d^l$, stop. x^* est la solution optimale de $(MOILP_E)$.

Étape 3 Soit

$$(P_l)\begin{cases} \max \phi(x) = dx \\ s.c. \\ x \in \mathcal{X} \cap \mathbb{Z}^n - \bigcup\limits_{s=l}^{l} D_s \end{cases} \tag{4.7}$$

ou $D_s = \{x \in \mathbb{Z}^n | C\bar{x}^s \geq Cx\}$
Si (P_l) est irréalisable, stop. x^* est la solution optimale de $(MOILP_E)$. Sinon, soit x^{l+1} la solution optimale de (P_l). Si $dx^{l+1} \leq d^l$, stop. x^* est une solution optimale de $(MOILP_E)$. Sinon, poser $l = l + 1$ et aller en (1).

Cet algorithme est l'extension de la méthode de SYLVA et CREMA [114] au problème d'optimisation sur l'ensemble efficient.

4.2.3 Le test d'efficience

À l'étape 1 de l'algorithme 8, l'algorithme teste l'efficience d'une solution et à l'étape 2 une solution efficiente est obtenue à partir d'une solution non-efficiente. Également, dans les algorithmes de ABBAS et CHAABANE et celui de OUAÏL et al. est supposé l'existence d'un test d'efficience, qui permet en plus de générer une solution efficiente

qui domine la solution testée si elle n'est pas elle même efficiente. Ce test peut être efficacement effectué et a été donné par J. G. Ecker et I. Kouada [39]. Si on veut testé l'efficience d'une solution réalisable quelconque x^*, le teste consiste en la résolution du programme :

$$(EK_{x^*}) \begin{cases} \max \sum_{i=1}^{r} w_i \\ s.c. \\ x \in \mathcal{X} \cap \mathbb{Z}^n \\ c^i x - w_i = c^i x^*, \ i = 1, \ldots, r \\ w_i \geq 0, \ i = 1, \ldots, r \end{cases} \tag{4.8}$$

Théorème 4.2.1. *Soit une solution réalisable $x^* \in \mathcal{X} \cap \mathbb{Z}^n$:*

1. *Si la solution optimale de (4.8) a une valeur optimale de zéro alors le vecteur Cx^* est non-dominé.*
2. *Sinon, soit la nouvelle solution $\bar{x}$ obtenue après résolution de (4.8) alors $\bar{x} \neq x^*$ et le vecteur $C\bar{x}$ domine le vecteur Cx^* et $C\bar{x}$ est non-dominé*

Démonstration. Soit $(\bar{w}, \bar{x})$ la solution optimale obtenue de la résolution du programme (4.8), alors :

1. Soit $\sum_{i=1}^{r} \bar{w}_i = 0 \Rightarrow \bar{w}_i = 0, \forall i = \in \{1, \ldots, r\}$.
   Sachant que toute solution $\bar{x}$ de (4.8) vérifie $C\bar{x} \geq Cx^*$ donc il n'existe pas de solution $\bar{x}$ dans $\mathcal{X} \cap \mathbb{Z}^n$ telle que $\forall i \in \{1, \ldots, r\}, \ c^i\bar{x} \geq cx^*$ avec au moins une inégalité stricte, il s'ensuit que le vecteur Cx^* est non-dominé.
2. Soit $\sum_{i=1}^{r} \bar{w}_i > 0 \Rightarrow \exists i \in \{1, \ldots, r\}, \ \bar{w}_i > 0$ donc pour un certain indice $i \in \{1, \ldots, r\}, \ c^i\bar{x} > c^i x^*$ et $c^i\bar{x} = c^i x^*$ pour les autres indices $i \in \{1, \ldots, r\}$ de ce faite $C\bar{x}$ domine Cx^*.

   Puisque $\bar{w}$ est atteint au maximum donc il n'existe pas un $x \in \mathcal{X} \cap \mathbb{Z}^n$ tel que $\forall i \in \{1, \ldots, r\}, \ cx \geq c^i\bar{x}$ avec au moins une inégalité stricte , il s'ensuit que le vecteur $C\bar{x}$ est non-dominé.

□

4.3 Optimisation biobjectif sur l'ensemble efficient

Les problèmes de l'optimisation sur l'ensemble efficient sont très pratiques vu qu'ils permettent de réduire le temps de calcul et la confusion due au nombre très important de solution efficientes. Cependant, supposant qu'il existe deux décideurs ou plusieurs qui expriment plusieurs fonctions de préférence, la solution de ce problème fournissant qu'une solution peut ne pas être de meilleur compromis entre les décideurs. Une solution à mi-chemin entre l'optimisation sur l'ensemble efficient et sa génération serrait la génération d'un sous-ensemble. On propose donc un moyen d'obtenir une partie de l'ensemble efficient qui est générée suivant les préférences de décideur.

Soit deux vecteurs réels d^1 et d^2, le problème d'optimisation biobjectif sur l'ensemble efficient d'un MOILP s'écrit comme suit :

$$\begin{cases} \max d^1 x \\ \max d^2 x \\ s.c. \\ x \in \mathcal{X}_E \end{cases} \tag{4.9}$$

où $\mathcal{X}_E$ est l'ensemble efficient du MOILP (3.4).

Ce nouveau type de problème peut être un bon modèle dans au moins deux types de situations réelles. Le premier type peut se produire lorsqu'un décideur dispose de deux fonctions de préférence pour évaluer les solutions efficientes d'un problème multiobjectif.

Le second type de situation se produit quant il y a deux décideurs. Par exemple, considérant l'existence de deux firmes ayant adhéré à un projet commun. Ce projet nécessite l'optimisation de plusieurs objectifs contradictoires, mais ces deux entreprises ont également des fonctions de profit individuelles à maximiser puisqu'elles ne participent pas au projet de la même manière. Le problème consiste à trouver des solutions qui optimisent deux fonctions de préférence par rapport à l'ensemble efficient de problèmes multiobjectifs.

Aussi, faisant l'analogie avec l'application pratique proposée par H. P. Benson [12] : Supposant un problème de planification de la production d'une entreprise qui dispose de dix usines pour produire un certain type de produit. L'objectif de l'entreprise est de maximiser sa fonction de profit. Cependant, chaque usine vise également à maintenir des niveaux d'emploi élevés. Aussi l'entreprise est une filiale d'une multinationale donc la

planification de production doit aussi tenir compte de la fonction de profit de l'entreprise-mère. Ce problème se modélise le mieux avec le programme (4.9).

4.3.1 Méthodologie de résolution et algorithme

La résolution de (4.9) fournit le sous-ensemble $\mathcal{X}_C \subset \mathcal{X}_E$. Notant que $\mathcal{X}_C$ est l'intersection de $\mathcal{X}_E$ et $\mathcal{X}'_E$ où $\mathcal{X}'_E$ est l'ensemble efficient du problème :

$$\begin{cases} \max & d^1 x \\ \max & d^2 x \\ s.c. & \\ x \in \mathcal{X} \cap \mathbb{Z}^n \end{cases} \tag{4.10}$$

La méthode naïve pour résoudre le problème consisterait à générer les ensembles efficients $\mathcal{X}_E$ et $\mathcal{X}'_E$ au complet puis à sélectionner les éléments de l'intersection. La méthode qu'on propose est une procédure de B&C basée sur un processus de B&B et renforcée par deux types de coupes permettant de sonder les nœuds plus rapidement.

À chaque nœud l le PL suivant est résolu en utilisant la méthode du simplexe ou dual du simplexe :

$$(PL_l) \begin{cases} \max & d^1 x \\ s.c. & \\ x \in \mathcal{X}_l \end{cases}$$

où $\mathcal{X}_0 = \mathcal{X}$ et pour $l > 0$, $\mathcal{X}_l$ est un sous-ensemble de $\mathcal{X}$.

Si (PL_l) n'a pas de solution, le nœud l est sondé, sinon, deux cas se présentent :

— Le premier cas arrive quant la solution optimale obtenue $x^{*(l)}$ n'est pas entière : un processus de branchement normal est alors suivi ;

— L'autre cas se produit lorsque la solution $x^{*(l)}$ est entière : deux ensembles $\mathcal{H}_l$ et $\mathcal{H}'_l$ sont alors construits et les coupes efficientes (4.11) et (4.12) sont éventuellement ajoutés aux nœuds successeurs de l.

$$\sum_{j \in \mathcal{H}_l} x_j \geq 1 \tag{4.11}$$

$$\mathcal{H}_l = \left\{ j \in \mathcal{N}_l | \exists i \in \{1, \ldots, r\}; \bar{c}^i_j > 0 \right\} \cup \left\{ j \in \mathcal{N}_l |\ \bar{c}^i_j = 0,\ \forall i =\in \{1, \ldots, r\} \right\}$$

$$\sum_{j\in\mathcal{H}'_l} x_j \geq 1 \tag{4.12}$$

$$\mathcal{H}'_l = \left\{j \in \mathcal{N}_l |\ \bar{d}^2_j > 0\right\} \cup \left\{j \in \mathcal{N}_l |\ \bar{d}^1_j = 0 \text{ et } \bar{d}^2_j = 0\right\}$$

Ainsi, le domaine correspondant du nœud l_1, le successeur de l, est obtenu en appliquant (4.11) et (4.12) à $\mathcal{X}_l$

$$\mathcal{X}_{l_1} = \mathcal{X}^1_{l_1} \cap \mathcal{X}^2_{l_1}. \tag{4.13}$$

où $\mathcal{X}^1_{l_1} = \left\{x \in \mathcal{X}_l |\ \sum_{j\in\mathcal{H}_l} x_j \geq 1\right\}$ et $\mathcal{X}^2_{l_1} = \left\{x \in \mathcal{X}_l |\ \sum_{j\in\mathcal{H}'_l} x_j \geq 1\right\}$.

Chaque solution entière $x^{*(l)}$ est testée en utilisant le test proposé par Ecker et Kouada [39]. Ainsi, pour vérifier l'appartenance de $x^{*(l)}$ à l'ensemble $\mathcal{X}_E$ on résout le programme :

$$(T^1_{x^{*(l)}}) \begin{cases} \max \sum_{i=1}^{r} w_i \\ s.c. \\ c^i x - w_i = c^i x^{*(l)}, \quad i = 1, \ldots, r \\ x \in \mathcal{X} \cap \mathbb{Z}^n, \\ w_i \geq 0, \quad i = 1, \ldots, r \end{cases}$$

et pour tester son appartenance à $\mathcal{X}'_E$ on résout :

$$(T^2_{x^{*(l)}}) \begin{cases} \max v_1 + v_2 \\ s.c. \\ d^1 x - v_1 = d^1 x^{*(l)}, \\ d^2 x - v_2 = d^2 x^{*(l)}, \\ x \in \mathcal{X} \cap \mathbb{Z}^n \\ v_1 \geq 0, v_2 \geq 0 \end{cases}$$

D'après [39], $x^{*(l)} \in \mathcal{X}_E$ si et seulement si $(T^1_{x^{*(l)}})$ a une valeur maximale de zéro et $x^{*(l)} \in \mathcal{X}'_E$ si et seulement si la valeur maximale de $(T^2_{x^{*(l)}})$ est aussi zéro. Par conséquent si $(T^1_{x^{*(l)}})$ et $(T^2_{x^{*(2)}})$ ont une valeur maximale de zéro alors $x^{*(l)}$ appartient à $\mathcal{X}_C$.

L'algorithme 9 résume la méthode de résolution.

4.3.2 Résultats théoriques

Les outils théoriques suivants montrent que l'algorithme 9 fournit l'ensemble des solutions du programme (4.9) en un nombre fini d'itérations.

Algorithme 9 : Optimisation BiObjectif sur l'Ensemble Efficient Discret

Résultat : $\mathcal{X}_C$ l'ensemble solution de (4.9)

Initialisation $l = 0$, $\mathcal{X}_0 = \{x \in \mathbb{R}^n | Ax \leq b,\ x \geq 0\}$ et $\mathcal{X}_C = \emptyset$;

Tant qu' *il existe un nœud non-sondé* l **faire**

- résoudre (PL_l) ;
- **Si** (PL_l) *a une solution optimale* $x^{*(l)}$ **alors**
  - **Si** $x^{*(l)}$ *est entière* **alors**
    - résoudre $(T^1_{x^{*(l)}})$;
    - **Si** *la valeur optimale de* $(T^1_{x^{*(l)}})$ *est 0* **alors**
      - résoudre $(T^2_{x^{*(l)}})$;
      - **Si** *la valeur optimale de* $(T^2_{x^{*(l)}})$ *est 0* **alors**
        - $\mathcal{X}_C = \mathcal{X}_C \cup \left\{x^{*(l)}\right\}$;
      - **FinSi**
    - **FinSi**
    - construire les ensembles $\mathcal{H}_l$ et $\mathcal{H}'_l$;
    - **Si** $\mathcal{H}_l = \emptyset$ *ou* $\mathcal{H}'_l = \emptyset$ **alors**
      - Sonder le nœud l
    - **Sinon**
      - Ajouter les coupes (4.11) et (4.12) au successeurs de l ;
    - **FinSi**
  - **Sinon**
    - choisir un indice s tel que $x^{*(l)}_s$ est fractionnaire. Ensuite, diviser le programme (PL_l) en deux sous-programmes, en ajoutant respectivement les contraintes $x_s \leq \left\lfloor x^{*(l)}_s \right\rfloor$ et $x_s \geq \left\lfloor x^{*(l)}_s \right\rfloor + 1$ pour obtenir (LP_{l_1}) et (LP_{l_2}) ($l_1 \geq l+1$, $l_2 > l+1$ et $l_1 \neq l_2$) ;
  - **FinSi**
- **Sinon**
  - Sonder le nœud l ;
- **FinSi**

Fin

Théorème 4.3.1. *Supposant que* $\mathcal{H}_l \neq \emptyset$ *et* $\mathcal{H}'_l \neq \emptyset$ *pour une solution optimale entière*

$x^{*(l)}$. Si $x \neq x^{*(l)}$ et $x \in \mathcal{X}_C$ dans le domaine $\mathcal{X}_l$, alors $x \in \mathcal{X}_{l_1}$ (l_1 est le successeur de l).

Démonstration. Soit $x \neq x^{*(l)}$ une solution entière du domaine $\mathcal{X}_l$ telle que $x \notin \mathcal{X}_{l_1}$, deux cas peuvent se présenter :

— $x \notin \mathcal{X}^1_{l_1}$ ce qui veut dire $x \notin \left\{x \in \mathcal{X}_l |\ \sum_{j\in\mathcal{H}_l} x_j \geq 1\right\}$.
Donc les composantes du vecteur x vérifient les inégalités

$$\begin{aligned} \sum_{j\in\mathcal{H}_l} x_j &< 1 \\ \sum_{j\in\mathcal{N}_l\setminus\mathcal{H}_l} x_j &\geq 1, \end{aligned}$$

ce qui veut dire $x_j = 0$ pour tout $j \in \mathcal{H}_l$, et $x_j \geq 1$ pour au moins un indice $j \in \mathcal{N}_l \setminus \mathcal{H}_l$.
D'après le dernier tableau du simplexe au point $x^{*(l)}$, l'égalité suivante est vrai pour chaque objectif $i \in \{1, \ldots, r\}$:

$$c^i x = \sum_{j\in\mathcal{B}_l} \overline{c}^i_j x_j + \sum_{j\in\mathcal{N}_l} \overline{c}^i_j x_j, \quad \text{où } c^i x^{*(l)} = \sum_{j\in\mathcal{B}_l} \overline{c}^i_j x_j.$$

Donc, on peut écrire :

$$\begin{aligned} c^i x - c^i x^{*(l)} &= \sum_{j\in\mathcal{N}_l} \overline{c}^i_j x_j \\ &= \sum_{j\in\mathcal{H}_l} \overline{c}^i_j x_j + \sum_{j\in\mathcal{N}_l\setminus\mathcal{H}_l} \overline{c}^i_j x_j \\ &= \sum_{j\in\mathcal{N}_l\setminus\mathcal{H}_l} \overline{c}^i_j x_j. \end{aligned}$$

Donc, $c^i x \leq c^i x^{*(l)}$ pour tout objectif $i \in \{1, \ldots, r\}$, avec $c^i x < c^i x^{*(l)}$ pour au moins un objectif $r \in \{1, \ldots, r\}$ puisque $\overline{c}^i_j \leq 0$ pour tout $j \in \mathcal{N}_l \setminus \mathcal{H}_l$.
On conclut que la solution x n'appartient pas à l'ensemble $\mathcal{X}_E$ ce qui implique $x \notin \mathcal{X}_C$.

— $x \notin \mathcal{X}^2_{l_1}$ de la même manière ceci implique $x \in \left\{x \in \mathcal{X}_l |\ \sum_{j\in\mathcal{N}_l\setminus\mathcal{H}'_l} x_j \geq 1\right\}$. Donc les composantes du vecteur x satisfont :

$$\begin{aligned} \sum_{j\in\mathcal{H}'_l} x_j &< 1, \\ \sum_{j\in\mathcal{N}_l\setminus\mathcal{H}'_l} x_j &\geq 1, \end{aligned}$$

Il s'en suit que $x_j = 0$ pour tout $j \in \mathcal{H}'_l$, et $x_j \geq 1$ pour au moins un indice $j \in \mathcal{N}_l \setminus \mathcal{H}'_l$. D'après la table du simplexe de la solution $x^{*(l)}$, les égalités suivantes sont satisfaites :

$$\begin{aligned} d^2x &= \sum_{j\in\mathcal{B}_l} \bar{d}^2_j x_j + \sum_{j\in\mathcal{N}_l} \bar{d}^2_j x_j \quad \text{où} \quad \sum_{j\in\mathcal{B}_l} \bar{d}^2_j x_j = d^2x^{*(l)} \\ &= d^2x^{*(l)} + \sum_{j\in\mathcal{H}'_l} \bar{d}^2_j x_j + \sum_{j\in\mathcal{N}_l\setminus\mathcal{H}'_l} \bar{d}^2_j x_j \\ &= d^2x^{*(l)} + \sum_{j\in\mathcal{N}_l\setminus\mathcal{H}'_l} \bar{d}^2_j x_j. \end{aligned}$$

Donc $d^2x < d^2x^{*(l)}$ et $d^1x \leq d^1x^{*(l)}$ puisque $x^{*(l)}$ est l'optimum du programme (LP_l) alors $x \notin \mathcal{X}_{E'}$ et $x \notin \mathcal{X}_C$.

Puisque $x \notin \mathcal{X}_C$ dans les deux cas, alors x n'est pas une solution du programme (4.9).

□

Proposition 4.3.2. *Supposant que $\mathcal{H}_l = \emptyset$ ou $\mathcal{H}'_l = \emptyset$ alors le domaine $\mathcal{X}_l \cap \mathbb{Z}^n$ ne contient aucune autre solution de $\mathcal{X}_C$ autre que $x^{*(l)}$.*

Démonstration.

- Supposant $\mathcal{H}_l = \emptyset$, alors $\forall i \in \{1, \ldots, r\}$, $\forall j \in \mathcal{N}_l$, on a $\bar{c}^i_j \leq 0$ et $\exists i \in \{1, \ldots, r\}$ tel que $\bar{c}^i_j < 0, \forall j \in \mathcal{N}_l$. Donc, $x^{*(l)}$ domine tout les points du domaine $\mathcal{X}_l \cap \mathbb{Z}^n$.
- Maintenant supposant que $\mathcal{H}'_l = \emptyset$, alors $\forall j \in \mathcal{N}_l$, $\bar{d}^2_j < 0$ ou $\bar{d}^2_j = 0$ et $\bar{d}^1_j < 0$, ajouté au faite que $\bar{d}^1_j < 0$, $\forall j \in \mathcal{N}_l$ puisqu'on est à l'optimum de (PL_l), $x^{*(l)}$ devient la meilleur solution de $\mathcal{X}_l \cap \mathbb{Z}^n$.

□

Théorème 4.3.3. *L'algorithme 9 se termine en un nombre fini d'itérations et l'ensemble $\mathcal{X}_C$ contient toutes les solutions de (4.9).*

Démonstration. L'ensemble $\mathcal{X}_l \cap \mathbb{Z}^n$ de toutes les solutions réalisables du problème (4.9) étant supposé borné dénombrable, donc il contient un nombre fini de solutions. De ce faite, la cardinalité des ensembles $\mathcal{X}_E$ et $\mathcal{X}_{E'}$ et $\mathcal{X}_C$ est un nombre fini. Ainsi, l'arbre de recherche dans le domaine $\mathcal{X}_l \cap \mathbb{Z}^n$ a un nombre fini de branches et l'algorithme se termine en un nombre fini d'étapes.

Les règles de stérilisation sont utilisées sans perte d'éléments dans $\mathcal{X}_C$. À chaque étape l de l'algorithme 9, si une solution entière $x^{*(l)}$ est trouvée, les coupes éliminent $x^{*(l)}$ et toute solution dont le vecteur objectif est dominé par le vecteur critère de $x^{*(l)}$ (voir

proposition 4.3.2). La première règle de stérilisation intervient donc lorsque l'ensemble $\mathcal{H}_l$ ou $\mathcal{H}'_l$ est vide. Dans ce cas, le nœud actuel peut être sondé puisque le reste du domaine ne contient pas des solutions appartenant à $\mathcal{X}_C$. La deuxième règle de stérilisation intervient dans le cas trivial où le problème réduit devient irréalisable. □

4.3.3 Résultats expérimentaux

L'algorithme 9 a été implémenté dans l'environnement Visual Studio 2010. Les programmes linéaires et linéaires en nombre entiers sont résolus à l'aide de la librairie IBM CPLEX 12.6 pour les programmes en C++ (ILOGCPLEX). Pour effectuer les tests, nous avons utilisé un ordinateur doté d'un processeur Intel Pentium avec 2,53 GHz et de 8 Go de mémoire.

Les fonctions objectifs et les coefficients des contraintes sont uniformément distribués et générés dans des domaines différents : Chaque composante du vecteur b et les composantes des matrices A et C ont été généré aléatoirement dans distributions uniformes discrètes des intervalles [50, 100], [1, 30] et [-10, 10], respectivement. Les vecteurs d^1 et d^2 sont générés de la même manière que C.

Pour éviter la génération de problème irréalisable, toutes les contraintes de chaque problème sont du type " $\leq$ ". De plus, comme tous les coefficients de A sont positifs, une région réalisable bornée est assurée.

Les problèmes ont été regroupés en six catégories en fonction du nombre de variables, nombre de contraintes et nombre de fonctions-objectifs. Dans chaque catégorie, le nombre de fonctions-objectifs r passe de 3 à 8. Pour chaque catégorie de problèmes, 50 instances ont été résolues.

Les résultats des expérimentations obtenus sont résumés dans le tableau 4.1 où les statistiques sur le temps de calcul (temp CPU en secondes) y sont rapportées. Les deux dernières colonnes μ et ρ font référence à la moyenne de $|\mathcal{X}_E|$ et à la moyenne de $\frac{|\mathcal{X}_C|}{|\mathcal{X}_E|}$, respectivement. Les éléments et la cardinalité de $\mathcal{X}_E$ sont calculés à l'aide de l'algorithme présenté dans l'article [27].

$r\times$	$m \times n$	temp CPU (secondes) Moyen	Min.	Max.	μ	ρ
3	10× 10	0.28	0.13	0.45	35.24	0.14
	10× 20	3.34	1.78	13.43	35.10	0.14
	20× 20	8.88	0.92	63.49	65.46	0.08
	20×30	11.31	1.64	36.84	66.11	0.09
	30×30	30.44	5.55	113.53	95.98	0.06
	30× 40	84.65	31.07	288.86	94.23	0.04
	40× 40	134.57	32.89	309.45	125.80	0.04
	40×50	157.73	34.56	433.92	125.20	0.03
	50×50	202.20	68.99	479.77	154.78	0.03
	50×60	203.40	95.00	479.77	259.40	0.03
	60×60	228.61	165.60	392.78	184.92	0.02
	60×70	310.23	123.54	609.23	185.02	0.02
	70×70	390.82	81.02	713.86	214.47	0.02
	70×80	292.62	127.08	558.62	214.55	0.02
	80×80	312.25	94.37	739.46	245.54	0.02
	80×90	350.63	74.69	539.61	245.23	0.02
	90×90	312.26	104.15	498.58	273.78	0.02
	90×100	399.51	217.42	725.66	275.33	0.02
	100×100	385.82	263.61	771.68	304.80	0.01
4	10× 10	1.41	0.49	3.68	48.28	0.17
	10× 20	3.50	7.18	9.83	45.64	0.12
	20× 20	15.39	4.03	46.96	85.20	0.08
	20×30	21.08	5.37	57.29	87.10	0.03
	30×30	31.14	6.58	48.91	124.10	0.04
	30× 40	96.71	54.71	144.21	125.16	0.04
	40× 40	142.48	12.80	441.43	168.86	0.03
	40×50	165.12	10.11	597.38	165.95	0.03
	50×50	193.56	67.92	403.40	206.61	0.03
	50×60	310.21	151.512	447.21	360.00	0.05
	60×60	258.81	118.114	381.22	430.88	0.04
	60×70	335.81	81.21	830.13	430.64	0.04
	70×70	1698.84	795.42	2628.38	499.34	0.03
	70×80	1996.99	454.60	3772.34	500.41	0.04
	80×80	2330.11	1715.02	4898.02	571.23	0.04
	80×90	2962.14	2004.15	4760.58	572.24	0.04
	90×90	3424.17	365.19	7258.68	639.00	0.02
	90×100	2560.14	768.07	4888.01	641.27	0.03
	100×100	4334.21	652.04	12073.88	711.49	0.03

$r\times$	$m \times n$	temp CPU (secondes) Moyen	Min.	Max.	μ	ρ
5	10× 10	1.76	1.45	38.76	55.45	0.10
	10× 20	22.01	2.02	71.04	56.74	0.11
	20× 20	55.62	7.32	196.72	55.64	0.06
	20×30	63.37	2.01	189.53	107.15	0.06
	30×30	105.63	12.63	314.84	157.66	0.04
	30× 40	108.55	11.67	238.44	56.51	0.04
	40× 40	95.88	10.08	196.04	208.00	0.04
	40×50	171.28	67.02	255.41	209.01	0.04
	50×50	165.06	19.80	412.60	256.20	0.02
	50×60	164.41	107.78	258.46	617.69	0.08
	60×60	193.00	116.16	241.51	737.87	0.08
	60×70	133.12	47.042	187.70	737.62	0.07
	70×70	160.21	69.32	264.74	856.46	0.05
	70×80	161.47	116.61	196.78	858.25	0.06
	80×80	217.85	83.31	361.50	979.63	0.05
	80×90	109.25	15.41	194.07	977.41	0.05
	90×90	262.81	105.75	440.98	1096.48	0.04
	90×100	269.43	114.42	452.83	1218.49	0.05
	100×100	323.08	163.65	590.63	1842.00	0.04
6	10× 10	3.03	0.39	10.94	67.00	0.10
	10× 20	39.78	15.78	94.01	66.44	0.09
	20× 20	66.32	5.17	282.71	126.79	0.05
	20×30	189.16	53.06	661.83	127.35	0.06
	30×30	92.32	4.17	309.10	188.67	0.05
	30× 40	111.68	25.49	983.48	186.58	0.03
	40× 40	142.95	12.83	474.68	247.54	0.03
	40×50	228.75	15.01	568.25	201.23	0.03
	50×50	193.56	67.92	403.40	206.11	0.03
	50×60	204.42	207.59	526.59	927.27	0.11
	60×60	304.18	264.65	1518.78	1105.81	0.09
	60×70	186.89	670.78	459.89	1106.03	0.09
	70×70	183.29	224.67	282.67	1285.22	0.08
	70×80	188.49	885.67	253.62	1288.00	0.08
	80×80	207.81	318.45	431.18	1465.65	0.07
	80×90	1466.03	721.54	289.13	1686.84	0.07
	90×90	1312.65	332.48	2872.48	1643.52	0.06
	90×100	2162.81	1172.07	3149.72	1647.83	0.06
	100×100	1574.78	195.28	2751.47	1824.42	0.05

$r\times$	$m \times n$	temp CPU (secondes) Moyen	Min.	Max.	μ	ρ
7	10× 10	4.72	0.85	10.45	76.4	0.08
	10× 20	14.93	7.47	60.00	76.5	0.08
	20× 20	89.35	18.75	191.43	147.6	0.04
	20×30	81.71	9.25	293.93	146.5	0.05
	30 ×30	193.24	7.21	279.32	218.4	0.04
	30×40	124.20	22.56	252.11	217.9	0.04
	40× 40	144.99	9.14	331.40	287.9	0.03
	40× 50	149.37	52.53	317.58	287.0	0.02
	50×50	161.94	7.88	531.84	357.3	0.02
	50×60	403.40	67.92	704.35	1284.2	0.01
	60×60	758.37	115.88	1414.97	1534.5	0.01
	60×70	1076.53	336.29	1846.52	1533.8	0.01
	70×70	2028.14	195.02	4653.68	1784.0	0.09
	70×80	960.48	50.87	2577.43	1787.5	0.01
	80×80	3530.17	381.61	9114.20	2035.2	0.08
	80×90	1569.78	708.29	2460.12	2035.8	0.09
	90×90	2316.11	1028.56	4889.80	2218.8	0.07
	90×100	1974.98	315.94	7845.17	2286.2	0.07
	100×100	1824.91	511.32	4602.09	2535.8	0.07
8	10× 10	0.18	0.03	0.37	87.73	0.08
	10× 20	16.47	0.71	60.18	87.62	0.08
	20× 20	7.39	1.20	20.18	169.99	0.06
	20×30	32.92	6.88	154.09	168.85	0.05
	30×30	62.25	7.17	128.51	248.21	0.03
	30× 40	272.71	47.81	607.82	247.61	0.03
	40× 40	106.78	6.81	442.09	329.28	0.03
	40× 50	103.12	18.53	291.50	328.67	0.02
	50× 50	93.66	0.76	307.25	408.84	0.02
	50×60	292.14	196.96	398.75	1694.28	0.16
	60×60	452.22	71.77	1374.76	2022.22	0.13
	60×70	1550.77	216.31	3051.36	2023.82	0.13
	70×70	496.24	216.91	738.42	2353.24	0.11
	70×80	604.30	82.03	1948.63	2354.61	0.11
	80×80	578.00	88.23	1782.96	2684.21	0.10
	80×90	2414.12	304.49	6129.03	2685.44	0.10
	90×90	1148.57	373.37	2236.29	3011.05	0.08
	90×100	1408.70	693.38	2801.51	3016.63	0.09
	100×100	1748.84	347.71	3221.59	3346.69	0.08

D'après les résultats des expérimentations présentées dans le tableau 4.1, il apparaît que ρ ne dépasse pas un maximum de 0,17 et présente une tendance décroissante par rapport à la taille du problème. Ces expériences aléatoires illustrent l'utilité du calcul de $\mathcal{X}_C$ puisque la cardinalité de $\mathcal{X}_C$ est beaucoup plus petite que celle de $\mathcal{X}_E$, ce qui explique le caractère raisonnable des temps d'exécution.

4.4 Optimisation multiobjectif sur l'ensemble efficient

Il s'agit d'étendre ce qui a été présenté dans la section précédente à plus de deux objectifs, donc soit à résoudre :

$$\begin{cases} \max d^k x, \quad k = 1, \ldots, q \\ s.c. \\ x \in \mathcal{X}_E \end{cases} \tag{4.14}$$

où $q > 2$ et $\mathcal{X}_E$ est l'ensemble efficient du MOILP (3.4). La résolution de (4.14) fournit le sous-ensemble $\mathcal{X}_C \subset \mathcal{X}_E$. Notant donc l'ensemble efficient du problème :

$$\begin{cases} \max d^k x, \quad k = 1, \ldots, q \\ s.c. \\ x \in \mathcal{X} \cap \mathbb{Z}^n \end{cases} \tag{4.15}$$

par $\mathcal{X}'_E$.

À chaque nœud l le PL suivant est résolu en utilisant la méthode du simplexe ou dual du simplexe :

$$(PL_l) \begin{cases} \max d^1 x \\ s.c. \\ x \in \mathcal{X}_l \end{cases}$$

où $\mathcal{X}_0 = \mathcal{X}$ et $\mathcal{X}_l$ est un sous-ensemble de $\mathcal{X}$.

De la même manière, si le (PL_l) n'a pas de solution, le nœud l est sondé, sinon deux cas se présentent :

- Le premier cas est que la solution optimale $x^{*(l)}$ n'est pas entière : l'ensemble réalisable du nœud est divisé en deux sous-ensembles exclusifs et complémentaires ;
- Si la solution $x^{*(l)}$ est entière : deux ensembles $\mathcal{H}_l$ et $\mathcal{H}'_l$ sont alors construits et les coupes efficientes (4.16) et (4.17) sont éventuellement ajoutées aux nœuds successeurs de l.

$$\sum_{j \in \mathcal{H}_l} x_j \geq 1 \tag{4.16}$$

$$\mathcal{H}_l = \left\{ j \in \mathcal{N}_l | \exists i \in \{1,\dots,r\}; \bar{c}_j^i > 0 \right\} \cup \left\{ j \in \mathcal{N}_l |\ \bar{c}_j^i = 0,\ \forall i \in \{1,\dots,r\} \right\}$$

$$\sum_{j \in \mathcal{H}'_l} x_j \geq 1 \tag{4.17}$$

$$\mathcal{H}'_l = \left\{ j \in \mathcal{N}_l | \exists k \in \{1,\dots,q\}; \bar{d}_j^k > 0 \right\} \cup \left\{ j \in \mathcal{N}_l |\ \bar{d}_j^k = 0,\ \forall k \in \{1,\dots,q\} \right\}$$

Ainsi, le domaine correspondant du nœud l_1, le successeur de l, est obtenu en appliquant (4.16) et (4.17) à $\mathcal{X}_l$

$$\mathcal{X}_{l_1} = \mathcal{X}_{l_1}^1 \cap \mathcal{X}_{l_1}^2. \tag{4.18}$$

où

$$\mathcal{X}_{l_1}^1 = \left\{ x \in \mathcal{X}_l |\ \sum_{j \in \mathcal{H}_l} x_j \geq 1 \right\} \text{ et } \mathcal{X}_{l_1}^2 = \left\{ x \in \mathcal{X}_l |\ \sum_{j \in \mathcal{H}'_l} x_j \geq 1 \right\}.$$

Ensuite, la solution entière obtenue $x^{*(l)}$ est testée pour vérifier son appartenance à l'ensemble $\mathcal{X}_E$ et pour son appartenance à l'ensemble $\mathcal{X}'_E$ en résolvant :

$$(T^2_{x^{*(l)}}) \begin{cases} \max \sum_{k=1}^{q} v_k \\ s.c. \\ d^k x - v_k = d^k x^{*(l)},\ k \in \{1,\dots,q\} \\ \\ x \in \mathcal{X} \cap \mathbb{Z}^n \\ v_k \geq 0,\ k = 1,\dots,q \\ \end{cases}$$

Pareillement, les outils théoriques précédents peuvent être étendus au multiobjectif :

Théorème 4.4.1. *Supposant que $\mathcal{H}_l \neq \emptyset$ et $\mathcal{H}'_l \neq \emptyset$ pour une solution optimale entière $x^{*(l)}$. Si $x \neq x^{*(l)}$ et $x \in \mathcal{X}_C$ dans le domaine $\mathcal{X}_l$, alors $x \in \mathcal{X}_{l_1}$ (l_1 est un successeur de l).*

Démonstration. Soit $x \neq x^{*(l)}$ une solution entière du domaine $\mathcal{X}_l$ telle que $x \notin \mathcal{X}_{l_1}$, deux cas peuvent se présenter :

— $x \notin \mathcal{X}_{l_1}^1$ ce qui veut dire $x \notin \left\{ x \in \mathcal{X}_l |\ \sum_{j \in \mathcal{H}_l} x_j \geq 1 \right\}$ et la démonstration est la même que pour le théorème de la section précédente.

— $x \notin \mathcal{X}_{l_1}^2$ de la même manière ceci implique $x \in \left\{ x \in \mathcal{X}_l |\ \sum_{j \in \mathcal{N}_l \setminus \mathcal{H}'_l} x_j \geq 1 \right\}$. Donc les composantes du vecteur x vérifient les inégalités

$$\sum_{j \in \mathcal{H}'_l} x_j < 1$$
$$\sum_{j \in \mathcal{N}_l \setminus \mathcal{H}'_l} x_j \geq 1,$$

ce qui veut dire $x_j = 0$ pour tout $j \in \mathcal{H}'_l$, et $x_j \geq 1$ pour au moins un indice $j \in \mathcal{N}_l \setminus \mathcal{H}'_l$.
D'après le dernier tableau du simplexe au point $x^{*(l)}$, on a l'égalité suivante est vrai pour chaque objectif $k \in \{1, \ldots, q\}$:

$$d^k x = \sum_{j \in \mathcal{B}_l} \overline{d}^k_j x_j + \sum_{j \in \mathcal{N}_l} \overline{d}^k_j x_j, \quad \text{où } d^k x^{*(l)} = \sum_{j \in \mathcal{B}_l} \overline{d}^k_j x_j.$$

Donc, on peut écrire :

$$\begin{aligned} d^k x - d^k x^{*(l)} &= \sum_{j \in \mathcal{N}_l} \overline{d}^k_j x_j \\ &= \sum_{j \in \mathcal{H}'_l} \overline{d}^k_j x_j + \sum_{j \in \mathcal{N}_l \setminus \mathcal{H}'_l} \overline{d}^k_j x_j \\ &= \sum_{j \in \mathcal{N}_l \setminus \mathcal{H}'_l} \overline{d}^k_j x_j. \end{aligned}$$

Donc, $d^k x \leq d^k x^{*(l)}$ pour tout critère $k = 1, \ldots, q$, avec $d^k x < d^k x^{*(l)}$ pour au moins un critère $k \in \{1, \ldots, q\}$ puisque $\overline{d}^k_j \leq 0$ pour tout $j \in \mathcal{N}_l \setminus \mathcal{H}'_l$.
On conclut que la solution x n'appartient pas à l'ensemble $\mathcal{X}'_E$ ce qui implique $x \notin \mathcal{X}_C$.
Puisque $x \notin \mathcal{X}_C$ dans les deux cas, alors x n'est pas solution du programme (4.14).

□

4.5 Conclusion

Les modèles d'optimisation sur l'ensemble efficient actuels présentent deux principales limites, la première est de ne prendre en compte qu'une seule préférence et la deuxième est de donner une seule solution alors que plusieurs solutions sont de meilleur compromis, quant est-il lorsque le décideur exprime plusieurs préférences ou bien qu'il y ait plusieurs décideurs ? Pour répondre à cette question, on a présenté dans ce chapitre une méthode exacte pour résoudre un nouveau type de problème qui a un réel intérêt pratique et qui peut modéliser plusieurs situations réelles. Ainsi, nous avons présenté une contribution qui introduit un nouveau type de problème dans la classe des problèmes d'optimisation sur l'ensemble efficient qui est l' « optimisation multiobjectif sur l'ensemble efficient » et on a proposé un algorithme pour la résolution de ce nouveau problème. Cette contribution a fait l'objet d'une publication dans le journal "*Journal of Industrial and Management Optimization*" (DOI : 10.3934/jimo.2019102).

Chapitre 5

Optimisation d'un Programme Linéaire plus Linéaire Fractionnaire Discret à Objectif Multiple

Sommaire

5.1 Introduction

Les programmes linéaires plus linéaires fractionnaires à objectif multiple sont des problèmes NP-difficiles. Les articles traitant ce problème ne proposent pas une méthode exacte capable de générer une solution efficiente dans le contexte général du problème, les quelques papiers de la littérature ne traitent que la version flou du problème sans pour autant donner une procédure qui assure l'efficience des solutions trouvées. Ainsi S. Pramanik et al. ont traité le problème dans sa version flou par la technique de programmation par but [87], alors que dans [91], les auteurs ont proposé une méthode qui dérive une solution supposée efficiente de l'optimum de la transformation de Tchebychev des objectifs. Dans [101], les auteurs proposent de transformer le problème en programme linéaire multiobjectif en utilisant les séries de Taylor de premier ordre, puis ont résolu le problème en agrégeant le programme par la méthode des poids pondérés .

Pour la principale contribution de cette thèse, on propose une méthode pour générer l'ensemble des solutions efficientes d'un programmes linéaire plus linéaire fractionnaire discret a objectif multiple par une méthode d'exploration **B&B** combinée à une technique de troncature qui permet de retirer les solutions inefficientes de la recherche.

Un programme linéaire plus linéaire fractionnaire discert à objectif multiple (MultiObjective Integer Linear plue Linear Fractional Program **MOILLFP**) peut être formulé d'une manière générique par :

$$(MOILLFP)\begin{cases} \max Z_i(x) = \quad h^i x + \dfrac{c^i x + \alpha^i}{d^i x + \beta^i},\ i = 1, \ldots, r \\ s.c. \\ x \in \mathcal{X} \cap \mathbb{Z}^n \end{cases} \tag{5.1}$$

où $r \geq 2$; c^i, d^i et h^i sont des $n-$vecteurs colonnes, α^i, β^i sont des scalaires pour tout $i \in \{1, 2, \ldots, r\}$; $\mathcal{X} = \{x \in \mathbb{R}^n | Ax \leq b, x \geq 0\}$; A est une $m \times n$ matrice de réels et $b \in \mathbb{R}^m$. Dans ce qui suit on suppose que $\mathcal{X}$ est un polyèdre compact non-vide dans $\mathbb{R}^n$ et que $\mathcal{D} = \mathcal{X} \cap \mathbb{Z}^n$ est non-vide, on suppose aussi que $d^i x + \beta^i > 0$ sur $\mathcal{X}$ pour tout $i \in \{1, 2, \ldots, r\}$.

Ce chapitre est organisé comme suit : après cette introduction, la section 5.2 suivante décrit la méthode de résolution et l'algorithme, la section 5.3 contient quelques résultats théoriques, la section 5.4 est réservée à un exemple illustratif et la section 5.5 donne quelques remarques conclusives.

5.2 Description de l'algorithme de résolution

Afin de générer l'ensemble efficient d'un MOILLFP, nous proposons de suivre un processus de **B&B** combiné à une méthode de troncature qui tire parti du lien entre le MOILLFP et le **MOILFP** suivant :

$$(MOILFP)\begin{cases} \max g_i(x) \quad = \quad h^i x,\ i = 1, \ldots, r \\ \max f_i(x) \quad = \quad \dfrac{f_i^1(x)}{f_i^2(x)},\ \ i = 1, \ldots, r \\ s.c. \\ x \in \mathcal{X} \cap \mathbb{Z}^n \end{cases} \tag{5.2}$$

où $f_i^1(x) = c^i x + \alpha^i, f_i^2(x) = d^i x + \beta^i$ pour tout $i \in \{1, 2, \ldots, r\}$.

Théorème 5.2.1. *Toute solution efficiente $x^* \in \mathcal{X} \cap \mathbb{Z}^n$ de (5.1) est également une solution efficiente de (5.2).*

Démonstration. Soit x^* une solution efficiente de (5.1) et supposant que x^* est une solution dominée de (5.2) alors

$$\exists \bar{x} \in \mathcal{X} \cap \mathbb{Z}^n : \forall i \in \{1, \ldots, r\},\ g_i(\bar{x}) \geq g_i(x^*) \text{ et } f_i(\bar{x}) \geq f_i(x^*)$$

et $g_{i_0}(\bar{x}) > g_{i_0}(x^*)$ ou$f_{i_0}(\bar{x}) > f_{i_0}(x^*)$ pour au moins un $i_0 \in \{1, \ldots, r\}$.

Ainsi, en additionnant terme par terme, on peut écrire :

$$\exists \bar{x} \in \mathcal{X} \cap \mathbb{Z}^n : \forall i \in \{1, 2, \ldots, r\},\ g_i(\bar{x}) + f_i(\bar{x}) \geq g_i(x^*) + f_i(x^*)$$

et $g_{i_0}(\bar{x}) + f_{i_0}(\bar{x}) > g_{i_0}(x^*) + f_{i_0}(x^*)$ pour au moins un $i_0 \in \{1, 2, \ldots, r\}$.

$$\Longrightarrow \exists \bar{x} \in \mathcal{X} \cap \mathbb{Z}^n : \forall i \in \{1, \ldots, r\},\ Z_i(\bar{x}) \geq Z_i(x^*)$$

et $Z_{i_0}(\bar{x}) > Z_{i_0}(x^*)$ pour au moins un $i_0 \in \{1, \ldots, r\}$.

□

Ce dernier résultat garantit que l'ensemble efficient de (5.1) noté $\mathcal{S}_E$ est contenu dans l'ensemble efficient du programme (5.2) noté $\mathcal{S}'_E$ ($\mathcal{S}_E \subset \mathcal{S}'_E$). Puisque, la contraposée de ce théorème est vrai alors chaque solution dominée de (5.2) est aussi dominée de (5.1).

La méthode de résolution proposée parcourt toutes les solutions de l'ensemble $\mathcal{S}'_E$, et pour cela l'algorithme proposé suit la même stratégie d'exploration que la méthode proposée par M. E-A Chergui et M. Moulaï pour résoudre les MOILFP [26]. Leur algorithme consiste en une méthode exacte basée sur le processus de **B&C** utilisant une coupe efficiente. La coupe est dite efficiente car elle permet de retirer de l'exploration des solutions entières dominées. Cette méthode peut être résumée comme suit :

Le processus de branchement La recherche de solutions efficientes est une exploration implicite structurée en arbre où n'importe quelle fonction g_i ou f_i $i \in \{1, ..., r\}$ peut être choisie pour être optimisée à chaque nœud. Le choix d'une fonction g_i ou f_i ne changera pas l'ensemble des résultats $\mathcal{S}_E$ mais modifiera l'arbre de recherche et changera l'évolution de $\mathcal{S}_E$ mais pas le résultat final.

Les fonctions g_i sont linéaires et les fonction f_i sont linéaires fractionnaires, Il est possible de choisir une fonction linéaire fractionnaire et d'utiliser une procédure

du simplexe pour la résolution [75, 19].

Ainsi, à chaque nœud l de l'arbre la fonction linéaire g_1 est optimisée sur le sous-domaine correspondant au nœud $\mathcal{X}_l$.

$$(P_l) \begin{cases} \max & g_1(x) \\ s.c. & x \in \mathcal{X}_l \end{cases}$$

Après avoir résolu le programme, trois cas peuvent se présenter :

(P_l) **n'a pas de solution** Comme pour un processus classique **B&B** le nœud n'a pas de descendant et il est sondé.

(P_l) **a une solution non-entière** $x^{*(l)}$ Dans ce cas, l'approche est la même que celle du **B&B** pour la programmation en nombres entiers, ce qui signifie que le nœud a deux descendants l_1 et l_2 tels que : Soit j un indice où $x_j^{*(l)}$ n'est pas un entier alors pour l_1 on met $\mathcal{X}_{l_1} = \left\{x \in \mathcal{X}_l | x_j \leq \left\lfloor x_j^{*(l)} \right\rfloor\right\}$ et pour l_2, $\mathcal{X}_{l_2} = \left\{x \in \mathcal{X}_l | x_j \geq \left\lceil x_j^{*(l)} \right\rceil\right\}$.

(P_l) **a une solution entière** $x^{*(l)}$ Cette solution est ensuite comparée aux solutions de l'ensemble $\mathcal{S}_E$ et l'ensemble $\mathcal{S}_E$ est donc mis à jour : Si le vecteur critère de $x^{*(l)}$ domine le vecteur critère de l'une des solutions de $\mathcal{S}_E$, cette solution est éjectée. Si aucun vecteur critère des solutions de $\mathcal{S}_E$ ne domine le vecteur critère de $x^{*(l)}$ alors $x^{*(l)}$ est ajouté à $\mathcal{S}_E$.

Le nœud dans ce cas n'a pas de descendant ou en a un seul selon la direction de croissance des fonctions g_i, f_i $i \in \{2, \ldots, r\}$.

La méthode de troncature Si à un certain nœud l la solution de (P_l) est entière, alors le domaine restant peut contenir des solutions efficientes. La méthode de troncature utilisée permet de supprimer les solutions dont le vecteur critère est dominé par le vecteur critère de $x^{*(l)}$.

Soit donc $x^{*(l)}$ une solution entière obtenue après avoir résolu par une procédure simplexe le programme (P_l) et soit $\mathcal{B}_l$ et $\mathcal{N}_l$ les ensembles des indices de variables de base et de variables hors-base respectivement de $x^{*(l)}$. Dans ce qui suit, les notations suivantes sont utilisées :

$$\boldsymbol{\eta^{i(l)}} = h^i - h^i_{\mathcal{B}_l} \mathcal{B}_l^{-1} A,$$

$$\nu^{i(l)} = c^i - c^i_{\mathcal{B}_l} \mathcal{B}_l^{-1} A,$$

$$\mu^{i(l)} = d^i - d^i_{\mathcal{B}_l}\mathcal{B}_l^{-1}A.$$

Définissant la direction de croissance de chaque fonction-objectif g_i, f_i $i \in \{1, \ldots, r\}$ pour une solution de base $x^{*(l)}$ comme suit :

$$\boldsymbol{\rho^{t(l)}} = \begin{cases} \boldsymbol{\eta^{t(l)}} & \text{si } t \in \{1, \ldots, r\} \\ \boldsymbol{\gamma^{t-r(l)}} & \text{si } t \in \{r+1, \ldots, 2r\} \end{cases}$$

où $\boldsymbol{\gamma^{i(l)}} = f_i^2(x^{*(l)})\nu^{i(l)} - f_i^1(x^{*(l)})\mu^{i(l)}$, indique le gradient réduit de la fonction f_i, $\quad i = 1, \ldots, r$.

Dans la section suivante, nous allons prouver que $\boldsymbol{\rho^{t(l)}}$ définit réellement une direction de croissance pour la fonction g_t et que la contrainte

$$\sum_{j \in \mathcal{H}_l} x_j \geq 1 \tag{5.3}$$

où

$$\mathcal{H}_l = \left\{ j \in \mathcal{N}_l | \exists t \in \{2, ..., 2r\}; \boldsymbol{\rho}_j^{t(l)} > 0 \right\} \cup \left\{ j \in \mathcal{N}_l | \boldsymbol{\rho}_j^{t(l)} = 0 \ , \forall t \in \{1, \ldots, 2r\} \right\}$$

est une coupe valide qui permet d'éliminer les solutions qui sont dominées par $x^{*(l)}$. Ainsi, après avoir trouvé une solution entière, nous mettons à jour $\mathcal{S}_E$ et calculons $\mathcal{H}_l$. Si $\mathcal{H}_l = \emptyset$ cela signifie que le domaine restant ne contient aucune solution efficiente et que le nœud l peut être sondé (démontré dans la proposition 5.3.3). Au contraire, si $\mathcal{H}_l \neq \emptyset$ le nœud l a un successeur l_0 avec $\mathcal{X}_{l_0} = \left\{ x \in \mathcal{X}_l | \sum_{j \in \mathcal{H}_l} x_j \geq 1 \right\}$.

Ainsi, il existe deux règles de stérilisation de nœuds, la première règle est lorsque le programme correspondant (P_l) n'est pas réalisable et la seconde est lorsque $\mathcal{H}_l = \emptyset$.

Toute cette procédure peut être résumé par l'algorithme suivant :

Algorithme 10 : Résolution d'un programme linéaire plus linéaire fractionnaire discret à objectif multiple

Résultat : Un ensemble $\mathcal{S}_E$ contenant toutes les solutions efficientes du programme (5.1)

Initialisation $l = 0$,$\mathcal{X}_l = \{x \in \mathbb{R}^n | Ax \leq b \text{ et } x \geq 0\}$ et $\mathcal{S}_E = \emptyset$;

Tant qu' *il y a un nœud non sondé* l **faire**

résoudre (P_l) en utilisant la méthode du simplexe ou dual simplexe ;

$$(P_l) \begin{cases} \max & g_1(x) \\ s.c. & x \in \mathcal{X}_l \end{cases}$$

si (P_l) *a une solution optimale* $x^{*(l)}$ **alors**

si $x^{*(l)}$ *est une solution entière* **alors**

Mettre-à-jour $\mathcal{S}_E$;

Construire l'ensemble $\mathcal{H}_l$;

si $\mathcal{H}_l = \emptyset$ **alors**

Sonder le nœud l

sinon

Ajouter la coupe (5.3) aux successeurs de l ;

fin

sinon

Choisir un indice j tel que $x_j^{*(l)}$ est fractionnaire. Ensuite, diviser le programme (P_l) en deux sous-programmes en ajoutant respectivement les contraintes $x_j \leq \left\lfloor x_j^{*(l)} \right\rfloor$ et $x_j \geq \left\lceil x_j^{*(l)} \right\rceil$ pour obtenir (P_{l_1}) et (P_{l_2}) ($l_1 >\geq l+1$, $l_2 > l+1$ et $l_1 \neq l_2$) ;

fin

sinon

Sonder le nœud l ;

fin

Fin

5.3 Résultats théoriques

Les résultats théoriques suivants montrent que l'algorithme 10 trouve toutes les solutions efficientes de (5.1) en un nombre fini d'itérations.

Théorème 5.3.1. *Supposant que pour la solution entière actuelle $x^{*(l)}$, $\mathcal{H}_l \neq \emptyset$. Si x est une solution efficiente entière dans le domaine $\mathcal{X}_l \setminus \{x^{*(l)}\}$, alors $x \in \mathcal{X}_{l+1} = \left\{x \in \mathcal{X}_l | \sum_{j\in\mathcal{H}_l} x_j \geq 1\right\}$.*

Démonstration. Soit x une solution entière dans l'ensemble $\mathcal{X}_l \setminus \left\{x^{*(l)}\right\}$ telle que $x \notin \mathcal{X}_{l+1}$, alors $x \notin \{x \in \mathcal{X}_l | \sum\limits_{j\in H_l} x_j \geq 1\}$, cela implique

$$x \in \{x \in \mathcal{X}_l | \sum_{j\in\mathcal{H}_l} x_j < 1\}.$$

Par conséquent, les inégalités suivantes sont satisfaites :

$$\begin{aligned} \sum_{j\in\mathcal{H}_l} x_j &< 1 \\ \sum_{j\in\mathcal{N}_l\setminus\mathcal{H}_l} x_j &\geq 1. \end{aligned}$$

Il s'ensuit que $x_j = 0$ pour tous $j \in \mathcal{H}_l$ et $x_j \geq 1$ pour au moins un indice $j \in \mathcal{N}_l \setminus \mathcal{H}_l$.

À partir du tableau simplexe optimal correspondant à la solution $x^{*(l)}$, la valeur actualisée de chaque fonction objectif est écrite pour l'indices hors-base $j \in \mathcal{N}_l$ comme suit :

$$\begin{aligned} g_i(x) &= g_i(x^{*(l)}) + \boldsymbol{\delta}_j(h^i_j - h^i_{\mathcal{B}_l}\mathcal{B}_l^{-1}A_j), \\ f^1_i(x) &= f^1_i(x^{*(l)}) + \boldsymbol{\delta}_j(c^i_j - c^i_{\mathcal{B}_l}\mathcal{B}_l^{-1}A_j), \\ f^2_i(x) &= f^2_i(x^{*(l)}) + \boldsymbol{\delta}_j(d^i_j - d^i_{\mathcal{B}_l}\mathcal{B}_l^{-1}A_j), \end{aligned}$$

où $\boldsymbol{\delta}_j = \dfrac{x^{*(l)}_{\mathcal{B}_{l_{(s)}}}}{A^s_j} = \min\limits_{i=1,\ldots,m}\left\{\dfrac{x^{*(l)}_{\mathcal{B}_{l_{(i)}}}}{A^i_j} | \ A^i_j > 0\right\}$.

Ainsi, nous avons pour tous les $i \in \{1,\ldots,r\}$:

$$g_i(x) = g_i(x^{*(l)}) + \boldsymbol{\delta}_j\eta^{i(l)}_j,$$

$$f_i(x) = \frac{f^1_i(x)}{f^2_i(x)} = \frac{f^1_i(x^{*(l)}) + \boldsymbol{\delta}_j\nu^{i(l)}_j}{f^2_i(x^{*(l)}) + \boldsymbol{\delta}_j\mu^{i(l)}_j},$$

Donc :

$$g_i(x) - g_i(x^{*(l)}) = \boldsymbol{\delta_j}\eta_j^{i(l)}$$

et

$$\begin{aligned} f_i(x) - f_i(x^{*(l)}) &= \frac{f_i^1(x^{*(l)}) + \boldsymbol{\delta}_j\nu_j^{i(l)}}{f_i^2(x^{*(l)}) + \boldsymbol{\delta}_j\mu_j^{i(l)}} - \frac{f_i^1(x^{*(l)})}{f_i^2(x^{*(l)})} \\ &= \boldsymbol{\delta}_j \frac{\left[f_i^2(x^{*(l)})\nu_j^{i(l)} - f_i^1(x^{*(l)})\mu_j^{i(l)}\right]}{f_i^2(x^{*(l)})\left[f_i^2(x^{*(l)}) + \boldsymbol{\delta}_j\mu_j^{i(l)}\right]} \end{aligned}$$

Puisque nous avons déjà désigné $\boldsymbol{\rho^{t(l)}}$ comme le vecteur de direction de croissance du critère t du programme (5.2) et $\boldsymbol{\gamma}^i$ comme le vecteur gradient réduit pour les fonctions fractionnaires linéaires de ce programme :

$$\boldsymbol{\gamma}^i = f_i^2(x^{*(l)})\nu^{i(l)} - f_i^1(x^{*(l)})\mu^{i(l)}$$

Comme les composantes $\boldsymbol{\rho_j^{t(l)}} \leq 0$, pour chaque indice $j \in \mathcal{N}_l \backslash \mathcal{H}_l$ alors

$$g_i(x) - g_i(x^{*(l)}) \leq 0, \quad i = 1, \ldots, r$$

$$f_i(x) - f_i(x^{*(l)}) \leq 0, \quad i = 1, \ldots, r$$

et pour au moins un critère, il est strictement inférieur à zéro, donc $\exists i \in \{1, \ldots, r\}$, $g_i(x) - g_i(x^{*(l)}) < 0$ ou $f_i(x) - f_i(x^{*(l)}) < 0$.
Par conséquent, en additionnant terme par terme les g_i et les f_i on obtient $Z_i(x) \leq Z_i(x^{*(l)})$ pour tous les critères $i \in \{1, ..., r\}$, avec $Z_i(x) < Z_i(x^{*(l)})$ pour au moins un critère $i \in \{1, ..., r\}$.

□

Corollaire 5.3.2. *La contrainte* $\sum_{j \in \mathcal{H}_l} x_j \geq 1$ *définie une coupe valide.*

Démonstration. Ceci est visible car $\sum_{j \in \mathcal{H}_l} x_j \geq 1$ est une contrainte valide en vertue du théorème ci-dessus, puisque toutes les solutions efficientes dans le domaine courant $\mathcal{X}_l$ remplissent cette contrainte. En outre, la solution entière actuelle $x^{*(l)}$ne satisfait pas cette contrainte puisque $x_j = 0$ pour tous les $j \in \mathcal{H}_l$. □

Proposition 5.3.3. *Si* $\mathcal{H}_l = \emptyset$ *pour la solution entière courante* $x^{*(l)}$*, alors* $\mathcal{X}_l \setminus \left\{x^{*(l)}\right\}$ *ne contient aucune solution efficiente.*

Démonstration. $\mathcal{H}_l = \emptyset$ veut dire que $x^{*(l)}$ est une solution entière optimale pour tout les critères, donc $x^{*(l)}$ est un point idéal dans le sous-domaine courant $\mathcal{X}_l$ par conséquent $\mathcal{X}_l \setminus \left\{x^{*(l)}\right\}$ ne contient aucune autre solution efficiente. □

Théorème 5.3.4. *L'algorithme 10 trouve toute les solutions efficientes du programme (5.1) en un nombre fini d'itérations.*

Démonstration. Chaque fois qu'une solution optimale entière $x^{*(l)}$ est trouvée, la coupe efficiente est ajoutée et l'ensemble $\mathcal{S}_E$ est mis à jour. Ainsi, selon le théorème 5.3.1 et son corollaire 5.3.2, au moins la solution $x^{*(l)}$ est éliminée du domaine au nœud courant mais aucune solution efficiente n'est omise, par conséquent chaque solution efficiente de (5.2) est parcourue et ainsi tous les éléments de $\mathcal{S}_E$ sont trouvés (puisque $\mathcal{S}_E \subset \mathcal{S}'_E$).

De plus, comme $\mathcal{D}$ l'ensemble des solutions entières réalisables de MOILLFP, est un ensemble fini borné non-vide, la cardinalité de l'ensemble $\mathcal{S}'_E$ est également un nombre fini et il en va de même pour $\mathcal{S}_E$, et comme l'algorithme 10 parcourt tous les éléments de $\mathcal{S}_E$, il s'agit d'un algorithme fini.

□

5.4 Exemple Illustratif

Soit à résoudre le MOILLFP suivant :

$$\begin{cases} \max & 2x_1 & +x_2 & +\dfrac{x_2+3}{x_1+3x_2+5} \\ \max & -x_1 & +3x_2 & +\dfrac{4x_1-2x_2}{2x_1+3} \\ s.t: & & & \\ & 2x_1 & +x_2 & \leq 18 \\ & 3x_1+x_2 \leq 24 & & \\ 1 \leq & x_1 & \leq 5 & \\ 10 \leq & x_2 & \leq 18 & \\ x_1, & x_2, & \text{sont des entiers} & \end{cases}$$

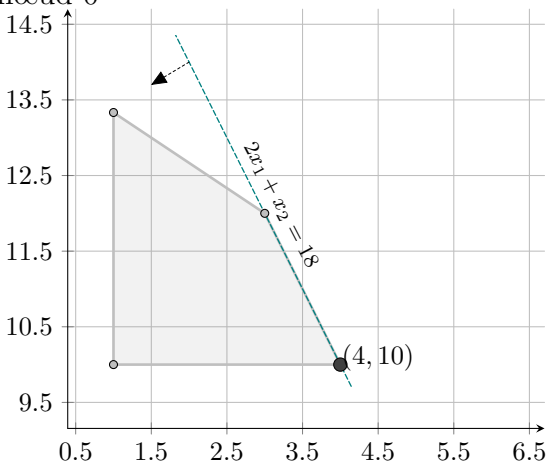
Initialisation : Poser $l = 0$, $\mathcal{X}_0 = \{(x_1, x_2) \in \mathbb{R}^2 | 2x_1 + x_2 \leq 18,\ 3x_1 + x_2 \leq 24,\ 1 \leq x_1 \leq 5,\ 10 \leq x_2 \leq 18\ \}$.

Nœud 0 : Le graphique 5.1 représente le domaine $\mathcal{X}_0$ et la solution optimale obtenue après résolution du PL $\{\max 2x_1 + x_2 | x \in \mathcal{X}_0\}$.

TABLE 5.1 – Table du simplexe du nœud 0.

$\mathcal{B}_0$	x_3	x_7		RHS
x_1	1/2	1/2		4
x_2	0	−1		10
x_4	−1	2		4
x_5	−3/2	−1/2		2
x_6	1/2	1/2		3
x_8	−1/2	−1/2		1
x_9	0	1		8
$\boldsymbol{\rho^{1(0)} = \eta^{1(0)}}$	−1	0	g_1=	18
$\nu^{1(0)}$	0	1		13
$\mu^{1(0)}$	−1/2	5/2		39
$\nu^{2(0)}$	−2	−4		−4
$\mu^{2(0)}$	−1	−1		11
$\boldsymbol{\rho^{2(0)} = \eta^{2(0)}}$	1/2	7/2	g_2=	26
$\boldsymbol{\rho^{3(0)}}$	13/2	13/2	f_1=	1/3
$\boldsymbol{\rho^{4(0)}}$	−26	−48	f_2 =	-4/11

FIGURE 5.1 – Illustration graphique du nœud 0



La solution trouvée $(4, 10)$ est entière donc $\mathcal{S}_E = \{(4, 10)\}$, A partir de la table du simplexe on trouve $\mathcal{H}_0 = \{3, 7\}$. la coupe $x_3 + x_7 \geq 1$ est ajouté ce qui est équivalent à la contrainte additionnelle $x_1 \leq 7/2$.

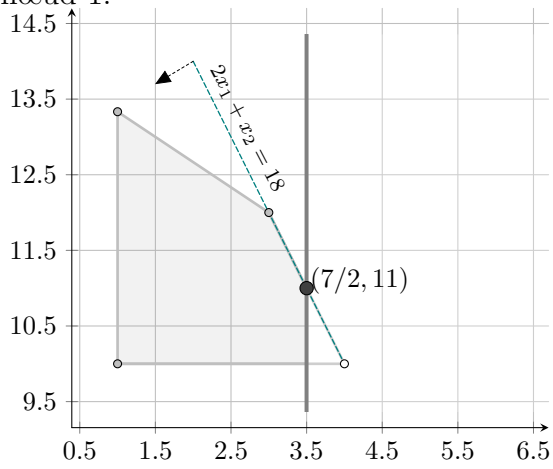
Nœud 1 : Après avoir ajouté la coupe $x_3 + x_7 \geq 1$ à la table correspondante au nœud 0 on obtient la représentation graphique 5.2.

$(\frac{7}{2}, 11)$ n'est pas entière donc le nœud a deux successeurs.

TABLE 5.2 – Table du simplexe du nœud 1.

$\mathcal{B}_1$	x_3	x_{10}		RHS
x_1	0	1/2		7/2
x_2	1	−1		11
x_4	−3	2		2
x_5	−1	−1/2		5/2
x_6	0	1/2		5/2
x_7	1	−1		1
x_8	0	−1/2		3/2
x_9	−1	1		7
$\boldsymbol{\rho^{1(1)} = \eta^{1(1)}}$	0	−1	g_1=	18
$\nu^{1(1)}$	−1	1		14
$\mu^{1(1)}$	−3	5/2		83/2
$\nu^{2(1)}$	2	−4		−8
$\mu^{2(1)}$	0	−1		10
$\boldsymbol{\rho^{2(1)} = \eta^{2(1)}}$	−3	7/2	g_2=	59/2
$\boldsymbol{\rho^{3(1)}}$	167/2	13/2	f_1=	28/83
$\boldsymbol{\rho^{4(1)}}$	20	−48	f_2 =	-4/5

FIGURE 5.2 – Illustration graphique du nœud 1.



Nœud 2 : Après avoir ajouté la contrainte $x_1 \leq \left\lfloor \frac{7}{2} \right\rfloor$ la table 5.3 est obtenue.

La solution $(3, 12)$ est entière et son vecteur critère domine celui de la solution se trouvant dans $\mathcal{S}_E$ donc $\mathcal{S}_E = \{(3, 12)\}$ et $\mathcal{H}_2 = \{4, 11\}$.

Nœud 3 : Après avoir ajouté la contrainte $x_1 \geq \left\lceil \frac{7}{2} \right\rceil$ le programme devient irréalisable.

Nœud 4 : La table 5.4 est la table du simplexe correspondante à ce nœud.

TABLE 5.3 – Table du simplexe du nœud 2.

$\mathcal{B}_2$	x_4	x_{11}		RHS
x_1	0	1		3
x_2	1/3	−1/6		12
x_3	−1/3	−4/3		0
x_5	−1/3	−7/3		3
x_6	0	1		2
x_7	1/3	−1/6		2
x_8	0	−1		2
x_9	−1/3	1/6		6
x_{10}	0	−2		1
$\boldsymbol{\rho}^{1(2)} = \boldsymbol{\eta}^{\mathbf{1(2)}}$	−1/3	−4/3	$g_1=$	18
$\nu^{1(2)}$	−1/3	1/6		15
$\mu^{1(2)}$	−1	1		44
$\nu^{2(2)}$	1/6	−16/3		−12
$\mu^{2(2)}$	0	−2		9
$\boldsymbol{\rho}^{2(2)} = \boldsymbol{\eta}^{\mathbf{2(2)}}$	−1	3	$g_2=$	33
$\boldsymbol{\rho}^{3(2)}$	1/3	−23/3	$f_1=$	15/44
$\boldsymbol{\rho}^{4(2)}$	3/2	−72	$f_2 =$	-4/3

FIGURE 5.3 – Illustration graphique du nœud 2.

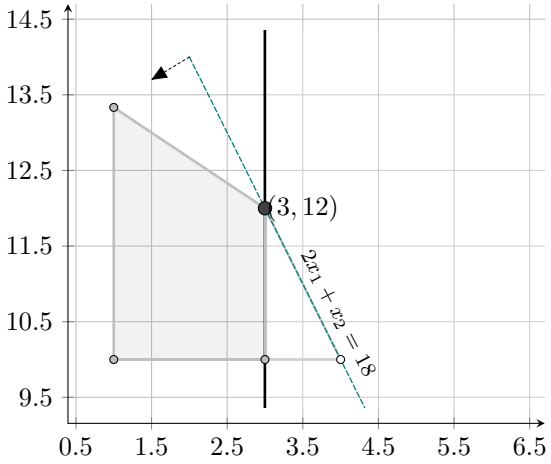
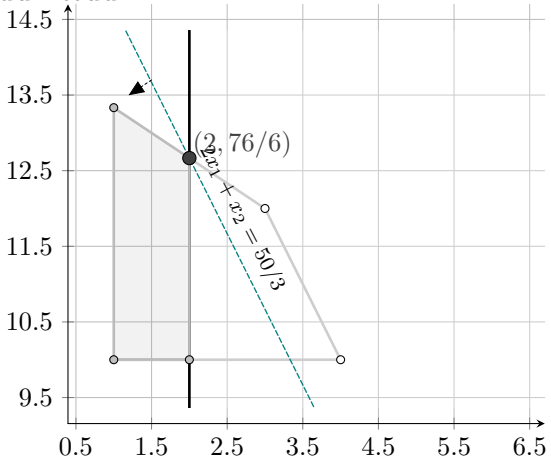


TABLE 5.4 – Table du simplexe du nœud 4.

$\mathcal{B}_4$	x_4	x_{12}		RHS
x_1	0	1		2
x_2	1/3	−1/6		76/6
x_3	−1/3	−4/3		4/3
x_5	−1/3	−7/3		16/3
x_6	0	1		1
x_7	1/3	−1/6		16/6
x_8	0	−1		3
x_9	−1/3	1/6		16/3
x_{10}	0	−2		3
x_{11}	0	−1		1
$\boldsymbol{\rho}^{1(4)} = \boldsymbol{\eta}^{\mathbf{1(4)}}$	−1/3	−4/3	$g_1=$	50/3
$\nu^{1(4)}$	−1/3	1/6		47/3
$\mu^{1(4)}$	−1	1		45
$\nu^{2(4)}$	1/6	−16/3		−52/3
$\mu^{2(4)}$	0	−2		7
$\boldsymbol{\rho}^{2(4)} = \boldsymbol{\eta}^{\mathbf{2(3)}}$	−1	3	$g_2=$	36
$\boldsymbol{\rho}^{3(4)}$	92/3	−49/6	$f_1=$	8/23
$\boldsymbol{\rho}^{4(4)}$	7/6	−72	$f_2 =$	-52/21

FIGURE 5.4 – Illustration graphique du nœud 4.



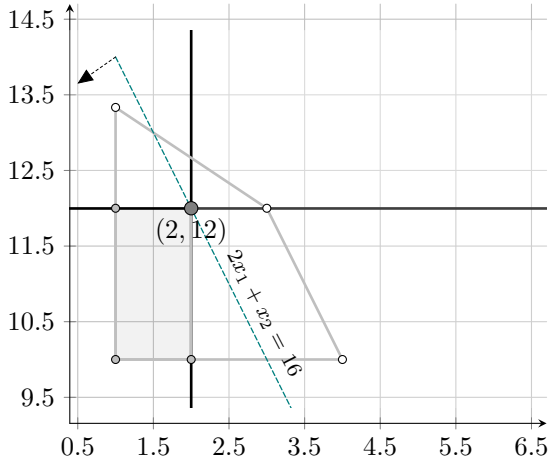
la solution trouvée après résolution $(2, \frac{76}{6})$ n'est pas entière et un branchement classique est suivie créant les deux nœuds 5 et 6.

Nœud 5 : Après avoir ajouté la contrainte $x_2 \leq \left\lfloor \frac{76}{6} \right\rfloor$ on obtient la table 5.5.

TABLE 5.5 – Table du simplexe du nœud 5.

$\mathcal{B}_5$	x_{12}	x_{13}		RHS
x_1	1	0		2
x_2	0	1		12
x_3	-2	-1		2
x_4	-2	-3		2
x_5	-3	-1		6
x_6	1	0		1
x_7	0	1		2
x_8	-1	0		3
x_9	0	-1		6
x_{10}	-2	0		3
x_{11}	-1	0		1
$\boldsymbol{\rho^{1(5)} = \eta^{1(5)}}$	-2	-1	$g_1 =$	16
$\nu^{1(5)}$	0	-1		15
$\mu^{1(5)}$	-1	-3		43
$\nu^{2(5)}$	-4	2		-16
$\mu^{2(5)}$	-2	0		7
$\boldsymbol{\rho^{2(5)} = \eta^{2(5)}}$	1	-3	$g_2 =$	34
$\boldsymbol{\rho^{3(5)}}$	15	2	$f_1 =$	15/43
$\boldsymbol{\rho^{4(5)}}$	-60	14	$f_2 =$	-16/7

FIGURE 5.5 – Illustration graphique du nœud 5.



$(2, 12)$ est entière donc $\mathcal{S}_E = \{(3, 12), (2, 12)\}$, $\mathcal{H}_5 = \{12, 13\}$.

Nœud 6 : Après avoir ajouté la contrainte $x_2 \geq \left\lceil \frac{75}{6} \right\rceil$ la table 5.6 résulte. $(\frac{3}{2}, 13)$ est la solution trouvée, les nœuds 8 et 9 sont créés.

TABLE 5.6 – Table du simplexe du nœud 6.

$\mathcal{B}_6$	x_4	x_{13}		RHS
x_1	1/2	3/2		3/2
x_2	0	−1		13
x_3	−1	−2		2
x_5	−3/2	−7/2		13/2
x_6	1/2	3/2		1/2
x_7	0	−1		3
x_8	−1/2	−3/2		7/2
x_9	0	1		5
x_{10}	−1	−3		4
x_{11}	−1/2	−3/2		3/2
x_{12}	−1/2	−3/2		1/2
$\boldsymbol{\rho}^{1(6)} = \boldsymbol{\eta}^{\mathbf{1(6)}}$	−1	0	g_1=	16
$\nu^{1(6)}$	0	1		16
$\mu^{1(6)}$	−1/2	3/2		91/2
$\nu^{2(6)}$	−2	−8		−20
$\mu^{2(6)}$	−1	−3		6
$\boldsymbol{\rho}^{2(6)} = \boldsymbol{\eta}^{\mathbf{2(6)}}$	1/2	9/2	g_2=	75/2
$\boldsymbol{\rho}^{3(6)}$	8	43/3	f_1=	32/91
$\boldsymbol{\rho}^{4(6)}$	−32	−108	f_2=	-10/3

FIGURE 5.6 – Illustration graphique du nœud 6.

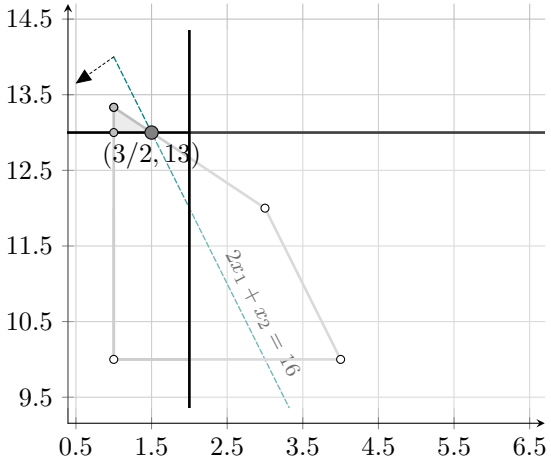
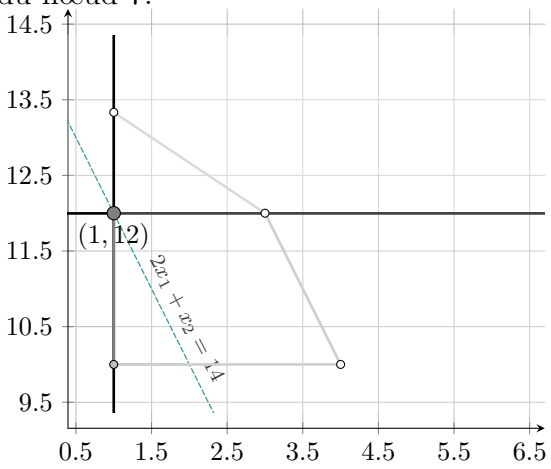


TABLE 5.7 – Table du simplexe du nœud 7.

$\mathcal{B}_7$	x_{13}	x_{14}		RHS
x_1	0	1		1
x_2	1	0		12
x_3	−1	−2		4
x_4	−3	−2		4
x_5	−1	−3		9
x_6	0	1		0
x_7	1	0		2
x_8	0	−1		4
x_9	−1	0		6
x_{10}	0	−2		5
x_{11}	0	−1		2
x_{12}	0	−1		1
$\boldsymbol{\rho}^{1(7)} = \boldsymbol{\eta}^{\mathbf{1(7)}}$	−1	−2	g_1=	14
$\nu^{1(7)}$	−1	0		15
$\mu^{1(7)}$	−3	−1		42
$\nu^{2(7)}$	2	−4		−20
$\mu^{2(7)}$	0	−2		5
$\boldsymbol{\rho}^{2(7)} = \boldsymbol{\eta}^{\mathbf{2(7)}}$	−3	1	g_2=	35
$\boldsymbol{\rho}^{3(7)}$	3	15	f_1=	5/14
$\boldsymbol{\rho}^{4(7)}$	10	−60	f_2=	-4

FIGURE 5.7 – Illustration graphique du nœud 7.



Nœud 7 : Après avoir ajouté la coupe $x_{12}+x_{13} \geq 1$ to the table 5.5 la table 5.7 résulte.

$(1, 12)$ est la solution obtenue, elle est dominée alors $\mathcal{S}_E$ ne change pas et $\mathcal{H}_7 = \{13, 14\}$, la coupe $x_{14} + x_{13} \geq 1$ est ajouté à la table 5.7.

Nœud 8 : Après avoir ajouté la contrainte $x_1 \leq \left\lfloor \frac{3}{2} \right\rfloor$ la table 5.8 résulte.

$(1, \frac{40}{3})$ est la solution trouvée, les nœuds sont 11 et 12 créés.

Nœud 9 : Après avoir ajouté la contrainte $x_2 \geq \left\lceil \frac{3}{2} \right\rceil$ à la table 5.6 le problème devient irréalisable.

Nœud 10 : Après avoir ajouté la contrainte cut $x_{13}+x_{14} \geq 1$ à la table 5.7 le problème devient irréalisable.

Nœud 11 : Après avoir ajouté la contrainte $x_2 \leq \left\lfloor \frac{40}{3} \right\rfloor$ à la table 5.8 la table 5.9 est obtenue.

TABLE 5.8 – Table du simplexe du nœud 8.

$\mathcal{B}_8$	x_4	x_{14}		RHS
x_1	0	1		1
x_2	1/3	−16		40/3
x_3	−1/3	−4/3		16/6
x_5	−1/3	−7/3		46/6
x_6	0	1		0
x_7	1/3	−1/6		10/3
x_8	0	−1		4
x_9	−1/3	1/6		28/6
x_{10}	0	−2		5
x_{11}	0	−1		2
x_{12}	0	−1		1
x_{13}	1/3	−1/6		1/3
$\boldsymbol{\rho}^{1(8)} = \boldsymbol{\eta}^{\mathbf{1(8)}}$	−1/3	−4/3	$g_1=$	46/3
$\nu^{1(8)}$	−1/3	1/6		49/3
$\mu^{1(8)}$	−1	1		46
$\nu^{2(8)}$	1/6	−16/3		−68/3
$\mu^{2(8)}$	0	−2		5
$\boldsymbol{\rho}^{2(8)} = \boldsymbol{\eta}^{\mathbf{2(8)}}$	-1	3	$g_2=$	39
$\boldsymbol{\rho}^{3(8)}$	1	−26/3	$f_1=$	11/31
$\boldsymbol{\rho}^{4(8)}$	5/6	−72	$f_2 =$	-68/15

FIGURE 5.8 – Illustration graphique du nœud 8.

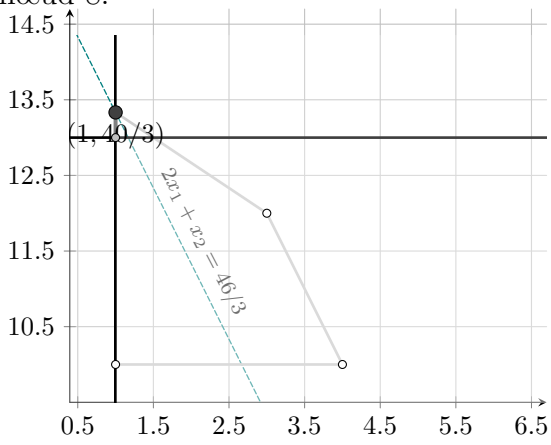
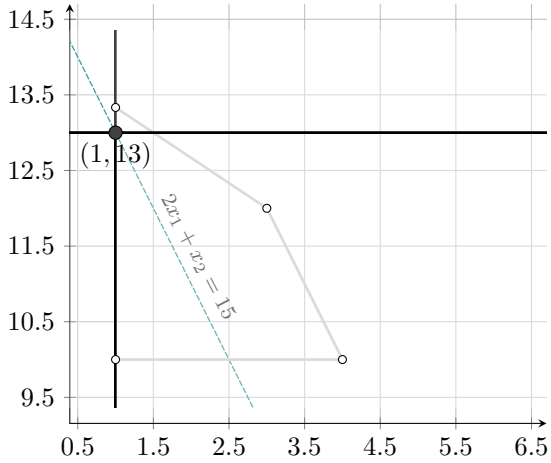


TABLE 5.9 – Table du simplexe du nœud 11.

$\mathcal{B}_{11}$	x_{14}	x_{15}		RHS
x_1	0	1		1
x_2	0	1		13
x_3	-2	-1		3
x_4	-2	-3		1
x_5	-3	-1		8
x_6	1	0		0
x_7	0	1		3
x_8	-1	0		4
x_9	0	-1		5
x_{10}	-2	0		5
x_{11}	-1	0		1
x_{12}	-1	0		1
x_{13}	0	1		0
$\boldsymbol{\rho^{1(11)} = \eta^{1(11)}}$	-2	-1	$g_1=$	15
$\nu^{1(11)}$	0	-1		16
$\mu^{1(11)}$	-1	-3		45
$\nu^{2(11)}$	-4	2		-22
$\mu^{2(11)}$	-2	0		5
$\boldsymbol{\rho^{2(11)} = \eta^{2(11)}}$	1	-3	$g_2=$	38
$\boldsymbol{\rho^{3(11)}}$	16	3	$f_1=$	16/45
$\boldsymbol{\rho^{4(11)}}$	-64	10	$f_2 =$	-22/5

FIGURE 5.9 – Illustration graphique du nœud 11.



$(1, 13)$ est obtenue, $\mathcal{S}_E = \{(3, 12), (2, 12), (1, 13)\}$, $\mathcal{H}_{11} = \{14, 15\}$ et on joute la coupe $x_{14} + x_{15} \geq 1$ à la table 5.9.

Nœud 12 : Après avoir ajouté la contrainte $x_2 \geq \left\lceil \frac{40}{3} \right\rceil$ à la table 5.8 le problème devient irréalisable.

Nœud 13 : Après avoir ajouté la contrainte cut $x_{14} + x_{15} \geq 1$ à la table 5.9 le problème devient irréalisable.

Donc l'ensemble des solutions efficientes final est $\mathcal{S}_E = \{(3, 12), (2, 12), (1, 13)\}$.

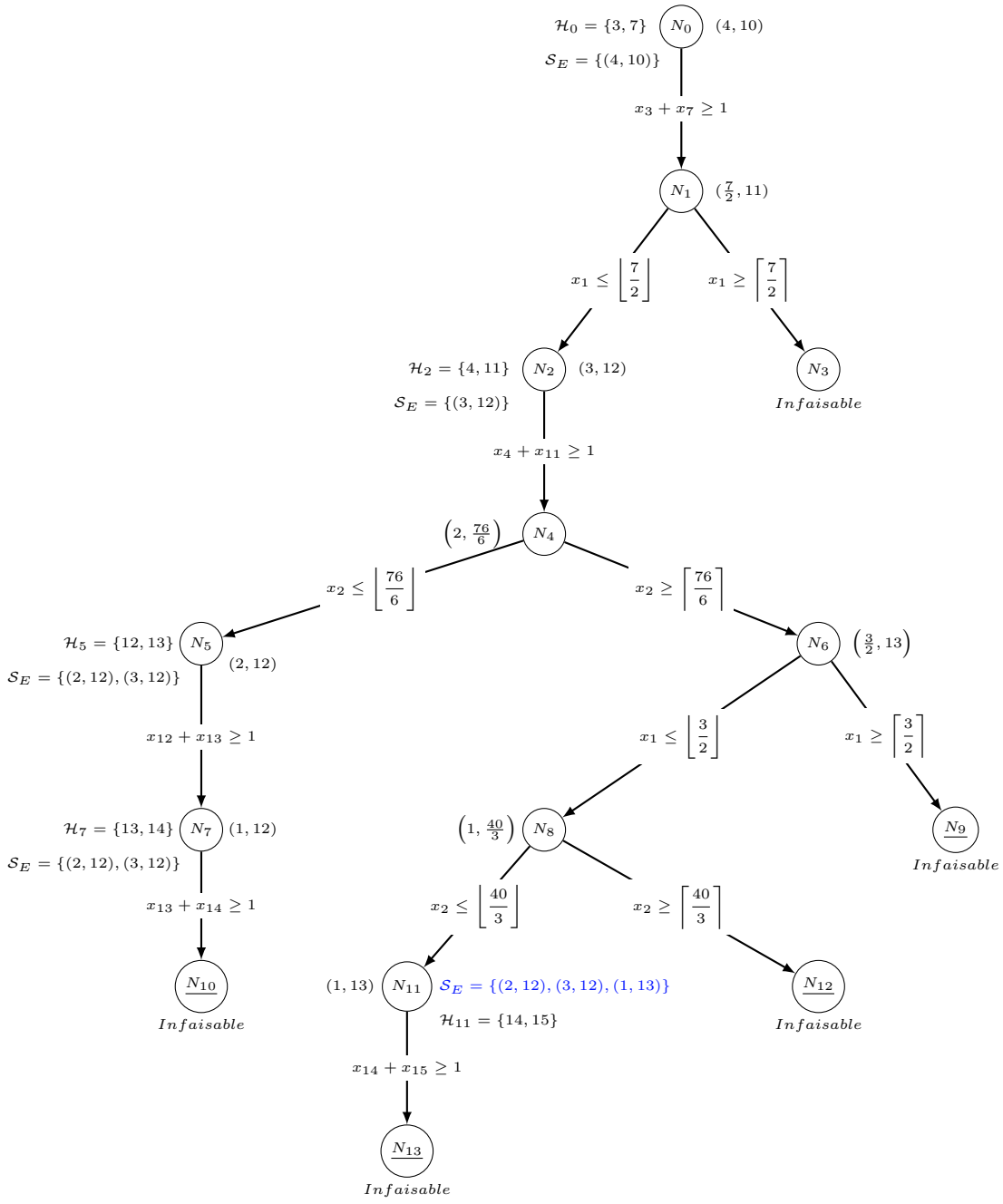


FIGURE 5.10 – Arbre de recherche de l'exemple illustratif

5.5 Conclusion

Dans ce chapitre, nous avons présenté une méthode exacte qui permet de générer l'ensemble efficient des problèmes de programmation linéaire plus linéaire fractionnaire discret à objectif multiple. La méthode présentée utilise le principe du **B&C**, c'est-à-dire qu'elle est basée sur la combinaison de l'exploration du **B&B** et d'une méthode de coupe, les coupes utilisées permettent d'éliminer les solutions inefficientes tandis que le processus de **B&B** explore le domaine implicitement.

Conclusion générale

Dans cette thèse, on s'est intéressé à l'optimisation discrète à objectif multiple. Le but principale été d'aborder les programmes linéaires plus linéaires fractionnaires discrets à objectif multiple. Beaucoup de problèmes pratiques intéressant se modélisent par un programme linéaire plus linéaire fractionnaire, c'est pour cela que beaucoup d'efforts ont été consentit pour l'analyse et l'optimisation de ces programmes. Cependant, la base théorique et les méthodes algorithmiques sur cette classe de programmes reste insuffisantes. La fonction linéaire plus linéaire fractionnaire n'étant pas convexe ni convexes généralisée comme la fonction linéaire fractionnaire, il est difficile de trouver son optimum global même sur un ensemble convexe, si on ajouté à cette difficulté la multiplicité des objectifs et la non-convexité du domaine réalisable, l'optimisation de cette fonction qui est originalement un problème NP-difficile devient encore plus inabordable. Ceci explique la pauvreté de la littérature sur le sujet ce qui est davantage motivant pour explorer cette classe de problèmes.

Ainsi comme contribution principale de cette thèse nous avons présenté une méthode de résolution exacte d'un problème d'*«optimisation de programme linéaire plus linéaire fractionnaire discret à objectif multiple»*, cette méthode consiste en une procédure de **B&C** qui associe le schéma de recherche par séparation et évaluation de **B&B** à une coupe efficiente. La coupe est dite efficiente car elles permet d'élaguer des solutions dont le vecteur critère est dominé. Cette méthode permet de générer d'une façon exhaustive toutes les solutions efficientes sans pour autant énumérer toutes les solutions entières du domaine réalisable.

La méthode proposée présente deux principaux avantages : Le premier est de n'utiliser aucune technique non-linéaire pour résoudre un problème non-linéaire, le second est de ne

pas explorer explicitement le domaine en évitant l'exploration de zones inintéressantes. Cette contribution a fait l'objet d'une publication dans un journal international dont le DOI est 10.1007/s10479-020-03581-0.

De ce travail découle aussi une contribution secondaire à l'optimisation multiobjectif en nombre entiers. Cette contribution consiste en la formulation d'un nouveau type de problème et la proposition d'une méthode de sa résolution, ce problème pourrait être désigné par le problème d'*«optimisation multiobjectif sur l'ensemble efficient »*. La résolution de ce problème permet de donné une partie de l'ensemble efficient générée à partir de plusieurs fonctions de préférence. Donc cette partie de l'ensemble efficient serra distribuée au gré du décideur et l'exploration du domaine lors de la résolution pourrait se restreindre seulement aux régions d'intérêt pour le décideur. En plus, le problème de l'*«optimisation multiobjectif sur l'ensemble efficient »* peut être vu en soit comme un modèle. En effet, c'est par exemple le cas quand il y a plusieurs décideurs différents ou bien dans le cas où le décideur exprime plusieurs préférences conflictuelles.

À la lumière du problème d'*optimisation multiobjectif sur l'ensemble efficient*, une des perspectives de recherche dans le domaine de l'*optimisation d'un programme linéaire plus linéaire fractionnaire discret à objectif multiple* est d'optimiser plusieurs fonctions de préférences linéaires sur l'ensemble efficient de ce programme afin de réduire le temps de calcul et d'obtenir une partie de l'ensemble efficient avec une cardinalité raisonnable. Aussi, puisque les solutions efficientes d'un programme linéaire plus linéaire fractionnaire sont liées aux solutions efficientes d'un programme linéaire factionnaire du même domaine et que celle-ci sont connecté [97], il serait intéressant d'étudier la possibilité de connexion entre les solutions efficientes.

Pareillement, d'autre sujets peuvent être abordés tels que :

— l'optimisation d'un programme fractionnaire quadratique discret et à objectif multiple ;
— l'optimisation d'un programme de somme d'une fonction quadratique plus d'une fraction quadratique discret à objectif multiple ;
— l'optimisation d'un programme min-max linéaire fractionnaire à objectif multiple ;
— l'optimisation d'un programme de somme d'une fonction linéaire plus linéaire fractionnaire stochastique discret et à objectif multiple etc...

Références bibliographiques

[1] M. Abbas et D. Chaabane. « Optimizing a linear function over an integer efficient set ». In : *European Journal of Operational Research* 174.2 (2006), p. 1140-1161.

[2] M. Abbas et M. Moulaï. « Integer linear fractional programming with multiple objective ». In : *Journal of the Italian Operations Research Society* 32.103-104 (2002), p. 15-38.

[3] S. Aggarwal et I. Sharma. « Maximization of the transmission rate of a discrete, constant channel ». In : *Unternehmensforschung* 14.1 (1970), p. 152-155.

[4] Y. Almogy et O. Levin. « Parametric analysis of a multi-stage stochastic shipping problem ». In : *Operational Research* 69 (1970), p. 359-370.

[5] M. Avriel et al. *Generalized concavity*. T. 63. Siam, 2010.

[6] E. B. Bajalinov. *Linear-Fractional Programming Theory, Methods, Applications and Software*. T. 84. Springer Science & Business Media, 2013.

[7] M Baluta, V Dobrescu et G Paun. « Algorithm for Solving a Combinatorial Problem of Optimal Selection ». In : *Economic Computation and Economic Cybernetics Studies and Research* 1 (1979), p. 87-95.

[8] E. M. Barriga et al. « Material requirements planning for a manufactured housing facility ». In : *Journal of architectural engineering* 11.3 (2005), p. 91-98.

[9] C. R. Bector, S Chandra et C Singh. « Duality in multiobjective fractional programming ». In : *Generalized Convexity and Fractional Programming with Economic Applications*. Springer, 1990, p. 232-241.

[10] E. J. Bell. *Primal-dual decomposition programming*. Rapp. tech. DTIC Document, 1965.

[11] H. P. Benson. « A finite, nonadjacent extreme-point search algorithm for optimization over the efficient set ». In : *Journal of Optimization Theory and Applications* 73.1 (1992), p. 47-64.

[12] H. P. Benson. « Optimization over the efficient set ». In : *Journal of Mathematical Analysis and Applications* 98.2 (1984), p. 562-580.

[13] B. Bereanu. « Programme de risque minimal en programmation linéaire stochastique ». In : *Comptes rendus hebdomadaires des séance de l'académie des sciences"* 259.5 (1964), p. 981.

[14] P. Bertier et B. Roy. « Procédure de résolution pour une classe de problèmes pouvant avoir un caractère combinatoire ». In : *Cahiers du Centre d'études de recherche opérationnelle* 6 (1964), p. 202-208.

[15] D. Bhati, P. Singh et R. Arya. « A taxonomy and review of the multi-objective fractional programming (MOFP) Problems ». In : *International Journal of Applied and Computational Mathematics* 3.3 (2017), p. 2695-2717.

[16] G. R. Bitran. « Linear multiple objective programs with zero–one variables ». In : *Mathematical Programming* 13.1 (1977), p. 121-139.

[17] N. Boland, H. Charkhgard et M. Savelsbergh. « A new method for optimizing a linear function over the efficient set of a multiobjective integer program ». In : *European journal of operational research* 260.3 (2017), p. 904-919.

[18] S. P. Bradley et S. C. Frey Jr. « Fractional programming with homogeneous functions ». In : *Operations Research* 22.2 (1974), p. 350-357.

[19] A. Cambini et L. Martein. « A modified version of Martos's algorithm ». In : *Methods of Operation Research* 53 (1985), p. 33-44.

[20] A. Cambini, L. Martein et S. Schaible. « On Maximizing a Sum of Ratios ». In : *Journal of Information and Optimization Sciences* 10.1 (1989), p. 65-79. eprint : http://www.tandfonline.com/doi/pdf/10.1080/02522667.1989.10698952. url : http://www.tandfonline.com/doi/abs/10.1080/02522667.1989.10698952.

[21] D. Chaabane, B. Brahmi et Z. Ramdani. « The augmented weighted Tchebychev norm for optimizing a linear function over an integer efficient set of a multicriteria linear program ». In : *International Transactions in Operational Research* 19.4 (2012), p. 531-545.

[22] D. Chaabane et M. Pirlot. « A method for optimizing over the integer efficient set ». In : *Journal of industrial and management optimization* 6.4 (2010), p. 811.

[23] L. G. Chalmet, L Lemonidis et D. J. Elzinga. « An algorithm for the bicriterion integer programming problem ». In : *European Journal of Operational Research* 25.2 (1986), p. 292-300.

[24] A. Charnes et W. W. Cooper. « Goal programming and multiple objective optimizations: Part 1 ». In : *European journal of operational research* 1.1 (1977), p. 39-54.

[25] A. Charnes et W. W. Cooper. « Programming with linear fractional functionals, 9, 181-186, 1962 ». In : *Naval Research Logistics Quarterly* 9 (1962), p. 181-186.

[26] M. E.-A. Chergui et M. Moulaï. « An Exact Method for a Discrete Multiobjective Linear Fractional Optimization ». In : *JAMDS* 2008 (2008).

[27] M. E.-A. Chergui, M. Moulaï et F. Z. Ouaïl. « Solving the multiple objective integer linear programming problem ». In : *Modelling, Computation and Optimization in Information Systems and Management Sciences*. Springer, 2008, p. 69-76.

[28] J. Clímaco, C. Ferreira et M. E. Captivo. « Multicriteria integer programming: An overview of the different algorithmic approaches ». In : *Multicriteria analysis*. Springer, 1997, p. 248-258.

[29] C. A. Coello. « An updated survey of GA-based multiobjective optimization techniques ». In : *ACM Computing Surveys (CSUR)* 32.2 (2000), p. 109-143.

[30] B. D. Craven. *Fractional programming*. T. 4. Heldermann, 1988.

[31] J. Crouzeix, J. Ferland et S. Schaible. « An algorithm for generalized fractional programs ». In : *Journal of Optimization Theory and Applications* 47.1 (1985), p. 35-49.

[32] G. B. Dantzig. « Linear programming: The story about how it began ». In : *History of Mathematical Programming* (1991), p. 19-31.

[33] G. B. Dantzig. « Maximization of a linear function of variables subject to linear inequalities ». In : *Activity analysis of production and allocation* 13 (1951), p. 339-347.

[34] G. B. Dantzig, R. Fulkerson et S. Johnson. « Solution of a large-scale traveling-salesman problem ». In : *Journal of the operations research society of America* 2.4 (1954), p. 393-410.

[35] J. P. Dauer. « Optimization over the efficient set using an active constraint approach ». In : *Zeitschrift für Operations Research* 35.3 (1991), p. 185-195.

[36] C. Derman. « On sequential decisions and Markov chains ». In : *Management Science* 9.1 (1962), p. 16-24.

[37] W. Dinkelbach. « On nonlinear fractional programming ». In : *Management Science* 13.7 (1967), p. 492-498.

[38] D. Z. Du, P. M. Pardalos et W. Wu. « History of Optimization ». In : *Encyclopedia of Optimization*. Springer, 2008, p. 1538-1542.

[39] J. G. Ecker et I. A. Kouada. « Finding efficient points for linear multiple objective programs ». In : *Mathematical Programming* 8.1 (1975), p. 375-377.

[40] J. G. Ecker et J. H. Song. « Optimizing a linear function over an efficient set ». In : *Journal of Optimization Theory and Applications* 83.3 (1994), p. 541-563.

[41] F. Y. Edgeworth. *Mathematical psychics: An essay on the application of mathematics to the moral sciences*. T. 10. Kegan Paul, 1881.

[42] M. Ehrgott et S. Ruzika. « Improved ε-constraint method for multiobjective programming ». In : *Journal of Optimization Theory and Applications* 138.3 (2008), p. 375.

[43] G. W. Evans. « An overview of techniques for solving multiobjective mathematical programs ». In : *Management Science* 30.11 (1984), p. 1268-1282.

[44] A. Fakhri et M. Ghatee. « Minimizing the sum of a linear and a linear fractional function applying conic quadratic representation: continuous and discrete problems ». In : *Optimization* 65.5 (2016), p. 1023-1038.

[45] J. E. Falk et S. W. Palocsay. « Recent Advances in Global Optimization ». In : sous la dir. de C. A. Floudas et P. M. Pardalos. Princeton, NJ, USA : Princeton University Press, 1992. Chap. Optimizing the Sum of Linear Fractional Functions, p. 221-258. URL : http://dl.acm.org/citation.cfm?id=137305.137325.

[46] P. de Fermat. *Abhandlungen uber maxima und minima*. Akademische verlagsgesellschaft mbh, 1934.

[47] L. R. Ford et D. R. Fulkerson. *A simple algorithm for finding maximal network flows and an application to the Hitchcock problem*. Rapp. tech. RAND CORP SANTA MONICA CA, 1955.

[48] H. Frenk et S. Schaible. « Fractional programmingFractional Programming ». In : *Encyclopedia of Optimization*. Sous la dir. de C. A. Floudas et P. M. Pardalos. Boston, MA : Springer US, 2009, p. 1080-1091. URL : https://doi.org/10.1007/978-0-387-74759-0_189.

[49] J. B. Frenk et S. Schaible. « Fractional programming ». In : *Handbook of generalized convexity and generalized monotonicity*. Springer, 2005, p. 335-386.

[50] R. W. Freund et F. Jarre. « Solving the sum-of-ratios problem by an interior-point method ». In : *Journal of Global Optimization* 19.1 (2001), p. 83-102.

[51] R. Freund et F Jarre. « An interior-point method for multifractional programs with convex constraints ». In : *Journal of Optimization Theory and Applications* 85.1 (1995), p. 125-161.

[52] M. H. Goedhart et J. Spronk. « Financial planning with fractional goals ». In : *European Journal of Operational Research* 82.1 (1995), p. 111-124.

[53] R. E. Gomory. « An Algorithm for Integer Solutions to Linear Programs, Princeton, 1958 ». In : *RL Graves and P. Wolfe, Recent Advances in Mathematical Programming, New York* (1963).

[54] R. E. Gomory et al. « Outline of an algorithm for integer solutions to linear programs ». In : *Bulletin of the American Mathematical society* 64.5 (1958), p. 275-278.

[55] M. Gugat. *Fractional semi-infinite programming*. na, 1994.

[56] J. Hoskins et R Blom. « Optimal allocation of warehouse personnel: a case study using fractional programming ». In : *FOCUS (UK)* 3.2 (1984), p. 13-21.

[57] S. Jain, A Mangal et P. R. Parihar. « Solution of a multi-objective linear plus fractional programming problem containing non-differentiable term ». In : *International Journal of Mathematical Sciences & Engineering Applications* 2.2 (2008), p. 221-229.

[58] S. Jain et K. Lachhwani. « Linear plus fractional multiobjective programming problem with homogeneous constraints using fuzzy approach ». In : *Operations Research* 2.1 (2010), p. 41-49.

[59] J. M. Jorge. « An algorithm for optimizing a linear function over an integer efficient set ». In : *European Journal of Operational Research* 195.1 (2009), p. 98-103.

[60] M. Jünger et al. *50 Years of integer programming 1958-2008: From the early years to the state-of-the-art*. Springer Science & Business Media, 2009.

[61] H. Karloff. *Linear programming*. Springer Science & Business Media, 2008.

[62] N Karmarkar. « A new polynomial-time algorithm for linear programming ». English. In : *Combinatorica* 4.4 (1984), p. 373-395. URL : http://dx.doi.org/10.1007/BF02579150.

[63] L. G. Khachiyan. « A polynomial algorithm in linear programming ». In : *Doklady Academii Nauk SSSR*. T. 244. 1979, p. 1093-1096.

[64] A. Khurana et S. Arora. « The sum of a linear and a linear fractional transportation problem with restricted and enhanced flow ». In : *Journal of Interdisciplinary Mathematics* 9.2 (2006), p. 373-383. eprint : http://dx.doi.org/10.1080/09720502.2006.1070045. URL : http://dx.doi.org/10.1080/09720502.2006.10700450.

[65] D. Klein et E. Hannan. « An algorithm for the multiple objective integer linear programming problem ». In : *European Journal of Operational Research* 9.4 (1982), p. 378-385.

[66] H. Konno et N. Abe. « Minimization of the Sum of Three Linear Fractional Functions ». In : *J. of Global Optimization* 15.4 (déc. 1999), p. 419-432. URL : http://dx.doi.org/10.1023/A:1008376731013.

[67] H. Konno et H. Watanabe. « Bond portfolio optimisation problems and their applications to index traking : a partial optimization approach ». In : *Journal of the Operations Research Society of Japan* 39.3 (1996), p. 295-306. URL : http://ci.nii.ac.jp/naid/110001184450/en/.

[68] J. S. Kornbluth et R. E. Steuer. « Multiple objective linear fractional programming ». In : *Management Science* 27.9 (1981), p. 1024-1039.

[69] H. W. Kuhn et A. W. Tucker. « Nonlinear Programming ». In : *Proceedings of the 2nd Berkeley Symposium on Mathematical Statistics and Probability*. 1951, p. 481-492.

[70] T. Kuno. « A Branch-and-bound Algorithm for Maximizing the Sum of Several Linear Ratios ». In : *J. of Global Optimization* 22.1-4 (jan. 2002), p. 155-174. URL : http://dx.doi.org/10.1023/A:1013807129844.

[71] M. Laumanns, L. Thiele et E. Zitzler. « An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method ». In : *European Journal of Operational Research* 169.3 (2006), p. 932-942.

[72] B. Lokman et M. Köksalan. « Finding all nondominated points of multi-objective integer programs ». In : *Journal of Global Optimization* 57.2 (2013), p. 347-365.

[73] R. T. Marler et J. S. Arora. « Survey of multi-objective optimization methods for engineering ». In : *Structural and multidisciplinary optimization* 26.6 (2004), p. 369-395.

[74] L. Martein et L. Carosi. « The Sum of a Linear and a Linear Fractional Function: Pseudoconvexity on the Nonnegative Orthant and Solution Methods. » In : *Bulletin of the Malaysian Mathematical Sciences Society* 35 (2012).

[75] B. Martos. « Hyperbolic programming ». In : *Nav. Res. Log. Quart* (1964), p. 135-155.

[76] G. Mavrotas. « Effective implementation of the ε-constraint method in multi-objective mathematical programming problems ». In : *Applied mathematics and computation* 213.2 (2009), p. 455-465.

[77] G. Mavrotas et D. Diakoulaki. « Multi-criteria branch and bound: A vector maximization algorithm for mixed 0-1 multiple objective linear programming ». In : *Applied mathematics and computation* 171.1 (2005), p. 53-71.

[78] G. Mavrotas et K. Florios. « An improved version of the augmented ε-constraint method (AUGMECON2) for finding the exact pareto set in multi-objective integer programming problems ». In : *Applied Mathematics and Computation* 219.18 (2013), p. 9652-9669.

[79] S Misra et C Das. « The sum of a linear and linear fractional function and a three-dimensional transportation problem ». In : *Opsearch* 18.3 (1981), p. 139-157.

[80] K. M. Mjelde. *Methods of the allocation of limited resources*. John Wiley & Sons, 1983.

[81] F. Z. Ouaï l, M. E. A. Chergui et M. Moulaï. « Optimizing over an Integer Efficient Set ». In : ().

[82] M. Özlen et M. Azizoğlu. « Multi-objective integer programming: a general approach for generating all non-dominated solutions ». In : *European Journal of Operational Research* 199.1 (2009), p. 25-35.

[83] B. B. Pal, B. N. Moitra et U. Maulik. « A goal programming procedure for fuzzy multiobjective linear fractional programming problem ». In : *Fuzzy sets and systems* 139.2 (2003), p. 395-405.

[84] V. Pareto. *Cours d'économie politique*. T. 1. Librairie Droz, 1964.

[85] G. Pentzaropoulos et D. Giokas. « Cost-performance modelling and optimization of network flow balance via linear goal programming analysis ». In : *Computer Communications* 16.10 (1993), p. 645-652.

[86] J. Philip. « Algorithms for the vector maximization problem ». In : *Mathematical programming* 2.1 (1972), p. 207-229.

[87] S. Pramanik, P. P. Dey et P. P. Vidyapith. « Multi-objective linear fractional programming problem based on fuzzy goal programming ». In : *International Journal of Mathematical Archive*. Citeseer. 2011.

[88] A. Przybylski, X. Gandibleux et M. Ehrgott. « A two phase method for multi-objective integer programming and its application to the assignment problem with three objectives ». In : *Discrete Optimization* 7.3 (2010), p. 149-165.

[89] M. Rao. « Cluster analysis and mathematical programming ». In : *Journal of the American statistical association* 66.335 (1971), p. 622-626.

[90] R. Rockafellar. « Convex Analysis, Princeton University Press, Princeton, 1970 ». In : *MATH Google Scholar* (1997).

[91] S. Sadia et al. « Solving Multi-Objective Linear plus Linear Fractional Programming Problem ». In : *Journal of Statistics Applications & Probability* 4.2 (2015), p. 253.

[92] S. Sayin. « Optimizing over the efficient set using a top-down search of faces ». In : *Operations Research* 48.1 (2000), p. 65-72.

[93] S. Schaible. « An note on the sum of linear and linear fractional function ». In : *Naval Research Quarterly* 24.V1 (1969), p. 691-962.

[94] S. Schaible. « Fractional programming: applications and algorithms ». In : *European Journal of Operational Research* 7.2 (1981), p. 111-120.

[95] S. Schaible. « Fractional programming: applications and algorithms ». In : *European Journal of Operational Research* 7.2 (1981), p. 111-120.

[96] S. Schaible. *Fractional programming with sums of ratios*. Università deglistudi di Pisa, Dipartimento di statistica e matematicaapplicataall'economia, 1994.

[97] S. Schaible. « Handbook of Global Optimization ». In : sous la dir. d'in R Horst et P. M. P. (eds.) Kluwer Academic Publishers, 1995. Chap. Fractional programming, p. 495 -608.

[98] S. Schaible et J. Shi. « Fractional programming: the sum-of-ratios case ». In : *Optimization Methods and Software* 18 (2003), p. 219-229.

[99] S. Schaible et J. Shi. « Recent developments in fractional programming: single-ratio and max-min case ». In : *Nonlinear analysis and convex analysis* 493506 (2004).

[100] R. G. Schroeder. « Linear programming solutions to ratio games ». In : *Operations Research* 18.2 (1970), p. 300-305.

[101] P Sing, S. D. Kumar et R. Singh. « Fuzzy multi-objective linear plus linear fractional programming problem: approximation and goal programming approach ». In : *International Journal of Mathematics and Computers in Simulation* 5.5 (2011), p. 395-404.

[102] W. Stadler. « A survey of multicriteria optimization or the vector maximum problem, part I: 1776–1960 ». In : *Journal of Optimization Theory and Applications* 29.1 (1979), p. 1-52.

[103] I. M. Stancu-Minasian. « A eighth bibliography of fractional programming ». In : *Optimization* 66.3 (2017), p. 439-470.

[104] I. M. Stancu-Minasian. « A fifth bibliography of fractional programming ». In : *Optimization* 45.1-4 (1999), p. 343-367.

[105] I. M. Stancu-Minasian. « A fourth bibliography of fractional programming ». In : *Optimization* 23.1 (1992), p. 53-71.

[106] I. M. Stancu-Minasian. *A ninth bibliography of fractional programming*. 2019.

[107] I. M. Stancu-Minasian. « A seventh bibliography of fractional programming ». In : *Adv. Model. Optim* 15.2 (2013), p. 309-386.

[108] I. M. Stancu-Minasian. « A sixth bibliography of fractional programming ». In : *Optimization* 55.4 (2006), p. 405-428.

[109] I. M. Stancu-Minasian. *Fractional programming: theory, methods and applications*. T. 409. Springer Science & Business Media, 2012.

[110] I. M. Stancu-Minasian. « Methods For Solving Linear Fractional Programming Problems ». English. In : Mathematics and Its Applications 409 (1997), p. 62-132. URL : http://dx.doi.org/10.1007/978-94-009-0035-6_4.

[111] I. M. Stancu-Minasian. « On the multiple minimum risk problem ». In : *Bulletin mathématique de la Société des Sciences Mathématiques de la République Socialiste de Roumanie* (1979), p. 427-437.

[112] I. M. Stancu-Minasian et S Tigan. « On some fractional programming models occurring in minimum-risk problems ». In : *Generalized Convexity and Fractional Programming with Economic Applications*. Springer, 1990, p. 295-324.

[113] R. E. Steuer. *Multiple criteria optimization: theory, computation, and application*. T. 233. Wiley New York, 1986.

[114] J. Sylva et A. Crema. « A method for finding the set of non-dominated vectors for multiple objective integer linear programs ». In : *European Journal of Operational Research* 158.1 (2004), p. 46-55.

[115] S. F. Tantawy. « A new method for solving linear fractional programming problems ». In : *Australian Journal of Basic and Applied Sciences* 1.2 (2007), p. 105-108.

[116] J Teghem et P. Kunsch. « A survey of techniques for finding efficient solutions to multi-objective integer linear programming ». In : *Asia-Pacific Journal of Operational Research* 3.2 (1986), p. 95-108.

[117] A. Teterev. « On a generalization of linear and piecewise linear programming ». In : (1970).

[118] P. M. Vitanyi et M. Li. *An introduction to Kolmogorov complexity and its applications*. T. 34. 10. Springer Heidelberg, 1997.

[119] J. Von Neumann. « Uber ein okonomsiches gleichungssystem und eine verallgemeinering des browerschen fixpunktsatzes ». In : *Erge. Math. Kolloq.* T. 8. 1937, p. 73-83.

[120] Y. Yamamoto. « Optimization over the efficient set: overview ». In : *Journal of Global Optimization* 22.1-4 (2002), p. 285-317.

[121] W. Zhang et M. Reimann. « A simple augmented ε-constraint method for multi-objective mathematical integer programming problems ». In : *European Journal of Operational Research* 234.1 (2014), p. 15-24.

[122] W. Ziemba, F. Brooks-Hill et C Parkan. « Calculating Investment Portfolios when the Returns are Normally Distributed ». In : *University of British Columbia Working Paper, Vancouver, British Columbia, Canada* (1971).

[123] S. Zionts. « A survey of multiple criteria integer programming methods ». In : *Annals of Discrete Mathematics*. T. 5. Elsevier, 1979, p. 389-398.