N^0 d'ordre : 11/2022-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 de Doctorat en Mathématiques

Présentée pour l'obtention du grade de Docteur

Spécialité : Recherche opérationnelle et Management (ROM)

Présentée par :

Yacine Chaiblaine

Intitulé :

Optimisation non linéaire sur l'ensemble efficient d'un problème vectoriel non linéaire

Soutenue publiquement le 17/02/2022, devant le jury composé de :

M. Meziane Aïder	Professeur à l'USTHB	Président ;
M. Mustapha Moulaï	Professeur à l'USTHB	Directeur de Thèse ;
M. Mohamed Aïdene	Professeur à l'UMMTO	Examinateur ;
M. Mouaouia Cherif Bouzid	Maître de conférence/A, à l'ENST	Examinateur ;
M. Nedjmeddine Kantour	Maître de conférence/B à l'USTHB	Invité ;

Résumé

La programmation multiobjectif joue un rôle très important dans la résolution de problèmes réels, car les décideurs ont souvent plusieurs fonctions objectifs à optimiser et le but est de trouver des solutions dites efficaces. Plusieurs approches ont été proposées dans la littérature permettant de trouver toutes les solutions efficaces à un problème multiobjectif. Cependant, le nombre de ces solutions peut être considérable et les trouver toutes peut entraîner des coûts de calcul considérables. De plus, le décideur aura des difficultés à choisir une solution parmi celles-ci. Pour contourner ce problème, on peut considérer une nouvelle fonction à optimiser sur les solutions efficaces. Cette approche est appelée optimisation sur l'ensemble efficace. Plusieurs méthodes sont proposées dans la littérature pour résoudre ce type de problème, essentiellement dans le cas linéaire. Néanmoins, de nombreux problèmes réels sont modélisés par des fonctions non linéaires. Parmi les formes non linéaires les plus utilisées, on trouve la programmation fractionnaire. La variante du problème d'optimisation sur l'ensemble efficace dans le cas fractionnaire n'a pas reçu beaucoup d'attention. Dans cette Thèse, nous contribuons principalement à la résolution d'un problème d'optimisation quadratique sur l'ensemble efficace d'un problème multiobjectif linéaire fractionnaire en nombres entiers ($MOILFP$). Nous proposons également de résoudre un problème où deux décideurs ont chacun une fonction fractionnaire à optimiser sur l'ensemble efficace d'un $MOILFP$. De plus, nous utilisons l'approche d'optimisation sur l'ensemble efficace pour résoudre un problème d'optimisation non convexe qui est la somme de fractions.

Mots clés : Optimisation Multiobjectif, Optimisation sur l'ensemble efficace, Optimisation en nombres entiers, Optimisation Fractionnaire, Optimisation non linéaire.

Abstract

Multiobjective programming plays a very important role in the solution of real problems, since decision makers often have several objective functions to optimize and the goal is to find the so-called efficient solutions. Several approaches have been proposed in the literature to find the set of all efficient solutions to a multiobjective problem. However, the number of these solutions can be considerable and finding them all is not obvious. Moreover, the decision-maker will face difficulties to choose a solution among these solutions. To overcome these difficulties, one can consider a new function to be optimized over the efficient solutions, this approach is called optimization over the efficient set. Several methods were proposed in literature to solve this type of problem, mainly in the linear case. However, many real world problems are modeled by nonlinear functions. Among the most commonly used nonlinear forms is fractional programming. The fractional variant of the optimization over the efficient set problem has not received much attention. In this thesis, we mainly contribute to the solution of a quadratic optimization problem over the efficient set of a multiobjective integer linear fractional problem ($MOILFP$). We also propose to solve a problem with two decision makers where, each have a fractional function to optimize over the efficient set of a $MOILFP$. Moreover, we use the optimization over the efficient set approach to solve a non-convex fractional programming problem which is the sum of ratios.

Keywords : Multiobjective optimization, Optimization over the efficient set, Integer Programming, Fractional Programming, Nonlinear optimization.

مُلَخَّص

تلعب البرمجة متعددة الأهداف دورًا مهمًا للغاية في حل المشكلات الحقيقية ، نظرًا لأن صانعي القرار غالبًا ما يكون لديهم العديد من الوظائف الموضوعية للتحسين ، والهدف هو إيجاد ما يسمى بالحلول الفعالة. تم اقتراح العديد من الأساليب في الأدبيات التي تسمح بإيجاد جميع الحلول الفعالة لمشكلة متعددة الأهداف. ومع ذلك ، يمكن أن يكون عدد هذه الحلول كبيرًا وإيجادها جميعًا ليس بالأمر السهل. علاوة على ذلك ، سيواجه صانع القرار صعوبات في اختيار الحل فيما بينهم. للتغلب على هذه المشكلة ، يمكن للمرء أن يفكر في وظيفة جديدة يتم تحسينها على الحلول الفعالة ويسمى هذا النهج التحسين على المجموعة الفعالة. تم اقتراح عدة طرق في الأدبيات لحل هذا النوع من المشاكل ، خاصة في الحالة الخطية. ومع ذلك ، فإن العديد من مشاكل العالم الحقيقي يتم تصميمها بواسطة وظائف غير خطية. من بين الأشكال غير الخطية الأكثر استخدامًا هي البرمجة الكسرية. لم يحظ التحسين على مشكلة المجموعة الفعالة في الحالة الكسرية باهتمام كبير. في هذه الأطروحة ، نساهم بشكل أساسي في حل تحسين وظيفة تربيعية على مجموعة فعالة من برمجة الأعداد الصحيحة متعددة الأهداف الكسرية الخطية MOILFP . نقترح أيضًا حل مشكلة حيث يكون لكل من صانعي القرار وظيفة كسرية للتحسين على مجموعة فعالة MOILFP . علاوة على ذلك ، نستخدم التحسين على المجموعة الفعالة لحل مشكلة من البرمجة الكسرية وهي مجموع الكسور.

الكلمات الرئيسية. التحسين متعدد الأهداف ، التحسين على المجموعة الفعالة ، البرمجة الصحيحة ، البرمجة الجزئية ، التحسين غير الخطي.

بسم الله الرحمن الرحيم

Remerciements

Je tiens à remercier mon directeur de Thèse, monsieur le professeur Mustapha Moulaï, pour sa patience, ses conseils et son soutien. J'ai beaucoup bénéficié de la richesse de ses connaissances. Je lui suis extrêmement reconnaissant pour la confiance qu'il m'a accordée au fil des années.

Mes sincères remerciements vont également au Dr Nedjmeddine Kantour, qui n'a jamais hésité à m'aider et à partager ses connaissances.

Mon profond respect va au professeur Meziane Aïder, qui m'a fait l'honneur d'accepter de présider mon jury et au Pr. Aïdene Mohamed et au Dr. Bouzid Mouaouia Cherif pour avoir accepté d'examiner ma Thèse.

Je suis reconaissant envers mes parents, dont l'amour et le soutien constants me permettent de rester motivé et confiant.

Je tiens également à remercier tous mes anciens professeurs et toute personne ayant contribué à mon éducation et apprentissage jusqu'à aujourd'hui.

Je dédie ce travail,
A mes parents,
A mon frère Samy,
A mes sœurs,
A mes cousins Adel, Wassim & Zinedine,
A mes neveux, Abderhaman, Dina & Lina, Amine,
Aicha et Youcef,
A mes proches et à toutes les personnes que j'aime.

Table des matières

1 Introduction générale **1**

1.1 Motivation 2

1.2 Contributions de la Thèse 3

1.3 Plan de la Thèse 4

I Notions préliminaires **5**

2 Optimisation fractionnaire **6**

2.1 Introduction 6

2.2 Applications 7

2.2.1 Un problème de sac à dos (Ishii, Ibaraki et Mine) 8

2.2.2 Un problème de transport maritime (Bitran et Novaes) 8

2.2.3 Un problème de couverture (Arora, SWARUP et al.) 8

2.3 Méthodes de résolution 9

2.3.1 Méthode de Charnes and Cooper 9

2.3.1.1 Exemple 10

2.3.2 Méthode de simplexe 11

2.3.3 Méthode de Dinkelbach 12

2.4 Programmation linéaire fractionnaire en nombres entiers 13

2.4.1 Branch & Bound 13

3 Optimisation sur l'ensemble efficace d'un problème multiobjectif en nombres entiers **15**

3.1 Introduction 15

3.2 Définitions 17

3.2.1 Test d'efficacité 19

3.3 Méthodes de résolution 20

3.3.1 Méthode de Chergui & Moulaî 20
3.3.2 Methode de Sylva & Crema 23
3.4 L'optimisation sur l'ensemble efficace 23
3.4.1 Méthode de Jorge 24
3.4.1.1 Variantes de la méthode de Jorge 25
3.4.2 Méthode de Drici & Moulaï 25

II Travaux de recherche (contributions) 27

4 Optimisation sur l'ensemble efficace d'un MOILFP 28
4.1 Introduction 28
4.2 Outils théoriques 30
4.2.1 Test d'efficacité pour les programmes fractionnaires linéaires multiobjectifs 30
4.3 Description de la méthode 31
4.3.1 Validation de l'algorithme 34
4.4 Exemple 35
4.5 Étude expérimentale 38
4.6 Conclusion 43

5 Optimisation biobjectif sur l'ensemble efficace du MOILFP 44
5.1 Introduction 44
5.2 Définitions et préliminaires 46
5.3 Méthodologie, algorithme et résultats théoriques 48
5.4 Résultats théoriques 53
5.5 Exemple didactique 57
5.6 Étude expérimentale 65
5.7 Conclusion 68

6 Optimisation de la somme des fractions 69
6.1 Introduction 69
6.2 Préliminaires 70
6.2.1 Les gradients réduits des objectifs fractionnaires 72
6.3 Algorithmes 74
6.3.1 Algorithme brute-force pour *ISOLRP* 74

6.3.2 L'algorithme proposé : une méthode exacte pour *ISOLRP* 74
6.4 Un exemple didactique 77
6.5 Étude expérimentale 84
6.6 Conclusion 90

7 Conclusion générale **91**
7.1 Optimisation sur l'ensemble d'un problème multiobjectif linéaire fractionnaire 91
7.2 Optimisation de deux fonctions fractionnaire sur l'ensemble efficace d'un multiobjectif linéaire fractionnaire 92
7.3 L'optimisation de la somme des fractions 92

Bibliographie **93**

Table des figures

3.1 Une représentation des différents types de solutions dans l'espace des critères. 18
3.2 Une représentation de différents types de solutions dans l'espace de décision. 18

4.1 Organigramme de la méthode proposée QPoverMOILFP. 33
4.2 La région admissible $\mathcal{D}^{(0)}$. 36
4.3 La région admissible $\mathcal{D}^{(1)}$ 37
4.4 Temps CPU pour les méthodes concurrentes (k=3) 43
4.5 Temps CPU pour des instances aléatoires de taille moyenne 43

5.1 Organigramme de la méthode proposée Biobjectif sur l'enssemble efficace d'un MOILFP 52
5.2 Arbre de recherche de l'exemple 65
5.3 Comparaison des temps d'exécution entre l'algorithme de brute-force et la méthode proposée pour des instances BIOLFP/MOILFP 68

6.1 Organigramme de la méthode proposée ISOLRP-Solver 76
6.2 Comparaison de temps d'exécution entre ISOLRP-Solver et brute-force avec 3 fractions 89
6.3 Temps d'exécution moyen de ISOLRP-Solver pour les instances 3DKP 89

Liste des tableaux

4.1 Temps d'exécution des instances aléatoires de petite taille. 40
4.2 Temps d'exécution d'instances aléatoires de taille moyenne de la méthode suggérée. 41
4.3 Temps d'exécution de la méthode proposée avec les instances KP, AP et GMO. 42

5.1 Tableau simplexe optimale pour noeud 0. 58
5.2 Tableau simplexe optimale pour noeud 1. 59
5.3 Tableau de simplexe optimal du Noeud 3. 60
5.4 Tableau de simplexe optimal du Noeud 4. 61
5.5 Tableau de simplexe optimal du Noeud 6. 62
5.6 Tableau de simplexe optimal du Noeud 7. 63
5.7 Tableau de simplexe optimal du Noeud 8. 64
5.8 Temps d'exécution des instances aléatoire BIOLFP/MOILFP 67

6.1 Tableau de simplexe optimal du Noeud 0 78
6.2 Tableau de simplexe optimal du Noeud 1. 79
6.3 Tableau de simplexe optimal du Noeud 2 80
6.4 Tableau de simplexe optimal du Noeud 5. 81
6.5 Tableau de simplexe optimal du Noeud 6. 82
6.6 Tableau de simplexe optimal du Noeud 8. 83
6.7 Tableau de simplexe optimal du Noeud 11. 84
6.8 Performance de l'algorithme 1 et de l'algorithme 2 sur des instances aléatoires 86
6.9 Les performances de ISOLRP-Solver 88

Liste des Algorithmes

1 Charnes & Cooper 10
2 Simplexe Fractionnaire 12
3 Algorithme de Dinkelbakh 13
4 Procédure de Branch & Bound 14

5 Méthode du simplexe pour résoudre un MOILFP 22
6 Méthode de Sylva & Crema 23
7 Méthode de Jorge 24
8 Méthode de Drici & Moulaï 26

9 QPoverMOILFP 34
10 Algorithme brute-force pour IQP/MOILFP 38

11 Optimisation biobjectif sur un programme multiobjectif fractionnaire linéaire en nombres entiers 53

12 Algorithme brute-force pour *ISOLRP*. 74
13 ISOLRP-Solver 75

Liste des Abréviations

B

BOILFP **B**i**O**bjective **I**nteger **L**inear **F**ractional **P**roblem

F

FP **F**ractional **P**roblem, problème fractionnaire

I

ILFP Integer Linear Fractional Program

IQP Integer Quadratic Program

ISOLRP Integer Sum Of Linear Ratios Problem

K

KP **K**napsack **P**roblem, problème de sac à dos

M

MOP **M**ulti**o**bjective **P**roblem

MOLFP **M**ulti**o**bjective **L**inear **P**roblem

MOILFP **M**ulti**o**bjective **I**nteger **L**inear **P**roblem

MOFP **M**ulti**o**bjective **F**ractional **P**roblem

MOLFP **M**ulti**o**bjective **L**inear **F**ractional **P**roblem

MOILFP **M**ulti**o**bjective **I**nteger **L**inear **F**ractional **P**roblem

MOKP **M**ulti**o**bjective **K**napsack **P**roblem

MOFKP **M**ulti**o**bjective **F**ractional **K**napsack **P**roblem

N

$\mathcal{NP}$ **N**ondeterministic **P**olynomial time (classe de complexité)

L

LP **L**inear **P**roblem

LFP **L**inear **F**ractional **P**roblem

Q

QP	**Q**uadratic **P**rogramming
QIP	**Q**uadratic **I**nteger **P**rogramming

S

SORP	**S**um **O**f **R**atios **P**roblem
SOLRP	**S**um **O**f **L**inear **R**atios **P**roblem

Chapitre 1

Introduction générale

"The seeker after the truth is not one who studies the writings of the ancients and, following his natural disposition, puts his trust in them, but rather the one who suspects his faith in them and questions what he gathers from them, the one who submits to argument and demonstration, and not to the sayings of a human being whose nature is fraught with all kinds of imperfection and deficiency. Thus the duty of the man who investigates the writings of scientists, if learning the truth is his goal, is to make himself an enemy of all that he reads, and, applying his mind to the core and margins of its content, attack it from every side. He should also suspect himself as he performs his critical examination of it, so that he may avoid falling into either prejudice or leniency." **Ibn al-Haytham (Alhazen).**

De nombreux problèmes de décision nécessitent l'optimisation simultanée de plusieurs objectifs. Souvent, les objectifs sont contradictoires (c'est-à-dire qu'un objectif ne peut être amélioré sans détériorer au moins un autre objectif). Ces problèmes sont appelés programmation multiobjectifs (Multiobjective programs *MOP*s). Contrairement à la programmation mono-objectif, on ne peut pas comparer deux solutions réalisables directement en évaluant la valeur objectif de chaque solution. La méthode la plus courante adoptée dans la programmation multiobjectifs consiste à introduire la notion de dominance et, au lieu d'avoir une seule solution optimale, on obtient un ensemble de solutions appelées solutions efficaces.

Au cours de la dernière décennie, plusieurs chercheurs se sont intéressés aux *MOP*s. Et plusieurs approches de résolution pour plusieurs types de *MOP*s ont été proposées dans la littérature. Parmi les types de *MOP* les plus étudiés figure la programmation multiobjectifs en nombres entiers. On peut les retrouver dans de nombreux problèmes réels puisque, dans la réalité, les variables de décision sont des entiers. Par exemple, une quantité entière, une

relation binaire 0-1, etc. Ces problèmes appartiennent à la classe $\mathcal{NP}$-difficile, et trouver tout l'ensemble efficace est une tâche difficile.

L'approche qui consistent à trouver l'ensemble des solutions efficaces présente plusieurs inconvénients :

— L'ensemble efficace est énorme, et trouver toutes ces solutions peut prendre énormément de temps.
— Souvent, le décideur n'a besoin que d'une seule solution qu'il mettra effectivement en œuvre.
— Le décideur aura du mal à choisir entre des solutions qui sont théoriquement incomparables.

Pour contourner cet inconvénient, plusieurs chercheurs ont introduit une nouvelle fonction que nous appellerons la fonction d'utilité et qui est indépendante des objectifs principaux. Ils proposent d'optimiser cette fonction sur toutes les solutions efficaces. En d'autres termes, choisir la meilleure solution efficace par rapport à la fonction d'utilité que le décideur choisit. Ces problèmes sont appelés "Optimisation sur l'ensemble efficace".

La programmation non linéaire, quant à elle, a reçu beaucoup d'attention car les fonctions non linéaires sont utilisées pour modéliser la majorité des problèmes du monde réel. En particulier, la programmation fractionnaire qui est une forme non linéaire intrigante dans laquelle la fonction objectif apparaît comme un rapport de fonctions. De nombreux problèmes d'ingénierie et économiques prennent en compte la minimisation d'un rapport de fonctions physiques et économiques, telles que coût/temps, coût/volume, coût/profit, ou d'autres quantités mesurant l'efficacité. Ce chapitre présente les problématiques qui seront abordés dans de cette Thèse. Les problèmes présentés dans ce chapitre sont tous liés aux problèmes fractionnaires.

1.1 Motivation

La plupart des recherches sont consacrées à l'optimisation d'une fonction linéaire sur l'ensemble efficace d'un problème multiobjectif linéaire en nombres entiers. Cependant, malgré le fait que la programmation non linéaire est plus générale et que de nombreux problèmes pratiques aboutissent à des modèles non linéaires. Quelques travaux sont consacrés au cas non linéaire de ce problème, parmi lesquels on trouve (Drici, Ouail et Moulaï, 2018 ; Moulaï et Drici, 2018 ; Moulaï et Mekhilef, 2021). D'autre part, la programmation

fractionnaire joue un rôle très important dans la résolution de nombreux problèmes pratiques, et qui peut être considérée comme une généralisation de la programmation linéaire, puisque tout programme linéaire peut être écrit sous forme d'un programme fractionnaire. Cependant, quant à la programmation sur l'ensemble efficace d'un programme multiobjectif, à notre connaissance, il n'existe qu'une seule méthode dans la littérature basé sur le simplexe (Zerdani et Moulai, 2011). Dans cette Thèse, nous contribuons principalement à la programmation linéaire fractionnaire en abordant ces deux questions :

— Peut-on étendre une des méthodes conçues pour résoudre un problème d'optimisation sur l'ensemble efficace d'un multiobjectif linéaire à la résolution d'un problème d'optimisation sur l'ensemble efficace d'un problème multiobjectif fractionnaire ?
— Peut-on trouver un sous-ensemble de solutions efficaces d'un problème multiobjectif fractionnaire dans le cas où il existe deux décideurs et chacun d'eux a sa propre fonction d'utilité ?

De plus, l'optimisation de la somme des fractions est l'un des problèmes les plus difficiles de la programmation fractionnaire en variables continues. Ce problème peut être vu comme un problème d'optimisation non linéaire sur l'ensemble efficace d'un *MOILFP*. A notre connaissance, il n'existe pas de méthode dans la littérature qui résout directement ce problème dans le cas où les variables sont des entiers.

1.2 Contributions de la Thèse

Nos propositions aux questions ci-dessus ont été valorisé par 3 publications. Deux d'entre elles sont en cours de révision (au moment de la rédaction du présent document). Nos contributions sont énumérées ci-dessous :

— Chaiblaine, Yacine, and Mustapha Moulaï. "An exact method for optimizing a quadratic function over the efficient set of multiobjective integer linear fractional program." Optimization Letters (2021) : 1-15.
— Chaiblaine, Yacine, Mustapha Moulaï, and Yasmine Cherfaoui. "An exact method for optimizing two linear fractional functions over the efficient set of a Multiobjective Integer Linear Fractional Program." arXiv preprint arXiv :2003.05364 (2020).
— Optimizing the sum of ratios (en cours de révision).

1.3 Plan de la Thèse

Cette Thèse est composée de deux parties, la première partie est consacrée aux notions préliminaires des problèmes traités dans cette Thèse. Cette partie comporte en deux chapitres :

— Introduction à la programmation fractionnaire, ces applications, quelques méthodes de résolutions et ces variantes.

— Notions préliminaire sur la programmation multiobjectif en nombres entiers en général et la programmation sur l'ensemble efficace en particulier. Nous présentons également quelques méthodes de résolutions.

La deuxième partie est consacrée aux contributions scientifiques, développées tout au long de ce projet de Thèse :

— Optimisation quadratique sur l'ensemble efficace d'un problème multiobjectif fractionnaire. Ce travail à été valorisé par une publication (voir Chaiblaine et Moulaï, 2021).

— Optimisation biobjectif sur l'ensemble efficace d'un problème multiobjectif fractionnaire. Ce travail est en cours de révisions et disponible en pré-impression Chaiblaine, Moulaï et Cherfaoui, 2020).

— Optimisation discrète de la somme des fractions. Ce travail est en cours de révision.

Première partie:

Notions préliminaires

Chapitre 2

Optimisation fractionnaire

"Life is an optimization problem",
[*"La vie est un problème d'optimisation"*],
Anonyme.

"L'optimisation c'est l'art de comprendre un problème réel, de pouvoir le transformer en un modèle mathématique que l'on peut étudier afin d'en extraire les propriétés structurelles et de caractériser les solutions du problème. Enfin c'est l'art d'exploiter cette caractérisation afin de déterminer des algorithmes qui les calculent mais aussi de mettre en évidence les limites sur l'efficience et l'efficacité de ces algorithmes."
Vangelis Th. PASCHOS.

Sommaire

2.1 Introduction 6
2.2 Applications 7
2.3 Méthodes de résolution 9
2.4 Programmation linéaire fractionnaire en nombres entiers 13

Dans ce chapitre, nous présentons quelques principes fondamentaux des problèmes fractionnaires en général et de la programmation linéaire en nombres entiers fractionnaires en particulier, qui seront au centre de cette Thèse. Nous commencerons par définir le problème, puis nous passerons en revue certaines applications du monde réel et enfin nous discuterons de certaines méthodes de résolution.

2.1 Introduction

Les programmes fractionnaires (Fractional Programming *FP*) apparaissent fréquemment dans de nombreux problèmes où un rapport entre deux fonctions doit être optimisée.

Neumann, 1945 a introduit un modèle d'équilibre pour une économie en expansion, qui est l'un des premiers programmes fractionnaires (bien qu'il ne soit pas appelé ainsi). Voici leurs formulation générale :

$$(FP)\begin{cases} \max f(x) = \dfrac{N(x)}{D(x)}, \\ s.c., \\ x \in \mathcal{S}, \end{cases} \tag{2.1}$$

avec $N(x)$ et $D(x)$ sont des fonctions et $\mathcal{S}$ est l'ensemble de solutions admissibles. Les problèmes de FP se posent lorsqu'il apparaît nécessaire d'optimiser un rapport par exemple : profitabilité d'un projet (gains/coûts), un rapport d'efficacité, etc. Nous nous intéressons à un type de problème fractionnaire appelé programmation linéaire fractionnaire (Linear-Fractional Programming LFP), dans lequel les fonctions $N(x)$ et $D(x)$ sont toutes les deux des fonctions linéaires. Les problèmes LFP ont été introduits pour la première fois par Martos, 1964 et leur formulation générale est la suivante :

$$(LFP)\begin{cases} \max f(x) = \dfrac{px+\alpha}{qx+\beta}, \\ s.c., \\ Ax \leq b, \end{cases} \tag{2.2}$$

où p, q, x sont des vecteurs de $\mathbb{R}^n$, A est une matrice $m \times n$, b un vecteur de $\mathbb{R}^m$, α, β sont des réels. On suppose que $qx + \beta > 0$ tout au long de cette Thèse.

2.2 Applications

Dans de nombreux problèmes du monde réel, la mesure de la qualité, de la rentabilité, de la probabilité, etc., est décrite comme une fonction fractionnaire. Par conséquent, la programmation fractionnaire modélise plusieurs problèmes du monde réel, tels que le problème de découpe (Gilmore et Gomory, 1961), la planification des systèmes électriques (Wang et al., 2018), l'allocation optimale des ressources en eau (Ren et al., 2016). Dans cette section, nous en verrons quelques-unes présentées dans (Bajalinov, 2013) et (Stancu-Minasian, 2012).

2.2.1 Un problème de sac à dos (Ishii, Ibaraki et Mine)

Étant donné un ensemble de n objets numérotés de $1, \ldots, n$, chacun ayant un poids w_i et une fonction de profitabilté à maximiser $f(x) = \frac{\sum p_i x_i}{\sum q_i x_i}$, ainsi qu'une capacité de poids maximale W. Le problème est formulé comme suit :

$$(FK) \begin{cases} \max f(x) = \dfrac{\sum_{i=1}^{n} p_i x_i}{\sum_{i=1}^{n} q_i x_i}, \\ s.c., \\ \sum_{j=1}^{n} w_i x_i \leq W, \\ x_i \in \{0,1\}, \qquad \forall i \in \{1, \ldots, n\}. \end{cases} \tag{2.3}$$

2.2.2 Un problème de transport maritime (Bitran et Novaes)

Supposons qu'au port A, nous devions charger un navire d'une capacité de transport limitée C avec n types de marchandises et les transporter vers le port B. Notre objectif est de déterminer la quantité de chaque type de marchandise à charger pour que le profit réalisé par unité de coût de transport soit maximal. Soit U_j la quantité maximale disponible de la j^{me} marchandise, et P_j et d_j, $j = \{1, \ldots, n\}$, sont le profit réalisé par unité de cette marchandise et le coût du transport respectivement. Si W_j représente le poids de l'unité du j^{me} bien, et x_j est une variable de décision, nous obtenons le modèle suivant :

$$(TM) \begin{cases} \max f(x) = \dfrac{\sum_{j=1}^{n} p_i x_i}{\sum_{j=1}^{n} d_j x_j}, \\ s.c., \\ \sum_{j=1}^{n} W_j x_j \leq C, \\ 0 \leq x_j \leq U_j, \qquad \forall j = \{1, \ldots, n\}. \end{cases} \tag{2.4}$$

2.2.3 Un problème de couverture (Arora, SWARUP et al.)

Une compagnie aérienne doit effectuer m vols et dispose de n équipages. On note par $c_j > 0$ le coût payé par la compagnie lorsque le j^{me} équipage est utilisé sur un vol, et par d_j la commission reçue par la compagnie. Soit $\beta > 0$ le montant d'argent reçu par la compagnie pour chaque équipage utilisé. La compagnie souhaite répartir les équipages sur les m vols de telle sorte que $\frac{\sum c_j}{\sum d_j + \beta}$ soit minimal. Soit $I = \{1, \ldots, m\}$ l'ensemble des vols, $J = \{1, \ldots, n\}$ l'ensemble des équipages, et $P_j \subset I$ l'ensemble des vols qui peuvent être effectués par le j^{me}

équipage. Il est nécessaire de déterminer un ensemble J^* d'équipages pour couvrir les m vols et qui minimise le rapport. Soit $a_{ij} = 1$ si le vol i est couvert par l'équipage j et 0 sinon. Posons la variable x_j associé à l'équipage j, qui vaut 1 si l'équipage j est affecté et 0 sinon. Le modèle de ce problème est formulé comme suit :

$$(PC) \begin{cases} \max f(x) = \dfrac{\sum c_j}{\sum d_j + \beta'}, \\ s.c., \\ \sum_{j=1}^{n} a_j x_j \geq 1, \\ x_j \in \{0,1\} \forall j = \{1,\ldots,n\}. \end{cases} \tag{2.5}$$

2.3 Méthodes de résolution

Plusieurs méthodes sont proposées dans la littérature pour résoudre les problèmes fractionnaires. Nous citons à titre d'exemple, la transformation de Charnes et Cooper, 1962 qui convertit un programme linéaire fractionnaire LFP en un programme linéaire. Cependant, cette transformation ne peut pas être appliquée directement au programme linéaire fractionnaire en nombres entiers (Integer Linear Fractional Program $ILFP$). Dinkelbach, 1967 a proposé une méthode utilisant la programmation paramétrique pour résoudre un problème de programmation fractionnaire. Chergui et Moulaï, 2008 ont proposé une méthode exacte pour énumérer tout l'ensemble efficace d'un $MOILFP$. Le lecteur peut trouver plus de détails sur les différentes applications et méthodes de la programmation fractionnaire dans : l'étude de Schaible, 1981, la bibliographie de Stancu-Minasian, 2019, et le livre de Bajalinov sur la programmation fractionnaire (Bajalinov, 2013).

2.3.1 Méthode de Charnes and Cooper

Charnes et Cooper, 1962 ont montré que tout problème de programmation linéaire fractionnaire peut être converti en un problème de programmation linéaire. D'après un LFP (2.2) on introduit un vecteur de variables y et une variable t telle que : $t = \dfrac{1}{D(x)}$ et $y_i = \dfrac{x_i}{D(x)}, i = \{1,\ldots,n\}$. On peut écrire la fonction objectif en fonction des nouvelles variables $L(y) = py + \alpha$. Ensuite, on multiplie toutes les contraintes par $1/D(x)$ ($D(x)$ est supposé être strictement positive) et on ajoute la contrainte $\sum_{i=1}^{n} q_i y_i + \beta t = 1$. On obtient

donc le programme linéaire suivant :

$$(TCC)\begin{cases} \max L(y) = py + \alpha t, \\ Ay - bt \leq 0, \\ qy + \beta t = 1, \\ t, y_i \geq 0, \forall i \in \{1, \ldots, n\}. \end{cases} \tag{2.6}$$

Théorème 2.1. *Si la solution (y^*, t^*) est optimale pour 2.6, alors la solution x^* est optimale pour LFP (2.2) avec :*

$$x^* = \frac{y^*}{t^*}. \tag{2.7}$$

L'algorithme qui permet de trouver la solution optimale d'un *LFP* en utilisant la transformation de Charnes & Cooper peut être résumé comme suit :

Algorithme 1 : Charnes & Cooper

Résultat : La solution optimale x_{opt} pour un problème *LFP* et sa valeur associée f_{opt}

Etape 1 : Trouver le problème linéaire (TCC) par la transformation (2.6).

Etape 2 : Résoudre (TCC).

Si (TCC) *a une solution optimale* $(y*, t^*)$ **alors**

  Retourner $x^* = \frac{y^*}{t}$ $f_{opt} = L(y^*, t^*)$.

Sinon

  Retourner Le problème n'admet pas de solution.

Finsi

2.3.1.1 Exemple

Prenons le programme fractionnaire suivant :

$$(P)\begin{cases} \max \dfrac{x_1 - 4}{-x_2 + 2}, \\ s.c., \\ -x_1 + 4x_2 \leq 0, \\ 2x_1 + x_2 \leq 8, \\ x_1 \geq 0,\ x_2 \geq 0. \end{cases}$$

En appliquant la transformation de Charnes & Cooper, on obtient le programme linéaire suivant :

$$(PT)\begin{cases} \max y_1 - 4t, \\ s.c., \\ -y_1 + 4y_2 \leq 0, \\ 2y_1 + y_2 - 8t \leq 0, \\ -y_2 + 2t = 1, \\ y_1 \geq 0,\ y_2 \geq 0, t \geq 0. \end{cases}$$

La solution optimale du programme linéaire est : $y^* = (2,0), t^* = 1/2$. Et donc la solution optimale de (P) est $x^* = (4,0)$.

2.3.2 Méthode de simplexe

Cambini et Martein, 1986 ont proposé une procédure basé sur le simplexe qui est une amélioration de la procédure de Martos, 1964. Cette procédure utilise le gradient réduit pour se rapprocher de l'optimum pour une base $\mathcal{B}$ et une solution associée x^* en utilisant les notations suivantes :

$$\begin{aligned} \bar{p} &= p - p_{\mathcal{B}}\mathcal{B}^{-1}A, \\ \bar{q} &= q - q_{\mathcal{B}}\mathcal{B}^{-1}A, \\ \bar{\alpha} &= px^* + \alpha, \\ \bar{\beta} &= qx^* + \beta. \end{aligned}$$

Le gradient réduit γ du problème LFP (2.2) peut s'écrire comme suit :

$$\gamma = \bar{\beta}\bar{p} - \bar{\alpha}\bar{q}.$$

Le gradient réduit donne le sens de croissance de la fonction objectif $f(x^*)$, donc si pour tout indice j de l'ensemble des indices hors base $\mathcal{N}$, $\gamma_j \leq 0$ alors la fonction $f(x)$ ne peut pas être améliorée. Et donc l'algorithme de résolution peut être résumé comme suit :

Algorithme 2 : Simplexe Fractionnaire

Résultat : La solution optimale x_{opt} pour un problème LFP et sa valeur associée f_{opt}

Étape 1 : Trouver la solution niveau optimale $\bar{x}$.

Si *($\bar{x}$ n'existe pas)* **alors**

Le problème est irréalisable.

Retourner $f_{opt} = -\infty$.

Sinon

Finsi

Aller à l'étape 2.

Étape 2 : Calculer le gradient réduit $\bar{\gamma}$ associé à $\bar{x}$

Soit $J = \{j | \bar{\gamma}_j > 0\}$.

Si $J = \emptyset$ **alors**

Retourner $x_{opt} = \bar{x}$ et $f_{opt} = f(\bar{x})$.

Sinon

Sélectionner un indice k tel que $\gamma_k = max(\gamma)$. Aller à l'étape 3.

Finsi

Étape 3 : La variable x_k entre dans la base et une variable quitte la base. Aller à l'étape 2.

2.3.3 Méthode de Dinkelbach

La méthode de Dinkelbach, 1967 est une méthode paramétrique utilisée pour la résolution d'un programme fractionnaire en général. Dans le cas linéaire la méthode consiste à résoudre une séquence de problème linéaire. Prenons un LFP (2.1), et considérons la fonction suivante :

$$F(\lambda) = \max_{x \in \mathcal{S}} \{N(x) - \lambda D(x)\}, \lambda \in \mathbb{R}.$$

Théorème 2.2. Dinkelbach, *1967* *La solution x^* est optimal pour le problème LFP (5.1) si et seulement si $F(\lambda^*)_{x \in \mathcal{S}} = \{N(x) - \lambda^* D(x)\}$, avec $\lambda^* = \dfrac{N(x^*)}{D(x^*)}$.*

Ce théorème donne également une procédure pour calculer la solution optimale du problème de programmation fractionnaire résumé dans l'algorithme 3.

Algorithme 3 : Algorithme de Dinkelbakh

Résultat : La solution optimale x_{opt} pour un problème LFP

Etape 1 : Trouver une solution réalisable $x^{(0)} \in \mathcal{S}$, poser $k = 0$.

Etape 2 : Trouver $x^{(1)} = argmax_{x \in \mathcal{S}} \left\{ N(x) - \lambda^{(k)} D(x) \right\}$.

Etape 3 : **Si** $(F(\lambda^k) = 0)$ **alors**

  Retourner $x_{opt} = x^{(k)}$.

Sinon

  Poser $\lambda^{k+1} = \dfrac{N(x^{(k)})}{D(x^{(k)})}$ et $k = k + 1$.

  Aller à **Etape 2**.

Finsi

2.4 Programmation linéaire fractionnaire en nombres entiers

Un programme linéaire fractionnaire en nombres entiers $(ILFP)$ est un LFP avec les les contraintes d'intégrités. Un problème $ILFP$ s'écrit généralement comme suit :

$$(ILFP) \begin{cases} \max f(x) = \dfrac{px + \alpha}{qx + \beta}, \\ s.c., \\ Ax \leq b, \\ x \in \mathbb{N}. \end{cases} \tag{2.8}$$

Avec $\mathbb{N}$ l'ensemble des entiers naturelle.

Pour résoudre un $ILFP$, nous pouvons combiner une procédure de Branch & Bound avec n'importe quelle méthode de résolution d'un problème LFP.

2.4.1 Branch & Bound

Branch & bound (B&B) est une procédure utilisée pour les problèmes d'optimisation discrets. La procédure B&B utilisée pour résoudre les problèmes d'optimisation en nombres

entiers peut être résumée comme suit :

Algorithme 4 : Procédure de Branch & Bound

Résultat : La solution optimale entière x_{opt}.

Initialisation : R=$\{L_0\}$, où L_0 le programme principale relaxé (Sans contraintes d'intégrité). $x_{opt} = []$ et $f_{opt} = -\infty$

Tant que *R est non vide* **faire**

Prendre L un noeud de R.

Résoudre L.

Si *L est irréalisable* **alors**

sonder le noeud L.

Finsi

Si *L a une solution optimale entière* x_l **alors**

Si $f_l > f_{opt}$ **alors**

$x_{opt} = x_l$ et $f_{opt} = f_l$.

Finsi

Finsi

Si *L a une solution continue* **alors**

Choisir un indice r tel que x_r n'est pas un entier. Ensuite, ajouter deux noeuds à l'arbre R, en ajoutant, respectivement à chaque noeud, les contraintes $x_r \leq \lfloor x_r \rfloor$ et $x_r \geq \lceil x_r \rceil$.

Finsi

Fintq

La solution optimale est obtenue lorsque tous les noeuds crées sont sondés. L'exploration de l'arborescence créée se fait selon quatre stratégies différentes :

1. **En largeur :** Cette stratégie favorise les noeuds les plus proche de la racine, ce qui engendre moins de séparations. Cependant les deux stratégies suivantes s'avèrent plus efficaces.
2. **En profondeur :** Dans cette stratégie, on commence par explorer le noeud le plus éloigné de la racine, c'est à dire le plus récemment créé.
3. **Le meilleur d'abord :** Cette stratégie consiste à explorer le noeud possédant la meilleure borne, ce qui permet d'éviter l'exploration des sous-problèmes qui possèdent une mauvaise évaluation par rapport à la solution optimale.
4. **Hybride :** Cette stratégie consiste à utiliser une combinaison des stratégies ci-dessus.

Chapitre 3

Optimisation sur l'ensemble efficace d'un problème multiobjectif en nombres entiers

"Obvious" is the most dangerous word in mathematics,
[*"Évident" est le mot le plus dangereux en mathématiques*],
Eric T., Bell.

Sommaire

3.1 Introduction 15
3.2 Définitions 17
3.3 Méthodes de résolution 20
3.4 L'optimisation sur l'ensemble efficace 23

3.1 Introduction

L'optimisation multiobjectif en nombres entiers consiste à optimiser simultanément plusieurs fonctions conflictuelles sur un espace discret. Par conséquent, il n'y a pas une seule solution entière à qualifier d'optimale. Néanmoins, le but est de trouver un ensemble de solutions entières de compromis qui contient les solutions qui réalisent un compromis entre ces objectifs conflictuels, appelées solutions efficaces. Un problème multiobjectif peut être formulé comme suit :

$$(MOP)\begin{cases} \max \quad F = (f_1(x), f_2(x), \ldots, f_k(x)), \\ s.c, \\ \qquad\qquad x \in \mathcal{S}. \end{cases} \tag{3.1}$$

Où $k \geq 2$ est le nombre d'objectifs et $\mathcal{S}$ l'ensemble des solutions admissibles dans l'espace de décision. On note par $\mathcal{Y}$ l'image de $\mathcal{S}$ par la fonction vectorielle $F = (f_1(x), f_2(x), \ldots, f_k(x))$ représentant l'image de l'ensemble réalisable dans l'espace critère. Cette formulation est générale, mais les problèmes multiobjectifs peuvent être classés suivant de la nature des fonctions objectifs et/ou de la région admissible. Par exemple :

- Si les fonctions objectifs et la région admissible sont convexes alors on dit que c'est un problème multiobjectif convexe.
- Si les fonctions objectifs et les fonctions formant la région admissible sont linéaire alors on dit que c'est un problème multiobjectif linéaire (MultiObjective Linear Program *MOLP*). Si de plus la région admissible contient que les solutions entières alors on dit que c'est un problème multiobjectif linéaire en nombres entiers (MultiObjective Integer Linear Program *MOILP*).
- Si les fonctions objectifs sont linéaire fractionnaire alors on dit que c'est un problème multiobjectif linéaire fractionnaire (Multiobjective Linear Fractional Program *MOLFP*) et la variante en nombres entiers est appelée (Multiobjective Integer Linear Fractional Program *MOILFP*)

Cependant, une recherche exhaustive de toutes ses solutions efficaces n'est pas pratique, car le nombre de ces solutions est parfois énorme et le décideur sera confronté à un autre problème, celui de choisir des solutions applicables parmi un ensemble de solutions théoriquement équivalentes. Pour surmonter ce problème, plusieurs chercheurs ont envisagé d'inclure une fonction d'utilité à optimiser sur l'ensemble efficace. Cette approche a donné naissance à un autre type de problèmes, appelés optimisation sur l'ensemble efficace. Elle a été considérée pour la première fois par (Philip, 1972). Plus tard, plusieurs chercheurs se sont intéressés à ce sujet, dont nous citons, (Abbas et Chaabane, 2006), où ils ont introduit une méthode exacte pour résoudre le problème d'optimisation d'une fonction linéaire sur l'ensemble efficace d'un problème linéaire multiojectif en nombres entiers *MOILP*. D'autres part, (Jorge, 2009) a proposé une autre approche pour résoudre ce problème en utilisant le même principe utilisé par la méthode de (Sylva et Crema, 2004) pour énumérer l'ensemble efficace du problème *MOILP*. Plusieurs travaux récents s'attaquent au même problème et proposent des méthodes plus efficace, ex., (Boland, Charkhgard et Savelsbergh, 2017 ; Lokman, 2021 ; Sierra Altamiranda et Charkhgard, 2019).

3.2 Définitions

Le concept de dominance joue un rôle important dans l'optimisation multiobjectif car il n'existe pas une solution unique pouvant être qualifiée d'optimale, mais, un ensemble de solutions appelées efficaces.

Définition 3.1. Soit $x^* \in \mathcal{S}$ et son image dans l'espace critère $z^* = (f_1(x^*), .., f_k(x^*))$. On dit que z^* **domine** une autre solution $y = (y_1, .., y_k) \in \mathcal{Y}$, si et seulement si, $f_i(x^*) \geq y_i, \forall i = 1..k$ et il existe au moins un j, $0 \leq j \leq k$, tel que, $f_j(x^*) > y_j$.

On peut étendre cette définition dans l'espace de décision, on dit que x^* **domine** x' si le vecteur critère de x^* domine le vecteur critère de x'.

Définition 3.2. On dit qu'une solution $x \in \mathcal{S}$ est **efficace** si il n'existe aucune solution $x' \in \mathcal{S}$ tel que x' **domine** x, i.e., x est **efficace**, si et seulement si, $\nexists x' \in \mathcal{S}$, tel que, $f_i(x') \geq f_i(x)$, $\forall i = 1..k$ et $\exists j$, $0 \leq j \leq k$, $f_i(x') > f_i(x)$. Le vecteur critère de x est appelé un point non dominé (et parfois Pareto). On note $\mathcal{X}_E$ l'ensemble des solutions efficaces et $\mathcal{Y}_E$ son image dans l'espace critère.

Définition 3.3. On dit qu'une solution $x \in \mathcal{S}$ est **faiblement efficace** si il n'existe aucune solution $x' \in \mathcal{S}$, tel que, $f_i(x) > f_i(x'), \forall i = 1..k$.

L'ensemble des solutions non dominées d'un problème d'optimisation multiobjectif est limité par un vecteur critère appelé point nadir z^{nad} et un vecteur appelé point idéal z^{id}, si ceux-ci sont finis (voir Miettinen, 2012).

Définition 3.4. Les composantes z^i du point idéal z^{id} sont obtenues en maximisant chacune des fonctions objectifs individuellement, en tenant compte des contraintes, c'est-à-dire en résolvant les problèmes suivants aux contraintes, c'est-à-dire en résolvant :

$$\begin{aligned} z_i^{id} = \max \quad & (f_i(x)) \quad i = 1..k \\ s.c, \quad & \\ & x \in \mathcal{S}. \end{aligned} \tag{3.2}$$

Définition 3.5. Le point nadir est le vecteur composé des pires valeurs d'objectif sur l'ensemble de solutions non dominées. Analytiquement, le vecteur objectif nadir est exprimé par :

$$\begin{aligned} Z_i^{nad} = \min \quad & f_i(x). \quad i = 1..k \\ s.c, \quad & \\ & x \in \mathcal{X}_E. \end{aligned} \tag{3.3}$$

S'il existe une solution réalisable, telle que son vecteur critère $z = z^{id}$ alors, il est clair que, l'ensemble efficace se réduit à cette solution. Plus précisément, les objectifs ne sont pas conflictuelles et donc le problème peut être réduit à un problème mono-objectif.

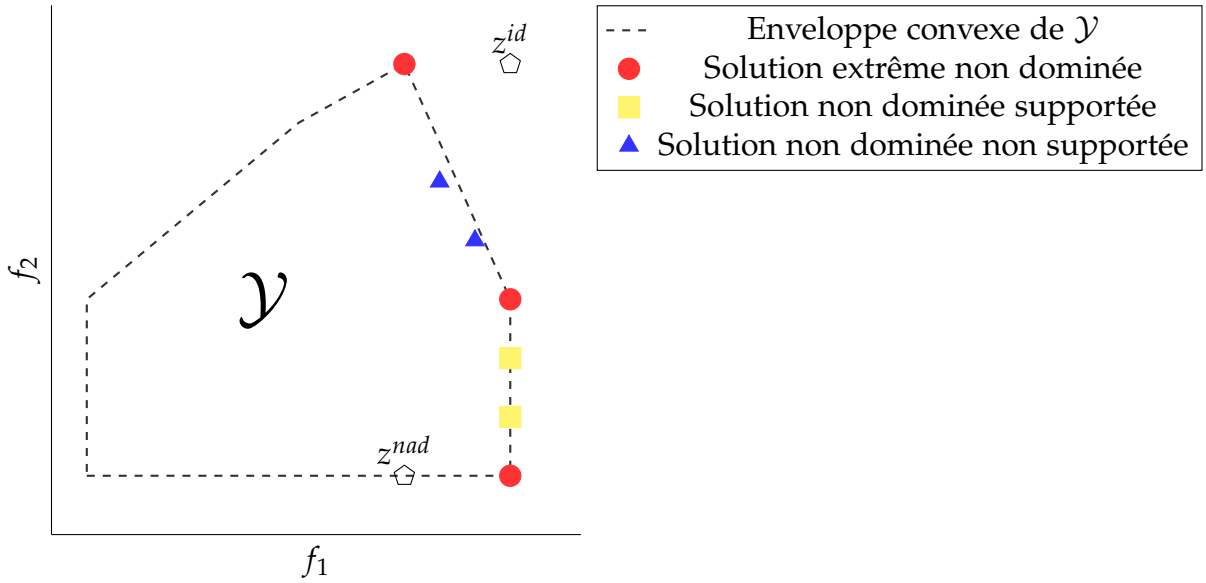

FIGURE 3.1 – Une représentation des différents types de solutions dans l'espace des critères.

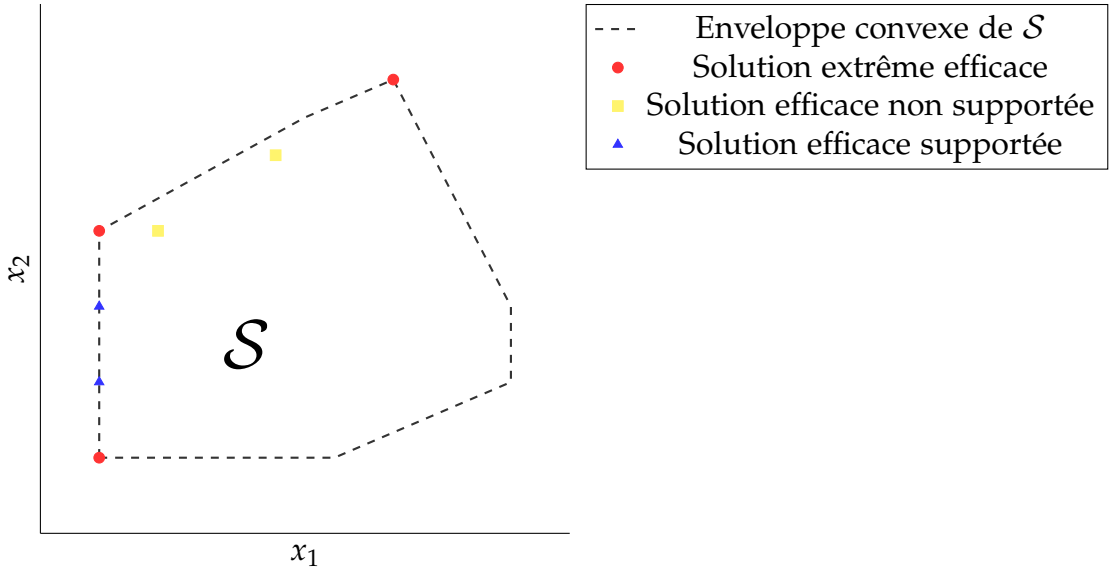

FIGURE 3.2 – Une représentation de différents types de solutions dans l'espace de décision.

Soit le programme P_λ suivant :

$$(P_\lambda) : \begin{cases} \max \quad (\lambda_1 f_i(x) + \lambda_2 f_2(x) + \cdots + \lambda_k f_k(x)), \\ s.c, \\ \qquad\qquad x \in \mathcal{S}. \end{cases} \tag{3.4}$$

Où $\lambda_i > 0, \forall i = 1..k$.

Théorème 3.1. *(Miettinen, 2012) La solution optimale de P_λ (3.4) est une solution efficace.*

Définition 3.6. La solution optimale x^* de (3.4) est appelée solution efficace supportée (Elle est sur l'enveloppe convexe de $\mathcal{S}$). De plus, si la solution x^* est un point extrême de $\mathcal{S}$, alors, la solution est appelée solution extrême efficace.

Il existe des solutions efficaces qui ne sont pas supportées dans la programmation multiobjectif en nombres entiers, contrairement dans le cas continue, et ce sont ces solutions qui rendent le problème difficile. Une représentation de $\mathcal{S}$ et $\mathcal{Y}$ avec deux variables de décisions et deux objectifs sont présentés dans les figures (3.2) et (3.1) respectivement.

3.2.1 Test d'efficacité

Soit x^* une solution efficace d'un problème multiobjectif quelconque (3.1) et considérons le test de Benson, 2009 $Tb(x^*)$ suivant :

$$Tb(x^*) \quad \begin{cases} \max \quad \Psi = \sum_{i=1}^{k} \psi_i, \\ s.c. \\ \qquad f_i(x) - \psi_i = f_i(x^*),\ i \in \{1, \ldots, k\}, \\ \qquad x \in \mathcal{S}, \\ \qquad \psi_i \geq 0, \forall i \in \{1, \ldots, k\}. \end{cases} \tag{3.5}$$

Théorème 3.2. *x^* est efficace d'un problème MOILP si et seulement si la valeur optimale de $Tb(x^*)$ est nulle, i.e., $\psi_i^* = 0$, $\forall i = 1..k$.*

Théorème 3.3. *Soit (ψ^*, x^*) la solution optimale d'un test d'efficacité $Tb(x)$ d'une solution $x \in \mathcal{S}$, alors, x^* est une solution efficace.*

3.3 Méthodes de résolution

Dans cette section, nous présentons quelques méthodes exactes pour générer l'ensemble efficace d'un problème multiobjectif linéaire entier *MOILFP* et d'un problème linéaire fractionnaire *MOILFP* qui sont définis comme suit :

$$(MOILP)\begin{cases} z^i = \max \quad (c^i x) \quad i = 1..k, \\ \quad s.c, \\ \qquad\qquad Ax \leq b, \\ \qquad\qquad x \in \mathbb{N}. \end{cases} \tag{3.6}$$

$$(MOILFP)\begin{cases} z^i = \max \quad (\dfrac{c^i x + \alpha^i}{d^i x + \beta^i}) \quad i = 1..k, \\ \quad s.c, \\ \qquad\qquad Ax \leq b, \\ \qquad\qquad x \in \mathbb{N}. \end{cases} \tag{3.7}$$

où le nombre d'objectifs est $k \geq 2$, $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, on a $c^i = (c^i_1, c^i_2, \ldots, c^i_n) \in \mathbb{R}^{1\times n}$, $d^i = (d^i_1, d^i_2, \ldots, d^i_n) \in \mathbb{R}^{1\times n}$, α^i, β^i sont des constantes réelles.

3.3.1 Méthode de Chergui & Moulaî

Cette méthode Chergui et Moulaï, 2008 est basée sur le simplexe, la procédure de Branch & Bound et une coupe efficace pour résoudre un *MOILFP*. A Chaque itération l, on résout un problème mono-objectif LFP en maximisant un des objectif, le choix de l'objectif n'a pas d'influence sur la solution finale.

$$(LFP^1)\begin{cases} z^1 = \max \quad (\dfrac{c^1 x + \alpha^1}{d^1 x + \beta^1}), \\ \quad s.c, \\ \qquad\qquad Ax \leq b. \end{cases} \tag{3.8}$$

Soit x^*_l la solution entière obtenue après la résolution du problème (3.8), on note $\mathcal{B}_l$ l'ensemble des indices des variables de base, et $\mathcal{N}_l$ l'ensemble des indices des variables hors base de x^*_l.
Soit $\overline{\gamma}^i_j$ la j^{eme} composante du vecteur gradient réduit $\overline{\gamma}^i$ défini pour chaque critère i, $i \in$

$\{1, \ldots, r\}$ par :
$\overline{\gamma}^i = \overline{\beta}^i \overline{c}^i - \overline{\alpha}^i \overline{d}^i$ où $\overline{c}^i$, $\overline{d}^i$, $\overline{\alpha}^i$ et $\overline{\beta}^i$ sont les valeurs mises à jour de c^i, d^i, α^i et β^i respectivement. On définit l'ensemble $\mathcal{H}_l$ en x_l^* par :

$$\mathcal{H}_l = \{j \in \mathcal{N}_l, \mid \exists i \in \{1, \ldots, r\} \ : \ \overline{\gamma}_j^i > 0\} \cup \{j \in \mathcal{N}_l, \mid \ \overline{\gamma}_j^i = 0, \ \forall i \in \{1, \ldots, r\}\}$$

Ainsi, l'ensemble $\mathcal{X}_{l+1}$ est définit par $\mathcal{X}_{l+1} = \{x \in \mathcal{X}_l \mid \sum_{j \in \mathcal{H}_l} x_j \geq 1\}$.

Algorithme de la méthode de Chergui et Moulai :

L'algorithme de génération de toutes les solutions entières efficaces du programme (6.2) est présenté dans les l'algorithme suivant :

Algorithme 5 : Méthode du simplexe pour résoudre un MOILFP

Résultat : L'ensemble des solutions efficace $\mathcal{X}_E$

Etape (0) **: Initialisation (**$l = 0$**)**

$R = \{\mathcal{X}_0 = \{x, Ax \leq b\}\}$ ($\mathcal{X}_E = \emptyset$ ($\mathcal{X}_E$ étant l'ensemble des solutions efficaces du problème multiobjectif (3.7)).

Etape (1) **: Etape générale**

Tant que *il existe un noeud l non sondé dans l'arbre R* **faire**

Résoudre le programme linéaire fractionnaire correspondant au noeud l.

Si *le programme n'a pas de solution* **alors**

Le noeud l est sondé.

Sinon

Soit $\widetilde{x}_l$ la solution optimale obtenue.

Si $\widetilde{x}_l$ *n'est pas entière* **alors**

aller à l'étape $(1a)$.

Sinon

aller à l'étape $(1b)$.

Finsi

Finsi

Fintq

Etape (1a) :

Appliquer le processus de branch and bound, en ajoutant deux noeuds à R.

Aller à l'étape 1.

Etape (2b) :

Si le vecteur $f(\widetilde{x}_l)$ n'est pas dominé par le vecteur $f(x)$ pour toute solution $x \in \mathcal{X}_E$, alors $\mathcal{X}_E = \mathcal{X}_E \cup \{\widetilde{x}_l\}$.

S'il existe $x \in \mathcal{X}_E$ tel que $f(\widetilde{x}_l)$ domine $f(x)$, alors $\mathcal{X}_E = \mathcal{X}_E \backslash \{x\} \cup \{\widetilde{x}_l\}$.

Déterminer les ensembles $\mathcal{N}_l$ et $\mathcal{H}_l$.

- Si $\mathcal{H}_l = \emptyset$, alors le noeud correspondant est sondé. Aller à l'étape (1).
- Sinon, rajouter la coupe $\sum_{j \in \mathcal{H}_l} x_j \geq 1$ au programme (3.8). Aller à l'étape (1).

3.3.2 Methode de Sylva & Crema

La méthode de Sylva et Crema, 2004 est basée sur la résolution d'une séquence de programmes linéaires en nombres entiers qui optimisent une combinaison linéaire des critères à chaque étape. À chaque fois, un nouvel ensemble de contraintes est appliqué, assurant la découverte d'une nouvelle solution efficace. Enfin, la méthode renvoie l'ensemble de toutes les solutions non dominées de programmation linéaire discrète multiobjectifs. La méthode est résumé dans l'algorithme 6 :

Algorithme 6 : Méthode de Sylva & Crema

Résultat : $\mathcal{X}_E$ l'ensemble efficace du $MOILFP$

Choisir $\lambda > 0$ et les pas $\epsilon_i > 0$ et M^i les bornes sup de z^i, i = 1, ... k, $l = 0$,

$\mathcal{S}_l = \{x, Ax <= b, x \in \mathbb{N}\}$ $\mathcal{X}_E = \emptyset$.

Résoudre : $P^l_\lambda : max\,\{\lambda cx : x \in \mathcal{S}_l\}$.

Tant que P^l_λ *est réalisable* **faire**

Soit x^l la solution optimale de P^l_λ. Ajouter x^l à l'ensemble $\mathcal{X}_E$.

Poser l=l+1 et ajouter les contraintes suivantes à $\mathcal{S}_l$:

$$c^i x \geq (c^i x^l + \epsilon_i) y^i_l - M_i(1 - y^i_l), \forall i = 1..k.$$

$$\sum_{i=1}^{k} y^i_l = 1.$$

Fintq

3.4 L'optimisation sur l'ensemble efficace

Un problème d'optimisation sur l'ensemble peut s'écrire comme suit :

$$(P) \begin{cases} \max\ \phi(x), \\ s.c., \\ \qquad x \in \mathcal{X}_E, \end{cases}$$

où, ϕ est une fonction et $\mathcal{X}_E$ est l'ensemble des solutions efficaces d'un problème multiobjectif.

3.4.1 Méthode de Jorge

La méthode de Jorge, 2009 résout le problème d'optimisation d'une fonction linéaire sur l'ensemble efficace d'un *MOILP*. La méthode est itérative : à chaque étape, nous maximisons la fonction d'utilité sur l'ensemble des solutions admissibles du *MOILP*. Ensuite, nous testons l'efficacité de la solution. Si elle n'est pas efficace, le test d'efficacité renvoie une solution efficace qui sera la borne inférieure du problème. En utilisant les coupes utilisées par (Sylva et Crema, 2004), nous éliminons cette solution du domaine ainsi que toutes les solutions dominées par celle-ci. La méthode de Jorge est présentée dans l'algorithme 7.

Algorithme 7 : Méthode de Jorge

Résultat : La solution optimale x_{opt}

Initialisation : Poser $\phi_{inf} = -\infty$, $x_{opt} = \emptyset$ $\mathcal{S}_0 = \{Ax <= b, x \in \mathbb{N}\}$, $\phi_{sup} = +\infty$, $l = 0$, $(R_0) : \{max_{x \in \mathcal{S}} \ \phi(x)\}$.

Tant que R_l *a une solution optimale* x^l **faire**

- **Si** (x^l) *est efficace* **alors**
  - **Si** $\phi(x^l) > \phi_{inf}$ **alors**
    - $\phi_{inf} = \phi(x^l)$
    - $x_{opt} = x^l$
  - **Finsi**
  - **Retourner** x_{opt}
- **Sinon**
  - Résoudre (T_l) :

$$(T_l) : \max\{\phi(x) \backslash Cx = Cx^l, x \in \mathcal{S}\}.$$

  - Soit $\hat{x}^l$ la solution optimale de T_l.
  - **Si** $\phi(\hat{x}^l) > \phi_{inf}$ **alors**
    - Poser $\phi_{inf} = \phi(\hat{x}^l)$ et $x_{opt} = \hat{x}^l$.
  - **Finsi**
  - **Si** $\phi_{inf} = \phi_{sup}$, **alors**
    - **Stop**. x_{opt} est la solution optimale.
  - **Finsi**
  - Construire $(\mathcal{S}_{l+1})$:

$$\mathcal{S}_{l+1} = \mathcal{S}_l \cup \{Cx > C\hat{x}^l\}.$$

  - Poser l=l+1.
- **Finsi**

Fintq

3.4.1.1 Variantes de la méthode de Jorge

La méthode de Jorge peut être utilisée pour résoudre un problème d'optimisation d'une fonction convexe quelconque sur l'ensemble efficace d'un $MOILP$. Les étapes restent presque les mêmes sauf qu'au lieu d'optimiser une fonction linéaire à chaque étape, nous optimisons la fonction convexe en utilisant une méthode appropriée. Mahdi et Chaabane, 2015 ont considéré le cas où la fonction d'utilité est fractionnaire.

De plus, un certain nombre de méthodes récentes utilisent des techniques pour améliorer la méthode de Jorge. Lokman, 2021 propose une méthode pour éviter d'ajouter des contraintes et des variables de décision au programme R_l à chaque itération, ce qui ralentit la méthode. La méthode consiste à établir des bornes qui englobent tous les scénarios possibles pour les améliorations à apporter (au lieu d'utiliser des variables binaires). Ensuite, si ces bornes n'ont pas déjà été résolues, on les résout. Lorsqu'il y a moins de quatre objectifs, la méthode est efficace ; cependant, lorsqu'il y a plus de quatre objectifs, le nombre de bornes explose, et la méthode ralentit considérablement. Boland, Charkhgard et Savelsbergh, 2017, quant à eux, propose de diviser l'espace de recherche en rectangles afin de trouver la solution optimale efficace. Des tests ont montré que leur méthode est rapide et peut résoudre des problèmes de taille considérable.

3.4.2 Méthode de Drici & Moulaï

La méthode de (Drici, Ouail et Moulaï, 2018) est une méthode basé sur le simplexe en utilisant des coupes efficaces, elle permet de résoudre le problème d'optimisation d'une fonction fractionnaire sur l'ensemble efficace d'un problème $MOILP$, qui s'écrit comme suit :

$$(ILFP)_E \begin{cases} \max f(x) = \dfrac{p^T x + \alpha}{q^T x + \beta} \\ sc. \\ x \in \mathcal{X}_E \subset \mathcal{D}. \end{cases} \tag{3.9}$$

où p, q sont des vecteur de $\mathbb{R}^n$, α, $\beta \in \mathbb{R}$ et $\mathcal{X}_E$ représente l'ensemble des solutions efficaces du problème $(MOILP)$. On suppose que le facteur $q^T x + \beta$ est positif pour tout $x \in \mathcal{S}$. La méthode est résumé dans l'algorithme 8.

Algorithme 8 : Méthode de Drici & Moulaï

Résultat : La solution optimale x_{opt} pour un problème $ILFP_E$ et sa valeur associée ϕ_{opt}

Etape (0) **: Initialisation (**$l = 0$**)**
$\mathcal{S} = \{\mathcal{X}_0 = \{x, Ax \leq b\}\}$, $\phi_{opt} = -\infty$, $x_{opt} = \emptyset$
Etape (1) **: Etape générale**
Tant que *il existe un noeud l non sondé dans l'arbre R* **faire**
  Résoudre le programme linéaire fractionnaire correspondant au noeud l
  $(LFP_l) : \{max \ \ \phi(x), x \in \mathcal{S}\}$.
  Si *le programme n'a pas de solution* **alors**
    Le noeud l est sondé.
  Sinon
    Soit $\widetilde{x}_l$ la solution optimale obtenue.
    Si $\phi_{opt} \geq \phi(\widetilde{x}^{*(l)})$ **alors**
      Sonder le noeud l
    Finsi
    Si $\widetilde{x}_l$ *n'est pas entière* **alors**
      aller à l'étape $(1a)$.
    Sinon
      aller à l'étape $(1b)$.
    Finsi
  Finsi
Fintq
Etape (1a) :
Appliquer le processus de branch and bound. (Ajouter deux noeuds à R).
Aller à l'étape 1.
Etape (2b) :

— Si $\widetilde{x}_l$ est efficace, le noeud l est sondé. Mettre à jour la valeur de ϕ_{opt} si nécessaire, et aller à l'**étape 1**.
— Sinon, soit $\bar{x}_{(l)}$ une solution efficace retourné par le test. Mettre à jour la valeur de ϕ_{opt} si nécessaire.

Déterminer les ensembles $\mathcal{N}_l$ et
$\mathcal{H} = \{j \in N_l \mid \ni i \in \{1, \ldots, r\}; \bar{c}_j^i > 0\} \cup \{j \in N_l \mid \bar{c}_j^i = 0, \ \forall i \in \{1, \ldots, r\}\}$.

– Si $\mathcal{H}_l = \emptyset$, alors le noeud correspondant est sondé. Aller à l'étape (1).
– Sinon, rajouter la coupe $\sum_{j \in \mathcal{H}_l} x_j \geq 1$ au programme LFP_l Aller à l'étape (1).

Deuxième partie:

Travaux de recherche (contributions)

Chapitre 4

Optimisation d'une fonction quadratique sur l'ensemble efficace d'un problème multiobjectif fractionnaire

"It's true. I doubt. I doubt because I seek the truth. Doubt has served me well."
"C'est vrai. Je doute. Je doute parce que je cherche la vérité. Le doute m'a bien servi.",
Ibn al-Haytham.

Sommaire

4.1 Introduction 28
4.2 Outils théoriques 30
4.3 Description de la méthode 31
4.4 Exemple 35
4.5 Étude expérimentale 38
4.6 Conclusion 43

4.1 Introduction

L'optimisation sur l'ensemble efficace d'un *MOILFP* n'a pas reçu beaucoup d'attention. En fait, à notre connaissance, Zerdani et Moulai, 2011 sont les seuls à avoir proposé une méthode basée sur le simplexe pour optimiser une fonction linéaire sur l'ensemble efficace d'un *MOILFP*.

La plupart des méthodes d'optimisation sur l'ensemble efficace considèrent une fonction linéaire à optimiser sur l'ensemble efficace. Cependant, dans de nombreux domaines d'application en finance, économie, production et opérations, etc., la fonction d'utilité est modélisée comme une fonction quadratique (Gupta, 1995). Par conséquent, nous sommes intéressés

par l'optimisation d'une fonction quadratique sur l'ensemble efficace d'un *MOILFP*. L'optimisation sur l'ensemble efficace d'un problème *MOILP* devient un cas particulier, puisque l'optimisation d'une fonction linéaire est équivalent à l'optimisation d'une fonction fractionnaire avec un dénominateur constant. Par conséquent, cela permet de modéliser une plus grande classe de problèmes d'optimisation sur l'ensemble efficace. Par exemple, considérons la variante multiobjectif suivante du problème du sac à dos fractionnaire (Ishii, Ibaraki et Mine, 1977) :

$$(MOFKP) \begin{cases} \max\ f^i(x) = \dfrac{c^i x + \alpha^i}{d^i x + \beta^i}, \quad i \in \{1, .., r\}, \\ s.c., \\ \qquad \sum_{j=1}^{n} B_j x_j \leq W. \end{cases}$$

Le décideur veut appliquer la solution efficace du MOFKP, qui maximise le profit global donné par la fonction quadratique suivante :

$$\sum_{i=1}^{n} \sum_{j=1}^{n} p_{ij} x_i x_j,$$

où p est la matrice des bénéfices. Le problème consiste à optimiser une fonction quadratique sur l'ensemble efficace de *MOILFP*. Le problème abordé peut également être projeté sur un cas particulier de programmes de maximisation multiplicatif considérant deux termes (voir Mahmoodian, Charkhgard et Zhang, 2020, Shao et Ehrgott, 2014). De plus, ces derniers ont une application importante en théorie des jeux, comme le calcul de la solution de négociation de Nash, lorsque l'on considère par exemple deux joueurs ayant des fonctions d'utilités fractionnaires linéaires (voir Nash Jr, 1950, Saghand, Charkhgard et Kwon, 2019, Sierra-Altamiranda et al., 2020).

Dans ce chapitre, nous présentons une méthode exacte, qui peut être vue comme une extension de la méthode de (Jorge, 2009), pour résoudre le problème de l'optimisation d'une fonction quadratique sur l'ensemble efficace d'un *MOILFP*. Cette dernière est itérative, où à chaque itération le domaine réalisable est réduit, permettant de trouver la solution optimale sans énumérer explicitement toutes les solutions efficaces du problème.

4.2 Outils théoriques

Le problème d'optimisation d'une fonction quadratique sur l'ensemble efficace de $MOILFP$ s'écrit comme suit :

$$(Q/MOILFP)\begin{cases} \max\ \phi(x) = \frac{1}{2}x^tQx + qx, \\ s.c., \\ \qquad x \in \mathcal{X}_E, \end{cases} \tag{4.1}$$

avec x est un vecteur de dimension n, Q est une matrice semi-définie négative $n \times n$, q est un vecteur de dimension n et $\mathcal{X}_E$ est l'ensemble de toutes les solutions efficaces du $MOILFP$ suivant :

$$(MOILFP)\begin{cases} \max\ f^i(x) = \frac{c^ix + \alpha^i}{d^ix + \beta^i}, \quad i \in \{1, .., r\}, \\ s.c., \\ \qquad x \in \mathcal{D} = S \cap \mathbb{Z}^n. \end{cases} \tag{4.2}$$

Où r est le nombre de fonctions objectives, $S = \{x \in \mathbb{R}^n / Ax \leq b, x \geq 0\}$, $c^i, d^i \in \mathbb{R}^n, A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m$, $\alpha^i, \beta^i \in \mathbb{R}$ et $\mathcal{D}$ est supposé être non vide et convexe.

4.2.1 Test d'efficacité pour les programmes fractionnaires linéaires multiobjectifs

Pour tester l'efficacité d'une solution x^* d'un $MOILFP$, considérons le programme linéaire en variables mixtes suivant :

$$MM(x^*)\begin{cases} \max\ \Psi = \sum_{i=1}^{k} \psi_i, \\ s.c., \\ \qquad [c^i - Z_i(x^*)d^i]x - \psi_i = f_i(x^*)\beta^i - \alpha^i,\ i \in \{1, \ldots, k\}, \\ \qquad x \in \mathcal{D}, \\ \qquad \psi_i \geq 0, \forall i \in \{1, \ldots, k\}. \end{cases} \tag{4.3}$$

Théorème 4.1. *(Jahanshahloo et al., 2017) Une solution réalisable x^* du problème $MOILFP$ est efficace, si et seulement si, la valeur objectif optimale du problème $MM(x^*)$ est nulle.*

Théorème 4.2. *(Jahanshahloo et al., 2017) Supposons que x^0 soit une solution réalisable arbitraire d'un $MOILFP$. Soit $t = 0$ et on résout le problème $MM(x^t)$. Si la valeur optimale n'est pas 0, alors, x^t n'est pas efficace. Dans ce cas, on pose $t = t + 1$ et on résout $MM(x^t)$ jusqu'à ce*

qu'une solution efficace soit trouvée. La séquence $\{x^t\}_{1\leq t\leq\infty}$ *converge certainement vers une solution réalisable efficace du* $MOILFP$.

4.3 Description de la méthode

La méthode proposée consiste à résoudre à chaque itération un problème relaxé $(R^{(l)})$ du problème principal Q/MOILFP, en utilisant une méthode de programmation quadratique en variables mixtes, où $(R^{(l)})$ est défini comme suit :

$$(R^{(l)}) : \max \left\{ \phi(x) \mid x \in \mathcal{D}^{(l)}. \right\}$$

Où l est l'indice de l'itération courante. On initialise $\mathcal{D}^{(0)} = \mathcal{D}$, $x_{opt} = \emptyset$, $\phi_{opt} = -\infty$ et $\phi_{sup} = \infty$. Si $(R^{(0)})$ est infaisable, alors Q/MOILFP est également infaisable. Sinon, si $x^{*(0)}$ est la solution optimale de $(R^{(0)})$, on pose $\phi_{sup} = \phi(x^{*(0)})$. Ensuite, nous testons l'efficacité de $x^{*(0)}$ en résolvant $(MM(x^0))$. Si $x^{*(0)}$ est efficace alors $x^{*(0)}$ est la solution optimale du problème principal et l'algorithme s'arrête. Sinon, soit $\hat{x}^{(0)}$ la solution efficace obtenue par $(MM(x^{(0)}))$. Nous pouvons avoir une solution équivalente à $\hat{x}^{(0)}$ qui n'a pas, nécessairement, la même valeur de ϕ. Par conséquent, à chaque itération l de l'algorithme, nous résolvons le problème suivant :

$$\left(T(\hat{x}^{(l)})\right) : \max \left\{ \phi(x) \mid x \in \mathcal{D}, \frac{c^i x + \alpha^i}{d^i + \beta^i} = f^i(\hat{x}^{(l)}),\ \forall\, i \in \{1, \ldots, r\} \right\},$$

où $f^i(\hat{x}^{(l)}) = \dfrac{c^i \hat{x}^{(l)} + \alpha^i}{d^i \hat{x}^{(l)} + \beta^i}$. Soit $\tilde{x}^{(l)}$ la solution optimale de $(T(\hat{x}^{(l)}))$, si $\phi_{opt} < \phi(\tilde{x}^{(l)})$, on met à jour $x_{opt} = \tilde{x}^{(l)}$ et $\phi_{opt} = \phi(\tilde{x}^{(l)})$.

L'étape suivante de l'algorithme consiste à ajouter une coupe pour réduire $\mathcal{D}$, de telle sorte qu'elle ne contienne pas $\tilde{x}^{(l)}$ et toutes les solutions qu'elle domine, où $f^i(x) > f^i(\tilde{x}^{(l)})$ pour au moins un critère. Cette contrainte peut être écrite sous une forme différente en utilisant la même idée proposée par (Jorge, 2009) pour résoudre le problème de l'optimisation d'une fonction linéaire sur l'ensemble efficace d'un $MOILP$. Cette contrainte peut être écrite dans le cas fractionnaire comme suit :

$$\frac{c^i x + \alpha^i}{d^i x + \beta^i} \geq \left(\frac{c^i \tilde{x}^{(l)} + \alpha^i}{d^i \tilde{x}^{(l)} + \beta^i} + \epsilon^i \right) y_i^{(l)} - M^i \left(1 - y_i^{(l)}\right) \quad \forall i \in \{1, \ldots, r\}. \tag{4.4}$$

Où ϵ^i et $-M^i$ sont des constantes qui représentent l'incrément minimal de f^i et la borne inférieur de f^i respectivement.

De plus, afin de forcer, au moins un critère, à être amélioré à chaque itération l, nous ajoutons la contrainte suivante :

$$\sum_{i=1}^{r} y_i^{(l)} \geq 1.$$

Remarquons que si $y_i^{(l)} = 1$ une amélioration stricte sur f^i est imposée.

La contrainte 4.4 peut être reformulée comme suit :

$$(c^i + M^i d^i)x - \beta^i(z_i^{(l)} + \epsilon^i + M^i)y_i^{(l)} - (z_i^{(l)} + \epsilon^i + M^i)d^i x y_i^{(l)} \geq -\beta^i M^i - \alpha^i.$$

Cette contrainte n'est pas linéaire à cause du terme $d^i x y_i^{(l)}$. Cependant, puisque $y_i^{(l)} \in \{0,1\}$ nous pouvons la transformer en une contrainte linéaire en introduisant une nouvelle variable $w_i^{(l)} = d^i x y_i^{(l)}$, telle que, $w_i^{(l)} = d^i x$ si $y_i^{(l)} = 1$, et 0 sinon. Par conséquent, le domaine d'itération (l) est défini comme suit :

$$\mathcal{D}^{(l)} = \mathcal{D}^{(l-1)} \cap \begin{cases} (c^i + M^i d^i)x + \beta^i(z_i + \epsilon^i + M^i)y_i^{(l)} - (z_i^* + \epsilon^i + M^i)w_i^{(l)} \geq -\alpha^i - M^i\beta^i, \\ \sum_{i=1}^{r} y_i^{(l)} \geq 1, \\ w_i^{(l)} \leq \tau_i y_i^{(l)}, \\ w_i^{(l)} \leq d^i x, \\ w_i^{(l)} \geq d^i x - (1 - y_i^{(l)})\tau_i, \\ y_i^l \in \{0,1\}, \\ w_i^{(l)} \geq 0. \end{cases} \tag{4.5}$$

Où τ_i est borne supérieur de $d^i x$.

À chaque itération l de l'algorithme, on optimise une fonction quadratique sur le domaine $\mathcal{D}^{(l)}$ et on désigne sa valeur optimale par ϕ_{sup}. En ce qui concerne la condition d'arrêt, le processus itératif s'arrête lorsqu'au moins une des conditions suivantes est vérifiée : si le problème est infaisable, si la valeur de ϕ_{sup} est inférieure à ϕ_{opt} ou si la solution optimale est efficace. L'algorithme se termine avec ϕ_{opt} et sa solution associée x_{opt} comme sorties. Sinon, il réduit successivement le domaine en construisant un nouveau domaine $\mathcal{D}^{(l+1)}$ à partir du $\mathcal{D}^{(l)}$ en utilisant les coupes définies ci-dessus. La méthode proposée peut être résumée par l'algorithme (9) et l'organigramme (4.1). Nous appelons cet algorithme QPoverMOILFP.

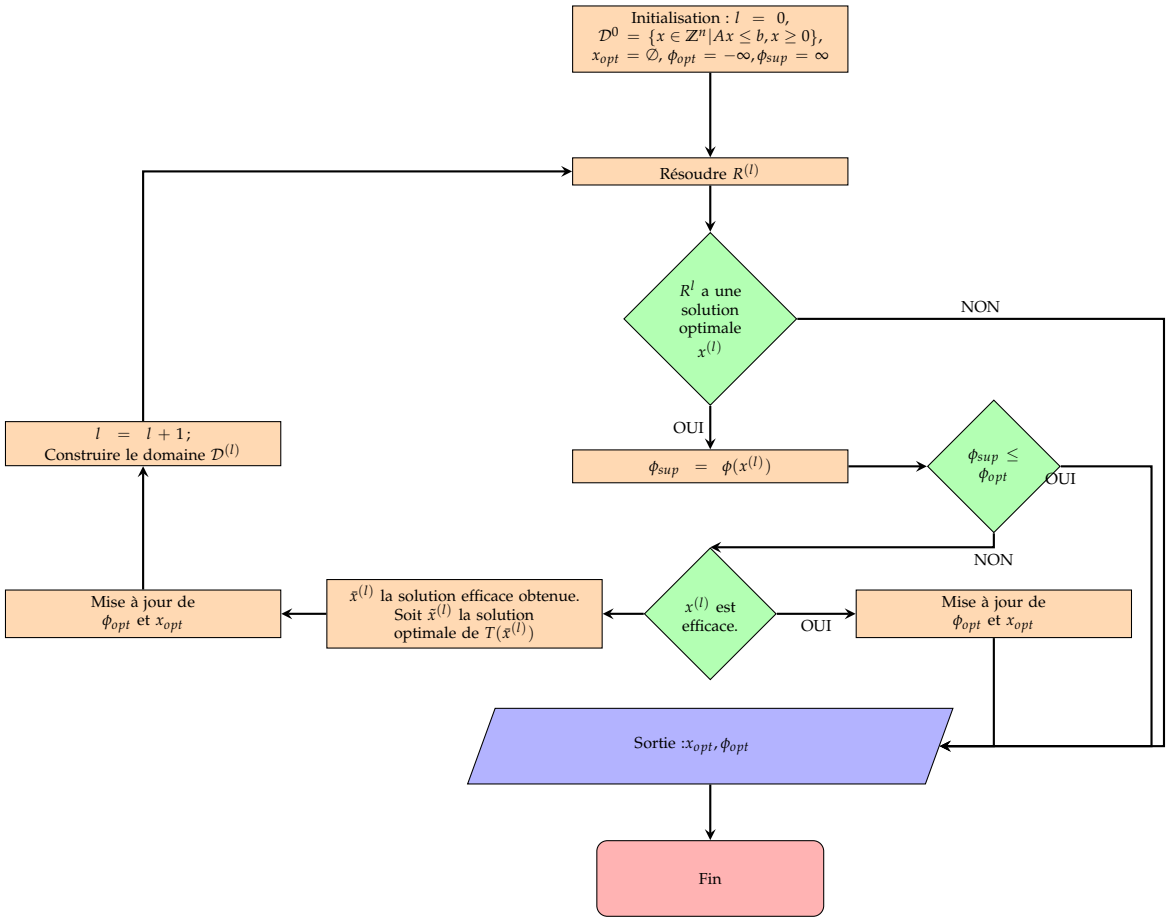

FIGURE 4.1 – Organigramme de la méthode proposée QPoverMOILFP.

Algorithme 9 : QPoverMOILFP

Résultat : La solution optimale x_{opt} pour le problème Q/MOILFP et sa valeur associée ϕ_{opt}.

Initialisation : $l = 0$, $\mathcal{D}^0 = \{x \in \mathbb{Z}^n | Ax \leq b \text{ et } x \geq 0\}$, $x_{opt} = \emptyset$, $\phi_{opt} = -\infty, \phi_{sup} = \infty$ et $continue = vrai$.

Tant que $\neg$ *la condition de terminaison est remplie* **faire**

- Solve $(R^{(l)})$ (4.3) avec n'importe quelle méthode QIP ;
- **Si** (R^l) *a la solution optimale* $x^{*(l)}$ **alors**
  - $\phi_{sup} = \phi(x^{*(l)})$;
  - **Si** $\phi_{sup} > \phi_{opt}$ **alors**
    - Résoudre $MM(x^{*(l)})$ et poser $\bar{x}^{(l)}$ la solution efficace obtenue par le test d'efficacité ;
    - Trouver une solution alternative à $\bar{x}^{(l)}$ en résolvant $(T(\hat{x}^{(l)}))$ et poser $\tilde{x}^{(l)}$ sa solution optimale ;
    - **Si** $\phi_{opt} \leq \phi(\tilde{x}^{(l)})$ **alors**
      - $\phi_{opt} = \phi(\tilde{x}^{(l)})$;
      - $x_{opt} = \tilde{x}^{(l)}$;
    - **Finsi**
    - $l = l + 1$;
    - Construire le domaine $\mathcal{D}^{(l)}$ comme définie dans (4.5) ;
  - **Sinon**
    - **Retourner** $x^{*(l)}$ *et* $\phi(x^{*(l)})$
  - **Finsi**
- **Sinon**
  - **Retourner** $x^{*(l)}$ *et* $\phi(x^{*(l)})$
- **Finsi**

Fintq

4.3.1 Validation de l'algorithme

Proposition 4.1. L'algorithme (QPoverMOILFP) fournit une solution optimale en un nombre fini d'itérations.

Démonstration. L'ensemble des solutions efficaces $\mathcal{X}_E$ est fini puisque la région réalisable $\mathcal{D}$ est un ensemble borné. À chaque itération de l'algorithme, une nouvelle solution efficace est trouvée car les solutions efficaces déjà rencontrées jusqu'à l'itération l deviennent infaisables

pour le programme $R^{(l)}$. Ainsi, l'algorithme se termine en un nombre fini d'itérations (Au plus $|\mathcal{X}_E|$ itérations). De plus, à chaque itération l, nous avons trois conditions pour terminer l'algorithme :

— Condition 1 : Le problème devient infaisable, ce qui signifie que le domaine est vide.
— Condition 2 : La valeur ϕ_{sup} est inférieure à ϕ_{opt} , ce qui signifie que $\phi_{opt} > \phi(x), \forall x \in \mathcal{D}^{(l)}$.
— Condition 3 : La solution optimale du programme $R^{(l)}$ est efficace, ce qui signifie qu'elle est optimale pour le problème Q/MOILFP dans le sous-domaine $\mathcal{D}^{(l)}$.

Il est évident que, l'exploration du sous-domaine restant pour les conditions 2 et 3, n'est pas nécessaire. Par conséquent, l'algorithme fournit la solution optimale pour Q/MOILFP en un nombre fini d'itérations. □

4.4 Exemple

Pour illustrer le fonctionnement de cet algorithme, nous considérons, par exemple, le problème suivant :

$$(Q/MOILFP)\begin{cases} \max\ \phi(x) = x_1^2 + x_2^2 + 2x_1x_2 - 9x_2, \\ s.c., \\ \qquad x \in \mathcal{X}_E. \end{cases}$$

Où $\mathcal{X}_E$ est la solution efficace du problème $MOILFP$ suivant :

$$(MOILFP)\begin{cases} \max\ f^1(x) = \dfrac{-4x_1 + 3x_2 - 3}{2x_1 + x_2 + 2}, \\ \max\ f^2(x) = \dfrac{x_1 + 2x_2 + 1}{x_1 + 3x_2 + 1}, \\ s.c., \\ \qquad x_1, x_2 \in \mathcal{D}. \end{cases}$$

Où $\mathcal{D} = \{-x_1 + x_2 \leq 0, 2x_1 + 3x_2 \leq 11, x_1 \geq 1, x_1, x_2 \in \mathbf{N}\}$.

Nous fixons M^i, τ_i et ϵ^i pour cet exemple $M = (-6, 10)\tau = (111, 88)$, $\epsilon = (0.1, 0.1)$, $l = 0$ et on initialise $\phi_{sup} = \inf$, $\phi_{inf} = -\inf$, $\phi_{opt} = \inf$ et $\mathcal{D}^{(0)} = \mathcal{D}$ représenté dans la Figure 4.2 et on résout $R^{(0)}$:

$$R^{(0)}\begin{cases} \max\ \phi(x) = x_1^2 + x_2^2 + 2x_1x_2 - 9x_2, \\ s.c., \\ x_1, x_2 \in \mathcal{D}^{(0)}. \end{cases}$$

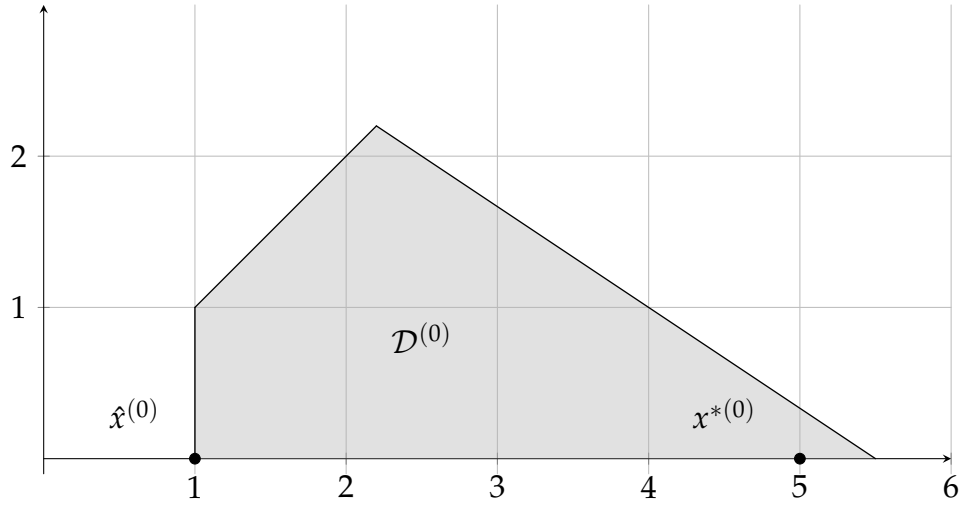

FIGURE 4.2 – La région admissible $\mathcal{D}^{(0)}$.

La solution optimale est $x^{*(0)}$ =(5,0), avec $\phi^{*(0)} = 25$. Fixez $\phi_{sup} = 25$. Le programme $(MM(x^{(0)}))$ retourne la solution efficace $\hat{x}^{(1)}$ = (1,0). Dans l'étape suivante, nous recherchons une autre solution efficace alternative avec un meilleur résultat de $\phi(x)$, en résolvant le programme suivant :

$$T(\hat{x}^{(l)}) \begin{cases} \max\ \phi(x) = 2x_1^2 + 14x_2^2 + 11x_1x_2 - 6x_1 + 10x_2, \\ s.c., \\ \qquad 4x_1 + 2x_2 + 10 = \frac{15}{7}(2x_1 + 3x_3 + 4), \\ \qquad 5x_1 + 5x_2 + 3 = 2(x_1 + 3x_3 + 9), \\ \qquad x_1, x_2 \in \mathcal{D}. \end{cases}$$

La solution optimale est $\tilde{x}^{(1)}$ (c'est-à-dire qu'aucune solution alternative avec une meilleure valeur de ϕ n'est trouvée). Définissez $\phi_{opt} = 1$ et $x_{opt} = (1, 0)$.
Construire le domaine $\mathcal{D}^{(1)}$ tel que défini dans 4.5, définir $l = l + 1$. Le domaine $\mathcal{D}^{(1)}$ est représenté sur la figure 4.3, où Δ_1 représente la première contrainte dans le cas où $y_1^1 = 1$ (c'est-à-dire que le premier objectif doit être amélioré) et Δ_2 représente la deuxième contrainte dans le cas où $y_2^1 = 1$ (c'est-à-dire que le deuxième objectif doit être amélioré). Remarquez que si $y_2^1 = 1$, le domaine devient infaisable. Par conséquent, on peut remarquer que la figure 4.3 ne montre que le domaine réduit dans le cas où $y_1^1 = 1$. Le domaine obtenu $\mathcal{D}^{(1)}$, en utilisant la construction donnée ci-dessus, est le suivant :

$$\mathcal{D}^{(1)} = \mathcal{D}^{(0)} \cap \begin{cases} 16,093x_1 + 3,046x_2 - 15,3930y_1^1 - 7,6965w_1^1 \leq -15,093, \\ -11x_1 - 32x_2 + 11.1y_2^1 - 11.1w_1^1 \leq 11, \\ y_1^1 + y_2^1 \geq 1, \\ w_1^1 \leq 110y_1^1, \\ w_2^1 \leq 88y_2^1, \\ w_1^1 \leq 2x_1 + x_2, \\ w_1^1 \leq 1x_1 + 3x_2, \\ w_1^1 \geq 2x_1 + x_2 + 110(1 - y_1^1), \\ w_2^1 \geq 1x_1 + 3x_2 + 88(1 - y_2^1), \\ y_1^1, y_2^1 \in \{0, 1\}, \\ w_1^1, w_2^1 \geq 0. \end{cases}$$

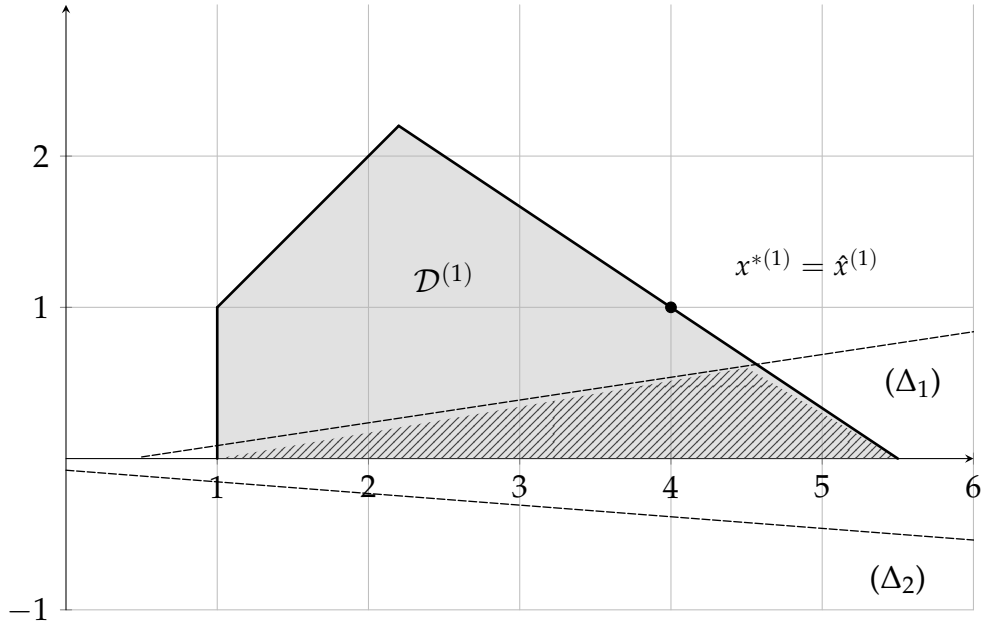

FIGURE 4.3 – La région admissible $\mathcal{D}^{(1)}$

La solution optimale de $(R^{(1)})$ est $x^{*(1)} = (4, 1)$ avec $\phi^{*(1)} = 16$, qui est efficace. De plus, puisque $\phi_{opt} < 16$, on actualise $x_{opt} = (4, 1)$, $\phi_{opt} = 16$ et l'algorithme s'arrête.

4.5 Étude expérimentale

Dans cette section, nous présentons l'étude expérimentale pour tester l'efficacité de la méthode suggérée. En effet, à notre connaissance, il n'existe aucune méthode dans la littérature qui traite le problème de l'optimisation d'une fonction quadratique sur l'ensemble efficace d'un *MOILFP*. Par conséquent, nous avons décidé de comparer la méthode proposée avec un algorithme brute-force (10), qui consiste à trouver l'ensemble efficace $\mathcal{X}_E$ en utilisant la méthode décrite par (Chergui et Moulaï, 2008), puis à sélectionner une solution dans $\mathcal{X}_E$ qui maximise la fonction de préférence du décideur.

Algorithme 10 : Algorithme brute-force pour IQP/MOILFP

Résultat : La solution optimale x_{opt} pour (4.1) et sa valeur ϕ_{opt}

Trouver l'ensemble efficace $\mathcal{X}_E$ pour le problème (4.2) ;

$x_{opt} = \arg\max \{\phi(x), x \in \mathcal{X}_E\}$ $\phi_{opt} = \phi(x_{opt})$;

La méthode proposée (9) et l'algorithme brute-force (10) sont implémentés dans l'environnement Matlab r2016a, avec un Intel Core i5 de 2,7 GHz et 16 GO de RAM. Nous avons réalisé les expériences à venir en utilisant des instances générées aléatoirement, catégorisées en fonction de leurs tailles. Pour chaque catégorie, nous avons généré trois instances.

Une instance aléatoire est générée uniformément de la manière suivante [1] : $A = (a_{i,j}) \in \mathcal{U}(0,30)$, $b = (b_i) \in \mathcal{U}(50,100)$, $q_j, c = (c_j^r), \alpha^k \in \mathcal{U}(-100,100)$, $d^k = (d_j^k), \beta^k \in \mathcal{U}(1,100)$ pour éviter que le dénominateur soit inférieur ou égal à zéro. La matrice Q est générée par la construction suivante : d'abord, nous générons aléatoirement $Q = (Q_{ij}) \in \mathcal{U}(-5,5)$, puis nous définissons $Q = -QQ^t$; ceci garantit que Q est semi-définie négative. En résolvant certaines instances, notamment de grande taille, nous avons remarqué qu'à chaque itération, la résolution du problème $T(x^{(l)})$ 4.3 prend un temps considérable, et renvoie souvent la même solution $x^{(l)}$ (c'est-à-dire qu'il n'y a pas de solution alternative à $x^{(l)}$). Afin de surmonter ce problème d'efficacité, nous avons utilisé $x^{(l)}$ comme solution initiale pour résoudre $T(x^{(l)})$.

L'expérience menée consiste en deux phases. Dans la première phase, les deux méthodes concurrentes sont comparées (c'est-à-dire l'algorithme QPoverMOILFP et l'algorithme brute-force). La deuxième phase de l'expérience consiste à résoudre des instances de taille moyenne en utilisant uniquement notre méthode. Le tableau (4.1) résume le temps d'exécution des deux méthodes en donnant le temps d'exécution minimum, maximum et moyen des algorithmes concurrents, ainsi que le nombre d'itérations minimum, maximum et moyen de

1. Nous désignons par $\mathcal{U}(I_{min},\ I_{max})$ une fonction qui génère uniformément un nombre entier aléatoire entre I_{min} et I_{max}.

notre méthode. La colonne $|\mathcal{X}_E|$ donne la taille moyenne de l'ensemble efficace et la dernière colonne μ représente le rapport entre le nombre moyen de solutions efficaces visitées par la méthode proposée sur la taille de l'ensemble efficace. Le tableau 4.1 montre que la méthode proposée génère en moyenne 3% de l'ensemble efficace, avec au moins 1% et un ratio maximum de 17%. De plus, le temps d'exécution de la méthode proposée est faible par rapport à l'algorithme brute-force : le temps moyen requis pour résoudre les instances, de taille $(5 \times 40 \times 20)$, est inférieur à 11(s). Contre, en moyenne, plus de 740(s) requis pour résoudre les mêmes instances, avec l'algorithme brute-force. De plus, la figure 4.4 montre que la méthode proposée offre une amélioration significative par rapport à l'algorithme brute-force en considérant l'évolution du temps d'exécution lorsque la taille des instances augmente.

Compte tenu de l'important coût de calcul requis par l'algorithme brute-force, nous avons choisi de procéder à la deuxième phase de cette étude pour les instances de taille moyenne avec la méthode proposée uniquement. Le tableau 4.2 contient des résultats supplémentaires. Nous avons également testé l'algorithme proposé avec trois ensembles d'instances : Le problème d'affectation (AP) et le problème de sac à dos (KP), générés dans (Kirlik et Sayin, 2014), et le problème général multiobjectif (GMP), généré dans (Kirlik et Sayin, 2015). Ces ensembles d'instances sont disponibles sur (*Multiobjective Optimization Library*). Nous avons couvert dans cette expérience un total de 10 instances pour chaque taille d'instance. Cependant, ces instances de données sont originellement générées selon une structure $MOILP$. Pour adapter ces dernières au problème traité, nous avons généré pour chaque instance $d^k = (d^k_j), \beta^k, \alpha^k$ et q de manière aléatoire, comme mentionné ci-dessus, et $Q = (Q_{ij}) \in \mathcal{U}(-20, 20)$ (au lieu de $\mathcal{U}(-5, 5)$), puis fixé $Q = -QQ^t$. Les résultats sont présentés dans le tableau 4.3.

Si l'on considère l'effet de la variation de la taille de l'instance sur le comportement de l'algorithme, comme le montrent les tableaux 4.1, 4.2 et 4.3, il a montré une cohérence assez élevée en termes de nombre d'itérations lorsque la taille de l'instance augmente (c'est-à-dire que le nombre de solutions efficaces visitées reste faible alors que la taille de l'ensemble efficace augmente). En outre, comme le montrent les tableaux (4.2, 4.3) et la figure (4.5), le temps d'exécution est raisonnable. Par ailleurs, la variation du nombre d'objectifs n'a pas d'effet significatif sur le coût de calcul.

TABLE 4.1 – Temps d'exécution des instances aléatoires de petite taille.

$r\times$	$m \times n$	**QPoverMOILFP**						**Brute-force**				
		Temps (s)			**Nombre d'itération**			**Temps (s)**			**moyenne** $\lvert\mathcal{X}_E\rvert$	μ
		min	max	moyenne	min	max	moyenne	min	max	moyenne		
3	5×5	0.03	0.08	0.04	1	1	1	0.36	0.8	0.6	17.33	0.07
	10×5	0.02	0.19	0.1	1	4	2.67	2.46	3.91	3.13	46.67	0.08
	20×10	0.06	0.2	0.11	1	1	1	15.15	23.89	20.06	33.67	0.03
	30×15	0.5	1.37	0.88	4	7	5.67	109.42	326.37	184.97	55.67	0.11
	40×20	6.19	57.53	24.75	2	11	6.33	181.12	379.02	302.55	64	0.1
	50×25	Â 1.1	10.54	6.87	3	11	6.33	347.5	2698.2	1389.1	68	0.1
5	5×5	0.10	0.14	0.11	1	2	1.67	0.54	3.09	1.42	62.67	0.03
	10×5	0.07	0.14	0.10	2	3	2.33	1.20	10.14	6.24	84.33	0.04
	20×10	0.05	0.17	0.11	1	4	2.33	16.05	94.17	50.02	177.67	0.01
	30×15	0.13	19.60	6.64	1	13	5	276.38	368.26	321.29	271.67	0.03
	40×20	0.18	25.94	10.83	1	9	4	596.64	992.88	742.14	316	0.01
	50×25	3.52	79.95	36.47	1	9	6.33	-	-	-	-	-
7	5×5	0.02	0.15	0.09	1	4	2	0.29	1.63	0.75	22.67	0.11
	10×5	0.03	0.15	0.08	1	3	1.67	6.17	19.72	12.76	427.67	0.01
	20×10	0.07	0.15	0.11	1	2	1.67	8.02	52.99	27.51	343.00	0.02
	30×15	0.080	31.72	10.64	1	6	2.67	94.54	967.48	389.35	1041.33	0.01
	40×20	0.22	1.28	0.67	1	3	2	228.34	1217.13	710.67	564.67	0.01
	50×25	0.63	8.94	4.53	1	3	1.33	1100.45	-	-	-	-

- : incapable de résoudre l'instance correspondante dans le délai prédéfini de 5000 secondes.

TABLE 4.2 – Temps d'exécution d'instances aléatoires de taille moyenne de la méthode suggérée.

$r\times$	$m \times n$	Temps (s)			Nombre de noeuds		
		min	max	moyenne	min	max	moyenne
3	60 ×30	2.34	27.57	10.94	1	8	3.33
	70 × 35	4.08	17.64	11.74	2	4	3
	80 ×40	28.11	34.88	31.73	4	5	4.33
	90 × 45	11.08	634.65	251.42	1	12	6.67
	100 × 50	23.12	710.29	284.84	2	6	4.67
	110 × 55	25.72	766.55	282.26	1	5	2.67
	120 × 60	320.60	1015.34	596.21	3	7	5
5	60 ×30	1.32	8.47	4.58	1	3	1.67
	70 × 35	3.7	36.27	15.66	1	7	3.33
	80 ×40	3.09	1100.31	376.11	1	13	6
	90 × 45	33.68	452.31	191.01	3	8	6
	100 × 50	29.43	399.85	214.49	2	7	5
	110 × 55	11.72	413.21	251.31	2	12	6.33
	120 × 60	43.09	373.27	556.27	2	4	3
7	60 ×30	1.27	216.28	135.26	2	4	3
	70 × 35	0.18	307.60	105.60	1	4	2.33
	80 × 40	19.61	313.95	143.62	1	6	3
	90 × 45	0.48	501.10	227.45	1	5	2.67
	100 × 50	23.59	654.83	370.56	3	7	4.67
	110 × 55	0.39	268.09	138.66	1	4	2.67
	120 × 60	34.50	986.71	398.71	2	7	4

TABLE 4.3 – Temps d'exécution de la méthode proposée avec les instances KP, AP et GMO.

Nom d'instance	r	Taille	Temps(s)			Nombre de noeud		
			min	max	moyenne	min	max	moyenne
GMO	3	10×5	0.12	0.95	0.44	3	12	6.40
		20×10	0.17	3.19	1.09	3	14	7.60
		30×15	0.11	29.00	4.66	2	18	7.00
		40×20	0.22	21.82	7.48	2	17	10.20
		50×25	0.75	49.74	11.45	2	18	9.20
	4	10×5	0.08	4.65	1.19	2	16	6.90
		20×10	0.17	9.50	2.27	2	18	8.20
		30×15	0.13	1188.89	127.33	2	31	11.10
		40×20	0.43	157.27	32.86	4	27	12.40
		50×25	0.17	65.81	17.56	2	18	9.60
	5	10×5	0.12	13.10	3.08	2	15	6.90
		20×10	0.19	34.51	4.86	3	24	9.20
		30×15	0.23	519.00	65.07	2	32	6.40
		40×20	0.68	689.49	142.25	5	31	16.60
KP	3	10	0.06	0.50	0.32	2	6	3.90
		20	0.27	3.26	1.27	2	13	6.60
		30	2.54	276.18	51.12	7	22	12.30
		40	35.37	507.86	252.49	3	18	9.30
		50	54.28	2365.91	832.43	3	9	5.30
	4	10	0.13	4.05	0.97	2	15	4.90
		20	0.56	24.14	6.14	3	20	10.70
		30	4.89	295.83	77.76	5	28	15.10
		40	23.97	1161.82	397.93	3	23	11.30
	5	10	0.05	0.98	0.53	2	9	5.2
		20	0.22	68.87	8.50	2	28	9.70
AP	3	5	0.22	7.65	1.08	2	4	2.60
		10	0.22	28.01	6.40	2	6	2.70

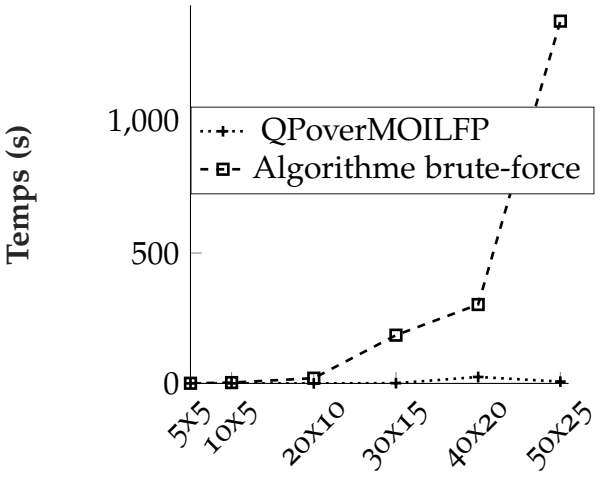

FIGURE 4.4 – Temps CPU pour les méthodes concurrentes (k=3)

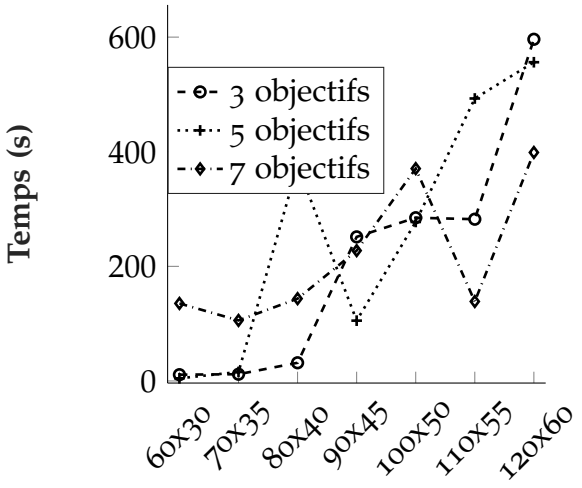

FIGURE 4.5 – Temps CPU pour des instances aléatoires de taille moyenne

4.6 Conclusion

Dans ce chapitre, nous avons présenté une méthode pour résoudre le problème d'optimisation d'une fonction quadratique sur l'ensemble efficace d'un problème multiobjectif en introduisant des coupes pour éliminer toutes les solutions dominées à chaque itération. Dans l'étude expérimentale, nous avons montré que la méthode permet de résoudre des instances de petite taille ainsi que des instances de taille moyenne. De plus, cette étude peut être étendue à d'autres fonctions de préférence, où la méthode QPoverMOILFP peut facilement être adaptée pour résoudre le problème d'optimisation d'une fonction d'utilité convexe sur l'ensemble efficace d'un *MOILFP*. Cependant, la méthode QPoverMOILFP ajoute progressivement des contraintes de disjonction qui diminuent considérablement l'efficacité de l'approche suggérée. Dans des travaux futurs, nous envisageons d'étudier l'adaptation d'autres schémas en utilisant des décompositions de l'espace des critères proposées dans la littérature pour surmonter cet inconvénient.

Ce travail a été publié (CHAIBLAINE et MOULAÏ, 2021) dans une revue internationale de renom : *Optimization Letters*.

Chapitre 5

L'optimisation de deux fonctions fractionnaire sur l'ensemble efficace d'un MOILFP

"You don't have to be a mathematician to have a feel for numbers."
"Il n'est pas nécessaire d'être mathématicien pour avoir le sens des chiffres.",
John Forbes Nash.

Sommaire

5.1 Introduction 44
5.2 Définitions et préliminaires 46
5.3 Méthodologie, algorithme et résultats théoriques 48
5.4 Résultats théoriques 53
5.5 Exemple didactique 57
5.6 Étude expérimentale 65
5.7 Conclusion 68

5.1 Introduction

L'optimisation d'une fonction non linéaire ou linéaire sur l'ensemble efficient est un domaine intéressant de la programmation multiobjectif (Miettinen, 2012). Il s'agit d'un moyen simple d'éviter d'énumérer toutes les solutions efficaces en les évaluant et en les distinguant les unes des autres à l'aide d'une fonction qui résume les préférences des décideurs.

Considéré pour la première fois par Philip, 1972, le problème de l'optimisation sur l'ensemble efficace a depuis attiré l'attention de plusieurs chercheurs, parmi lesquels Jorge, 2009 qui a proposé une méthode exacte qui résout successivement des programmes mono-objectif. Zerdani et Moulai, 2011 ont optimisé une fonction linéaire sur l'ensemble efficient entier de *MOILFP* en utilisant le test d'efficacité de (Ehrgott et al., 1997). Plus récemment,

Moulaï et Drici, 2018 ont également proposé une méthode exacte basé sur le branch-and-bound combiné avec des coupes efficaces. Liu et Ehrgott, 2018 ont présenté des algorithmes primaux et duaux pour résoudre ce problème. D'autres méthodes peuvent être trouvées dans l'étude de Yamamoto, 2002.

La plupart du temps, dans ce type de problèmes, nous avons plusieurs décideurs et chacun d'entre eux peut avoir une fonction d'utilité. Cherfaoui et Moulaï, 2021 ont traité le cas où il y a deux fonctions linéaires à optimiser sur l'ensemble efficace d'un problème multiobjectif linéaire en nombres entiers. Cependant, en pratique, la mesure d'une qualité, d'une rentabilité ou d'une probabilité, etc., sont formulées comme des fonctions fractionnaires. La programmation fractionnaire est rencontrée dans plusieurs domaines d'application tels que : le problème de découpe (Gilmore et Gomory, 1961), l'optimisation de la qualité des formes (Munson, 2007), les problèmes de clustering (Hansen et Jaumard, 1997), etc., d'autres applications de la programmation fractionnaire peuvent être trouvées dans (Schaible, 1977), la bibliographie de Stancu-Minasian, 2012.

Dans ce chapitre, nous présentons une généralisation de la méthode de (Cherfaoui et Moulaï, 2021). Nous considérons la généralisation avec deux fonctions fractionnaires linéaires (BiObjective Integer Linear Fractional Program *BOILFP*) que nous voulons optimiser sur l'ensemble efficace d'un *MOILFP*. Ce problème modélise la problématique pratique suivante : Une entreprise "A" souhaite sous-traiter une partie de sa production. Dans le cahier des charges, l'entreprise "A" a deux critères : optimiser la rentabilité et la qualité, chaque critère est représenté par une fonction linéaire fractionnaire. D'autre part, une entreprise "B" veut reprendre le projet et a en même temps plusieurs fonctions objectifs linéaires fractionnaires. L'entreprise "B" veut savoir s'il existe une ou plusieurs solutions avantageuses pour les deux entreprises. En d'autres termes, trouver les solutions qui sont efficaces à la fois pour le problème *BOILFP* et le problème *MOILFP*. Une façon de résoudre ce problème est d'énumérer les ensembles efficaces de *BOILFP* et *MOILFP* et de trouver l'intersection entre les deux ensembles efficaces. Cette méthode a un coût de calcul important comme nous le montrerons dans l'étude expérimentale.

La méthode proposée est une procédure exacte de branch and cut qui fournit les solutions efficaces exactes pour *BOILFP* et *MOILFP*. Cette procédure a l'avantage d'éviter l'exploration inutile du domaine en utilisant des coupes qui éliminent les solutions dominées. Elle utilise également des tests d'efficacité permettant de ne retenir que les bonnes solutions.

5.2 Définitions et préliminaires

Soit LFP^1 un programme linéaire fractionnaire définit comme suit :

$$(LFP^1)\begin{cases} \max f_1(x) = \dfrac{p^1x + \alpha^1}{q^1x + \beta^1}, \\ s.c., \\ x \in \mathcal{X}, \end{cases} \tag{5.1}$$

où $\mathcal{X} = \{x \in \mathbb{R}^n | Ax \leq b,\ x \geq 0\}$, p^1, q^1 sont des vecteurs, α^1 et β^1 sont des scalaires, b est un vecteur de taille m, A est une matrice $n \times m$ et x est le vecteur de décision.

L'ensemble $\mathcal{X}$ est supposé être un polytope convexe non vide et l'ensemble $\mathcal{D} = \mathcal{X} \cap \mathbb{Z}^n$ est également supposé non vide. Nous supposons également que $q^1x + \beta^1 > 0, \forall x \in \mathcal{X}$.

Rappelons la méthode 2 de (Cambini et Martein, 1986). Cette méthode utilise le gradient réduit pour se rapprocher de l'optimum pour une base $\mathcal{B}_l$ et une solution associée $x^{*(l)}$ en utilisant les notations suivantes :

$$\begin{aligned} \nu^{1(l)} &= p^1 - p^1_{\mathcal{B}_l}\mathcal{B}_l^{-1}A, \\ \mu^{1(l)} &= q^1 - q^1_{\mathcal{B}_l}\mathcal{B}_l^{-1}A, \\ P^1(x) &= p^1x^{*(l)} + \alpha^1, \\ Q^1(x) &= q^1x^{*(l)} + \beta^1. \end{aligned}$$

Le gradient réduit du problème (LFP^1) peut s'écrire comme suit :

$$\boldsymbol{\gamma^{1(l)}} = Q^1(x)\nu^{1(l)} - P^1(x)\mu^{1(l)}.$$

Le gradient réduit donne le sens de croissance de la fonction objectif $f_1(x)$, donc si pour tout indice j de l'ensemble des indices hors base $\mathcal{N}_l$, $\boldsymbol{\gamma}_j^{\mathbf{1(l)}} \leq 0$, alors la fonction $f_1(x)$ ne peut être améliorée.

Soit le problème multiobjectif linéaire fractionnaire suivant :

$$(MOILFP)\begin{cases} \max Z_1(x) = \dfrac{c^1x + c_0^1}{d^1x + d_0^1}, \\ \max Z_2(x) = \dfrac{c^2x + c_0^2}{d^2x + d_0^2}, \\ \vdots \\ \max Z_k(x) = \dfrac{c^kx + c_0^k}{d^kx + d_0^k}, \\ s.c. \\ x \in \mathcal{D}, \end{cases} \tag{5.2}$$

où $k \geq 2$, c^i, d^i sont des vecteurs de taille n, c_0^i, d_0^i sont des scalaires pour chaque $i \in \{1, 2, \ldots, k\}$. Tout au long de ce chapitre, nous supposons que $d^ix + d_0^i > 0$ sur $\mathcal{X}$ pour tout $i \in \{1, 2, \ldots, k\}$.

Certains chercheurs ont abordé ce problème, en particulier dans le cas continu. Kornbluth et Steuer, 1981 ont présenté une procédure basée sur le simplexe pour trouver tous les solutions extrêmes faiblement efficaces, Tandis que, Costa, 2007 a proposé une nouvelle technique pour optimiser une somme pondérée des fonctions objectifs linéaires fractionnaires. Martein et Stancu-Minasian, 1999 ont rédigé une étude sur les problèmes fractionnaires biobjectifs.

En général, étant donné la lourdeur des calculs, les auteurs évitent de générer l'ensemble efficace. Cependant, Chergui et Moulaï, 2008 ont utilisé une procédure de branchement et de coupe pour générer l'ensemble efficace. Dans ce chapitre, nous présentons une méthode exacte qui génère une partie de l'ensemble efficace de $MOILFP$ qui est définie par les préférences des décideurs. Supposons donc l'existence de deux fonctions d'utilité f_1 et f_2 :

$$(BOILFP)\begin{cases} \max f_1(x) = \dfrac{p^1x + \alpha^1}{q^1x + \beta^1}, \\ \max f_2(x) = \dfrac{p^2x + \alpha^2}{q^2x + \beta^2}, \\ s.c., \\ x \in \mathcal{D}. \end{cases} \tag{5.3}$$

où p^1, p^2, q^1, q^2 sont des vecteurs, α^1, β^1 et α^2, β^2 sont des scalaires. Dans tout ce

chapitre, nous supposons également que $q^1x + \beta^1 > 0$ et $q^2x + \beta^2 > 0$ sur $\mathcal{X}$ et les notations suivantes sont utilisées pour $\mathcal{B}_l$, $\mathcal{N}_l$ et $x^{*(l)}$:

$$\begin{array}{lll} P^s(x) = p^s x + \alpha^s, & Q^s(x) = q^s x + \beta^s, & s \in \{1,2\}, \\ z_i^1(x) = c^i x + c_0^i, & z_i^2(x) = d^i x + d_0^i, & i \in \{1,2,\ldots,k\}, \\ v^{s(l)} = p^s - p^s_{\mathcal{B}_l}\mathcal{B}_l^{-1}A, & \mu^{s(l)} = q^s - q^s_{\mathcal{B}_l}\mathcal{B}_l^{-1}A, & s \in \{1,2\}, \\ \eta^{i(l)} = c^i - c^i_{\mathcal{B}_l}\mathcal{B}_l^{-1}A, & \vartheta^{i(l)} = d^i - d^i_{\mathcal{B}_l}\mathcal{B}_l^{-1}A, & i \in \{1,2,\ldots,k\}. \end{array}$$

Par conséquent, le problème que nous proposons de résoudre est le suivant :

$$(B_E) \begin{cases} \max f_1(x) = \dfrac{p^1x + \alpha^1}{q^1x + \beta^1}, \\ \max f_2(x) = \dfrac{p^2x + \alpha^2}{q^2x + \beta^2}, \\ s.c., \\ x \in \mathcal{X}_E, \end{cases} \tag{5.4}$$

où $\mathcal{X}_E$ est l'ensemble des solutions efficaces du MOILFP, on note également $\mathcal{X}_{E'}$ l'ensemble des solutions efficaces du *BOILFP*. Le problème B_E a une réelle utilité pratique, pourtant il n'a pas été considéré. Il se pose, par exemple, lorsque deux entreprises doivent optimiser leur intérêt respectif tout en ayant le même espace de décision.

5.3 Méthodologie, algorithme et résultats théoriques

L'approche naïve pour résoudre B_E consiste à énumérer les éléments de $\mathcal{X}_E$ et de $\mathcal{X}_{E'}$ et à déterminer leur intersection. Dans ce qui suit, nous proposons une méthode qui évite l'énumération des deux et donne l'ensemble des solutions exactes de B_E. Cette méthode combine la technique de branch & bound et la méthode du branch & cut. L'ensemble des solutions de B_E est noté $\mathcal{X}_{BE}$.

Dans ce chapitre, nous allons présenter une méthode équivalente du branch & cut de (Chergui et Moulaï, 2008) pour éliminer les solutions qui n'appartiennent pas à $\mathcal{X}_E$ ou $\mathcal{X}_{E'}$. La méthode peut être résumée comme suit :

Le processus de branchement La recherche de solutions efficaces est une exploration implicite structurée comme un arbre où l'on peut choisir f_1 ou f_2 à optimiser à chaque noeud. Le choix de la fonction f_1 ou f_2 ne changera pas l'ensemble de résultats $\mathcal{X}_{BE}$,

cependant, il modifiera l'arbre de recherche et changera l'évolution de $\mathcal{X}_{BE}$ mais pas le résultat final.

À chaque noeud l de l'arbre, nous optimisons la fonction f_1 sur le sous-domaine correspondant au noeud $\mathcal{X}_l$.

$$(LFP_l)\begin{cases} \max f_1(x) = \dfrac{p^1 x + \alpha^1}{q^1 x + \beta^1}, \\ s.c., \\ x \in \mathcal{X}_l, \end{cases}$$

où $\mathcal{X}_0 = \mathcal{X}$ et $\mathcal{X}_l$ est un sous-ensemble de l'ensemble original $\mathcal{X}$ à explorer au noeud l de l'arbre.

Les fonctions f_1 et f_2 sont linéaires fractionnaires. Pour résoudre le (LFP_l), on peut utiliser la méthode présentée dans (Charnes et Cooper, 1962) ou une autre méthode simplexe comme celles présentées dans (Martos, 1964), (Cambini et Martein, 1986).

Après avoir résolu (LFP_l), trois cas peuvent se présenter :

(LFP_l) **n'a pas de solution** Comme pour un processus classique de Branch-and-Bound, le noeud n'a pas de descendant.

(LFP_l) **a une solution non entière** $x^{*(l)}$ **:** Si la solution de (LFP_l) n'est pas entière, on procède suivant l'approche conventionnelle du Branch-and-Bound pour la programmation en nombres entiers, ce qui signifie que le noeud a deux descendants l_1 et l_2 tels que : Soit r un indice où $x_r^{*(l)}$ n'est pas entier alors pour l_1 on met

$$\mathcal{X}_{l_1} = \left\{ x \in \mathcal{X}_l | x_r \leq \left\lfloor x_r^{*(l)} \right\rfloor \right\}$$

et pour l_2 on met $\mathcal{X}_{l_2} = \left\{ x \in \mathcal{X}_l | x_r \geq \left\lceil x_r^{*(l)} \right\rceil \right\}$.

(LFP_l) **a une solution entière** $x^{*(l)}$ **:** Si la solution $x^{*(l)}$ est entière alors deux étapes sont effectuées :

— ***La première étape consiste à tester l'appartenance de*** $x^{*(l)}$ ***à*** $\mathcal{X}_{BE}$ ***:***

Puisqu'il y a deux tests d'efficacité à prendre en compte :

$$(T^1_{x^{*(l)}}) \begin{cases} \max \sum_{i=1}^{i=k} w_i, \\ s.c., \\ \left(c^i - Z_i(x^{*(l)})d^i\right) x - w_i = Z_i(x^{*(l)})d^i_0 - c^i_0,\ i = \{1,\dots,k\}, \\ x \in \mathcal{D}. \end{cases}$$

$$(T^2_{x^{*(l)}}) \begin{cases} \max v_1 + v_2, \\ s.c., \\ \left(p^1 - f_1(x^{*(l)})q^1\right) x - v_1 = f_1(x^{*(l)})\beta^1 - \alpha^1, \\ \left(p^2 - f_2(x^{*(l)})q^2\right) x - v_2 = f_2(x^{*(l)})\beta^2 - \alpha^2, \\ x \in \mathcal{D}. \end{cases}$$

Le premier programme sert à tester l'efficacité dans le programme *MOILFP* et le second pour le programme *BOILFP*. Si les deux programmes ont un zéro comme valeur optimale, cela signifie que la solution $x^{*(l)}$ est efficace pour *MOILFP* et *BOILFP* et donc $x^{*(l)} \in \mathcal{X}_{BE}$.

— ***La seconde consiste à rechercher les descendants du noeud l :*** Dans ce cas, le noeud n'a soit aucun descendant, soit un descendant direct, selon le sens de croissance des fonctions $Z_i, i \in \{1,\dots,k\}$ et f_2. S'il n'y a pas d'amélioration pour toutes les fonctions Z_i dans le domaine $\mathcal{X}_l$, alors $x^{*(l)}$ domine tout le domaine $\mathcal{X}_l$ en termes de *MOILFP* ainsi le noeud est sondé. De même, s'il n'y a pas de solution qui soit au moins meilleure que $x^{*(l)}$ dans le critère f_2, alors la solution $x^{*(l)}$ domine tout le domaine $\mathcal{X}_l$ en termes de *BOILFP* et le noeud est sondé.

D'autre part, s'il existe une amélioration possible de l'une des fonctions $Z_i, i \in \{1,\dots,k\}$ et f_2, une solution efficace en termes de *MOILFP* et *BOILFP* pourrait encore exister dans $\mathcal{X}_l$ et le noeud a des descendants. Ainsi, la méthode du branch & cut que nous utilisons permet d'éliminer les solutions qui sont dominées par les vecteurs critères de $x^{*(l)}$.

Soit $\mathcal{B}_l$ et $\mathcal{N}_l$ les ensembles des indices des variables de base et des variables

hors base de $x^{*(l)}$ respectivement. Dans ce qui suit, nous utiliserons ces notations pour $j \in \mathcal{N}_l$:

$$\begin{aligned}\boldsymbol{\gamma^{2(l)}} &= Q^2(x^{*(l)})\nu^{2(l)} - P^2(x^{*(l)})\mu^{2(l)},\\ \boldsymbol{\lambda^{i(l)}} &= z^2(x^{*(l)})\eta^{i(l)} - z^1(x^{*(l)})\vartheta^{i(l)},\ i \in \{1,2,\dots,k\}.\end{aligned}$$

Puisque $\boldsymbol{\gamma^{1(l)}}$ and $\boldsymbol{\gamma^{2(l)}}$ représentent les vecteurs de gradient réduit pour f_1 et f_2, et $\boldsymbol{\lambda^{i(l)}}$ correspond aux vecteurs de gradient réduit pour le critère Z_i pour chaque $i \in \{1,2,\dots,k\}$. Nous définissons les ensembles $\mathcal{H}_l$ et $\mathcal{H}'_l$ pour chaque noeud l comme suit :

$$\mathcal{H}_l = \left\{ j \in \mathcal{N}_l | \exists i \in \{1,\dots k\}, \boldsymbol{\lambda_j^{i(l)}} > 0 \right\} \cup \left\{ j \in \mathcal{N}_l | \boldsymbol{\lambda_j^{i(l)}} = 0\ , \forall i \in \{1,\dots,k\} \right\}. \tag{5.5}$$

$$\mathcal{H}'_l = \left\{ j \in \mathcal{N}_l | \boldsymbol{\gamma_j^{2(l)}} > 0 \right\} \cup \left\{ j \in \mathcal{N}_l | \boldsymbol{\gamma_j^{2(l)}} = 0 \text{ and } \boldsymbol{\gamma_j^{1(l)}} = 0 \right\}. \tag{5.6}$$

Dans le théorème suivant 5.1, nous prouverons que les coupes :

$$\sum_{j \in \mathcal{H}_l} x_j \geq 1 \text{ et } \sum_{j \in \mathcal{H}'_l} x_j \geq 1.$$

sont valides et éliminent les solutions qui sont dominées par $x^{*(l)}$.

Après avoir trouvé une solution entière, nous mettons à jour $\mathcal{X}_{BE}$ et calculons $\mathcal{H}_l, \mathcal{H}'_l$. Si $\mathcal{H}_l = \emptyset$ ou $\mathcal{H}'_l = \emptyset$, cela signifie que le domaine restant ne contient aucune solution efficace, que ce soit pour le $MOILFP$ ou le $BOILFP$. Ainsi, le noeud l est sondé (voir proposition 5.1).

Sinon, si $\mathcal{H}_l \neq \emptyset$ et $\mathcal{H}'_l \neq \emptyset$ le noeud l a un successeur l_0 avec
$\mathcal{X}_{l_0} = \left\{ x \in \mathcal{X}_l | \sum_{j \in \mathcal{H}_l} x_j \geq 1,\ \sum_{j \in \mathcal{H}'_l} x_j \geq 1 \right\}$.

Ainsi, il existe deux règles de sondage des noeuds, la première est lorsque le programme correspondant (LFP_l) n'est pas réalisable et la seconde est lorsque $\mathcal{H}_l = \emptyset$ ou $\mathcal{H}'_l = \emptyset$.

Dans l'algorithme (11), les noeuds de l'arborescence peuvent être traités selon n'importe quelle stratégie d'exploration.

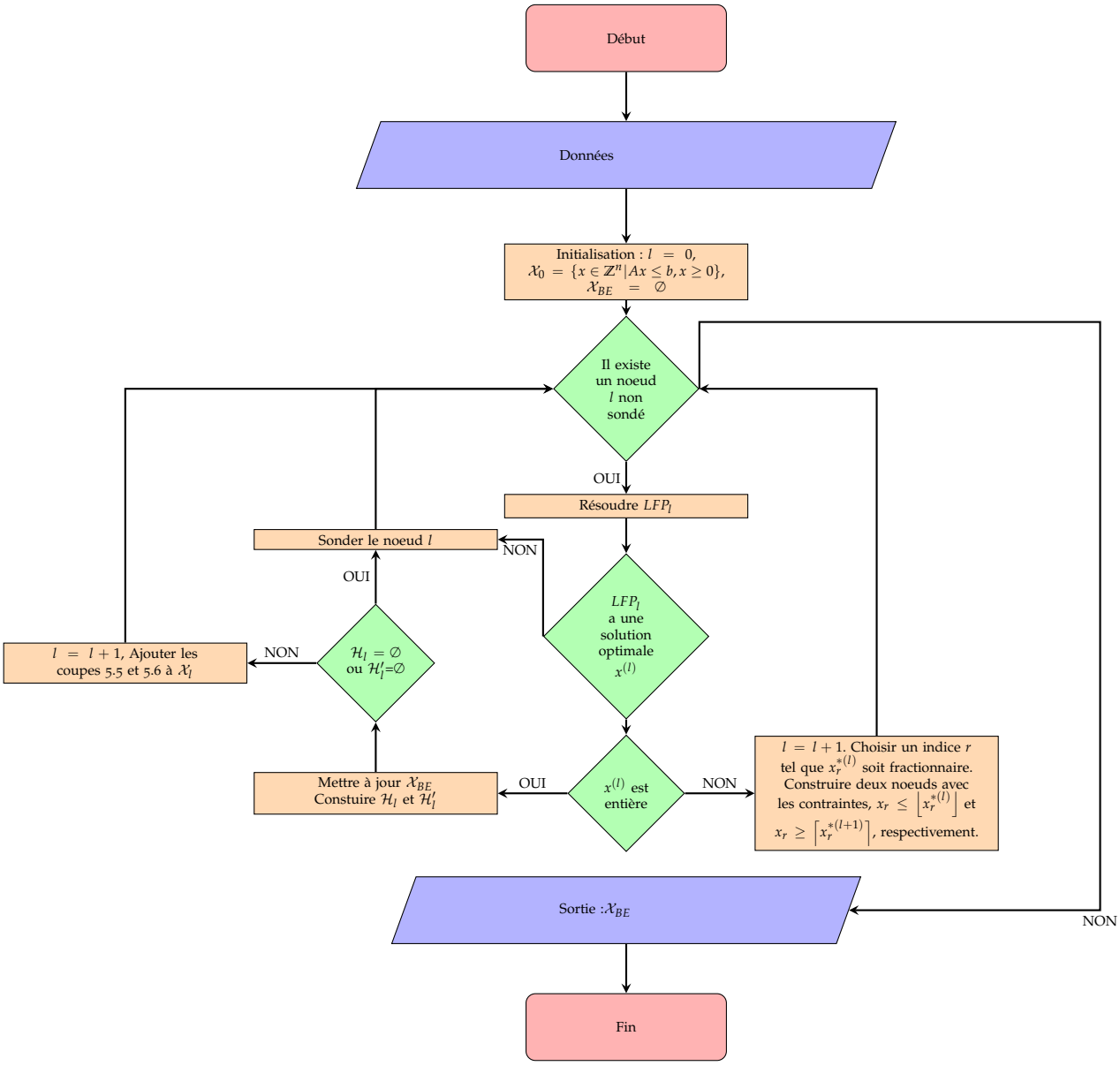

FIGURE 5.1 – Organigramme de la méthode proposée Biobjectif sur l'enssemble efficace d'un MOILFP

Algorithme 11 : Optimisation biobjectif sur un programme multiobjectif fractionnaire linéaire en nombres entiers

Résultat : L'ensemble $\mathcal{X}_{BE}$ contenant toutes les solutions efficaces du programme (5.4)

initialisation $l = 0$,$\mathcal{X}_l = \{x \in \mathbb{R}^n | Ax \leq b \text{ and } x \geq 0\}$ et $\mathcal{X}_{BE} = \emptyset$;

Tant que *(il y a un noeud non-sondé l)* **faire**

Résoudre (LFP_l) ;

$$(LFP_l) \begin{cases} \max & f_1(x), \\ s.c., & \\ & x \in \mathcal{X}_l. \end{cases}$$

Si (LFP_l) *a la solution optimale* $x^{*(l)}$ **alors**

Si $x^{*(l)}$ *est entière* **alors**

Mise à jour de $\mathcal{X}_{BE}$;

Construire deux ensembles $\mathcal{H}_l$, $\mathcal{H'}_l$;

Si $\mathcal{H}_l = \emptyset$ *ou* $\mathcal{H'}_l = \emptyset$ **alors**

Sondé le noeud l

Sinon

Ajouter les deux coupes (5.5) et (5.6) à l_0 le successeur de l ;

Finsi

Sinon

Choisir un indice r tel que $x_r^{*(l)}$ soit fractionnaire. Ensuite, divisez le programme (LFP_l) en deux sous-programmes, en ajoutant respectivement les contraintes $x_r \leq \left\lfloor x_r^{*(l)} \right\rfloor$ et $x_r \geq \left\lceil x_r^{*(l)} \right\rceil$ pour obtenir (LFP_{l_1}) et (LFP_{l_2}) ($l_1 \geq l+1$, $l_2 > l+1$ et $l_1 \neq l_2$) ;

Finsi

Sinon

Sonder le noeud l ;

Finsi

Fintq

5.4 Résultats théoriques

Les outils théoriques suivants montrent que l'algorithme donne l'ensemble de solutions $\mathcal{X}_{BE}$ du programme B_E (5.4) en un nombre fini d'itérations.

Théorème 5.1. *Supposons que $\mathcal{H}_l \neq \emptyset$ et $\mathcal{H}'_l \neq \emptyset$ à la solution entière courante $x^{*(l)}$. Si $x \in \mathcal{X}_{BE}$, $x \neq x^{*(l)}$ et $x \in \mathcal{X}_l$, alors $x \in \mathcal{X}_{l_1}$ (l_1 successeur de l).*

Démonstration. □

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

— $x \notin \mathcal{X}_{l_1}$ parce que $x \in \left\{x \in \mathcal{X}_l |\ \sum_{j \in \mathcal{N}_l / \mathcal{H}_l} x_j \geq 1\right\}$, cela implique que

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

Par conséquent, les inégalités suivantes se vérifient :

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

Il s'ensuit que $x_j = 0$ pour tout $j \in \mathcal{H}_l$ et $x_j \geq 1$ pour au moins un indice $j \in \mathcal{N}_l \setminus \mathcal{H}_l$. A partir du tableau du simplexe optimal correspondant à la solution $x^{*(l)}$, la valeur actualisée de chaque fonction objectif s'écrit en fonction des indices hors base $j \in \mathcal{N}_l$ comme suit :

$$z_i^1(x) = z_i^1(x^{*(l)}) + \delta_j \eta_j^{i(l)},$$
$$z_i^2(x) = z_i^2(x^{*(l)}) + \delta_j \vartheta_j^{i(l)},$$

où $\delta_j = \dfrac{x^{*(l)}_{\mathcal{B}_{l_{(r^*)}}}}{A_j^{r^*}} = \min \left\{ \dfrac{x^{*(l)}_{\mathcal{B}_{l_{(r)}}}}{A_j^r} |\ A_j^r > 0 \right\}$.

Ainsi, nous avons pour tous les $i \in \{1, 2, \ldots, k\}$:

$$Z_i(x) \quad = \frac{z_i^1(x)}{z_i^2(x)} = \frac{z_i^1(x^{*(l)}) + \delta_j \eta_j^{i(l)}}{z_i^2(x^{*(l)}) + \delta_j \vartheta_j^{i(l)}},$$

alors nous pouvons écrire :

$$
\begin{aligned}
Z_i(x) - Z_i(x^{*(l)}) &= \frac{z_i^1(x^{*(l)}) + \delta_j \eta_j^{i(l)}}{z_i^2(x^{*(l)}) + \delta_j \vartheta_j^{i(l)}} - \frac{z_i^1(x^{*(l)})}{z_i^2(x^{*(l)})} \\
&= \delta_j \frac{\left[z_i^2(x^{*(l)})(\eta_j^{i(l)}) - z_i^1(x^{*(l)})(\vartheta_j^{i(l)})\right]}{z_i^2(x^{*(l)})\left[z_i^2(x^{*(l)}) + \delta_j(\vartheta_j^{i(l)})\right]} \\
&= \delta_j \frac{\left[z_i^2(x^{*(l)})(\eta_j^{i(l)}) - z_i^1(x^{*(l)})(\vartheta_j^{i(l)})\right]}{z_i^2(x^{*(l)}) z_i^2(x)}.
\end{aligned}
$$

Nous avons :

$$\lambda^{i(l)} = z^2(x^{*(l)})\eta^{i(l)} - z^1(x^{*(l)})\vartheta^{i(l)},\ i \in \{1, \ldots, k\}$$

Comme les composants $\lambda_j^{i(l)} \leq 0$, pour tout indice $j \in \mathcal{N}_l \backslash \mathcal{H}_l$:

$$Z_i(x) - Z_i(x^{*(l)}) \leq 0, \quad i \in \{1, \ldots, k\}.$$

Pour au moins un critère $i_0 \in \{1, \ldots, r\}, Z_{i_0}(x) - Z_{i_0}(x^{*(l)}) < 0$.

Par conséquent, le vecteur critère $(Z_1(x), Z_2(x), \ldots, Z_k(x))$ est dominé par le vecteur critère $\left(Z_1(x^{*(l)}), Z_2(x), \ldots, Z_k(x^{*(l)})\right)$ alors x n'est pas dans $\mathcal{X}_E$. Donc x n'est pas dans $\mathcal{X}_{BE}$.

— $x \notin \mathcal{X}_{l_1}$ parce que $x \in \left\{x \in \mathcal{X}_l |\ \sum_{j \in \mathcal{N}_l / \mathcal{H}'_l} x_j \geq 1\right\}$, de la même manière, cela implique que :

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

Par conséquent, les inégalités suivantes se vérifient :

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

Il s'ensuit que $x_j = 0$ pour tout $j \in \mathcal{H}'_l$ et $x_j \geq 1$ pour au moins un indice $j \in \mathcal{N}_l \setminus \mathcal{H}'_l$. À partir du tableau du simplexe optimal correspondant à la solution $x^{*(l)}$, la valeur actualisée de chaque fonction objectif s'écrit selon les indices hors base $j \in \mathcal{N}_l$ comme suit :

$$P^2(x) = P^2(x^{*(l)}) + \delta_j \nu_j^{2(l)},$$
$$Q^2(x) = Q^2(x^{*(l)}) + \delta_j \mu_j^{2(l)},$$

où $\delta_j = \dfrac{x^{*(l)}_{\mathcal{B}_{l_{(r^*)}}}}{A^{r^*}_j} = \min\left\{ \dfrac{x^{*(l)}_{\mathcal{B}_{l_{(r)}}}}{A^r_j} |\ A^r_j > 0 \right\}$.

Ainsi, nous avons :

$$f_2(x) \ = \frac{P^2_i(x)}{Q^2(x)} = \frac{P^2(x^{*(l)}) + \delta_j \nu^{2(l)}_j}{Q^2_i(x^{*(l)}) + \delta_j \mu^{2(l)}_j},$$

alors, on peut écrire :

$$\begin{aligned} f_2(x) - f_2(x^{*(l)}) &= \frac{P^2(x^{*(l)}) + \delta_j \nu^{2(l)}_j}{Q^2(x^{*(l)}) + \delta_j \mu^{2(l)}_j} - \frac{P^2_i(x^{*(l)})}{Q^2(x^{*(l)})} \\ &= \delta_j \frac{\left[Q^2(x^{*(l)})(\nu^{2(l)}_j) - P^2(x^{*(l)})(\mu^{2(l)}_j)\right]}{Q^2(x^{*(l)})\left[Q^2_i(x^{*(l)}) + \delta_j(\mu^{2(l)}_j)\right]} \\ &= \delta_j \frac{\left[Q^2_i(x^{*(l)})(\nu^{2(l)}_j) - P^2(x^{*(l)})(\mu^{2(l)}_j)\right]}{Q^2(x^{*(l)})Q^2(x)}. \end{aligned}$$

Puisque nous avons déjà la notation suivante :

$$\boldsymbol{\gamma^{2(l)}} = Q^2(x^{*(l)})\nu^{2(l)} - P^2(x^{*(l)})\mu^{2(l)}.$$

Comme les composantes $\boldsymbol{\gamma^{2(l)}}_j \leq 0$, pour chaque indice $j \in \mathcal{N}_l \backslash \mathcal{H}'_l$.

$$f_2(x) - f_2(x^{*(l)}) \leq 0.$$

Donc, $f_2(x) \leq f_2(x^{*(l)})$, avec $f_1(x) \leq f_1(x^{*(l)})$ puisque $x^{*(l)}$ est le maximum du programme (LFP_l), alors $x \notin \mathcal{X}_{E'}$. Par conséquent, $x \notin \mathcal{X}_{BE}$.

Puisque, $x \notin \mathcal{S}_{BE}$ dans les deux cas, alors x n'est pas une solution du programme B_E.

Proposition 5.1. Supposons que $\mathcal{H}_l = \emptyset$ ou $\mathcal{H}'_l = \emptyset$ à la solution entière courante $x^{*(l)}$, alors il n'existe aucune solution dans le domaine restant qui ne soit pas dominée par $x^{*(l)}$.

Démonstration.

— Supposons que $\mathcal{H}_l = \emptyset$, alors $\forall i \in \{1, \ldots, k\}$, $\forall j \in \mathcal{N}_l$, nous avons $\overline{\gamma}^i_j \leq 0$ et $\exists i_0 \in \{1, \ldots, r\}$ tel que $\overline{\gamma}^{i_0}_j < 0, \forall j \in \mathcal{N}_l$. Donc, $x^{*(l)}$ domine tous les points x, $x \neq x^{*(l)}$ du domaine $\mathcal{D}_l$.

— Maintenant, supposons que $\mathcal{H}'_l = \emptyset$, alors $\forall j \in \mathcal{N}_l, \overline{\lambda}^2_j < 0$ ou $\overline{\lambda}^2_j = 0$ et $\overline{\lambda}^1_j < 0$, en ajoutant à cela $\overline{\lambda}^1_j < 0$, $\forall j \in \mathcal{N}_l$ puisque c'est une solution optimale pour (LP_l), $x^{*(l)}$ devient la solution la plus préférée dans le domaine $\mathcal{D}_l$.

□

Théorème 5.2. *L'algorithme se termine en un nombre fini d'itérations et l'ensemble $\mathcal{X}_{BE}$ contient toutes les solutions de B_E, si de telles solutions existent.*

Démonstration. Soit $\mathcal{D}$, l'ensemble des solutions entières réalisables du problème $MOILFP$, un ensemble fini borné contenu dans $\mathcal{X}$. Les ensembles efficaces $\mathcal{X}_{BE}$, $\mathcal{X}_E$ et $\mathcal{X}_{E'}$ ont une cardinalité finie. L'arbre de recherche aura donc un nombre fini de branches. L'algorithme se termine donc en un nombre fini d'étapes. Pour que $\mathcal{X}_{BE}$ contienne toutes les solutions de B_E, on utilise les règles de sondage sans perte d'éléments dans $\mathcal{X}_{BE}$. A chaque étape l de l'algorithme 11, si une solution entière $x^{*(l)}$ est trouvée, les coupes éliminent $x^{*(l)}$ et toutes les solutions dominées de la recherche (voir proposition 5.1).

Ainsi, la première règle de sondage est vérifiée lorsque l'ensemble $\mathcal{H}_l$ ou $\mathcal{H}'_l$ est vide. Dans ce cas, le noeud actuel peut être sonder puisque le reste du domaine ne contient que des solutions dominées, soit en termes de programme multiobjectif, soit en termes de $(BIOLFP)$. La deuxième règle est le cas trivial où le domaine réduit devient infaisable, que ce soit à cause des coupes précédentes ou du branchement. □

5.5 Exemple didactique

Considérons l'optimisation de deux fonctions d'utilité :

$$f_1(x) = \frac{-x_1 + x_2 - 3}{2x_1 + x_1 + 1} \text{ et } f_2(x) = \frac{-4x_1 + 3x_2 + 1}{2x_1 + x_2 + 2},$$

sur l'ensemble efficace $\mathcal{X}_E$ du problème $MOILFP$ présenté dans (Kornbluth et Steuer, 1981) :

$$\begin{cases} \max \dfrac{x_1 - 4}{-x_2 + 2}, \\ \max \dfrac{-x_1 + 4}{x_2 + 1}, \\ \max -x_1 + x_2, \\ s.c., \\ -x_1 + 4x_2 \leq 0, \\ 2x_1 - x_2 \leq 8, \\ x_1 \geq 0,\ x_2 \in \mathbb{N}. \end{cases}$$

Pour résoudre ce problème à l'aide de l'algorithme décrit ci-dessus :

Initialisation Tout d'abord, nous initialisons $l = 0$, $\mathcal{X}_{BE} = \emptyset$ et $\mathcal{X}_0 = \mathcal{X}$.

les itérations suivantes consistent à résoudre le problème attaché au noeud, ainsi pour le noeud 0, nous allons résoudre (LFP_0) :

$$(LFP_0)\begin{cases} \max f_1(x) = \dfrac{-x_1 + x_2 - 3}{2x_1 + x_1 + 1}, \\ s.c., \\ x \in \mathcal{X}_0. \end{cases}$$

Comme la méthode de résolution de (LFP_0) est une procédure basé sur le simplexe, elle donne un tableau du simplexe final, à partir duquel nous analyserons les résultats selon l'algorithme décrit.

Noeud 0 la résolution de (LFP_0) donne le tableau simplexe final 5.1 :

Table 5.1 – Tableau simplexe optimale pour noeud 0.

$\mathcal{B}_0$	x_3	x_4	RHS
x_1	1/7	4/7	32/7
x_2	2/7	1/7	8/7
$\nu^{1(0)}$	-1/7	3/7	-45/7
$\mu^{1(0)}$	-4/7	-9/7	79/7
$\boldsymbol{\gamma^{1(0)}}$	-37/7	-24/7	-45/79
$\nu^{2(0)}$	-2/7	13/7	-97/7
$\mu^{2(0)}$	-4/7	-9/7	86/7
$\boldsymbol{\gamma^{2(0)}}$	-80/7	5	-97/86
$\eta^{1(0)}$	-1/7	-4/7	4/7
$\vartheta^{1(0)}$	2/7	1/7	6/7
$\boldsymbol{\lambda^{1(0)}}$	-2/7	-4/7	2/3
$\eta^{2(0)}$	1/7	4/7	-4/7
$\vartheta^{2(0)}$	-2/7	-1/7	15/7
$\boldsymbol{\lambda^{2(0)}}$	1/7	8/7	-4/15
$\eta^{3(0)}$	-1/7	3/7	-24/7
$\vartheta^{3(0)}$	0	0	1
$\boldsymbol{\lambda^{3(0)}}$	-1/7	3/7	-24/7

Puisque la solution optimale $(\frac{32}{7}, \frac{8}{7})$ n'est pas entière, le problème est divisé en deux sous-problèmes en ajoutant les contraintes $x_1 \leq \lfloor \frac{32}{7} \rfloor$ et $x_1 \geq \lceil \frac{32}{7} \rceil$ respectivement à $\mathcal{X}_0$, et on obtient les deux programmes :

$$(LFP_1)\begin{cases} \max f_1(x) = \dfrac{-x_1 + x_2 - 3}{2x_1 + x_1 + 1}, \\ s.c., \\ x \in \mathcal{X}_0, \\ x_1 \leq 4. \end{cases} \qquad (LFP_2)\begin{cases} \max f_1(x) = \dfrac{-x_1 + x_2 - 3}{2x_1 + x_1 + 1}, \\ s.c., \\ x \in \mathcal{X}_0, \\ x_1 \geq 5. \end{cases}$$

Noeud 1 La résolution du problème (LFP_1) donne le tableau 5.2 :

TABLE 5.2 – Tableau simplexe optimale pour noeud 1.

$\mathcal{B}_1$	x_3	x_5	RHS
x_1	0	1	4
x_2	1/4	1/4	1
x_4	1/4	-7/4	1
$\nu^{1(1)}$	-1/4	3/4	-6
$\mu^{1(1)}$	-1/4	-9/4	10
$\boldsymbol{\gamma^{1(1)}}$	-4	-6	-3/5
$\nu^{2(1)}$	-3/4	13/4	-12
$\mu^{2(1)}$	-1/4	-9/4	11
$\boldsymbol{\gamma^{2(1)}}$	-45/4	35/4	-12/11
$\eta^{1(1)}$	0	-1	0
$\vartheta^{1(1)}$	1/4	1/4	1
$\boldsymbol{\lambda^{1(1)}}$	0	-1	0
$\eta^{2(1)}$	0	1	0
$\vartheta^{2(1)}$	-1/4	-1/4	2
$\boldsymbol{\lambda^{2(1)}}$	0	2	0
$\eta^{3(1)}$	-1/4	3/4	-3
$\vartheta^{3(1)}$	0	0	1
$\boldsymbol{\lambda^{3(1)}}$	-1/4	3/4	-3

La solution optimale trouvée $(4, 1)$ est entière, selon l'algorithme nous commençons par mettre à jour $\mathcal{S}$, puisque cet ensemble est vide $\mathcal{S} = \{(4, 1)\}$ puis nous calculons $\mathcal{H}_1$ et $\mathcal{H}'_1$: D'après le tableau 6.2 nous trouvons $\mathcal{H}_1 = \{5\}$ et $\mathcal{H}'_1 = \{5\}$. Donc le noeud 0 a un successeur noeud 3 avec $\mathcal{X}_3 = \mathcal{X}_1 \cap \{x_5 \geq 1\}$.

Noeud 2 Le programme (LFP_2) est infaisable.

Noeud 3 Après avoir résolu le programme linéaire fractionnaire correspondant, cela donne le tableau 5.3 :

TABLE 5.3 – Tableau de simplexe optimal du Noeud 3.

$\mathcal{B}_3$	x_3	x_6	RHS
x_1	0	1	3
x_2	1/4	1/4	3/4
x_4	1/4	-7/4	11/4
x_5	0	-1	1
$v^{1(3)}$	-1/4	3/4	-21/4
$\mu^{1(3)}$	-1/4	-9/4	31/4
$\boldsymbol{\gamma^{1(3)}}$	-13/4	-6	-21/31
$v^{2(3)}$	-3/4	13/4	-35/4
$\mu^{2(3)}$	-1/4	-9/4	35/4
$\boldsymbol{\gamma^{2(3)}}$	-35/4	35/4	-1
$\eta^{1(3)}$	0	-1	-1
$\vartheta^{1(3)}$	1/4	1/4	5/4
$\boldsymbol{\lambda^{1(3)}}$	1/4	-1	-4/5
$\eta^{2(3)}$	0	1	1
$\vartheta^{2(3)}$	-1/4	-1/4	7/4
$\boldsymbol{\lambda^{2(3)}}$	1/4	2	4/7
$\eta^{3(3)}$	-1/4	3/4	-9/4
$\vartheta^{3(3)}$	0	0	1
$\boldsymbol{\lambda^{3(3)}}$	-1/4	3/4	-9/4

La solution optimale $(3, \frac{3}{4})$ n'étant pas entière, les contraintes $x_2 \leq \lfloor \frac{3}{4} \rfloor$ et $x_1 \geq \lceil \frac{3}{4} \rceil$ sont ajoutées respectivement au Tableau 5.3 et les noeuds 4 et 5 sont créés.

Noeud 4 La résolution du problème donne le tableau 5.4 :

TABLE 5.4 – Tableau de simplexe optimal du Noeud 4.

$\mathcal{B}_4$	x_6	x_7	RHS
x_1	0	1	3
x_2	1	0	0
x_3	-4	-1	3
x_4	1	-2	2
x_5	0	-1	1
$\nu^{1(3)}$	-1	1	-6
$\mu^{1(3)}$	-1	-2	7
$\boldsymbol{\gamma^{1(3)}}$	-13	-5	-6/7
$\nu^{2(3)}$	-3	4	-11
$\mu^{2(3)}$	-1	-2	8
$\boldsymbol{\gamma^{2(3)}}$	-35	10	-11/8
$\eta^{1(3)}$	0	-1	-1
$\vartheta^{1(3)}$	1	0	2
$\boldsymbol{\lambda^{1(3)}}$	1	-2	-1/2
$\eta^{2(3)}$	0	1	1
$\vartheta^{2(3)}$	-1	0	1
$\boldsymbol{\lambda^{2(3)}}$	1	1	1
$\eta^{3(3)}$	-1	1	-3
$\vartheta^{3(3)}$	0	0	1
$\boldsymbol{\lambda^{3(3)}}$	-1	1	-3

La solution optimale trouvée $(3,0)$ est entière, cette solution est dominée par la solution $(4,1)$ alors $\mathcal{X}_{BE} = \{(4,1)\}$. D'après la table 5.4, nous trouvons $\mathcal{H}_4 = \{6,7\}$ et $\mathcal{H}'_4 = \{6\}$, donc le Noeud 4 a un successeur Noeud 6 avec $\mathcal{X}_6 = \mathcal{X}_4 \cap \{x_6 + x_7 \geq 1, x_6 \geq 1\}$.

Noeud 5 Le problème correspondant est infaisable et le noeud est sondé.

Noeud 6 En ajoutant les contraintes $x_6 \geq 1$ et $x_6 + x_7 \geq 1$ au tableau 5.4 et en résolvant, on obtient le tableau 5.5.

TABLE 5.5 – Tableau de simplexe optimal du Noeud 6.

$\mathcal{B}_6$	x_7	x_9	RHS
x_1	0	1	2
x_2	1	0	0
x_3	-4	1	2
x_4	1	-2	4
x_5	0	-1	2
x_6	-0	-1	1
x_8	-1	-1	0
$\nu^{1(6)}$	-1	1	-5
$\mu^{1(6)}$	-1	-2	5
$\boldsymbol{\gamma^{1(6)}}$	-10	-5	-1
$\nu^{2(6)}$	-3	4	-7
$\mu^{2(6)}$	-1	-2	6
$\boldsymbol{\gamma^{2(6)}}$	-25	10	-7/6
$\eta^{1(6)}$	0	-1	-2
$\vartheta^{1(6)}$	1	0	2
$\boldsymbol{\lambda^{1(6)}}$	2	-2	-1
$\eta^{2(6)}$	0	1	2
$\vartheta^{2(6)}$	-1	0	1
$\boldsymbol{\lambda^{2(6)}}$	2	1	2
$\eta^{3(6)}$	-1	1	-2
$\vartheta^{3(6)}$	0	0	1
$\boldsymbol{\lambda^{3(6)}}$	-1	1	-2

La solution optimale trouvée $(2,0)$ est entière, cette solution est dominée par la solution $(4,1)$ alors $\mathcal{X}_{BE} = \{(4,1)\}$. D'après la table 5.5, nous trouvons $\mathcal{H}_6 = \{7,9\}$ et $\mathcal{H}'_6 = \{9\}$, donc le noeud 6 a un successeur 7 avec $\mathcal{X}_7 = \mathcal{X}_6 \cap \{x_7 + x_9 \geq 1, x_9 \geq 1\}$.

Noeud 7 La résolution du problème correspondant produit le tableau 5.6 :

TABLE 5.6 – Tableau de simplexe optimal du Noeud 7.

$\mathcal{B}_7$	x_7	x_{10}	RHS
x_1	0	1	1
x_2	1	0	0
x_3	-4	1	1
x_4	1	-2	6
x_5	0	-1	3
x_6	0	-18	2
x_8	0	-1	1
x_9	-1	-1	1
x_{11}	-1	-1	0
$\nu^{1(7)}$	-1	1	-4
$\mu^{1(7)}$	-1	-2	3
$\boldsymbol{\gamma^{1(7)}}$	-7	-5	-4/3
$\nu^{2(7)}$	-3	4	-3
$\mu^{2(7)}$	1	-2	4
$\boldsymbol{\gamma^{2(7)}}$	-15	10	-3/4
$\eta^{1(7)}$	0	-1	-3
$\vartheta^{1(7)}$	1	0	2
$\boldsymbol{\lambda^{1(7)}}$	3	-2	-3/2
$\eta^{2(7)}$	0	1	3
$\vartheta^{2(7)}$	-1	0	1
$\boldsymbol{\lambda^{2(7)}}$	3	1	3
$\eta^{3(7)}$	-1	1	-1
$\vartheta^{3(7)}$	0	0	1
$\boldsymbol{\lambda^{3(7)}}$	-1	1	-1

La solution optimale trouvée $(1,0)$ est entière, cette solution n'est pas dominée par la solution $(4,1)$ donc $\mathcal{X}_{BE} = \{(4,1),(1,0)\}$. D'après la table 5.6, nous trouvons $\mathcal{H}_7 = \{7,10\}$ et $\mathcal{H}'_7 = \{10\}$, donc le noeud 7 a un successeur, le noeud 8 avec

$$\mathcal{X}_8 = \mathcal{X}_7 \cap \{x_7 + x_{10} \geq 1, x_{10} \geq 1\}$$

.

Noeud 8 La résolution du problème correspondant produit le tableau 5.7 :

TABLE 5.7 – Tableau de simplexe optimal du Noeud 8.

$\mathcal{B}_8$	x_7	x_{12}	RHS
x_1	0	1	0
x_2	1	0	0
x_3	-4	1	0
x_4	1	-2	8
x_5	0	-1	4
x_6	0	-1	3
x_8	0	-1	2
x_9	-1	-1	2
x_{10}	0	-1	1
x_{11}	-1	-1	1
x_{12}	-1	-1	0
$\nu^{1(8)}$	-1	1	-3
$\mu^{1(8)}$	-1	-2	1
$\boldsymbol{\gamma^{1(8)}}$	-4	-5	-3
$\nu^{2(8)}$	-3	4	1
$\mu^{2(8)}$	-1	-2	2
$\boldsymbol{\gamma^{2(8)}}$	-5	10	1/2
$\eta^{1(8)}$	0	-1	-4
$\vartheta^{1(8)}$	1	0	2
$\boldsymbol{\lambda^{1(8)}}$	-4	-2	-2
$\eta^{2(8)}$	0	1	4
$\vartheta^{2(8)}$	-1	0	1
$\boldsymbol{\lambda^{2(8)}}$	4	1	4
$\nu^{3(8)}$	-1	1	0
$\mu^{3(8)}$	0	0	1
$\boldsymbol{\lambda^{3(8)}}$	-1	1	0

La solution optimale trouvée $(0,0)$ est entière, cette solution n'est dominée ni par la solution $(4,1)$ ni par la solution $(0,0)$, elle ne domine pas les deux donc $\mathcal{X}_{BE} = \{(4,1),(1,0),(0,0)\}$. Dans la table 5.7, nous trouvons $\mathcal{H}_8 = \{7,13\}$ et $\mathcal{H}'_8 = \{13\}$, donc le noeud 8 a un successeur, le noeud 9 avec $\mathcal{X}_9 = \mathcal{X}_8 \cap \{x_7 + x_{10} \geq 1, x_{13} \geq 1\}$.

Noeud 9 Le problème correspondant est infaisable et le noeud est sondé.

Comme il ne reste aucun noeud, la recherche se termine par l'ensemble de solutions $\mathcal{X}_{BE} = \{(4,1),(1,0),(0,0)\}$. L'ensemble efficace de ce problème est $\mathcal{X}_E = \{(4,1),(3,0),(2,0),(1,0),(0,0)\}$

L'arbre de recherche est présenté dans la figure 5.2.

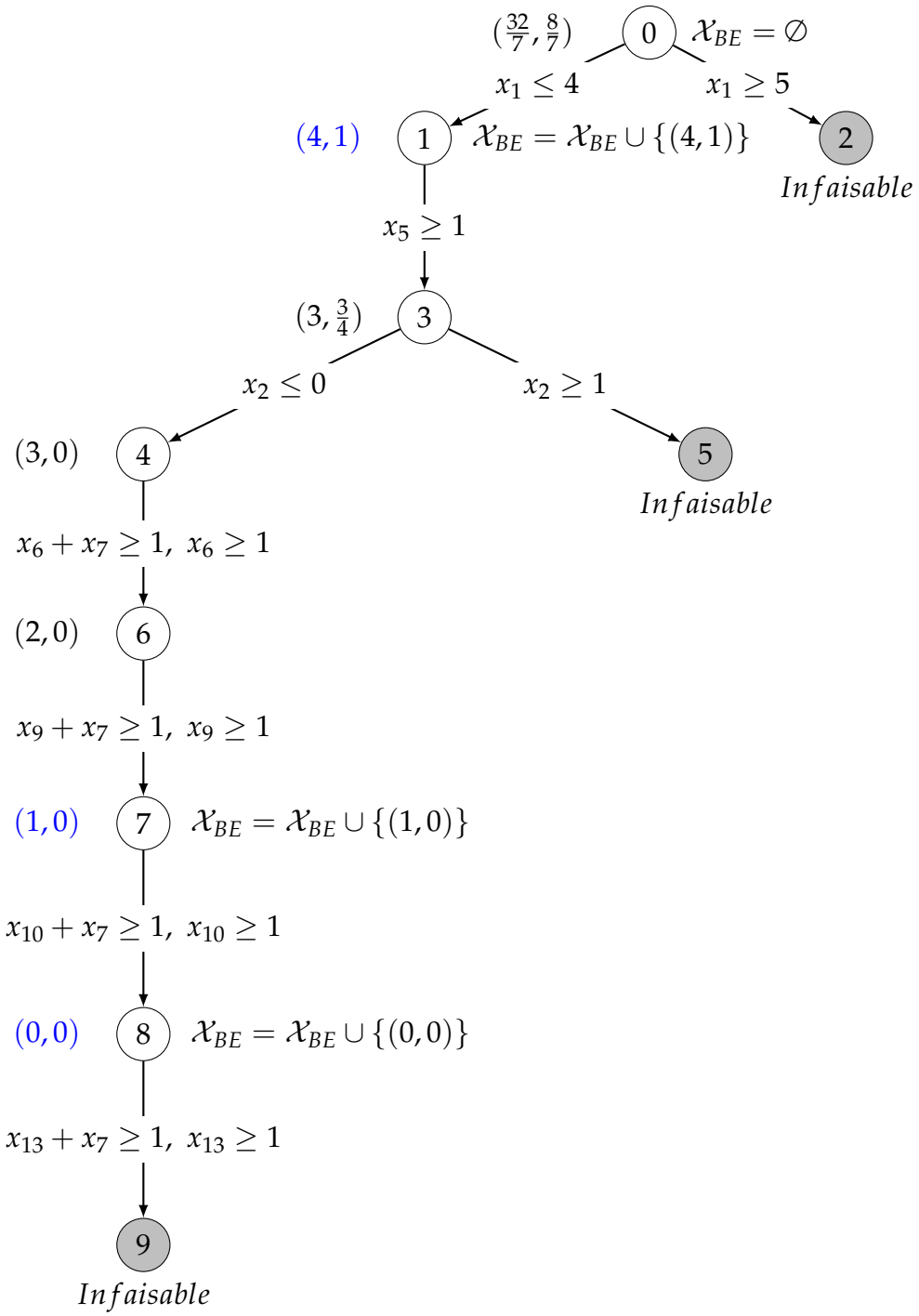

FIGURE 5.2 – Arbre de recherche de l'exemple

5.6 Étude expérimentale

L'algorithme proposé est implémenté dans Matlab r2017a. Les programmes linéaires et linéaires entiers sont résolus à l'aide de la bibliothèque IBM CPLEX 12.9 pour Matlab. Pour

effectuer les tests, nous avons utilisé un ordinateur avec un processeur Intel i7 5500u et $8GB$ de mémoire.

La méthode est testée sur des instances $MOILFP$ générés aléatoirement avec deux fonctions d'utilitées fractionnaires linéaires générées aléatoirement. Les fonctions objectifs et les coefficients des contraintes sont non corrélés et uniformément distribués. Chaque composante du vecteur b et les entrées des matrices A ont été tirées aléatoirement de distributions uniformes discrètes dans les intervalles [50, 100] et [1, 30] respectivement. Les vecteurs $p^1,\ p^2$ et $c^i,\ i = \overline{1,k}$ et les scalaires α^1, α^2 et c_0^i sont tirés aléatoirement de distributions uniformes discrètes dans l'intervalle [-10, 10], tandis que les vecteurs $q^1,\ q^2$ et $d^i,\ i = \overline{1,k}$ et les scalaires β^1, β^2 et d_0^i sont générés aléatoirement à partir de distributions uniformes discrètes dans l'intervalle [0, 10] pour éviter le cas indéfini du dénominateur nul. De plus, pour éviter l'infaisabilité, toutes les contraintes de chaque problème sont du type $\leq$ et 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, de contraintes et de fonctions objectifs. Dans chaque catégorie, le nombre de fonctions objectifs r=3,5,7. Pour chaque catégorie de problèmes, 10 instances ont été résolues.

Les résultats de calcul obtenus sont résumés dans le Tableau 5.8. Les statistiques du temps CPU (en secondes) et du nombre de noeuds sont indiquées. La dernière colonne $\tilde{\mathcal{X}}_E$ fait référence à la moyenne de $|\mathcal{X}_E|$, tandis que n et m font référence au nombre de variables et de contraintes respectivement. Les cas où le temps d'exécution nécessaire pour trouver l'ensemble efficace $\mathcal{X}_E$ est trop élevé sont notés " $-$ ".

TABLE 5.8 – Temps d'exécution des instances aléatoire BIOLFP/MOILFP

		Temps(s)			Nombre de noeuds			
$r\times$	$m \times n$	Moyenne	Max.	Min.	Moyenne	Max.	Min.	$\tilde{\mathcal{X}}_E$
3	10× 5	3.20	7.70	0.18	250	600	10	28
	10× 10	1.64	9.20	0.1	117.8	660	3	22.3
	25× 20	13.51	28.66	0.45	1257.1	2656	40	26.5
	30×25	12.29	25.35	3.83	1167.7	2242	336	26
	35×30	20.25	47.89	0.44	2182.7	5892	43	33.7
	40×35	14.29	46.48	0.67	1862.8	5804	71	37.6
	45×40	22.74	54.15	1.33	3193.2	7325	81	29.2
	50× 25	46.25	109.31	10.57	6061.5	13081	1405	-
	60× 30	61.87	222.27	11.70	8012.6	28371	1675	-
	70× 35	162.81	373.31	3.35	20018.5	42385	376	-
	80× 40	163.27	434.77	0.83	21606.8	52783	35	-
	90× 45	164.48	343.94	4.40	21545.1	42653	592	-
	100× 50	208.17	462.57	45.30	29060.2	60184	5880	-
	120× 60	451.76	1127.12	14.98	54032.4	128250	2106	-
	140× 70	608.84	1015.43	159.95	68329.22	115096	19820	-
	160× 80	517.30	1015.43	6.92	56799.22	115096	596	-
5	5× 5	0.57	1.81	0.06	55.4	191	3	39.3
	10× 5	2.15	6.23	0.11	262.44	725	8	149
	10× 10	1.72	3.96	0.31	218.9	496	38	42.5
	20× 20	4.43	12.08	0.48	571.7	1463	65	109.4
	30× 30	11.01	38.81	2.23	1477	5250	281	138
	40× 40	18.84	46.37	0.22	2138	4987	16	125
	50× 50	41.88	103.93	1.79	4677.5	10157	236	162.5
7	5× 5	0.26	0.56	0.06	19.5	32	4	36.7
	10× 5	2.45	6.98	0.06	309.1	824	3	221.7
	10× 10	1.40	2.74	0.15	199	381	18	110.5
	20× 20	6.00	18.57	0.52	958	3104	60	190.9
	30× 30	6.71	12.86	1.34	953.4	1733	200	174.3
	40× 40	30.34	54.32	17.82	4387.5	8030	2635	191.4
	50× 50	33.4	71.02	2.34	4591	9401	293	243.6
	50× 25	77.31	163.62	14.05	10553.5	20943	2146	-
	60× 30	55.75	164.07	2.008	8314.1	24890	351	-
	70× 35	93.86	284.15	0.404	12448.8	32799	40	-
	80× 40	176.06	350.88	40.98	22950.1	48044	6468	-
	90× 45	243.70	773.29	4.33	28888.8	88885	585	-
	100× 50	269.34	541.84	51.32	26847.9	52410	5782	-
	110× 55	517.56	847.28	163.40	51484.9	83932	15147	-
	120× 60	352.08	899.07	9.58	33535.2	79358	894	-
	130× 65	546.83	1522.45	19.69	50156	133845	1873	-
	140× 70	799.93	2243.55	45.90	65041.6	175900	5006	-

D'après les expériences de calcul présentées dans le tableau 5.8, nous observons que la méthode proposée résout les problèmes de petite et moyenne taille en un temps raisonnable.

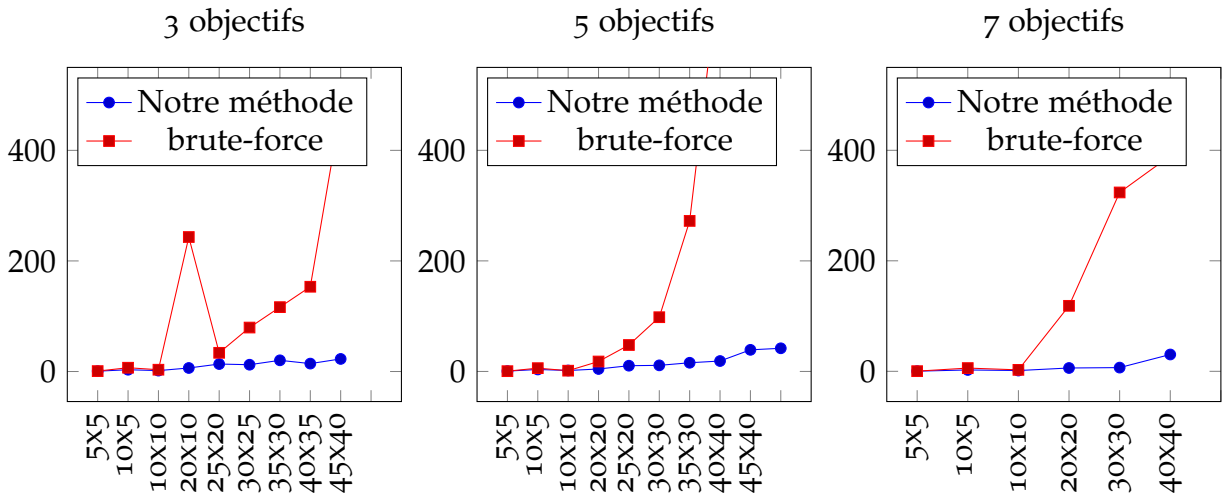

FIGURE 5.3 – Comparaison des temps d'exécution entre l'algorithme de brute-force et la méthode proposée pour des instances BIOLFP/MOILFP

La (Figure 5.3) résume le temps d'exécution moyen en utilisant notre méthode et la méthode force brute, qui consiste à trouver les deux ensembles efficaces $\mathcal{X}_E$ et $\mathcal{X}'_E$ en utilisant l'algorithme présenté dans (CHERGUI et MOULAÏ, 2008), puis à trouver l'intersection entre les deux. Remarquez que dans le cas où $\mathcal{X}_{BE}$ est vide, le problème a été remplacé par un autre. Nous observons que notre méthode est plus rapide que brute-force avec 3, 5 et 7 objectifs, surtout lorsque le nombre de variables et de contraintes augmente.

5.7 Conclusion

Dans ce chapitre, nous avons présenté une nouvelle méthode pour optimiser un *BIOLFP* sur l'ensemble efficace d'un problème *MOILFP* à l'aide d'un algorithme de branchement et de coupe (branch-and-cut). Ce dernier réduit le domaine de recherche et permet ainsi d'éviter d'explorer tous les ensembles efficaces $\mathcal{X}_E$ et $\mathcal{X}'_E$. De plus, le même algorithme peut être facilement étendu dans le cas où il y a plusieurs fonctions d'utilité à considérer, ce qui est un cas très courant dans des les problèmes réels. Les résultats expérimentaux ont montré que notre méthode donne un bon temps d'exécution pour les problèmes de petite et moyenne taille. En outre, par rapport à la méthode brute-force, notre proposition s'est avérée meilleure, surtout lorsque la taille du problème augmente.

Pour les travaux futurs, nous avons l'intention de remplacer les fonctions fractionnaires linéaires par d'autres fonctions non linéaires.

La première version de ce travail est disponible en pré-publication sur arXiv (CHAIBLAINE, MOULAÏ et CHERFAOUI, 2020)

Chapitre 6

Optimisation de la somme des fractions

"If I have seen further than others, it is by standing upon the shoulders of giants."
"Si j'ai vu plus loin que les autres, c'est en me tenant sur les épaules de géants.",
Sir Isaac Newton.

Sommaire

6.1 Introduction 69
6.2 Préliminaires 70
6.3 Algorithmes 74
6.4 Un exemple didactique 77
6.5 Étude expérimentale 84
6.6 Conclusion 90

6.1 Introduction

L'optimisation d'une somme de fractions linéaires *SOLRP*, qui consiste à optimiser la somme de K fonctions fractionnaires linéaires soumises à des contraintes convexes, se pose dans de nombreux domaines d'application, ex., en géométrie (Kuno et Masaki, 2013), optimisation de la qualité des formes (Munson, 2007), problèmes de regroupement (Rao, 1971), pour n'en citer que quelques-unes, d'autres applications peuvent être trouvées dans (Stancu-Minasian, 2017). Un cas particulier de *SOLRP* est lorsqu'il n'y a qu'une seule fraction (K=1), Charnes et Cooper, 1962 ont proposé une méthode qui transforme le problème en un programme linéaire. Cependant, pour $K \geq 2$ même pour la somme d'une fonction linéaire et d'une fonction fractionnaire le problème devient $\mathcal{NP}$-difficile (Freund et Jarre, 2001 ; Schaible, 1977 ; Matsui, 1996). De plus, le problème possède de multiples optima locaux, mais trouver la solution globalement optimale est assez difficile. L'utilité et la difficulté du problème *SOLRP* ont attiré de nombreux chercheurs : (Kuno, 2002 ; Benson, 2002 ; Freund et Jarre, 2001), même pour la somme de fractions non linéaires : (Benson, 2002 ; Gao, Mishra et Shi, 2012). D'autres méthodes peuvent être trouvées dans l'étude de

Schaible et Shi, 2003 et la biblographie de Stancu-Minasian, 2017 sur la programmation fractionnaire.

Les méthodes mentionnées ci-dessus résolvent les *SOLRP* avec des variables continues. Cependant, dans de nombreux problèmes du monde réel, la représentation d'une quantité ou d'une relation logique est souvent réalisée à l'aide de variables entières. En outre, la programmation en nombres entiers est utile pour résoudre un large éventail de problèmes, comme le démontrent Wolsey, 1998, Conforti, Cornuéjols, Zambelli et al., 2014 et Jünger et al., 2009. Malgré l'importance de la programmation en nombres entiers, le problème de la variante de *SOLRP* avec des variables entières n'a pas reçu beaucoup d'attention dans la littérature. En effet, à notre connaissance, il n'existe aucune méthode permettant de résoudre le problème de l'optimisation d'une somme de fractions linéaires avec des variables entières.

Dans ce chapitre, nous nous intéressons à la résolution du problème de l'optimisation de la somme de plusieurs fractions linéaires soumis à plusieurs contraintes linéaires, où les variables de décision sont des entiers. Ce problème peut être écrit comme suit :

$$(ISOLRP)\begin{cases} \max & F(x) = \sum_{i=1}^{K} \frac{N_i(x)}{D_i(x)}, \\ S.C., & \\ & x \in \mathcal{D}, \end{cases} \tag{6.1}$$

où $\mathcal{X} = \{x \in \mathbb{R}^n / Ax = b, x \geq 0\}$, $\mathcal{D} = \mathcal{X} \cap \mathbb{Z}^n$, $\mathbb{Z}$ est l'ensemble des nombres entiers, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$. $N_i(x) = c^i x + \alpha^i$ et $D_i(x) = d^i x + \beta^i$ pour $i = 1, \ldots, K$, où $c = (c^i) \in \mathbb{R}^{K \times n}$ et $d = (d^i) \in \mathbb{R}^{K \times n}$ sont des vecteurs lignes, $\alpha = (\alpha^i) \in \mathbb{R}^K$, $\beta = (\beta^i) \in \mathbb{R}^K$.
Nous supposons que : $\mathcal{X}$ est un polyèdre convexe non vide, $\mathcal{D}$ n'est pas vide, $d^i x + \beta^i > 0$ pour tout $x \in \mathcal{X}$ et $i \in \{1, 2, \ldots, K\}$.

Dans ce chapitre, nous proposons une méthode de branch & cut pour résoudre le *ISOLRP* en partant de l'hypoThèse qu'il existe une relation entre la solution globalement optimale du *ISOLRP* et une solution efficace à un problème multiobjectif spécifique. Le reste de ce chapitre est organisé comme suit : Tout d'abord, nous montrons la relation entre *ISOLRP* et un problème multiobjectif. Ensuite, nous démontrons comment résoudre le problème multiobjectif et introduisons une technique pour améliorer la méthode. Un exemple didactique est présenté afin d'illustrer les différentes étapes de la méthode. Enfin, une étude expérimentale est présentée pour montrer l'efficacité de la méthode proposée.

6.2 Préliminaires

La méthode présentée dans ce chapitre tire parti du lien entre le problème *ISOLRP* (6.3) et le problème suivant :

$$(MOILFP)\begin{cases} \max & (f_i(x) = \dfrac{N_i(x)}{D_i(x)},\ i = \{1,\ldots,K\}), \\ s.c., & \\ & x \in \mathcal{D}. \end{cases} \tag{6.2}$$

Proposition 6.1. Une solution optimale du problème (6.3) est une solution efficace du problème (6.2) mais l'inverse n'est pas vrai.

Démonstration. Soit x^* une solution optimale entière du problème (6.3). Supposons que x^* n'est pas une solution efficace entière du problème (6.2). Ainsi, il existe un point entier $\hat{x} \in \mathcal{D}$ tel que :

$$\begin{aligned} &\begin{cases} f_i(\hat{x}) \geq f_i(x^*),\ i = \{1,\ldots,K\}, \\ f_k(\hat{x}) > f_k(x^*),\ \text{ pour au moins un } k \in \{1,\ldots,K\}. \end{cases} \\ \Longrightarrow &\begin{cases} \sum\limits_{\substack{i=1 \\ i\neq k}}^{r} f_i(\hat{x}) \geq \sum\limits_{\substack{i=1 \\ i\neq k}}^{K} f_i(x^*), \\ f_k(\hat{x}) > f_k(x^*),\ \text{ pour au moins un } k \in \{1,\ldots,K\}. \end{cases} \\ \Longrightarrow\ & \sum_{i=1}^{K} f_i(\hat{x}) > \sum_{i=1}^{K} f_i(x^*) \\ \Longrightarrow\ & F(\hat{x}) > F(x^*). \end{aligned}$$

Par conséquent, x^* n'est pas une solution optimale entière pour le problème (6.3), contradiction.

L'inverse n'est pas vrai, car la somme des composantes des solutions non dominées n'est pas toujours la même. □

Le problème *ISOLRP* peut être donc formulé comme suit :

$$(ISOLRP)\begin{cases} \max & F(x) = \sum\limits_{i=1}^{K} \dfrac{N_i(x)}{D_i(x)}, \\ S.C., & \\ & x \in \mathcal{X}_E, \end{cases} \tag{6.3}$$

où, $\mathcal{X}_E$ est l'ensemble efficace du *MOILFP* associé.

Définition 6.1. Le point idéal $(\bar{f}_1,\ldots,\bar{f}_K)$ est le vecteur composé avec les meilleures valeurs objectifs sur l'espace de recherche, i.e., $\bar{f}_i = \max_{x\in\mathcal{D}}\{f_i(x)\}$.

Proposition 6.2. La valeur optimale du problème (6.3) F^* est inférieure ou égale à $\bar{F} = \sum\limits_{i=1}^{K} \bar{f}_i$.

Démonstration. Nous avons, $\bar{f}_i \geq f_i(x)\ \forall i \implies \bar{F} = \sum\limits_{i=1}^{K} \bar{f}_i \geq \sum\limits_{i=1}^{K} f_i(x) = F(x)\ \forall x.$ □

6.2.1 Les gradients réduits des objectifs fractionnaires

La méthode de Chergui et Moulaï, 2008 consiste à résoudre une séquence de programmes de maximisation fractionnaires linéaires continus 6.4 sur des régions réduites et séparées récursivement, notées $\mathcal{X}_l$, où $\mathcal{X}_0 = \mathcal{X}$ est le domaine réalisable du problème 6.2 sans les contraintes d'intégrité.

$$(LFP_l) \begin{cases} \max & f_1(x) = \dfrac{N_1(x)}{D_1(x)}, \\ s.c., & \\ & x \in \mathcal{X}_l, \end{cases} \tag{6.4}$$

Soit $\mathcal{I}_l$ et $\mathcal{N}_l$ désignant, respectivement, l'ensemble d'indices des variables de base et l'ensemble d'indices des variables hors base de la solution optimale x^* de 6.4. Pour tous les critères du problème MOILFP, nous définissons les coûts réduits

$$\begin{aligned} v_j^{i(1)} &= c^i_{I_l} B^{-1} a_j, \\ v_j^{i(2)} &= d^i_{I_l} B^{-1} a_j, \end{aligned}$$

où a_j est la colonne j^{me} de la matrice A et B est une matrice de base.

Soit $\boldsymbol{\rho}^i$ le gradient réduit de la fonction objectif f_i, $i = \{1, \ldots, K\}$, au tableau simplexe optimal du problème LFP_l, chaque composante de $\boldsymbol{\rho}^i$ est donnée par :

$$\boldsymbol{\rho}^i_j = N_i(x^*)(d^i_j - v_j^{i(2)}) - D_i(x^*)(c^i_j - v_j^{i(1)}),\ j \in \mathcal{N}_l. \tag{6.5}$$

Définition 6.2. *Directions d'amélioration (*Chergui *et* Moulaï, *2008)* Soit x^* une solution optimale entière du problème LFP_l, nous associons à x^* un ensemble de directions d'amélioration noté $\mathcal{H}_l$ défini par :

$$\mathcal{H}_l = \left\{ j \in \mathcal{N}_l | \exists i \in \{1, \ldots, K\} \text{ with } \boldsymbol{\rho}^i_j > 0 \right\} \cup \left\{ j \in \mathcal{N}_l | \boldsymbol{\rho}^i_j = 0, \forall i \in \{1, \ldots, K\} \right\}. \tag{6.6}$$

Proposition 6.3. Soit x^* une solution optimale entière du problème LFP_l, et $\mathcal{H}_l$ son ensemble associé de directions améliorantes, Si $\mathcal{H}_l = \emptyset$ alors $\forall x \in \mathcal{X}_l \setminus \{x^*\}$, x n'est pas efficace pour le problème *MOILFP*.

Démonstration. Si $\mathcal{H} = \emptyset$ à la solution entière actuelle x^*, alors il n'y a aucune amélioration possible des valeurs des critères fractionnaires. Par conséquent, x^* étant un point idéal dans $\mathcal{X}_l$, il s'ensuit qu'il n'existe pas de solution efficace dans le domaine $\mathcal{X}_l \setminus \{x^*\}$. □

Proposition 6.4. Soit x^* est une solution optimale entière du problème LFP_l, $\mathcal{H}_l$ son ensemble non vide de directions d'amélioration, et soit $\bar{x} \in \mathcal{X}_l$ une solution telle que $\bar{x} \neq x^*$, alors :

$\forall j \in \mathcal{H}_l : \bar{x}_j = 0 \Rightarrow \bar{x}$ n'est pas efficace pour le problème $MOILFP$.

Démonstration. Soit $\bar{x}$ une solution réalisable du problème LFP_l, telle que $\bar{x} \neq x^*$ et $\bar{x}_j = 0, \forall j \in \mathcal{H}_l$.

À partir du tableau simplexe correspondant à la solution entière optimale x^*, la valeur actualisée de chaque fonction objectif f_i est écrite dans les variables hors base $j \in \mathcal{N}_l$. Ainsi, l'égalité suivante pour tous les critères $i \in \{1, \ldots, K\}$:

$$N_i^1(\bar{x}) = N_i^1(x^{*(l)}) + \boldsymbol{\mu}_j(c_j^i - v_j^{i(1)}),$$
$$D_i^2(\bar{x}) = D_i^2(x^{*(l)}) + \boldsymbol{\mu}_j(d_j^i - v_j^{i(2)}).$$

où, $\boldsymbol{\mu}_j = \dfrac{x_{B_r}^{*(l)}}{a_{rj}} = \min\left\{ \dfrac{x_{B_i}^{*(l)}}{a_{ij}} |\ a_{ij} > 0,\ i \in I_l \right\}, j \in \mathcal{N}_l.$

$$f_i(\bar{x}) = \frac{N_i(\bar{x})}{D_i(\bar{x})} = \frac{N_i(x^*) + \boldsymbol{\mu}_j(c_j^i - v_j^{i(1)})}{D_i(x^*) + \boldsymbol{\mu}_j(d_j^i - v_j^{i(2)})},\ f_i(x^*) = \frac{N_i(x^*)}{D_i(x^*)}.$$

Donc, on peut écrire :

$$\begin{aligned} f_i(\bar{x}) - f_i(x^*) &= \frac{N_i(\bar{x})}{D_i(\bar{x})} = \frac{N_i(x^*) + \boldsymbol{\mu}_j(c_j^i - v_j^{i(1)})}{D_i(x^*) + \boldsymbol{\mu}_j(d_j^i - v_j^{i(2)})} - \frac{N_i(x^*)}{D_i(x^*)} \\ &= \boldsymbol{\mu}_j \frac{D_i(x^*)(c_j^i - v_j^{i(1)}) - N_i(x^*)(d_j^i - v_j^{i(2)})}{D_i(x^*)[D_i(x^*) + (d_j^i - v_j^{i(2)})]} \\ &= \boldsymbol{\mu}_j \frac{D_i(x^*)(c_j^i - v_j^{i(1)}) - N_i(x^*)(d_j^i - v_j^{i(2)})}{D_i(x^*)D_i(\bar{x})}, \end{aligned}$$

et en utilisant 6.5, nous obtenons :

$$f_i(\bar{x}) - f_i(x^*) = \frac{\boldsymbol{\mu}_j \boldsymbol{\rho}_j^i}{D_i(x^*)D_i(\bar{x})}.$$

Comme les termes $D_i(x^*),\ D_i(\bar{x})$ sont positifs et les composantes $\boldsymbol{\rho}_j^i \leq 0$, pour tout indice $j \in \mathcal{N} \setminus \mathcal{H}_l$ et il est strictement inférieur à zéro pour au moins un critère, on a :

$$f_i(\bar{x}) - f_i(x^*) \leq 0,\ i = \{1, \ldots, K\},$$

et pour au moins un $k \in \{1, \ldots, K\}$, on a $f_k(\bar{x}) - f_k(x^*) < 0$, donc, $f_i(\bar{x}) \leq f_i(x^*)$ pour tout critère $i \in \{1, \ldots, K\}$, avec $f_k(\bar{x}) < f_k(x^*)$ pour au moins un critère $k \in \{1, \ldots, K\}$, le vecteur

critère $(f_1(\bar{x}), \ldots, f_K(\bar{x}))$ est dominé par le vecteur critère $(f_1(x^*), \ldots, f_K(x^*))$. Ainsi, $\bar{x}$ n'est pas efficace. □

Nous pouvons facilement voir à partir de la proposition 6.4 que si nous ajoutons la contrainte $\sum_{j \in \mathcal{H}_l} x_j \geq 1$ au problème LFP_l, la solution x^* est écartée de la sous-région réalisable $\mathcal{X}_l$, et l'objectif fractionnaire est également optimisé sur le sous-ensemble réduit résultant, qui est :

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

6.3 Algorithmes

6.3.1 Algorithme brute-force pour *ISOLRP*

D'après la proposition 6.4, l'algorithme suivant qui résout le problème (6.3) :

Algorithme 12 : Algorithme brute-force pour *ISOLRP*.

Résultat : x_{opt} La solution optimale globale du problème (6.3).
Trouver l'ensemble efficace $\mathcal{X}_E$ du problème 6.2.
Calculer la valeur de F pour chaque solution dans $\mathcal{X}_E$ et trouver $x_{opt} = \underset{x \in \mathcal{X}_E}{arg\ max} F(x)$.

La faille de cette méthode est qu'elle trouve l'ensemble des solutions efficaces. Plus le nombre de solutions efficaces est grand, plus la méthode prend du temps, comme nous le montrerons plus tard. Il est plus pratique de ne pas passer explicitement par toutes les solutions efficaces.

6.3.2 L'algorithme proposé : une méthode exacte pour *ISOLRP*

L'idée de l'algorithme suivant est de passer par les solutions efficaces telles que décrites dans la méthode (Chergui et Moulaï, 2008), qui consiste en une séquence de programmes de maximisation linéaire fractionnaire continue. Nous maximisons la même fonction objectif sur des régions réduites et séparées de manière récursive, notées $\mathcal{X}_l$. Notre méthode considère la valeur maximale de F_{sup} que la fonction F peut prendre à chaque noeud. Ceci peut être fait en calculant le point idéal comme indiqué dans la proposition 6.2, et F_{inf} la valeur de la meilleure solution efficace x_{opt}, qui maximise F. A chaque étape de l'algorithme, si F_{sup} est inférieur à F_{inf} alors le noeud est sondé. Sinon, si la solution $x^{(l)}$ n'est pas entière, nous procédons au processus de branchement. Sinon, on utilise le test d'efficacité 4.3 pour trouver une solution efficace $\bar{x}^{(l)}$. Si F_{inf} est inférieur à $F(\bar{x}^{(l)})$, F_{inf} est remplacé par $F(\bar{x}^{(l)})$ et $x_{opt} = \bar{x}^{(l)}$, ensuite on ajoute une coupe 6.6 au noeud suivant.

L'algorithme se termine lorsqu'il n'y a plus de noeud dans l'arbre. En d'autres termes, toutes les sous-régions nouvellement créées ont été explorées. Par conséquent, x_{opt} est la solution optimale du problème (6.3), et $F_{opt} = F(x_{inf})$ sa valeur optimale. La méthode peut être résumée comme suit :

Algorithme 13 : ISOLRP-Solver

Résultat : La solution optimale x_{opt}
Initialisation $l = 0$, $\mathcal{X}_l = \{x \in \mathbb{R}^n | Ax \leq b \text{ et } x \geq 0\}$;
Tant que *il y a un noeud non sondé l* **faire**
  Résoudre $LFP_l : \{\max\ f_1(x), x \in \mathcal{X}_l\}$
  Si LFP_l *a une solution optimale* $x^{*(l)}$ **alors**
    Calculer le point idéal $\bar{F}$ du noeud l ;
    $F_{sup} = \sum_{i=1}^{K} \bar{F}_i$;
    Si $F_{sup} \leq F_{inf}$ **alors**
      $F_{opt} = F_{inf}$;
      sonder le noeud l ;
    Finsi
    Si $x^{*(l)}$ *est une solution entière* **alors**
      Soit $\hat{x}^{(l)}$ la solution optimale du $MM(x*(l))$;
      Si $F_{inf} \leq F(\hat{x}^{(l)})$ **alors**
        $F_{inf} = F(\hat{x}^{(l)})$ $x_{opt} = \hat{x}^{(l)}$
      Finsi
      Construire l'ensemble $\mathcal{H}_l$ (6.6) ;
      Si $\mathcal{H}_l = \emptyset$ **alors**
        Sonder le noeud l
      Sinon
        Ajouter la coupe $\sum_{j \in \mathcal{H}_l} x_j \geq 1$ à l_0 le successeur de l ;
      Finsi
    Sinon
      Choisir un indice r tel que $x_r^{*(l)}$ soit fractionnaire. Ensuite, diviser le programme LFP_l en deux sous-programmes, en ajoutant respectivement les contraintes $x_r \leq \left\lfloor x_r^{*(l)} \right\rfloor$ et $x_r \geq \left\lceil x_r^{*(l)} \right\rceil$ pour obtenir $\texttt{LFP}_{l_1}$ et $\texttt{LFP}_{l_2}$ ($l_1 \geq l+1$, $l_2 > l+1$ et $l_1 \neq l_2$) ;
    Finsi
  Sinon
    Sonder le noeud l ;
  Finsi
Fintq

En effet, à une étape générale de la méthode, la meilleure solution réalisable du programme (6.3) est x_{opt} et la meilleure valeur pour la fonction objectif est $F_{opt} = F(x_{opt})$, qui est utilisée comme une borne inférieure pour F au cours du processus. Lorsque le programme LFP_l est réalisable, trois cas peuvent se présenter :

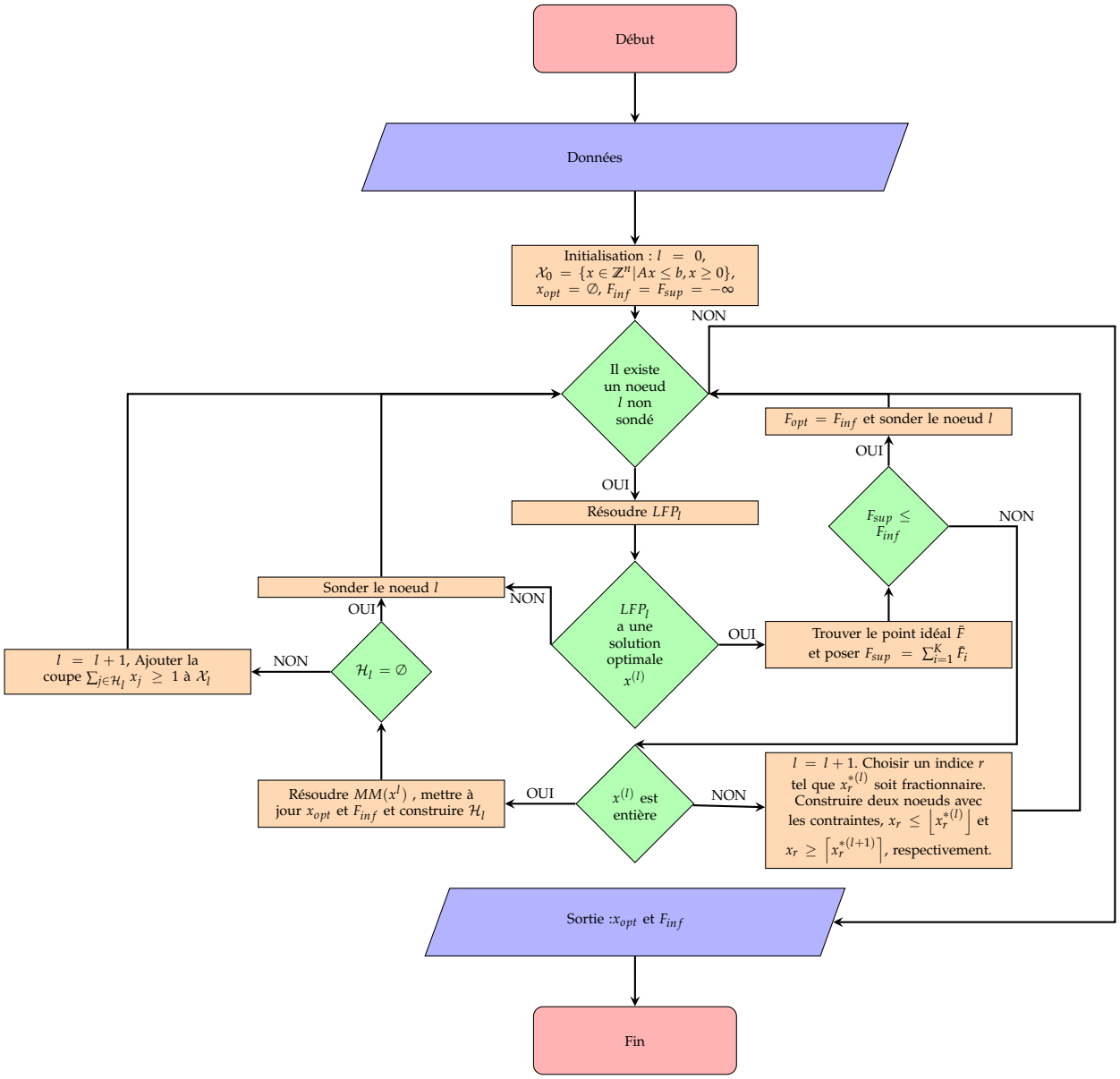

FIGURE 6.1 – Organigramme de la méthode proposée ISOLRP-Solver

— La solution x^* a des composantes fractionnaires,
— La solution x^* est entière mais pas efficace,
— La solution x^* est entière et efficace.

Dans le premier cas, deux sous-régions distinctes de $\mathcal{X}_l$ seront obtenues par les contraintes de branchement et ensuite explorées. Dans le second, en fonction des gradients réduits des objectifs fractionnaires linéaires du problème MOILFP, nous tentons de trouver des directions améliorantes et de réduire le sous-ensemble $\mathcal{X}_l$ en ajoutant la coupe efficace, et enfin dans le dernier cas, nous mettons à jour $F_{opt} = F(x^*)$, si nécessaire.

L'efficacité d'une solution entière x^* pour le problème $MOILFP$ est testée en résolvant le programme linéaire mixte 4.3, et lorsqu'elle est efficace, le domaine courant $\mathcal{X}_l$ est réduit par la coupe efficace, ou considéré comme un domaine exploré, si l'ensemble des directions améliorantes $\mathcal{H}_l$ est vide. Ce qui signifie qu'aucune amélioration pour les critères du problème $MOILFP$ ne peut être effectuée. La valeur de F_{opt} n'est pas mise à jour à moins que la solution x^* ne soit entière, efficace et que $F(x^*) > F_{opt}$. Les noeuds de l'arbre sont sondés dans les cas suivants :

— Le programme LFP_l est infaisable.
— L'ensemble $\mathcal{H}_l$ des directions d'amélioration est vide.
— La valeur optimale déjà trouvée F_{inf} est meilleure que la somme du point idéal F_{sup}.

L'algorithme se termine lorsqu'il n'y a plus de problème d'optimisation linéaire fractionnaire à résoudre. En d'autres termes, toutes les sous-régions créées ont été explorées, et alors x_{opt} est la solution optimale du problème (6.3) et $F_{opt} = F(x_{opt})$ sa valeur optimale.

Théorème 6.1. *Les algorithmes 12 et 13 convergent vers une solution optimale du programme 6.3 en un nombre fini d'itérations, si une telle solution existe.*

Démonstration. L'ensemble efficace de $MOILFP$ peut être calculé en un nombre fini d'itérations (Chergui et Moulaï, 2008). Le nombre de solutions efficaces est fini. Ainsi, l'algorithme 12 trouve la solution optimale global en un nombre fini d'itérations.

Le deuxième algorithme utilise plus de conditions pour s'arrêter plus tôt. À chaque noeud, le point idéal est calculé pour avoir une limite supérieure de F dans ce noeud en utilisant la proposition 6.4. À un noeud donné, si la valeur de F_{inf}, une solution efficace déjà trouvée, est supérieure à la borne supérieure de ce noeud. Ainsi, il n'est pas nécessaire de continuer à explorer le noeud. L'algorithme 13 converge vers une solution optimale de $ISOLRP$ en un nombre fini d'itérations. □

6.4 Un exemple didactique

Le problème considérable suivant sera utilisé pour illustrer l'utilisation de l'algorithme :

$$(ISOLR)\begin{cases} \max \left\{ \dfrac{-3x_1 - x_2 + 1}{2x_1 + 4x_2 + 2} + \dfrac{x_1 + 5x_2 - 3}{x_1 + 3x_2 + 1} + \dfrac{4x_1 + 2x_2 + 1}{x_1 + 3x_2 + 3} \right\}, \\ \text{S.C.,} \\ \qquad 4x_1 + 3x_2 \leq 10, \\ \qquad x_1 - x_2 \leq 2, \\ \qquad\quad x_2 \leq 2, \\ x_1, x_2 \geq 0, \text{ entiers.} \end{cases} \tag{6.7}$$

Initialisation : On met : $l = 0$, $\mathcal{X}_0 = \left\{(x_1, x_2) \in \mathbb{R}^2 | 4x_1 + 3x_2 \leq 10,\ x_1 - x_2 \leq 2,\ x_2 \leq 2, x_1, x_2 \geq 0\right\}$ et $fopt = -\infty$.

noeud 0 : Le tableau 6.1 est le tableau simplexe optimal après avoir résolu le programme linéaire fractionnaire $\left\{\max \dfrac{-3x_1 - x_2 + 1}{2x_1 + 4x_2 + 2} | x \in \mathcal{X}_0\right\}$.

$\mathcal{B}_0$	x_1	x_2	RHS
x_3	4	3	10
x_4	1	-1	2
x_5	0	1	2
c^1	-3	-1	1
d^1	2	4	2
c^2	1	5	-3
d^2	1	3	1
c^3	4	2	1
d^3	1	3	3
$\boldsymbol{\rho}^1$	-8	-6	
$\boldsymbol{\rho}^2$	4	14	
$\boldsymbol{\rho}^3$	11	3	

Table 6.1 – Tableau de simplexe optimal du Noeud 0

La solution trouvée est $(0, 0)$ c'est une solution entière alors $F_{opt} = \frac{-13}{6}$, $x_{opt} = (0, 0)$ et $\mathcal{H}_l = \{1, 2\}$ alors nous ajoutons la coupe $x_1 + x_2 \geq 1$ à la table 6.1.

Noeud 1 : Après avoir ajouté la coupe à la table 6.1, nous obtenons la table 6.2.

$\mathcal{B}_1$	x_1	x_6	RHS
x_2	1	-1	1
x_3	1	3	7
x_4	2	-1	3
x_5	-1	1	1
c^1	-2	-1	0
d^1	-2	4	6
c^2	-4	5	2
d^2	-2	3	4
c^3	2	2	3
d^3	-2	3	6
$\boldsymbol{\rho}^1$	-12	-6	
$\boldsymbol{\rho}^2$	-12	14	
$\boldsymbol{\rho}^3$	18	3	

TABLE 6.2 – Tableau de simplexe optimal du Noeud 1.

$(0,1)$ est la solution trouvée, F_{opt} est mis à jour puisque cette solution a une meilleure valeur que F_{opt} ,$F_{opt} = 1$, $x_{opt} = (0,1)$ et $\mathcal{H}_l = \{1,6\}$ puis on ajoute la coupe $x_1 + x_6 \geq 1$.

Noeud 2 : Le tableau suivant 6.3 donne le tableau simplexe pour ce noeud.

$\mathcal{B}_2$	x_5	x_7	RHS
x_1	-0.5	-0.5	0
x_2	1	0	2
x_3	-1	2	4
x_4	1.5	0.5	4
x_6	0.5	-0.5	1
c^1	-0.5	-1.5	-1
d^1	-3	1	10
c^2	-4.5	0.5	7
d^2	-2.5	0.5	7
c^3	06	2	5
d^3	-2.5	0.5	9
$\boldsymbol{\rho}^1$	-8	-14	
$\boldsymbol{\rho}^2$	-14	0	
$\boldsymbol{\rho}^3$	$\frac{25}{2}$	$\frac{31}{2}$	

TABLE 6.3 – Tableau de simplexe optimal du Noeud 2

La solution trouvée après résolution est $(0,2)$, la valeur de la solution est $\frac{47}{30}$, $fopt$ est mis à jour $F_{opt} = \frac{47}{30}$, $x_{opt} = (0,2)$ et $\mathcal{H}_l = \{5,7\}$ donc on ajoute la coupe $x_5 + x_7 \geq 1$.

Noeud 3 : Après résolution on obtient la solution $x_3 = (\frac{1}{2}, 1)$ qui n'est pas entière donc, on branche .

Noeud 4 : En ajoutant la contrainte $x_1 \leq 0$ le problème devient infaisable.

Noeud 5 : Nous ajoutons la contrainte $x_1 \geq 1$, le tableau suivant est donné pour ce noeud :

$\mathcal{B}_5$	x_5	x_9	RHS
x_1	0	-1	1
x_2	1	0	2
x_3	-3	4	0
x_4	1	1	3
x_6	1	-1	2
x_7	1	-2	2
x_8	0	-2	1
c^1	1	-3	-4
d^1	-4	2	12
c^2	-5	1	8
d^2	-3	1	8
c^3	-2	4	9
d^3	-3	1	10
$\boldsymbol{\rho}^1$	-4	-28	
$\boldsymbol{\rho}^2$	-16	0	
$\boldsymbol{\rho}^3$	7	31	

TABLE 6.4 – Tableau de simplexe optimal du Noeud 5.

La solution optimale est $(1,2)$ et sa valeur est $\frac{5}{3}$ que $F_{opt} = \frac{5}{3}$ et $x_{opt} = (1,2)$, aussi $\mathcal{H}_l = \{5,9\}$ alors nous ajoutons la coupe $x_5 + x_9 \geq 1$ à la table 6.4.

$\mathcal{B}_6$	x_9	x_{10}	RHS
x_1	-1	0	1
x_2	-1	1	1
x_3	7	-3	3
x_4	0	1	2
x_5	1	-1	1
x_6	-2	1	1
x_7	-3	1	1
x_8	-2	0	1
c^1	-4	1	-3
d^1	6	-4	8
c^2	6	-5	3
d^2	4	-3	5
c^3	6	-2	7
d^3	4	-3	7
$\boldsymbol{\rho}^1$	-14	-4	
$\boldsymbol{\rho}^2$	18	-16	
$\boldsymbol{\rho}^3$	14	7	

TABLE 6.5 – Tableau de simplexe optimal du Noeud 6.

Noeud 6 : La solution optimale est $(1,1)$, sa valeur est $\frac{383}{280}$, F_{opt} reste la même, et $\mathcal{H}_l = \{9,10\}$ alors on ajoute la coupe $x_9 + x_{10} \geq 1$ à la table 5.5.

Noeud 7 : Après la résolution, nous obtenons la solution $x_7 = (\frac{8}{5}, \frac{6}{5})$ qui n'est pas entière donc nous créons deux branches.

Noeud 8 : Après avoir ajouté la contrainte $x_1 \leq 1$, on obtient le tableau suivant :

$\mathcal{B}_8$	x_{11}	x_{12}	RHS
x_1	0	1	1
x_2	1	2	0
x_3	-3	-10	6
x_4	1	1	1
x_5	-1	-2	2
x_6	1	3	0
x_7	1	4	0
x_8	0	2	1
x_9	0	1	0
x^{10}	-1	-1	1
c^1	1	5	-2
d^1	-4	-10	4
c^2	-5	-11	-2
d^2	-3	-7	2
c^3	-2	-8	5
d^3	-3	-7	4
$\boldsymbol{\rho}^1$	-4	0	
$\boldsymbol{\rho}^2$	-16	-36	
$\boldsymbol{\rho}^3$	7	3	

TABLE 6.6 – Tableau de simplexe optimal du Noeud 8.

La solution optimale est $(1, 0)$, sa valeur est 0 alors F_{opt} reste le même, et $\mathcal{H}_l = \{11, 12\}$ alors nous ajoutons la coupe $x_{11} + x_{10} \geq 1$ à la table 5.7.

Noeud 9 : Après avoir ajouté la contrainte $x_{11} + x_{10} \geq 1$ le problème devient infaisable.

Noeud 10 : A partir du Noeud 7, nous ajoutons la contrainte $x_1 \geq 2$. On obtient ainsi la solution $x_{10} = (2, \frac{2}{3})$ est trouvée, ce qui n'est pas un nombre entier donc nous branchons.
Noeud 11 : Après avoir ajouté la contrainte $x_2 \leq 0$ nous obtenons le tableau suivant :

$\mathcal{B}_{11}$	x_{12}	x_{13}	RHS
x_1	-1	0	2
x_2	0	1	0
x_3	4	-3	2
x_4	1	1	0
x_5	0	-1	2
x_6	-1	1	1
x_7	-2	1	2
x_8	-2	0	3
x_9	-1	0	1
x_{10}	-1	-1	2
x_{11}	-2	-1	2
c^1	-3	1	-5
d^1	2	-4	6
c^2	1	-5	-1
d^2	1	-3	3
c^3	4	-2	9
d^3	1	-3	5
$\boldsymbol{\rho}^1$	-8	-14	
$\boldsymbol{\rho}^2$	4	-18	
$\boldsymbol{\rho}^3$	11	17	

TABLE 6.7 – Tableau de simplexe optimal du Noeud 11.

La solution optimale est $(2, 0)$, sa valeur est $\frac{5}{6}$ que F_{opt} reste la même , et $\mathcal{H}_l = \{12, 13\}$ alors on ajoute la coupe $x_{12} + x_{13} \geq 1$ à la table 5.7.

Noeud 12 : Le problème devient infaisable.

Noeud 13 : A partir du noeud 10, on ajoute la contrainte $x_2 \geq 1$ le problème devient infaisable.

L'algorithme s'arrête avec $F_{opt} = \frac{5}{3}$ et $x_{opt} = (1, 2)$.

6.5 Étude expérimentale

Nous avons réalisé une étude expérimentale pour évaluer les performances de la méthode proposée. Comme indiqué précédemment. A notre connaissance, aucune méthode ne

permet de résoudre ce problème, par conséquent, nous ne pouvons pas la comparer à une autre méthode. Les deux algorithmes (*Algorithme* 12 et *Algorithme* 13) pour résoudre le problème de la somme de fractions entiers sont codés sur Matlab 2017a et testés sur différentes instances.

Tout d'abord, nous avons réalisé une expérience aléatoire pour montrer l'efficacité de *Algorithm* 13 par rapport à *Algorithm* 12. Nous avons généré un total de 200 instances aléatoires de différentes tailles. Les instances sont divisées en différentes classes en fonction du nombre de fractions inclus dans la fonction objectif. Chaque classe a différentes sous-classes basées sur la taille du problème $n \times m$, où n est le nombre de variables et m est le nombre de contraintes. Une instance aléatoire est générée en utilisant la même approche que celle utilisée dans Jorge, 2009 : $A = (a_{i,j}) \in \mathcal{U}(0, 30)$, $b = (b_i) \in \mathcal{U}(50, 100)$, $q_j, c = (c_j^r), \alpha^k \in \mathcal{U}(-100, 100)$, $d^k = (d_j^k)$, et $\beta^k \in \mathcal{U}(1, 100)$ pour éviter que le dénominateur soit inférieur ou égal à zéro. Nous désignons par $\mathcal{U}(I_{min}, I_{max})$, une fonction qui génère uniformément un nombre entier aléatoire entre I_{min} et I_{max}. Nous avons généré 10 instances de chaque sous-classe. Chaque instance est résolue en utilisant les deux algorithmes.

Les résultats sont résumés dans le tableau 6.8, Où $r \times m \times n$ est la taille des instances de la sous-classe, minimum, maximum et le temps d'exécution moyen nécessaire pour résoudre le problème en utilisant les deux algorithmes, minimum, maximum et le nombre moyen d'itérations de *algortihm* 13. De plus, la moyenne des solutions efficaces rencontrées $\mathcal{X}_{EV}$ avec l'algorithme 13 et le nombre moyen de solutions efficaces $\mathcal{X}_E$.

TABLE 6.8 – Performance de l'algorithme 1 et de l'algorithme 2 sur des instances aléatoires

		Notre méthode(*algorithm* 13)							**Brute-force (*algorithm* 12)**			
		Temps(s)			**Nombre d'itération**				**Temps(s)**			
$r\times$	$m \times n$	min	max	moyenne	min	max	moyenne	moyenne $\mathcal{X}_{EV}$	min	max	moyenne	moyenne $\lvert\mathcal{X}_E\rvert$
3	5×5	0.02	0.92	0.20	2	49	25.70	3.20	0.14	5.27	1.17	17.30
	10×5	0.03	2.85	0.58	8	157	76.30	7.00	0.27	12.85	4.57	27.20
	20×10	0.16	1.79	0.61	42	439	153.40	7.40	4.74	39.46	16.43	34.60
	30×15	0.12	5.34	1.08	35	1007	216.30	7.90	17.41	81.35	46.55	39.60
	40×20	0.25	11.86	3.39	41	1783	517.70	9.70	45.39	511.19	180.57	46.60
	50×25	0.34	16.28	4.63	23	1912	595.90	8.10	208.94	655.26	361.09	57.40
5	5×5	0.04	0.64	0.26	8	106	53.60	11.40	0.07	0.59	0.33	28.50
	10×5	0.25	7.07	1.64	58	775	216.80	34.70	2.99	31.16	11.22	243.00
	20×10	0.09	11.56	2.42	20	1745	398.30	30.00	7.80	83.75	37.91	190.30
	30×15	0.19	31.33	8.49	28	3862	1039.50	45.20	44.37	204.57	98.21	306.80
	40×20	0.05	35.16	10.55	2	2471	1167.40	38.50	65.00	434.61	216.47	296.90
	50×25	0.52	61.37	16.35	76	4429	1256.50	36.20	260.71	-	-	425.70
7	5×5	0.02	0.46	0.27	2	100	56.10	15.50	0.11	1.25	0.33	49.10
	10×5	0.12	2.40	0.84	23	352	131.00	25.00	1.24	19.66	5.20	181.10
	20×10	0.21	6.12	1.82	29	823	268.70	28.20	6.45	51.83	24.42	427.80
	30×15	0.16	36.16	9.96	17	4025	1104.10	90.20	50.13	278.12	111.62	803.70
	40×20	0.19	42.92	15.10	20	3387	1290.30	66.40	126.92	627.47	243.60	757.40
	50×25	19.64	72.09	40.05	1125	3841	2425.33	183.67	162.06	-	-	2373.67

"-" : Impossible à résoudre dans le délai imparti (5000(s))

Le tableau 6.8 et la figure 6.2 montrent que *algortihme* 13 est plus rapide que *algortihme* 12 dans toutes les instances. Nous observons également que la méthode proposée offre une amélioration significative par rapport à l'algorithme brute-force en considérant l'évolution du temps d'exécution lors en augmentant de la taille des instances, de 98 secondes en moyenne avec *algortihme* 12 et 8,49 secondes avec *algortihme* 13 pour résoudre des instances de taille $5 \times 30 \times 15$ à plus de 5000 secondes en moyenne avec *algortihme* 12 et 16,35 secondes avec *algortihme* 13 pour résoudre des instances de taille $5 \times 50 \times 25$. Le temps d'exécution de *algortihme* 13 est raisonnable en tenant compte de la nature du problème (non-convexité de la fonction objectif et les variables sont des entiers).

Ensuite, afin de tester davantage la performance de l'algorithme proposé, nous avons testé notre algorithme avec deux ensembles d'instances : le problème du sac à dos tridimensionnel (3DKP) et le problème d'affectation (AP). Ces ensembles sont utilisés dans plusieurs travaux antérieurs, (Ozlen, Burton et MacRae, 2014 ; Boland, Charkhgard et Savelsbergh, 2016 ; Boland, Charkhgard et Savelsbergh, 2017). Puisque les ensembles sont construits pour la programmation linéaire multiobjectifs, nous avons généré la fonction objectif de manière aléatoire comme mentionné précédemment. Chaque instance est résolue avec 3, 5, 10 fractions.

Le tableau 6.9 résume les résultats de la deuxième expérience. Pour l'ensemble (3DKP), *algorithm* 13 a été capable de résoudre toutes les instances en un temps raisonnable, et nous pouvons remarquer que l'ajout de fractions supplémentaires a un faible impact sur le temps d'exécution, comme le montre la figure 6.3. Pour le deuxième ensemble (AP), nous remarquons que le temps d'exécution de *algorithm* 13 augmente très rapidement lorsque la taille de l'instance augmente. Cela est dû au fait que le nombre de solutions réalisables croît exponentiellement lorsque le nombre de variables augmente.

Lors de la résolution de l'ensemble (AP), nous avons remarqué que *algorithm* 13 peine la plupart du temps à trouver la solution réalisable entière (dans le processus de branchement). Par conséquent, nous pensons que, si nous utilisons des techniques de branch & bound plus sophistiquées pour trouver la solution entière, l'algorithme va être plus rapide.

TABLE 6.9 – Les performances de ISOLRP-Solver

Instance			**Notre méthode (*algorithm* 13)**		
Ensemble	r	n	Temps(s)	Nombre d'itération	$\mathcal{X}_{EV}$
3DKP	3	10	0.11	13	2
		20	0.57	102	8
		30	1.45	179	8
		40	2.14	251	8
		50	529.86	44342	185
	5	10	0.21	36	6
		20	0.04	2	1
		30	8.81	810	46
		40	31.26	2983	120
		50	368.75	19897	44
	10	10	0.03	2	1
		20	0.40	37	6
		30	1.49	115	17
		40	11.49	668	100
		50	6.38	314	23
AP	3	5	0.52	38	4
		10	11.34	653	35
		15	207.25	3969	35
	5	5	0.89	78	8
		10	25.23	811	43
		15	402.51	6814	57
	10	5	2.25	58	4
		10	62.49	1125	78
		15	578.41	7548	151

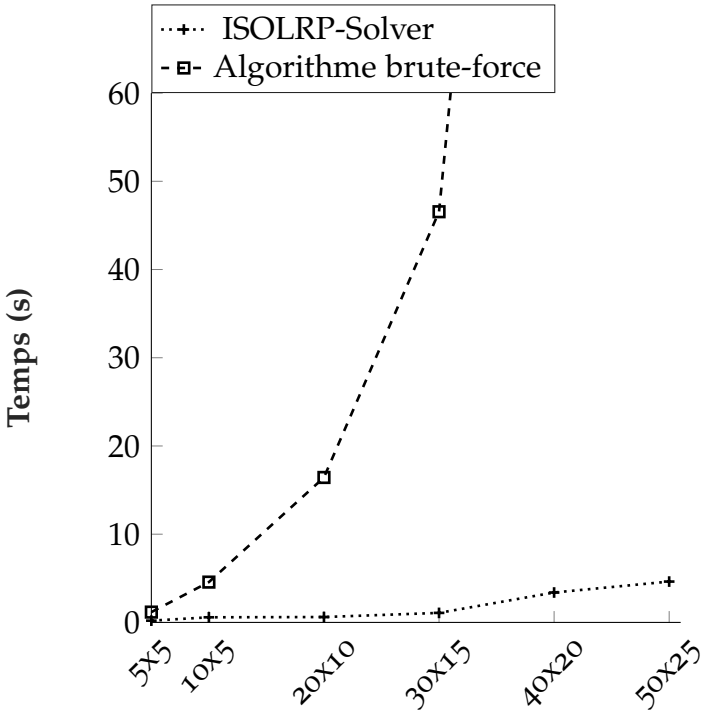

FIGURE 6.2 – Comparaison de temps d'exécution entre ISOLRP-Solver et brute-force avec 3 fractions

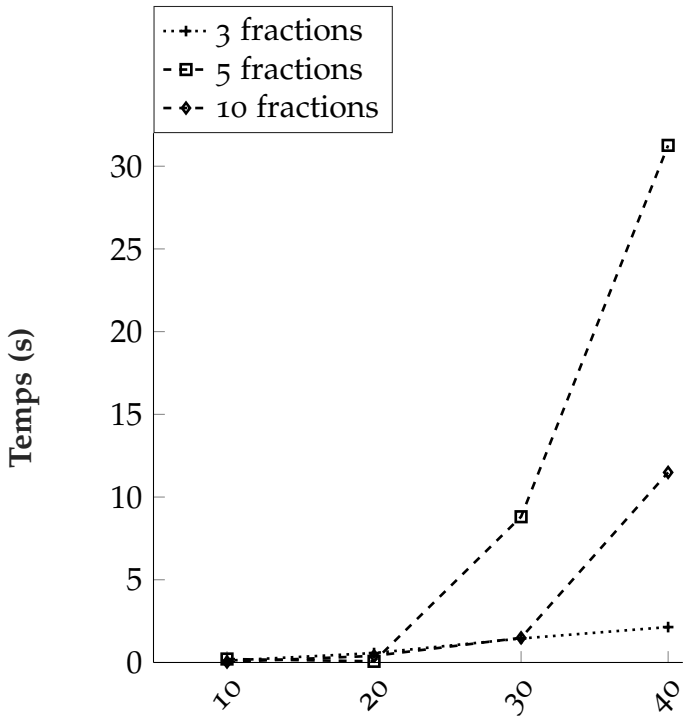

FIGURE 6.3 – Temps d'exécution moyen de ISOLRP-Solver pour les instances 3DKP

6.6 Conclusion

Dans ce chapitre, nous avons présenté une méthode exacte pour résoudre le problème de la somme de fractions linéaires avec des variables entières. L'étude expérimentale montre que la méthode permet de résoudre des instances de taille considérable en un temps raisonnable. Cependant, l'algorithme peut être amélioré en utilisant une technique de branchement plus sophistiquée afin de trouver une solution entière, ou en utilisant une autre technique pour parcourir l'ensemble efficace qui n'est pas basée sur l'algorithme du simplexe. Cela afin de pouvoir utiliser des solveurs comme Cplex ou Gurobi. Dans le cadre de travaux futurs, nous pouvons étendre l'algorithme pour résoudre le problème de la somme de fractions non linéaires. Cela conduira à une grande variété de problèmes découlant de la somme de fractions.

Chapitre 7

Conclusion générale

"We can only see a short distance ahead,
but we can see plenty there that needs to be done."

[*Nous ne pouvons voir qu'une courte distance devant nous,*
mais nous pouvons y voir beaucoup de travail à accomplir.]

Alan M., Turing

Dans la première partie de la Thèse, nous avons rappelé quelques généralités sur l'optimisation fractionnaire et l'optimisation multiobjectif et présenté quelques méthodes de résolution.

Nous présentons maintenant un résumé des conclusions de la partie contribution de cette Thèse, et quelques perspectives de recherche intéressantes encore ouvertes en rapport avec nos contributions.

7.1 Optimisation sur l'ensemble d'un problème multiobjectif linéaire fractionnaire

Nous avons proposé une méthode exacte qui trouve la solution exacte du problème d'optimisation d'une fonction sur l'ensemble efficace d'un problème multiobjectif linéaire fractionnaire en nombres entiers. La méthode est facile à mettre en œuvre et peut facilement résoudre des instances de taille moyenne. De plus, la méthode peut être utilisée pour résoudre un problème où la fonction d'utilité est convexe (pas nécessairement quadratique). Cependant, les contraintes disjonctives ajoutées à chaque itération ralentissent l'algorithme, surtout lorsque le nombre d'itérations est important. Plusieurs approches dans la littérature utilisent des techniques pour les inclure indirectement, il est donc intéressant d'utiliser une de ces techniques dans notre algorithme.

De plus, on peut considérer les fonctions fractionnaires non linéaires qui sont plus générales et ont de nombreuses applications.

7.2 Optimisation de deux fonctions fractionnaire sur l'ensemble efficace d'un multiobjectif linéaire fractionnaire

Nous avons proposé un problème qui résout un problème intéressant qui se pose lorsqu'il y a deux décideurs. La méthode proposée utilise des coupes efficaces et peut résoudre des instances de taille considérable. De plus, elle peut être facilement utilisée pour résoudre un problème où l'on doit trouver l'intersection entre deux ensembles efficaces de deux MOILFPs.

Cependant, la technique de branchement utilisée est très simple et il est donc possible d'améliorer cette méthode en utilisant une technique de Branch& Bound plus sophistiquée.

7.3 L'optimisation de la somme des fractions

Nous avons proposé une méthode exacte qui résout un problème non linéaire et non convexe à objectif unique et qui n'a jamais été traité dans la littérature. Ce problème est difficile à résoudre même dans le cas continu. Il peut être vu comme un problème d'optimisation sur l'ensemble efficace d'un $MOILFP$. La méthode peut résoudre des instances de taille considérable en un temps raisonnable. Cependant, la méthode pourrait être plus efficace si nous pouvions utiliser, par exemple, la technique d'exploration de solution efficace décrite par notre méthode (Chaiblaine et Moulaï, 2021), pour pouvoir utiliser des solveurs modernes comme Cplex ou Gurobi. Nous sommes également intéressés par l'extension de cette méthode pour résoudre le cas où les fractions sont non linéaires.

Bibliographie

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

Arora, S., K. SWARUP et al. (1977). « THE SET COVERING PROBLEM WITH LINEAR FRACTIONAL FUNCTIONAL. » In :

Bajalinov, E. B. (2013). *Linear-fractional programming theory, methods, applications and software*. T. 84. Springer Science & Business Media.

Benson, H. P. (2009). « Multi-objective Optimization : Pareto Optimal Solutions, Properties. » In : *Encyclopedia of optimization* 3, p. 489-493.

Benson, H. (2002). « Global optimization algorithm for the nonlinear sum of ratios problem ». In : *Journal of Optimization Theory and Applications* 112.1, p. 1-29.

Bitran, G. R. et A. Novaes (1973). « Linear programming with a fractional objective function ». In : *Operations Research* 21.1, p. 22-29.

Boland, N., H. Charkhgard et M. Savelsbergh (2016). « The L-shape search method for triobjective integer programming ». In : *Mathematical Programming Computation* 8.2, p. 217-251.

— (2017). « 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, p. 904-919.

Cambini, A et L Martein (1986). « A modified version of Martos' algorithm ». In : *Methods of Operation Research* 53, p. 33-44.

Chaiblaine, Y. et M. Moulaï (2021). « An exact method for optimizing a quadratic function over the efficient set of multiobjective integer linear fractional program ». In : *Optimization Letters*, p. 1-15.

Chaiblaine, Y., M. Moulaï et Y. Cherfaoui (2020). « An exact method for optimizing two linear fractional functions over the efficient set of a Multiobjective Integer Linear Fractional Program ». In : *arXiv preprint arXiv :2003.05364*.

Charnes, A. et W. W. Cooper (1962). « Programming with linear fractional functionals ». In : *Naval Research logistics quarterly* 9.3-4, p. 181-186.

Cherfaoui, Y. et M. Moulaï (2021). « Biobjective optimization over the efficient set of multiobjective integer programming problem ». In : *Journal of Industrial & Management Optimization* 17.1, p. 117.

Chergui, M. E.-A. et M. Moulaï (2008). « An exact method for a discrete multiobjective linear fractional optimization ». In : *Advances in Decision Sciences* 2008.

Conforti, M., G. Cornuéjols, G. Zambelli et al. (2014). *Integer programming*. T. 271. Springer.

Costa, J. P. (2007). « Computing non-dominated solutions in MOLFP ». In : *European Journal of Operational Research* 181.3, p. 1464-1475.

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

Drici, W., F. Z. Ouail et M. Moulaï (2018). « Optimizing a linear fractional function over the integer efficient set ». In : *Annals of Operations Research* 267.1, p. 135-151.

Ehrgott, M., H. W. Hamacher, K. Klamroth, S. Nickel, A. Schöbel et M. M. Wiecek (1997). « Equivalence of balance points and Pareto solutions in multiple-objective programming ». In : *Journal of Optimization Theory and Applications* 92.1, p. 209-212.

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

Gao, L., S. K. Mishra et J. Shi (2012). « An extension of branch-and-bound algorithm for solving sum-of-nonlinear-ratios problem ». In : *Optimization Letters* 6.2, p. 221-230.

Gilmore, P. C. et R. E. Gomory (1961). « A linear programming approach to the cutting-stock problem ». In : *Operations research* 9.6, p. 849-859.

Gupta, O. K. (1995). « Applications of quadratic programming ». In : *Journal of Information and Optimization Sciences* 16.1, p. 177-194.

Hansen, P. et B. Jaumard (1997). « Cluster analysis and mathematical programming ». In : *Mathematical programming* 79.1, p. 191-215.

Ishii, H., T. Ibaraki et H. Mine (1977). « Fractional knapsack problems ». In : *Mathematical Programming* 13.1, p. 255-271.

Jahanshahloo, G. R., B Talebian, F. H. Lotfi et J. Sadeghi (2017). « Finding a solution for Multi-Objective Linear Fractional Programming problem based on goal programming and Data Envelopment Analysis ». In : *RAIRO-Operations Research* 51.1, p. 199-210.

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

Jünger, M., T. M. Liebling, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi et L. A. Wolsey (2009). *50 Years of integer programming 1958-2008 : From the early years to the state-of-the-art*. Springer Science & Business Media.

Kirlik, G. et S. Sayin (2014). « A new algorithm for generating all nondominated solutions of multiobjective discrete optimization problems ». In : *European Journal of Operational Research* 232.3, p. 479-488.

— (2015). « Computing the nadir point for multiobjective discrete optimization problems ». In : *Journal of Global Optimization* 62.1, p. 79-99.

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

Kuno, T. (2002). « A branch-and-bound algorithm for maximizing the sum of several linear ratios ». In : *Journal of Global Optimization* 22.1-4, p. 155-174.

Kuno, T. et T. Masaki (2013). « A practical but rigorous approach to sum-of-ratios optimization in geometric applications ». In : *Computational optimization and applications* 54.1, p. 93-109.

Liu, Z. et M. Ehrgott (2018). « Primal and dual algorithms for optimization over the efficient set ». In : *Optimization* 67.10, p. 1661-1686.

Lokman, B. (2021). « Optimizing a linear function over the nondominated set of multiobjective integer programs ». In : *International Transactions in Operational Research* 28.4, p. 2248-2267.

Mahdi, S. et D. Chaabane (2015). « A linear fractional optimization over an integer efficient set ». In : *RAIRO-Operations Research* 49.2, p. 265-278.

Mahmoodian, V., H. Charkhgard et Y. Zhang (2020). « Multi-objective optimization based algorithms for solving mixed integer linear minimum multiplicative programs ». In : *Computers & Operations Research* 128, p. 105178.

Martein, A. C.-L. et I. Stancu-Minasian (1999). « A survey of bicriteria fractional problems ». In :

Martos, B. (1964). *Hyperbolic programming*. Rapp. tech. CARNEGIE INST OF TECH PITTSBURGH PA GRADUATE SCHOOL OF INDUSTRIAL ADMINISTRATION.

Matsui, T. (1996). « NP-hardness of linear multiplicative programming and related problems ». In : *Journal of Global Optimization* 9.2, p. 113-119.

Miettinen, K. (2012). *Nonlinear multiobjective optimization*. T. 12. Springer Science & Business Media.

Moulaï, M. et W. Drici (2018). « An indefinite quadratic optimization over an integer efficient set ». In : *Optimization* 67.8, p. 1143-1156.

Moulaï, M. et A. Mekhilef (2021). « Quadratic optimization over a discrete pareto set of a multi-objective linear fractional program ». In : *Optimization* 70.7, p. 1425-1442.

Multiobjective Optimization Library. http://home.ku.edu.tr/~moolibrary/. Accessed : 15-02-2021.

Munson, T. (2007). « Mesh shape-quality optimization using the inverse mean-ratio metric ». In : *Mathematical Programming* 110.3, p. 561-590.

Nash Jr, J. F. (1950). « The bargaining problem ». In : *Econometrica : Journal of the econometric society*, p. 155-162.

Neumann, J. v. (1945). « A model of general economic equilibrium ». In : *Review of Economic Studies* 13, p. 1-9.

Ozlen, M., B. A. Burton et C. A. MacRae (2014). « Multi-objective integer programming : An improved recursive algorithm ». In : *Journal of Optimization Theory and Applications* 160.2, p. 470-482.

Philip, J. (1972). « Algorithms for the vector maximization problem ». In : *Mathematical programming* 2.1, p. 207-229.

Rao, M. (1971). « Cluster analysis and mathematical programming ». In : *Journal of the American statistical association* 66.335, p. 622-626.

Ren, C. F., R. H. Li, L. D. Zhang et P. Guo (2016). « Multiobjective stochastic fractional goal programming model for water resources optimal allocation among industries ». In : *Journal of Water Resources Planning and Management* 142.10, p. 04016036.

Saghand, P. G., H. Charkhgard et C. Kwon (2019). « A branch-and-bound algorithm for a class of mixed integer linear maximum multiplicative programs : A bi-objective optimization approach ». In : *Computers & Operations Research* 101, p. 263-274.

Schaible, S. (1977). « A note on the sum of a linear and linear-fractional function ». In : *Naval Research Logistics Quarterly* 24.4, p. 691-693.

— (1981). « Fractional programming : applications and algorithms ». In : *European Journal of Operational Research* 7.2, p. 111-120.

Schaible, S. et J. Shi (2003). « Fractional programming : the sum-of-ratios case ». In : *Optimization Methods and Software* 18.2, p. 219-229.

Shao, L. et M. Ehrgott (2014). « An objective space cut and bound algorithm for convex multiplicative programmes ». In : *Journal of Global Optimization* 58.4, p. 711-728.

Sierra Altamiranda, A. et H. Charkhgard (2019). « A new exact algorithm to optimize a linear function over the set of efficient solutions for biobjective mixed integer linear programs ». In : *INFORMS Journal on Computing* 31.4, p. 823-840.

Sierra-Altamiranda, A., H. Charkhgard, M. Eaton, J. Martin, S. Yurek et B. J. Udell (2020). « Spatial conservation planning under uncertainty using modern portfolio theory and Nash bargaining solution ». In : *Ecological Modelling* 423, p. 109016.

Stancu-Minasian, I. (2017). « A eighth bibliography of fractional programming ». In : *Optimization* 66.3, p. 439-470.

— (2019). *A ninth bibliography of fractional programming*.

Stancu-Minasian, I. M. (2012). *Fractional programming : theory, methods and applications*. T. 409. Springer Science & Business Media.

Sylva, J. et A. Crema (2004). « A method for finding the set of non-dominated vectors for multiple objective integer linear programs ». In : *European Journal of Operational Research* 158.1, p. 46-55.

Wang, L., G. Huang, X. Wang et H. Zhu (2018). « Risk-based electric power system planning for climate change mitigation through multi-stage joint-probabilistic left-hand-side chance-constrained fractional programming : A Canadian case study ». In : *Renewable and Sustainable Energy Reviews* 82, p. 1056-1067.

Wolsey, L. A. (1998). *Integer programming*. T. 52. John Wiley & Sons.

Yamamoto, Y. (2002). « Optimization over the efficient set : overview ». In : *Journal of Global Optimization* 22.1, p. 285-317.

Zerdani, O. et M. Moulai (2011). « Optimization over an integer efficient set of a multiple objective linear fractional problem ». In : *Applied Mathematical Sciences*.