**N° d'ordre : 09/2018 - D/MT**

**REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE**

**Ministère de l'Enseignement Supérieur et de la Recherche Scientifique**

**Université des Sciences et de la Technologie Houari Boumediène**

**Faculté de Mathématiques**

**THESE**

Présentée pour l'obtention du **grade** de **DOCTEUR EN SCIENCES**

**En** : MATHÉMATIQUES

**Spécialité** : RECHERCHE OPÉRATIONNELLE : Mathématique de Gestion

**Par** : OUAÏL Fatma Zohra

**Sujet**

# Optimisation vectorielle discrète en avenir certain

Soutenue publiquement, le 05/05/2018 devant le jury composé de :

| | | |
|---|---|---|
| M. A. BERRACHEDI | Professeur à l' USTHB | Président |
| M. M. MOULAI | Professeur à l' USTHB | Directeur de thèse |
| M. MEA. CHERGUI | Professeur à l' USTHB | Co-directeur de thèse |
| M. M. AIDENE | Professeur à l'UMMTO | Examinateur |
| M. D. BENTERKI | Professeur à l'UFAS | Examinateur |
| M. MO. BIBI | Professeur à l'UAMB | Examinateur |
| M. D. CHAABANE | Professeur à l' USTHB | Invité |
| M. B. OUKACHA | Professeur à l'UMMTO | Invité |

# DÉDICACES

♢

*A mes*
*parents et*
*à mes grands*
*parents, A mon*
*époux et à mes ado-*
*rables filles Insaf & Yas-*
*mine, A ma tante Fadhila, A*
*mes frères, A mes amies,*
*A tous ceux qui m'ont*
*témoigné leurs*
*soutien, je dé-*
*die la pré-*
*sente*
*thèse.*

♢

# REMERCIMENTS

*Louange à Dieu, Clément et Miséricordieux, sans lui rien de tout cela n'aurait pu être*

J'aimerais tout d'abord remercier mon directeur de thèse, Professeur Moulaï Mustapha, pour la confiance qu'il m'a accordée en acceptant d'encadrer ce travail doctoral, pour ses multiples conseils et son soutien constant tout au long de ces années.

Je tiens à exprimer mes plus vifs remerciements à Professeur Chergui Mohamed El-Amine, mon co-directeur de thèse, qui fut pour moi un professeur attentif et disponible malgré ses nombreuses charges. Sa compétence, sa rigueur scientifique et sa clairvoyance m'ont beaucoup appris. Ils ont été et resteront des moteurs de mon travail de chercheur.

Je tiens aussi à exprimer l'honneur qui m'est fait par les membres du jury en acceptant d'évaluer mon travail, du temps qu'ils ont consacré à la lecture du document et leurs conseils avisés. Qu'ils trouvent ici toute ma gratitude : Professeur Berrachedi Abdelhafid (président), Professeurs Aidene Mohammed, Bibi Mohand Ouamar et Benterki Djamel (Examinateurs), Professeurs Djamel Chaabane et Oukacha Brahim (invités).

Mes remerciements vont aussi à ma famille qui, avec cette question récurrente, " quand est-ce que tu la soutiens cette thèse ? ", bien qu'angoissante en période fréquente de doutes, m'ont permis de ne jamais dévier de mon objectif final. Je tiens à remercier, en particulier, mes parents qui m'ont toujours soutenue dans cette aventure. Je leur serai toujours reconnaissante pour tout ce qu'ils ont bien voulu faire de moi. Je salue ici toutes leur détermination, leurs efforts et leur sens du sacrifice. Je remercie aussi mon cher époux pour son soutien quotidien indéfectible et son enthousiasme.

J'adresse toute ma gratitude à toutes mes amies et à toutes les personnes qui m'ont aidée dans la réalisation de ce travail. Je remercie tout le personnel de la Faculté de Mathématiques de m'avoir permis de travailler dans d'aussi bonnes conditions.

# TABLE DES MATIÈRES

**Page**

**Liste des tableaux** **vi**

**Table des figures** **vii**

**Introduction générale** **1**

**1 Éléments de la programmation mathématique** **5**

1.1 Introduction . . . . . 5

1.2 Convexité en optimisation . . . . . 7

1.2.1 Ensembles convexes . . . . . 7

1.2.2 Fonctions convexes . . . . . 9

1.3 Conditions d'optimalité . . . . . 10

1.3.1 Formule de Taylor . . . . . 10

1.3.2 Développement limité . . . . . 10

1.3.3 Généralités sur les matrices . . . . . 11

1.3.4 Direction admissible . . . . . 12

1.3.5 Conditions nécessaires d'optimalité . . . . . 12

1.3.6 Conditions de Karush-Kuhn-Tucker . . . . . 13

1.4 Conclusion . . . . . 14

**2 Programmation linéaire en nombres entiers** **15**

2.1 Introduction . . . . . 15

2.2 Programmation linéaire ([1], [2], [3]) . . . . . 15

2.2.1 Principes généraux de la programmation linéaire . . . . . 16

2.2.2 Conditions d'optimalité . . . . . 18

2.2.3 Algorithme du simplexe . . . . . 19
2.2.4 Dualité . . . . . 20
2.2.5 Dégénérescence . . . . . 23
2.3 Programmation linéaire en nombres entiers [1, 2, 4] . . . . . 24
2.3.1 Formulation mathématique et quelques propriétés . . . . . 25
2.3.2 Méthodes de résolution d'un ($PLNE$) . . . . . 26
2.4 Conclusion . . . . . 34

**3 Optimisation multiobjectif discrète 35**
3.1 Introduction . . . . . 35
3.2 Programmation multiobjectif discrète ([5],[6], [7]) . . . . . 36
3.2.1 Formulation Générale . . . . . 37
3.2.2 Concepts de base . . . . . 37
3.2.3 Illustration des définitions . . . . . 43
3.3 Tour d'horizon des méthodes de résolutions d'un programme MOILP 45
3.4 L'optimisation sur l'ensemble efficient d'un programme MOILP . . . 48
3.4.1 Formulation du problème et quelques résultats . . . . . 50
3.4.2 Méthode Abbas& Chaaabane ([8], [9]) . . . . . 51
3.4.3 Méthode Jorge [10] . . . . . 53
3.5 Conclusion . . . . . 55

**4 Optimisation sur l'ensemble efficient d'un problème MOILP 56**
4.1 Introduction . . . . . 56
4.2 Formulation mathématique . . . . . 56
4.3 Méthodologie . . . . . 59
4.4 Résultats fondamentaux . . . . . 61
4.5 Illustration numérique . . . . . 63
4.6 Expérimentation et étude comparative . . . . . 67
4.7 Conclusion . . . . . 72

**5 Une méthode exacte pour la résolution du problème MOIQP 73**
5.1 Introduction . . . . . 73

5.2 Définitions et notations . . . . . 76
5.3 Résultats théoriques . . . . . 77
5.4 Principe de la méthode . . . . . 81
5.5 Exemple numérique . . . . . 84
5.6 Expérimentation et résultats . . . . . 88
5.7 Conclusion . . . . . 94

**Conclusion générale** **96**

**Bibliographie** **98**

# LISTE DES TABLEAUX


| TABLE | | Page |
|---|---|---|
| 2.1 | Écriture canonique de $(PL)/B$ | 19 |
| 4.1 | Résutats pour $r = 3$ | 67 |
| 4.2 | Résultats pour $r = 4$ | 68 |
| 4.3 | Résultats pour $r = 5$ | 68 |
| 5.1 | Résultat pour les problèmes de tailles moyennes | 89 |
| 5.2 | Résultats pour les problèmes de grandes tailles | 90 |
| 5.3 | Autres résultats | 94 |

# TABLE DES FIGURES

**FIGURE** **Page**


3.1 Espace des décisions . . . . . 44
3.2 Espace des critères . . . . . 44
3.3 Solutions supportées et non supportées . . . . . 44

4.1 Domaine réalisable . . . . . 63
4.2 Comparaison de CPU(s) pour chaque méthode . . . . . 69
4.3 Comparaison de |Eff| pour chaque méthode . . . . . 70
4.4 Comparaison de |ILP| pour chaque méthode . . . . . 71

5.1 L'arbre de recherche de l'exemple numérique . . . . . 88
5.2 Moyenne du temps de calcul pour chaque $(n,m)$ et $r=3$ . . . . . 91
5.3 Moyenne du temps de calcul pour chaque $(n,m)$ et $r=5$ . . . . . 91
5.4 Moyenne du temps de calcul pour chaque $(n,m)$ et $r=7$ . . . . . 92
5.5 Sensibilité de CPU(s) /au nombre de critères (pbs de tailles moyennes) . 93
5.6 Sensibilité de CPU(s) /au nombre de critères (pbs de grandes tailles) . . 93

# INTRODUCTION GÉNÉRALE

La recherche opérationnelle s'est développée comme une discipline scientifique durant les années 1950, ayant depuis proposé des méthodes et des techniques adéquates permettant de résoudre plusieurs problèmes de décision rencontrés dans divers domaines (transport, télécommunication, économie ...etc). On admettait ainsi implicitement que pour aider les entreprises à mieux décider, il y avait une règle générale, une fonction objectif qui s'imposait aux yeux de tous pour caractériser la bonne direction dans laquelle il convenait de faire évoluer le système auquel on s'intéressait [11].

La programmation multiobjectif peut être considérée comme l'extension de la programmation classique à un seul objectif au cas où plusieurs fonctions objectif sont explicitement considérées. La difficulté principale de tels problèmes est liée à la présence de conflits entre les divers objectifs. En effet, les solutions optimales, pour un objectif donné, ne correspondent généralement pas à celles des autres objectifs pris indépendamment. De ce fait, il n'existe, la plupart du temps aucun point de l'espace de recherche où toutes les fonctions objectif sont optimales simultanément. On parle dans ce cas de problème d'optimisation multiobjectif qui définit un ensemble de solutions acceptables assurant un compromis entre les objectifs considérés.

Historiquement, cette discipline fut premièrement introduite en économie par Francis Y. Edgeworth [12] et Vilfredo Pareto [13]. Depuis lors, l'optimisation multiobjectif s'est développée à un rythme rapidement croissant, en particulier, au cours des quatre dernières décennies. Aujourd'hui, de nombreux systèmes d'aide à la décision intègrent des méthodes pour faire face à des objectifs contradictoires.

Il est opportun de mentionner que, sous la désignation commune d'approches à

critères multiples, deux branches distinctes apparaissent dans la littérature spécialisée :

(a) Les méthodes d'aide à la décision multi-attribut ;

(b) Les méthodes d'aide à la décision multiobjectif.

La première désignation fait généralement référence aux méthodes de sélection et de classement traitant d'un ensemble fini d'alternatives, explicitement connues à priori. Quant à la deuxième désignation, elle concerne les problèmes dans lesquels les alternatives sont implicitement définies par un ensemble de contraintes.

En fait, on distingue en réalité deux types de problèmes d'optimisation multiobjectif selon la nature des variables de décision : les problèmes en variables *discrètes* et les problèmes en variables *continues*. Un nombre infini de solutions dites *efficaces* ou *Pareto-optimales* peuvent être trouvées pour un problème d'optimisation multiobjectif en variables continues. Cependant, les problèmes d'optimisation multiobjectif en variables discrètes ont un nombre de solutions Pareto-optimal fini mais qui peut être très grand. De plus, l'ensemble de ces solutions est non convexe.

Le travail présenté dans cette thèse concerne la programmation mathématique multiobjectif en nombres entiers. L'intérêt de tels problèmes provient du fait qu'il existe un large éventail d'applications concrètes dont l'introduction des variables discrètes est obligatoire.

Notre premier objectif fut de réaliser une étude détaillée sur la programmation linéaire multiobjectif en nombres entiers (MOILP) en passant en revue l'important de la littérature existante. En fait, le nombre de solutions efficaces peut être très grand et la présentation de telles solutions à un décideur peut le confondre et rendre la sélection d'une solution préférée pratiquement impossible. Une approche qui atténue ce problème consiste à trouver la meilleure solution parmi les solutions efficaces (de préférence sans les énumérer toutes). Cette approche est connue comme l'optimisation sur l'ensemble efficient, qui est un problème d'optimisation globale . En effet, ce problème n'a pas reçu beaucoup d'attention vu sa difficulté (ensemble des solutions réalisables non convexe). Nous énumérons seulement quatre méthodes qui ont été proposées dans la littérature pour traiter ce problème. Dans cette thèse, nous proposons de résoudre ce même problème [14], en utilisant le principe de "branch &

bound" et le concept de coupes efficaces afin de générer la solution optimale. De plus, une étude comparative a été établie avec la méthode proposée par Jorge [10] montrant ainsi qu'en utilisant les mêmes conditions de programmation, notre méthode est meilleure.

Cependant, la configuration où toutes les fonctions objectif sont linéaires est en fait un cas particulier des problèmes rencontrés en optimisation. Les fonctions étudiées peuvent, bien sûr, sortir du cadre linéaire. Dans ce cas, il apparaît alors comme évident que la recherche des solutions efficaces est plus difficile que dans le cas linéaire. Notre deuxième contribution dans cette thèse concerne la programmation quadratique qui traite d'une partie spécifique des problèmes non linéaires : ceux qui font appel à des fonctions quadratiques et à des contraintes linéaires.

Comme exemple de problèmes proposés dans la littérature, nous citons, le problème d'affectation quadratique et le problème du sac-à-dos quadratique dans les versions aussi bien monoobjectif que multiobjectif. Si le problème général de la programmation multiobjectif quadratique continue a été investi par de nombreux chercheurs, force est de constater que le même problème considérant des variables discrètes (noté MOIQP) n'a pas encore reçu l'attention voulue alors que plusieurs applications ont été rapportées principalement dans le domaine de la finance, par exemple le problème célèbre de la sélection de portefeuilles [15].

La méthode proposée dédiée au problème MOIQP [16] est une généralisation de notre méthode de recherche de l'ensemble complet des solutions efficaces dans le cas linéaire [17]. Ceci est rendu possible grâce à l'utilisation des gradients des fonctions quadratiques qui sont supposées être différentiables sur le domaine des solutions réalisables.

L'organisation de cette thèse est la suivante :

Le chapitre 1 expose l'essentiel de la programmation mathématique nécessaire à la compréhension du reste de cette thèse.

Le chapitre 2 expose le cadre général de notre travail. Dans un premier temps, les principaux résultats concernant la programmation linéaire (en variables continues et en variables discrètes) sont décrits, notamment les méthodes de résolution classiques (méthode du simplexe, duale du simplexe, coupes de Gomory, la méthode

"branch & bound", les méthodes de points intérieurs, etc.).

Le chapitre 3 parle des définitions et des résultats liées à l'optimisation multiobjectif en nombres entiers. Aussi, nous présentons un tour d'horizon des méthodes d'optimisation multiobjectif discrète existantes dans la littérature. Aussi dans une dernière partie, nous abordons la problématique de l'optimisation d'un critère linéaire sur l'ensemble efficient d'un problème MOILP et nous détaillons quelques méthodes de résolution.

Le chapitre 4 aborde un problème d'optimisation globale. Une méthode exacte est proposée pour la détermination de la solution optimale pour le problème de l'optimisation d'un critère linéaire sur l'ensemble efficient d'un problème MOILP. Une étude comparative est rapportée pour évaluer les performances de la méthode proposée avec celle proposée par Jorge [10].

Dans le chapitre 5, une méthode exacte est proposée pour la résolution du problème MOIQP. La méthode a été programmée à l'aide du logiciel Matlab afin d'évaluer sa performance.

Le travail est achevé par une conclusion et des perspectives ouvrant la voie à de futurs travaux de recherche sur des classes de problèmes de programmation multiobjectif.

CHAPITRE 1

# ÉLÉMENTS DE LA PROGRAMMATION MATHÉMATIQUE

## 1.1 Introduction

Dans la première partie de ce chapitre, nous allons passer en revue quelques concepts mathématiques qui sont nécessaires pour étudier la programmation mathématique et en particulier les sujets abordés dans le reste de cette thèse.

Un problème de programmation mathématique est une classe spéciale de problèmes de décision où nous sommes concernés par l'utilisation efficace des ressources pour atteindre les objectifs souhaités.

Mathématiquement le problème peut être énoncé comme :

$$(P)\begin{cases} Min \quad f(x) \\ h_i(x) = 0 \\ g_j(x) \leq 0 \\ x \geq 0 \end{cases} \tag{1.1}$$

où :

- $x = (x_1, \cdots, x_n)^t \in \mathbb{R}^n$ ;
- $f$, $h_i$, et $g_j$ sont des fonctions réelles de $x$, $i = 1, \cdots, m_1$ et $j = 1, \cdots, m_2$.

Si les fonctions $f$, $h_i$ et $g_j$ sont toutes linéaires, le problème est connu sous

le nom de problème de programmation linéaire, sinon on dit qu'il s'agit d'un programme non linéaire.

L'intérêt pour la programmation non linéaire s'est développé presque simultanément avec l'intérêt croissant pour la programmation linéaire. Il a été rapidement reconnu que beaucoup de problèmes pratiques ne peuvent être représentés par un modèle de programmation linéaire. Par conséquent, les tentatives ont été faites pour développer des méthodes de résolution pour des programmes mathématiques plus généraux. En 1951, Kuhn & Tucker [18] ont fourni les conditions nécessaires pour trouver des solutions optimales aux problèmes de programmation non linéaire. Depuis lors, de nombreux auteurs ont développé des méthodes de résolution pour des programmes non linéaires de natures diverses.

En programmation mathématique, on distingue les branches suivantes :

1. **Programmation linéaire :** la discipline mathématique consacrée à la théorie et aux méthodes de résolution de problèmes concernant les extrema de fonctions linéaires sur des ensembles d'un espace vectoriel n-dimensionnel spécifié par des systèmes d'inégalités et d'égalités linéaires ;
2. **Programmation non linéaire :** La programmation non linéaire implique l'optimisation d'une fonction objectif non linéaire soumise à des contraintes linéaires et/ou non linéaires.
3. **Programmation convexe :** la branche de la programmation mathématique traitant de la théorie et des méthodes de résolution des problèmes de minimisation des fonctions convexes sur des ensembles convexes définis par des systèmes d'égalités et d'inégalités.
4. **Programmation quadratique :** la programmation quadratique est une branche de la programmation non linéaire. Elle est consacrée à la théorie et aux méthodes de résolution de problèmes de minimisation de fonctions quadratiques convexes sur des ensembles définis par des systèmes d'inégalités et d'égalités linéaires.
5. **Programmation en nombres entiers :** un problème de programmation en nombres entiers est un problème d'optimisation dans lequel certaines ou

toutes les variables sont astreintes à être entières.

6. **Programmation stochastique :** les programmes stochastiques sont des programmes mathématiques dans lesquels certaines des données incorporées dans l'objectif ou dans les contraintes sont incertaines. L'incertitude est généralement caractérisée par une distribution de probabilité sur les paramètres.

   Les notions et concepts de ce chapitre sont largement extraits des ouvrages de la littérature ([19], [20], [21]).

## 1.2 Convexité en optimisation

Les ensembles et fonctions convexes jouent un rôle important en optimisation. Dans cette section, nous présentons les concepts de base et les résultats fondamentaux concernant la convexité des ensembles et des fonctions dans $\mathbb{R}^n$.

### 1.2.1 Ensembles convexes

**Définition 1.1.** Soit l'ensemble $S \subset \mathbb{R}^n$. Si, pour tout $x_1$, $x_2 \in S$, nous avons :

$$\alpha x_1 + (1-\alpha)x_2 \in S \quad \forall \alpha \in [0,1]$$

Alors $S$ est dit ensemble convexe.

Cette définition indique, en géométrie, que pour deux points quelconques $x_1$, $x_2 \in S$, le segment de droite joignant $x_1$ et $x_2$ est entièrement contenu dans $S$.

Elle se généralise de la façon suivante, l'ensemble $S \subset \mathbb{R}^n$ est convexe si et seulement si pour tout $x_1, x_2, \cdots, x_m \in S$,

$$\sum_{i=1}^{m} \alpha_i x_i \in S$$

où : $\sum_{i=1}^{m} \alpha_i = 1$, $\alpha_i \geq 0$, $i = 1, \cdots, m$

Les théorèmes ci-dessous énoncent les propriétés algébriques et les propriétés topologiques. A savoir, l'intersection de deux ensembles convexes est convexe, la somme algébrique de deux ensembles convexes est convexe, l'intérieur d'un ensemble convexe est convexe et la fermeture d'un ensemble convexe est convexe.

**Théorème 1.1.** *Soient $S_1$ et $S_2$ deux ensembles convexes de $\mathbb{R}^n$, alors :*

- *$S_1 \bigcap S_2$ est convexe ;*
- *La somme $S_1 + S_2 = \{x_1 + x_2 \mid x_1 \in S_1, x_2 \in S_2\}$ est convexe.*

**Théorème 1.2.** *Soit S un ensemble convexe, l'adhérence de S et l'intérieur de S sont convexes.*

Soit $S \subset \mathbb{R}^n$. L'intersection de convexes étant convexe, on peut parler du plus petit convexe contenant $S$, qui est donc l'intersection de tous les convexes contenant $S$. C'est l'ensemble des points de $\mathbb{R}^n$ qui s'écrivent comme combinaisons convexes des points de $S$. On le note :

$$conv(S) = \left\{ x \in \mathbb{R}^n | x = \sum_{i=1}^{p} \alpha_i x_i, \ , x_i \in S, \ \alpha_i \geq 0, \ i = 1, \cdots, p \ et \sum_{i=1}^{p} \alpha_i = 1 \right\}$$

**Définition 1.2.** Soit $K$ une partie d'un espace métrique $E$. On dit que $K$ est compact s'il vérifie la propriété :

**Propriété de Borel-Lebesgue** : De tout recouvrement de $K$ par des ouverts on peut extraire un sous-recouvrement fini.

Ceci se traduit de la manière suivante : si $(U_i)_{i \in I}$ est une famille d'ouverts telle que $K \subset \bigcup_{i \in I} U_i$ alors il existe un sous-ensemble fini $J \subset I$ tel que $K \subset \bigcup_{i \in J} U_i$.

**Proposition 1.1.** *$S \subset \mathbb{R}^n$ est compact si et seulement s'il est fermé et borné.*

**Définition 1.3.** Un polyhèdre convexe $P$ est l'intersection d'un nombre fini de demi-espaces fermés et/ou d'hyperplans.
Un polyhèdre convexe $P$ est dit borné, s'il existe une valeur $\beta$ finie et positive telle que :

$$\left| x_j \right| \leq \beta \quad \forall j = 1 \ldots n, \ \ \forall x \in P \tag{1.2}$$

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

**Définition 1.4.** On dit que $x$ est un point extrême (ou sommet) de $P$, si :

$$x = \lambda y + (1 - \lambda) z, \ pour \ y, z \in P, \ 0 < \lambda < 1, \ alors \ x = y = z. \tag{1.3}$$

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

### 1.2.2 Fonctions convexes

**Définition 1.5.** Une fonction $f : S \subset \mathbb{R}^n \to \mathbb{R}$ est convexe si :

- Le domaine $S$ de $f$ est convexe et non vide ;
- $\forall (x_1, x_2) \in S^2$ et pour tout $\lambda$, $0 \leq \lambda \leq 1$, on a :

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2) \tag{1.4}$$

En d'autres termes, le graphe de la fonction $f$ se trouve toujours en dessous du segment reliant les points $(x_1, f(x_1))$ et $(x_2, f(x_2))$.

**Remarque 1.1.** *Si $f$ est une fonction linéaire, alors pour tout $(x_1, x_2)$ dans $S^2$ et pour tout $\lambda_1$ et $\lambda_2$ on a :*

$$f(\lambda_1 x_1 + \lambda_2 x_2) = \lambda_1 f(x_1) + \lambda_2 f(x_2) \tag{1.5}$$

Donc les fonctions convexes généralisent les fonctions linéaires de deux manières :

- Elles sont sous-linéaires, signifiant que l'égalité en (1.5) est remplacée par inégalité ;
- Seuls certains coefficients sont autorisés : ils doivent satisfaire $\sum_{i=1}^{i=p} \lambda_i = 1$.

**Définition 1.6.** L'épigraphe de $f$ est la partie de $\mathbb{R}^{n+1}$ définie par :

$$Epi(f) = \{(x, y) \in S \times \mathbb{R} | y \geq f(x)\} \tag{1.6}$$

**Théorème 1.3.** *La fonction $f$ est convexe si, et seulement si, son épigraphe est convexe dans $S \times \mathbb{R}$.*

**Définition 1.7.** $f$ est dite strictement convexe sur $S$ si : $\forall (x_1, x_2) \in S^2$ et pour tout $\lambda$, $0 < \lambda < 1$, on a :

$$f(\lambda x_1 + (1-\lambda)x_2) < \lambda f(x_1) + (1-\lambda) f(x_2) \tag{1.7}$$

**Définition 1.8.** $f$ est concave (strictement concave) si et seulement si $-f$ est convexe (strictement convexe) sur $S$.

**Théorème 1.4.** *(Inégalité de Jensen) Soient $f$ une fonction convexe, $(x_1, \cdots, x_n)$ un n-uplet de réels appartenant à l'intervalle de définition de $f$ et $(\lambda_1, \cdots, \lambda_n)$ un n-uplet de réels positifs tels que : $\sum_{i=1}^{n} \lambda_i = 1$, alors :*

$$f\left(\sum_{i=1}^{n} \lambda_i x_i\right) \leq \sum_{i=1}^{n} \lambda_i f(x_i). \tag{1.8}$$

## 1.3 Conditions d'optimalité

**Définition 1.9.** (vecteur gradient) Le gradient de $f$ en $a$ est le vecteur dont les composantes sont les dérivées partielles premières. En effet, si $f : U \subset \mathbb{R}^n \longrightarrow \mathbb{R}$, on définit le gradient par :

$$\nabla f(a) = \left(\frac{\partial f}{\partial x_1}(a), \cdots, \frac{\partial f}{\partial x_n}(a)\right)^t$$

Il est orthogonal à la tangente de la ligne de niveau de $f$ passant par $a$.

### 1.3.1 Formule de Taylor

La formule de Taylor, du nom du mathématicien Brook Taylor qui l'établit en 1712, permet l'approximation d'une fonction plusieurs fois dérivable au voisinage d'un point par un polynôme dont les coefficients dépendent uniquement des dérivées de la fonction en ce point.

Soit $f$ une fonction de classe $C^{n+1}$ sur $I$, $x_0$ et $x$ des points de $I$. On a :

$$f(x) = P_n(x) + \int_{x_0}^{x} \frac{(x-t)^n}{n!} f^{(n+1)}(t)\,\mathrm{d}t$$

Où :

$$P_n(x) = f(x_0) + \frac{(x-x_0)}{1!} f^{(1)}(x_0) + \cdots + \frac{(x-x_0)^n}{n!} f^{(n)}(x_0)$$

est l'approximation de Taylor à l'ordre $n$ ; et $R_n(x) = \int_{x_0}^{x} \frac{(x-t)^n}{n!} f^{(n+1)}(t)\,\mathrm{d}t$ est le reste intégral d'ordre $n$. On note $f^{(i)}$ la dérivée $i$ème de $f$.

### 1.3.2 Développement limité

**Définition 1.10.** Soit $f$ une fonction définie au voisinage de $x_0$ . On dit que $f$ admet un développement limité d'ordre $n$ au voisinage de $x_0$ , s'il existe une fonction

polynôme $P_n$ de degré inférieur ou égal à $n$, et une fonction $\epsilon$, définies au voisinage de $x_0$ telles que :

$$f(x) = P_n(x) + (x - x_0)^n \epsilon(x)$$

avec

$$\lim_{x \to x_0} \epsilon(x) = 0$$

$P_n(x)$ est la partie régulière et $(x-x_0)^n \epsilon(x)$ est le reste. Dans ce cas, on a des fonctions équivalentes : $f(x) \backsim P_n(x)$.

### 1.3.3 Généralités sur les matrices

Pour $n$, $p$ appartenant à $N^*$, on note $\mathcal{M}_{n,p}(\mathbb{R})$ l'ensemble des matrices à $n$ lignes et $p$ colonnes.

**Définition 1.11.** Soit $A \in \mathcal{M}_{n,n}(\mathbb{R})$ une matrice symétrique. $A$ est définie positive si et seulement si $x^t A x > 0$, $\forall x \in \mathbb{R}^n$, $x \neq 0$. $A$ est semi définie positive si $x^t A x \geq 0$, $\forall x \in \mathbb{R}^n$. $A$ est définie négative ou semi définie négative si $-A$ est définie positive ou semi définie positive. $A$ est indéfinie si elle est ni semi définie positive ni semi définie négative.

**Proposition 1.2.** *Soit $A \in \mathcal{M}_{n,n}(\mathbb{R})$ symétrique. Alors $A$ est définie positive si et seulement si toutes ses valeurs propres sont positives. $A$ est semi-définie positive si et seulement si toutes ses valeurs propres sont non négatives. $A$ est définie négative ou semi définie négative si et seulement si toutes ses valeurs propres sont négatives ou non-positives. $A$ est indéfinie si et seulement si elle possède à la fois des valeurs propres positives et négatives.*

**Théorème 1.5.** *(Théorème spectral) Une matrice $A$ est symétrique si et seulement si elle peut être factorisée sous la forme $A = Q \Lambda Q$ Où $Q$ est orthogonale et $\Lambda$ est la matrice diagonale des valeurs propres de $A$.*

### 1.3.4 Direction admissible

Une direction est appelée ***admissible*** en un point donné d'un ensemble $S$, si un déplacement suffisamment faible dans cette direction ne laisse pas $S$. Nous donnons la définition mathématique suivante :

**Définition 1.12.** Soit $S \subseteq \mathbb{R}^n$. $d$ est une direction admissible si $\exists \bar{\lambda} > 0$ t.q. :

$$x + \lambda d \in S, \quad \forall \lambda,\ 0 \leq \lambda \leq \bar{\lambda}$$

**Proposition 1.3.** *$f : \mathbb{R}^n \longmapsto \mathbb{R}$ de classe $C^1$. $x^*$ minimum local de $f$ sur $S$*
*$\Rightarrow \forall$ la direction admissible $d$ en $x^*$, on a $\nabla f(x^*)^t d \geq 0$.*

### 1.3.5 Conditions nécessaires d'optimalité

**Théorème 1.6.** *Si $f : S \subseteq \mathbb{R}^n \to \mathbb{R}$ est une fonction convexe et différentiable sur $S$ convexe alors, tout minimum local de $f$ est un minimum global.*

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

$$f(\lambda \overline{x} + (1-\lambda)x^*) \leq \lambda f(\overline{x}) + (1-\lambda) f(x^*) < \lambda f(x^*) + (1-\lambda) f(x^*) = f(x^*). \tag{1.9}$$

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

**Théorème 1.7.** *Soit $f : S \subseteq \mathbb{R}^n \to \mathbb{R}$ une fonction convexe et différentiable sur $S$ convexe et $x^* \in S$ tel que $\nabla f(x^*) = 0$. Alors, $x^*$ est un minimum global sur $S$.*

### 1.3.6 Conditions de Karush-Kuhn-Tucker

Considérons un problème d'optimisation avec des contraintes d'égalité et d'inégalité :

$$(P)\begin{cases} Min \quad f(x) \\ h_i(x) = 0, i = 1, \cdots, m_1 \\ g_j(x) \leq 0, j = 1, \cdots, m_2 \\ x \geq 0 \end{cases} \tag{1.10}$$

Notons $S$ le domaine des contraintes.

**Définition 1.13.** (Contraintes actives) Soient $g_j : \mathbb{R}^n \to \mathbb{R}$ et $h_i : \mathbb{R}^n \to \mathbb{R}$ :

- Une contrainte d'inégalité $g_j(x) \leq 0$ est dite *active* en $x^*$ si $g_j(x^*) = 0$ et elle est inactive si $g_j(x^*) < 0$ ;
- Par extension, une contrainte d'égalité $h_i(x) = 0$ sera dite active en $x^*$, si elle est vérifiée en $x^*$, c-à-d $h_i(x^*) = 0$ ;
- L'ensemble des indices des contraintes actives en $x^*$ est noté $I(x^*)$

**Définition 1.14.** (Contraintes qualifiées) Les contraintes du problème $(P)$ sont dites *qualifiées* au point $x^* \in S$ si $h_i$ et $g_j$ sont différentiables au point $x^*$ et si les gradients des contraintes actives en $x^*$ sont linéairement indépendants. Le point $x^*$ est dit dans ce cas *point régulier*.

On introduit le Lagrangien associé au problème $(P)$ :

$$L(x;\lambda,\mu) = f(x) + \sum_{i=1}^{m_1} \lambda_i h_i(x) + \sum_{j=1}^{m_2} \mu_j g_j(x), x \in \mathbb{R}^n, \lambda \in \mathbb{R}^{m_1}, \mu \in \mathbb{R}^{m_2}, \mu \geq 0 \tag{1.11}$$

**Théorème 1.8.** *(CN d'optimalité de Karush-Kuhn-Tucker) Soit $x^* \in S$ un point réalisable du problème $(P)$. Supposons $f$, $g_j$ et $h_i$ sont différentiables en $x^*$ et les contraintes sont qualifiées au point $x^*$. Si $x^*$ est un minimum local de $f$ sur $S$ alors il existe $\lambda^* \in \mathbb{R}^{m_1}$ et $\mu^* \in \mathbb{R}^{m_2}$ tels que :*

$$\begin{cases} \nabla_x L(x^*;\lambda^*,\mu^*) = 0 \\ h_i(x^*) = 0, i = 1, \cdots, m_1 \\ \mu_j^* g_j(x^*) = 0, j = 1, \cdots, m_2 \\ g_j(x^*) \leq 0, j = 1, \cdots, m_2 \\ \mu_j^* \geq 0, j = 1, \cdots, m_2;\ x^* \geq 0 \end{cases} \tag{1.12}$$

*Si de plus le problème* $(P)$ *est convexe, alors les conditions de* $KKT$ *sont suffisantes pour que* $x^*$ *soit un minimum global de* $f$.

## 1.4 Conclusion

Dans ce chapitre, nous avons passé en revue les principaux résultats liés aux problèmes de programmation mathématique. Nous détaillons dans le chapitre qui suit une classe très importante des programmes mathématiques à savoir, la programmation linéaire en nombres entiers.

CHAPITRE 2

# PROGRAMMATION LINÉAIRE EN NOMBRES ENTIERS

## 2.1 Introduction

Dans ce chapitre, nous commençons par donner l'essentiel de la programmation linéaire PL, étant donné que la programmation linéaire en nombres entiers PLNE est généralement considérée comme une extension de PL. Nous allons ensuite donner les principales propriétés d'un programme PLNE tout en mettant l'accent sur ses applications et techniques de résolution. Enfin, la dernière partie de ce chapitre est consacrée à la complexité de calcul de cette classe de problèmes.

## 2.2 Programmation linéaire ([1], [2], [3])

La *programmation linéaire* se réfère au processus de minimisation (ou de maximisation) d'une fonction objectif linéaire dont les variables sont soumises à un nombre fini de contraintes linéaires d'égalité et/ou d'inégalité. Bien que les fonctions linéaires sont des fonctions simples, elles sont présentes fréquemment dans les domaines de l'économie, de la planification, de la production, des réseaux, de l'ordonnancement et dans d'autres applications. Le développement de la théorie de la programmation linéaire est crédité à George B. Dantzig en 1947, grâce à ses travaux éminents qui ont donné naissance à la fameuse ***méthode du simplexe***.

### 2.2.1 Principes généraux de la programmation linéaire

Sans perte de généralités, la formulation du problème de programmation linéaire se décline comme suit :

$$(PL)\begin{cases} Max \quad z = c^t x \\ Ax = b \\ x \geq 0 \end{cases} \tag{2.1}$$

Où : $A$ est une $m \times n$-matrice réelle, $b$ est un $m$-vecteur réel, $c$ et $x$ sont deux $n$-vecteurs réels avec la restriction sur $x$, qui doit être non négatif. Le système $Ax = b$ est supposé de rang plein, c'est-à-dire le nombre de lignes de $A$ linéairement indépendantes est égal à $m < n$.

L'ensemble de toutes les solutions réalisables de $(PL)$ :

$$S = \{x \in \mathbb{R}^n | Ax = b,\ x \geq 0\} \tag{2.2}$$

est un ***polyèdre convexe***, par définition c'est une intersection d'un nombre fini d'hyperplans.

Un point $x \in S$ est dit un ***point extrême*** ou ***sommet*** du polyèdre $S$, s'ils n'existent pas $x_1,\ x_2\ \in S$, différents et un scalaire $\alpha$, $0 < \alpha < 1$ tels que : $x = \alpha x_1 + (1-\alpha)x_2$.

L'ensemble $S$ étant convexe et la fonction objectif dans $(PL)$ est continue , il y a trois possibilités mutuellement exclusives et collectivement exhaustives pour $(PL)$ :

1. Il existe au moins un sommet $\bar{x} \in S$ tel que : $c\bar{x} \geq cx$, $\forall x \in S$, dans ce cas $\bar{x}$ est une solution optimale pour $(PL)$. Soit $S^*$ l'ensemble de toutes les solutions optimales de $(PL)$, si $\bar{x}_1$ et $\bar{x}_2 \in S^*$, alors toute combinaison convexe de ces deux solutions est optimale pour $(PL)$ (une infinité) ;
2. Il n'existe aucune solution réalisable, i.e., le système de contraintes est incompatible, $S$ est vide ;
3. Le problème est non borné, $S$ est non borné.

Le cas $n$ř1 est le plus important d'un point de vue pratique, donc nous allons supposer dans ce qui suit que le domaine $S$ est borné et non vide.

Nous pouvons écrire le programme $(PL)$, sans perte de généralités de la manière suivante :

$$(PL)\begin{cases} Max \quad z = \left[c^B, c^N\right]\begin{bmatrix} x_B \\ x_N \end{bmatrix} \\ \left[A^B, A^N\right]\begin{bmatrix} x_B \\ x_N \end{bmatrix} = b \\ x_B \geq 0, \quad x_N \geq 0 \end{cases} \tag{2.3}$$

Où $B$ et $N$ sont appelés respectivement ensemble des indices de base et hors base. On appelle ***base*** un sous ensemble $B \subset \{1,\ldots,n\}$ d'indices de colonnes de $A$ tel que $A^B$ soit carrée non singulière. Le complémentaire de $B$ dans $\{1,\ldots,n\}$ est l'ensemble d'indices ***hors base*** associé noté $N$.

Les contraintes dans $(PL)$ peuvent êtres écrites sous la forme suivante :

$$A^B x_B + A^N x_N = b \tag{2.4}$$

Comme $det(A^B) \neq 0$, alors :

$$x_B = (A^B)^{-1}\left[b - A^N x_N\right] \tag{2.5}$$

La solution particulière de $(PL)$ : $x_B = (A^B)^{-1}b$, $x_N = 0$ est appelée ***solution de base*** .

Si de plus $x_B \geq 0$, la solution $(x_B, x_N)$ est appelée ***solution de base réalisable***.

Il est bien connu de l'algèbre linéaire qu'une solution de base réalisable correspond à un sommet du polyèdre convexe $S$. Notons qu'un sommet peut correspondre a plusieurs solutions de bases réalisables dégénérées. Deux bases sont appelées adjacentes si elles diffèrent seulement en une seule colonne.

Notez que le système d'équation $Ax = b$ peut ne pas avoir de solutions dans le cas général. Afin de s'assurer que des solutions de base existent, il est nécessaire de faire certaines hypothèses : (a) que $n > m$ ; (b) que les lignes de $A$ soient linéairement indépendantes ; (c) que le rang de $A$ est $m$, c'est à dire de rang plein. Ces conditions sont suffisantes pour l'existence d'au moins une solution de base.

### 2.2.2 Conditions d'optimalité

**Théorème 2.1.** *(théorème fondamental de la programmation linéaire)*
*Étant donné un programme linéaire écrit sous forme standard, avec une matrice A de rang m.*

*i) S'il admet une solution réalisable, il admet aussi une solution réalisable de base ;*

*ii) S'il admet une solution optimale, il admet une solution optimale de base ;*

*iii) S'il admet une solution réalisable et si la valeur de la fonction objectif est finie, il admet une solution optimale de base.*

Étant donnée une base réalisable $B$ du programme linéaire $(PL)$, $(PL)$ peut s'écrire comme suit :

$$(PL)\begin{cases} x_B + (A^B)^{-1}A^N x_N = (A^B)^{-1}b \\ \widehat{c}_N x_N = z - \pi b \\ x_B, x_N \geq 0 \end{cases}$$

Où :

- $\pi = c_B(A^B)^{-1}$ est dit *vecteur des potentiels relatif à la base* $B$.
- $\widehat{c} = c - \pi A$ est dit *vecteur des coûts réduits relatif à la base* $B$.

**Théorème 2.2.** *Si le vecteur coût $\widehat{c}$ relatif à une base réalisable B est négatif ou nul, la solution de base correspondante est solution optimale de $(PL)$. La base B est alors dite* ***base optimale****.*

Dans $(PL)$, il y au plus $C_m^n$ bases différentes, le problème de programmation linéaire peut être résolu de la façon suivante : pour une base donnée $B$, on vérifie si la solution correspondante est réalisable, c'est-à-dire si $x_B \geq 0$. Dans ce cas, nous calculons le coût réduit $\hat{c}$ et mettons à jour, si nécessaire, la valeur de la meilleure solution réalisable et passons à la base suivante. C'est le principe de l'algorithme du simplexe que nous allons détailler dans la section suivante.

### 2.2.3 Algorithme du simplexe

Nous avons vu à la section précédente, que si un problème de programmation linéaire sous forme standard a une solution optimale, alors il existe une solution de base réalisable qui est optimale. La méthode du simplexe est basée sur ce fait et recherche une solution optimale en passant d'une solution de base réalisable à une autre, le long des arêtes du domaine réalisable, toujours dans une direction d'augmentation de la fonction objectif (ou réduction). Finalement, une solution de base réalisable est atteinte à laquelle aucune des arêtes disponibles ne conduit à une augmentation de la fonction $z$ ; une telle solution de base est optimale et l'algorithme se termine. Dans cette section, nous présentons un développement détaillé de la méthode du simplexe.

Soit à résoudre le programme linéaire $(PL)$ suivant :

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

On définit l'application linéaire $col$ par :

$$\begin{array}{rcl} col : \{1,\ldots,m\} & \longrightarrow & \{1,\ldots,n\} \\ i & \longrightarrow & col(i) = \text{indice de la variable de base associée à la ligne } i. \end{array}$$

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

| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | $\cdots$ | 0 | $\widehat{A}^N = (A^B)^{-1}A^N$ | $\widehat{b} = (A^B)^{-1}b$ |
| 0 | $\ddots$ | | $\vdots$ | | |
| $\vdots$ | | $\ddots$ | 0 | | |
| 0 | $\cdots$ | 0 | 1 | | |
| 0 | $\cdots$ | 0 | 0 | $\widehat{c}_N = c_N - c_B(A^B)^{-1}A^N$ | $z - c_B(A^B)^{-1}b$ |

TABLE 2.1 – Écriture canonique de $(PL)/B$

**Etape 0** (Initialisation) : Trouver une solution de base réalisable $(x_B, x_N)$ avec $x_B \geq 0$ ; $x_N = 0$.

**Etape 1** (Test d'optimalité) : Si $\forall j \in N, \widehat{c}_j \leq 0$, alors **STOP** (la solution est optimale). Sinon aller à l'étape 2.

**Etape 2** (Choix de la variable entrante en base) : choisir une colonne $s$ telle que : $\widehat{c}_s = \underset{j \in N}{max}\, \widehat{c}_j$.

**Etape 3** (Choix de la variable sortante de la base) : Si $\forall i = \overline{1,m}, \widehat{A}_i^s < 0$, alors **STOP** (la fonction objectif n'est pas bornée), sinon, choisir une $r$, tel que :

$$\frac{\widehat{b}_r}{\widehat{A}_r^s} = \underset{i=\overline{1,m}}{min} \left\{ \frac{\widehat{b}_i}{\widehat{A}_i^s} | \widehat{A}_i^s > 0 \right\}$$

$$B = B \cup \{s\} \setminus col(r)$$

**Etape 4** (Pivotage) : Calculer une nouvelle solution de base réalisable. Soient $L_1, \ldots, L_m$ les $m$ premières lignes du tableau (TAB.2.1) correspondantes aux contraintes du problème et $L_{m+1}$ la $(m+1)$ ème ligne correspondante à la fonction objectif, alors les lignes du nouveau tableau sont calculées ainsi :

1. $L_i \leftarrow L_i - \dfrac{L_r \widehat{A}_i^s}{\widehat{A}_r^s} \quad i = \overline{1,m},\ i \neq r$
2. $L_{m+1} \leftarrow L_{m+1} - \dfrac{L_r \widehat{c}_s}{\widehat{A}_r^s}$
3. $L_r \leftarrow \dfrac{L_r}{\widehat{A}_r^s}$.

Aller à l'étape 1.

**Théorème 2.3.** *Si $S \neq \emptyset$ et que toutes les solutions de base réalisables ne sont pas dégénérées, alors l'algorithme du simplexe est fini, et une solution optimale est trouvée dans au plus $C_m^n$ itérations.*

### 2.2.4 Dualité

Pour chaque problème de programmation linéaire, il existe un problème associé, appelé programme linéaire *dual*, dans lequel les rôles des variables et des contraintes sont inversés. C'est-à-dire que pour chaque variable du programme linéaire original ou *primal*, il y a une contrainte dans le problème dual, et pour toute contrainte dans le primal, il y a une variable dans le dual.

Bien qu'il soit possible de définir un programme dual à n'importe quel programme linéaire, la symétrie des deux problèmes est plus évidente lorsque le programme linéaire est sous forme canonique. Nous nous référerons au programme $(P)$ comme étant le programme linéaire primal :

$$(P)\begin{cases} Max \quad z = c^t x \\ Ax \leq b \\ x \geq 0 \end{cases} \tag{2.6}$$

Le programme linéaire dual correspondant aura la forme :

$$(D)\begin{cases} Min \quad w = b^t y \\ A^t y \geq c \\ y \geq 0 \end{cases} \tag{2.7}$$

Cette définition possède les propriétés suivantes :

a) Elle est involutive : si l'on recherche le dual du problème $(D)$, on trouve le problème $(P)$ ; *le dual du dual est le primal* ;

b) Elle est caractérisée de façon simple :

- à toute contrainte représentée par une inéquation correspond une variable duale soumise à une contrainte de non négativité, et réciproquement ;
- à toute contrainte représentée par une équation correspond une variable duale de signe quelconque, et réciproquement ;
- la matrice de contraintes dans le dual est la transposée de la matrice dans le primal ;
- les seconds membres des contraintes de chacun des problèmes sont les coefficients de la fonction objectif de l'autre problème.

**Théorème 2.4.** *(Dualité faible) Soit x un point réalisable pour le problème primal sous forme standard, et soit y un point réalisable pour le problème dual. alors :*

$$z = c^t x \leq b^t y = w$$

**Corollaire 2.1.** *Si le primal est non borné, alors le dual n'est pas réalisable. Si le dual est non borné, alors le primal n'est pas réalisable.*

**Corollaire 2.2.** *Si $x$ est une solution réalisable pour le primal, $y$ est une solution réalisable pour le dual, et $c^t x = b^t y$, alors $x$ et $y$ sont optimaux pour leurs problèmes respectifs.*

**Théorème 2.5.** *(Dualité forte) Considérons une paire de programmes linéaires primal et dual. Si l'un des problèmes a une solution optimale, l'autre également, et les valeurs optimales des fonctions objectifs sont égales.*

Nous discutons ici d'une autre relation entre une paire de problèmes primal et dual qui ont des solutions optimales. Il existe une interdépendance entre les contraintes de non-négativité du primal ($x \geq 0$) et les contraintes dans le dual ($A^t y \geq c$). Aux solutions optimales des deux problèmes, il n'est pas possible d'avoir à la fois $x_j > 0$ et $(A^t y)_j > c_j$. Au moins une de ces contraintes doit être active : soit $x_j$ est nulle, soit la $j^{ième}$ variable duale est nulle. Cette propriété, appelée écarts complémentaires, peut être résumée par le théorème suivant :

**Théorème 2.6.** *(Théorème des écarts complémentaires) Soient $x$ et $y$ des solutions réalisables respectivement du primal et du dual. Une condition nécessaire et suffisante pour que $x$ et $y$ soient solutions optimales est qu'elles vérifient les relations suivantes :*

$$\begin{aligned} (c^t - y^t A)x &= 0 \\ y^t(Ax - b) &= 0 \end{aligned} \tag{2.8}$$

L'idée de l'algorithme dual du simplexe est résumée de la façon suivante : il part d'une solution de base "duale réalisable" mais irréalisable (pour le primal), et fait décroitre, de façon itérative, le nombre de variables négatives tout en maintenant à chaque itération les critères d'optimalité.

**Algorithme dual du simplexe**

Etape(0) Le problème est initialement sous forme canonique (les vecteurs $\hat{b}$ et $\hat{c}$ sont les écritures de $b$ et $c$ par rapport à la base de départ $B$, respectivement).

$\forall j, \hat{c}_j \leq 0$.

Etape(1) Si $\hat{b} \geq 0$, $\forall i = \overline{1,m}$, alors **STOP**, on est à l'optimum. Sinon, $\exists i$ t.q $\hat{b}_i < 0$.

Etape(2) Choisissez la ligne à pivoter (c'est-à-dire, variable qui quitte la base) par : $\hat{b}_r = \min_i\{\hat{b}_i | \hat{b}_i < 0\}$.

Si $\hat{a}_{rj} \geq 0$, $j = 1,2,...,n$, alors **STOP**; le problème primal n'est pas réalisable (dual non borné). Sinon $\exists \hat{a}_{rj} < 0$ pour certains indices $j$, $j = 1,...,n$.

**Etape(3)** Choisir la variable qui rentre en base :

$\frac{\hat{c}_s}{\hat{a}_{rs}} = Min_j\{\frac{\hat{c}_j}{\hat{a}_{rj}} | \hat{a}_{rj} < 0\}$.

**Etape(4)** Remplacez la variable de base de la ligne $r$ par la variable $s$ et rétablissez la forme canonique (c'est-à-dire, pivotez sur le coefficient $\hat{a}_{rs}$).

**Etape(5)**Aller à l'étape (1).

### 2.2.5 Dégénérescence

**Définition 2.1.** (Solution de base dégénérée)

- Une solution de base $x_B$ est dite primale dégénérée lorsque une ou plusieurs composantes de $x_B$ sont nulles.
- Une solution de base $x_B$ est dite dual dégénérée lorsque une ou plusieurs composantes de $c_N - \pi A^N$ sont nulles.
- Une solution de base $x_B$ est dite dégénérée si elle est primal ou dual dégénérée.

Ce phénomène peut se manifester pour le programme $(P)$ tout comme pour son dual, où il s'énonce comme suit : le vecteur de coût réduit $c_N - \pi A^N$ associé à une solution réalisable de base $x_B$, possède une composante nulle.

La version de la méthode du simplexe que nous avons décrite jusque là peut échouer et un phénomène de cyclage peut se produire sans amélioration de la valeur de la fonction objectif.

Supposons qu'à une itération donné, $x_s$ est la variable qui rentre en base et $x_r$ est la variable qui quitte la base.

$\hat{x}_s = \frac{\hat{b}_r}{\hat{A}^s_r}$ et $\bar{z} = \hat{z} + \hat{c}_r\hat{x}_r$

Où : $\bar{z}$ est la nouvelle valeur de la fonction objectif.

Dans un problème dégénéré, il se peut que $\hat{b}_r = 0$ et $\hat{x}_s = 0$, ainsi la valeur de la fonction objectif ne change pas.

Plusieurs techniques ont été développées garantissant la finitude de la méthode du simplexe même pour des problèmes dégénérés. L'une des méthodes a été proposée par Bland [22].

**Définition 2.2.** (Règle de Bland) La variable entrante est la première pour laquelle $\hat{c}_j > 0$ . La variable sortante est la première à atteindre tel que :

$$\frac{\widehat{b}_r}{\widehat{A}_r^s} = \underset{i=\overline{1,m}}{min} \left\{ \frac{\widehat{b}_i}{\widehat{A}_i^s} | \widehat{A}_i^s > 0 \right\}$$

$$B = B \cup \{s\} \setminus col(r)$$

Une autre façon de résoudre la dégénérescence dans la méthode du simplexe consiste à introduire de petites perturbations dans le vecteur $b$. Ces perturbations éliminent la dégénérescence, de sorte que la méthode progresse à chaque itération.

## 2.3 Programmation linéaire en nombres entiers [1, 2, 4]

Le but de la présente section est d'introduire les définitions de base de la programmation linéaire en nombres entiers (PLNE), ses propriétés ainsi que ses différentes applications. Nous présentons deux idées algorithmiques pour les résoudre, les méthodes de coupes et la méthode par "branch & bound".

Un problème de programmation linéaire en nombres entiers ($PLNE$) est un problème de programmation linéaire dans lequel toutes les variables de décisions sont soumises à prendre des valeurs entières. Si une partie seulement des variables est soumise à cette contrainte d'intégrité, il s'agit alors d'un problème de programmation linéaire en variables ***mixtes*** (PLM).

Les restrictions d'intégrité sur les variables de décision se présentent dans de nombreuses situations. Par exemple, dans le modèle linéaire de production [23], où nous devons produire des produits de $n$ types différents $P_1, \cdots, P_n$ , La production repose sur des matières provenant de différentes ressources $R_1, \cdots, R_m$ de telle

sorte que la production d'une unité d'un produit de type $P_j$ nécessite $a_{ij}$ unités de ressource $R_i$, pour $i = 1, \cdots, m$. Combien d'unités $x_j$ de chaque produit $P_j$ devrions-nous produire afin de maximiser le total des recettes provenant des ventes? En effet, les produits peuvent êtres des biens indivisibles qui ne peuvent être produits que dans de multiples de l'unité.

### 2.3.1 Formulation mathématique et quelques propriétés

Dans ce qui suit, nous présentons les concepts de base qui seront utiles pour l'étude des problèmes de programmation linéaire en entiers :

**Définition 2.3.** (Problème de programmation linéaire mixte (PLM))
Un problème de programmation linéaire mixte (PLM) est donné par les vecteurs $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, la matrice $A \in \mathbb{R}^{m \times n}$ et un nombre $p \in \{0, \cdots, n\}$. Le but est de trouver un vecteur $x \in \mathbb{R}^n$ résolvant le problème d'optimisation suivant :

$$(PLM) \begin{cases} Max \quad z = c^t x \\ Ax = b \\ x \geq 0 \\ x \in \mathbb{Z}^p \times \mathbb{R}^{n-p} \end{cases} \tag{2.9}$$

Si $p = n$, alors toutes les variables doivent être entières. Dans ce cas, nous parlons d'un programme linéaire en nombre entiers ($PLNE$) :

$$(PLNE) \begin{cases} Max \quad z = c^t x \\ Ax = b \\ x \geq 0 \\ x \in \mathbb{Z}^n \end{cases} \tag{2.10}$$

Si dans $PLNE$, toutes les variables sont limitées à prendre des valeurs dans l'ensemble $\mathbb{B} = \{0, 1\}$, nous avons un programme linéaire en $0-1$, ces variables sont souvent utilisées pour représenter des décisions :

$$(PLB) \begin{cases} Max \quad z = c^t x \\ Ax = b \\ x \geq 0 \\ x \in \mathbb{B}^n \end{cases} \tag{2.11}$$

Les applications de $PLB$ sont nombreuses, nous citons : le problème d'affectation, le problème du voyageur de commerce, le problème de recouvrement minimum, le problème de set-packing, le problème de partitionnement et le problème de l'arbre de poids minimum.

### 2.3.2 Méthodes de résolution d'un ($PLNE$)

Contrairement aux problèmes de programmation linéaire, les problèmes de programmation linéaire en nombres entiers sont très difficiles à résoudre. En fait, aucun algorithme général efficace n'est connu pour leur résolution.

Dans cette section, nous passons en revue les algorithmes de résolution de ($PLNE$). En fait, il existe deux catégories principales d'algorithmes :

a) Les algorithmes exacts qui sont garantis pour trouver une solution optimale, mais qui peuvent prendre un nombre exponentiel d'itérations, ils comprennent les méthodes de coupes planes, "branch & bound", "branch & cut" et la programmation dynamique.

b) Ce sont des algorithmes de recherche de solutions de bonne qualité, sans garantie d'optimalité, mais au profit d'un temps de calcul plus réduit.

Un bon nombres de méthodes de résolution des programmes linéaires en nombres entiers ont été proposé dans la littérature : methodes de coupes planes, méthodes "branch & bound", "branch & cut", etc. Ces méthodes seront détaillées et feront l'objet des sections suivantes [24], [25], [26] et [27] :

#### 2.3.2.1 Méthode de Coupes Planes

L'idée principale dans les méthodes de coupes planes est de résoudre le problème de programmation linéaire en nombres entiers en résolvant une séquence de problèmes de programmation linéaire comme suit : nous résolvons d'abord la relaxation du programme ($PLNE$), et trouvons la solution optimale $x^*$. Si $x^*$ est entière, alors c'est une solution optimale pour ($PLNE$). Sinon, une inégalité est construite, satisfaite par toutes les solutions entières de ($PLNE$), mais ne l'est pas par la solution $x^*$.

Nous ajoutons cette inégalité au problème de programmation linéaire ($PL$) pour obtenir une relaxation plus serrée, et nous itérons cette étape jusqu'à trouver la solution entière optimale pour ($PLNE$).

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

$$S = \{x \mid Ax = b,\ x \geq 0\} \tag{2.12}$$

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

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

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

$$\begin{aligned} S &= S_1 \supset S_2 \supset ...S_k \supset S_{k+1} \supset ...conv(D) \\ S_{k+1} &= S_k \cap \{x \mid \alpha_k x \leq \beta_k\} \end{aligned} \tag{2.13}$$

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

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

Généralement, les inégalités valides sont de deux types. Il y a celles portant sur des structures provenant de l'étude de problèmes combinatoires spécifiques, typiquement dérivées des techniques polyédrales, et il y a celles qui sont générales, dérivées de techniques algébriques. Dans cette deuxième classe nous trouvons, par

exemple, les coupes fractionnaires et les coupes mixtes de Gomory exposées dans les années soixante [25] et les coupes de Chvátal-Gomory de Chvátal [24].

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

### Coupes de Gomory

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

Soit $\overline{x}$ une solution optimale du problème relaxé ($PL$) trouvée par la méthode du simplexe. Si $\overline{x}$ est une solution entière, elle est optimale pour ($PLNE$). Sinon, choisir une variable $x_j$ telle que la valeur $\overline{x}_j$ est fractionnaire et considérer la ligne correspondante du tableau du simplexe, par exemple la ligne $i$ :

$$x_j + \sum_{k \in N} \widehat{a}_{ik} x_k = \overline{x}_j \tag{2.14}$$

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

La contrainte

$$\sum_{k \in N} f(\widehat{a}_{ik}) x_k \geq f(\overline{x}_j) \tag{2.15}$$

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

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

Dans la pratique, les coupes fractionnaires de Gomory convergent très lentement. D'autres coupes ont été introduites dont l'intention de les rendre plus performantes. En particulier, Gomory [25] lui même a donné les coupes suivantes :

**Proposition 2.1.** *Pour tout entier naturel t, l'inéquation*

$$\sum_{k \in N} f(t\widehat{a}_{ik}) x_k \geq f(t\overline{x}_j) \tag{2.16}$$

*est une coupe. De plus, si* $f(\widehat{a}_{ik}) < \frac{1}{2}$ *et* $\frac{1}{2} \leq tf(\widehat{a}_{ik}) < 1$*, alors la coupe 2.16 est plus profonde que la coupe 2.15.*

**Proposition 2.2.** *L'inéquation*

$$\sum_{k \in N} \min \left\{ f(\widehat{a}_{ik}), f(\overline{x}_j) \frac{1 - f(\widehat{a}_{ik})}{1 - f(\overline{x}_j)} \right\} x_k \geq f(\overline{x}_j) \tag{2.17}$$

*est une coupe. De plus, elle est plus profonde que la coupe 2.15.*

*De même, pour tout entier naturel t, l'inéquation*

$$\sum_{k\in N} \min\left\{f(t\widehat{a}_{ik}), f(t\overline{x}_j)\frac{1-f(t\widehat{a}_{ik})}{1-f(t\overline{x}_j)}\right\} x_k \geq f(t\overline{x}_j) \tag{2.18}$$

*est une coupe plus profonde que 2.16.*

### Coupes de Chvátal-Gomory

Chvátal considère un programme linéaire en nombres entiers

$$(P_1)\begin{cases} \max z = c^t x \\ Ax \leq b \\ x \geq 0,\ x \text{ vecteur entier} \end{cases} \tag{2.19}$$

avec des contraintes d'inégalités.

**Proposition 2.3.** *Pour tout vecteur $\lambda \in \mathbb{R}^m_+$, l'inégalité suivante est une coupe pour $(P_1)$ :*

$$\lfloor \lambda^t A \rfloor x \leq \lfloor \lambda^t b \rfloor \tag{2.20}$$

Il a été démontré que cette dernière coupe est équivalente à la coupe de Gomory [25]. C'est pour cette raison qu'on l'appelle coupe de Chvátal-Gomory.

### Coupe disjonctives

Le concept du problème de programmation disjonctive a été introduit par Balas [28].

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

$$Q_i = \{x \in \mathbb{R}^n \mid A_i x \leq b_i\},\ \ i = 0,1 \tag{2.21}$$

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

**Coupes de "split"**

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

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

$$\begin{cases} \max c^t x \\ x \in R \end{cases} \tag{2.22}$$

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

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

$$\begin{aligned} Q_0 &= R \cap \{x \in \mathbb{R}^n \mid \pi^t x \leq \pi_0\} \\ Q_1 &= R \cap \{x \in \mathbb{R}^n \mid \pi^t x \geq \pi_0 + 1\} \end{aligned} \tag{2.23}$$

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

$$x_i \leq \lfloor \widehat{x}_i \rfloor \text{ ou } x_i \geq \lceil \widehat{x}_i \rceil \tag{2.24}$$

**Coupes de "lift and project"**

Ce sont des coupes particulières des coupes de "split" pour les programmes linéaires à variables bivalentes ou mixtes [26], [29].

Soit le programme linéaire à variables bivalentes :

$$\begin{cases} \max\ Z = c^t x \\ Ax \leq b \\ x_j \in \{0,1\},\ j = 1, \ldots, n \end{cases} \tag{2.25}$$

- Résoudre le problème relaxé suivant :

$$\begin{cases} \max\ Z = c^t x \\ A'x \leq b' \end{cases} \tag{2.26}$$

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

- Considérer le problème disjonctif :

$$\begin{cases} A'x \leq b' \\ x_i = 0 \end{cases} \text{ ou } \begin{cases} A'x \leq b' \\ x_i = 1 \end{cases} \tag{2.27}$$

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

$$S_i = conv(\{A'x \leq b',\ x_i = 0\} \cup \{A'x \leq b',\ x_i = 1\}) \tag{2.28}$$

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

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

$$\begin{cases} \alpha = u^1 A' + v^1 e_i, \qquad \alpha = u^2 A' + v^2 e_i \\ \beta \leq u^1 A' + v^1 0, \qquad \beta \leq u^2 A' + v^2 \\ u^1,\ u^2 \geq 0 \\ \alpha,\ \beta,\ v^1,\ v^2 \ \textit{réels} \end{cases} \tag{2.29}$$

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

$$\begin{cases} \max \beta - \alpha \widehat{x} \\ \alpha = u^1 A' + v^1 e_i, \qquad \alpha = u^2 A' + v^2 e_i \\ \beta \leq u^1 A' + v^1 0, \qquad \beta \leq u^2 A' + v^2 \\ u^1,\ u^2 \geq 0 \\ \alpha,\ \beta,\ v^1,\ v^2 \ \textit{réels} \end{cases} \tag{2.30}$$

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

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

#### 2.3.2.2 Méthode Branch & Bound

La méthode par séparation et évaluation (*Branch* & *Bound*) est très efficace pour la résolution des problèmes de programmation linéaire en nombres entiers. Elle a été à l'origine développée par Land et Doig [30] pour résoudre un problème de programmation linéaire en nombres entiers et a été modifiée plus tard par Dakin [31].

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

La méthode par séparation et évaluation peut être considérée comme une méthode d'énumération raffinée dans laquelle plusieurs points entiers réalisables sont écartés sans les évaluer. Son principe repose sur trois notions distinctes : séparation du problème principal, relaxation des sous problèmes et stérilisation de l'arbre de recherche.

1. **Séparation :** Considérons le problème de programmation linéaire en nombres entiers $(Q)$ dont l'espace des solutions réalisables est noté $D(Q)$.
   L'ensemble de sous problèmes $(Q_1),(Q_2),\ldots,(Q_n)$ est une séparation de $(Q)$ si :

$$\bigcup_i D(Q_i) = D(Q) \; et \; \forall i, j \text{ avec } i \neq j, \; D(Q_i) \cap D(Q_j) = \emptyset.$$

2. **Relaxation :** Le domaine $S$ des solutions réalisables de la relaxation $(P)$ du problème $(Q)$ contient celui de $(Q)$ ce qui implique que :
   a) Si $(P)$ n'est pas réalisable, alors $(Q)$ ne l'est pas aussi ;
   b) La solution optimale de $(P)$ est une borne supérieure de la solution optimale de $(Q)$ ;
   c) Une solution optimale de $(P)$ réalisable pour $(Q)$ est une solution optimale de $(Q)$.

3. **Stérilisation :** Soit $(Q_k)$ un sous problème de $(Q)$ susceptible de mener à la solution optimale de $(Q)$.

   Trois critères peuvent autoriser à stériliser $(Q_k)$ :

   a) La relaxation de $(Q_k)$ n'est pas réalisable ;

   b) L'évaluation de la solution optimale de la relaxation de $(Q_k)$ est inférieure à celle de la meilleure solution réalisable trouvée antérieurement (soit $U$ son évaluation) ;

   c) la solution optimale de la relaxation de $(Q_k)$ est réalisable, donc optimale pour $(Q_k)$. Si en plus son évaluation est supérieure à $U$, alors cette dernière peut être actualisée.

Un problème de programmation linéaire en nombres entiers $(Q)$ peut avoir plusieurs solutions optimales, on parle dans ce cas de solutions alternatives dont la définition est la suivante :

**Définition 2.4.** Soit $x^*$ une solution optimale du problème $(Q)$. Une solution réalisable $x' \in D$ est dite alternative à $x^*$ si $cx' = cx^*$.

#### 2.3.2.3 Méthode "Branch & Cut"

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

## 2.4 Conclusion

Ce chapitre avait pour objectif de réaliser une étude détaillée de la programmation linéaire en nombres entiers. Ces outils sont étendus dans le chapitre suivant à la compréhension de la programmation multiobjectif en nombres entiers en passant en revue l'important de littérature dans ce domaine.

CHAPITRE 3

# OPTIMISATION MULTIOBJECTIF DISCRÈTE

## 3.1 Introduction

L'optimisation en nombres entiers est un domaine largement étudié par de nombreux chercheurs. Son importance provient du fait qu'elle peut être utilisée pour résoudre un large éventail de problèmes provenant du monde réel. Un bon aperçu de l'état de la technique est fourni par [32].

Cependant, les problèmes d'optimisation issus de la réalité sont la plupart du temps de nature multiobjectif car, plusieurs critères souvent contradictoires sont à considérer simultanément.

Contrairement à l'optimisation mono objectif, la solution d'un problème multiobjectif n'est pas unique, mais un ensemble de solutions, connu comme l'ensemble des solutions efficaces ou solution Pareto optimales. Toute solution de cet ensemble est optimale dans le sens qu'aucune amélioration ne peut être faite sur un critère sans dégradation d'au moins un autre critère.

Dans cette optique, nombreuses stratégies ont été proposées, consistant à caractériser totalement ou partiellement l'ensemble des solutions efficaces du problème de programmation linéaire multiobjectif en nombres entiers.

Les problème d'optimisation combinatoire multiobjectif peuvent être résolus à l'aide des méthodes proposées pour le cas continu, en l'occurrence (méthode de la

somme pondérée [33], $e - contrainte$[34], but programmé [35], etc., avec quelques limitations sur les capacités de ces méthodes à construire entièrement le front de Pareto. En effet, l'introduction de variables discrètes dans les problèmes de programmation multiobjectifs conduit à des problèmes plus difficiles à aborder, même si les fonctions objectifs et les contraintes sont linéaires. En fait, l'ensemble des solutions réalisables n'est plus convexe, et les difficultés vont au-delà de celles du passage de la programmation monoobjectif continue à la programmation monoobjectif discrète. L'une des principales limites de ces techniques réside dans l'impossibilité d'obtenir certaines solutions efficaces. Par conséquent, certaines solutions efficaces sont plus faciles à calculer que d'autres : ce sont les solutions supportées, qui optimisent une combinaison linéaire des objectifs. Les autres solutions efficaces sont dites non-supportées.

Ce chapitre a pour objectif principal de présenter le contexte de l'optimisation multiobjectif discrète. Nous introduisons, les notions fondamentales ainsi que les principaux résultats liées à la théorie de la programmation multiobjectif en nombres entiers. Ensuite, nous présentons un exemple illustratif des différentes définitions. Ultérieurement, nous allons donner une capture de quelques développements récents en optimisation multiobjectif linéaire en nombres entiers. Enfin, nous parlons du problème d'optimisation sur l'ensemble efficient d'un programme MOILP.

## 3.2 Programmation multiobjectif discrète ([5],[6], [7])

Dans les sections précédentes, nous n'avons considéré que les cas où le problème à traiter possédait un objectif unique à optimiser. Pour de nombreuses applications pratiques, ce n'est en fait pas le cas et il faut optimiser en même temps plusieurs objectifs pour un même problème. La fonction objectif n'est alors plus scalaire mais vectorielle, on parle dans ce cas de problème multiobjectif.

### 3.2.1 Formulation Générale

Le problème de programmation multiobjectif en nombres entiers est défini comme un problème de décision, qui consiste à optimiser (maximiser ou minimiser) simultanément $r$ fonctions à valeurs réelles notées $f^i$, $i = \overline{1,r}$ sur un domaine définit par le système de contraintes $g_j(x) \leq 0$ où $g_j$ est une fonction réelle définit sur $\mathbb{R}$, $j = \overline{1,m}$. Ce problème peut être formulé comme suit :

$$(MOIP) \begin{cases} "Optimiser" \quad f^i(x) \quad i = \overline{1,r} \\ g_j(x) \leq 0 \quad j = \overline{1,m} \\ x \in \mathbb{Z}^n \end{cases}$$

Le symbole " " signifie qu'il n'est généralement pas possible de trouver un vecteur $x \in \mathbb{Z}^n$ vérifiant les différentes contraintes et qui optimise simultanément les $r$ critères.

**Remarque 3.1.** *Sans perte de généralités, nous supposons dans la suite que toutes les fonctions objectif sont à maximiser.*

Si les $r$ critères sont linéaires et les contraintes sont linéaires en $x$, on parle de problème de programmation linéaire multiobjectif en nombres entiers (MOILP) :

$$(MOILP) \begin{cases} Max \quad z^i = c^i x \quad i = \overline{1,r} \\ x \in D \end{cases}$$

Où :

- $r$ le nombre d'objectifs ($r \geq 2$).
- $D = \{x \in \mathbb{Z}^n | Ax = b, x \geq 0\}$.

Avec : $A \in \mathbb{Z}^{m \times n}$, $b \in \mathbb{Z}^m$, $C = (c^1, c^2, \ldots, c^r)^t \in \mathbb{Z}^{r \times n}$.

### 3.2.2 Concepts de base

Soit le problème de programmation linéaire en nombres entiers à objectifs multiples suivant :

$$(P) \begin{cases} "Max" \quad z^i = c^i x \quad i = \overline{1,r} \\ x \in D \end{cases}$$

Où : $D = \{x \in \mathbb{Z}^n | Ax = b, x \geq 0\}$ est supposé borné et non vide.
On considère l'application linéaire $\varphi$ qui associe à chaque vecteur $x \in D$ son image $\varphi(x) = (c^1x, c^2x, \ldots, c^rx)$ dans l'espace des critères.

**Définition 3.1.** L'espace $\mathbb{Z}^n$ dans lequel se situe l'ensemble des actions $D$ est appelé *espace des décisions*. D'autre part, et dans le cadre multiobjetif, le décideur raisonne plutôt en terme d'évaluation d'une solution pour chaque objectif et se place naturellement dans l'espace :

$$\varphi(D) = \left\{z = Cx \in \mathbb{R}^r | x \in D\right\}$$

appelé *espace des critères*.

La résolution d'un problème d'optimisation multiobjectif conduit à l'obtention d'une multitude de solutions dites efficaces. En effet, les solutions n'admettent pas une relation d'ordre total mais une relation d'ordre partiel, car une solution pouvant être meilleure qu'une autre sur certains objectifs et moins bonne sur d'autres. Dans ce cas, la notion de dominance est introduite et est définit comme suit :

#### 3.2.2.1 Dominance

Soient $z$, $z'$ deux vecteurs de $\mathbb{R}^r$. On dit que $z$ *domine* $z'$ ssi :
$z \geq z'$ et $z \neq z'$ (i.e. $z^i \geq z'^i \ \forall i = \overline{1,r}$ et $z^i > z'^i$ pour au moins un $i$).
Autrement dit, $z$ domine $z'$ si et seulement si :

- $z$ est au moins aussi bon que $z'$ pour tous les critères et,
- $z$ est strictement meilleur que $z'$ pour au moins un critère.

#### 3.2.2.2 Efficacité

Une solution $x^*$ est dite *efficace* si et seulement s'il n'existe pas de $x \in D$ telle que :

$$Cx \geq Cx^*$$

avec au moins une inégalité stricte.
Autrement dit, une solution $x^*$ est efficace si et seulement si le vecteur critère qui lui est associé n'est dominé par aucun autre vecteur.

On note par $ESE(P)$ l'ensemble des solutions efficaces d'un problème de programmation linéaire multiobjectif en nombres entiers et par $SND(P)$ celui des solutions non dominées.

#### 3.2.2.3 Efficacité faible

Une solution $x^*$ est dite *faiblement efficace* si et seulement s'il n'existe pas de $x \in D$ telle que : $Cx > Cx^*$.

Parmi les points particuliers qui apparaissent dans l'espace des critères, on trouve : le point *idéal*, le point *nadir*, etc.

#### 3.2.2.4 Point idéal

Les coordonnées de ce point sont obtenues en optimisant chaque fonction objectif séparément : $\overline{z} = \left( \underset{x \in D}{Max}\, c^1 x, \underset{x \in D}{Max}\, c^2 x, \ldots, \underset{x \in D}{Max}\, c^r x \right) \in \mathbb{R}^r$.
Généralement, parce que les objectifs sont contradictoires, le point idéal ne correspond pas à une solution réalisable.

#### 3.2.2.5 Matrice des gains et point nadir

Soit $\overline{x_i}$ la solution optimale obtenue en optimisant le critère $z^i$ sur $D$. La matrice carrée de dimension $r$ suivante :

$$\begin{pmatrix} \overline{z_1} & z_{12} & \cdots & z_{1r} \\ z_{21} & \overline{z_2} & \cdots & z_{2r} \\ \vdots & \vdots & \vdots & \vdots \\ z_{r1} & z_{r2} & \cdots & \overline{z_r} \end{pmatrix}$$

est appelée *matrice des gains*.
Où :

- $\overline{z_i} = \underset{x \in D}{Max}\, c^i x = c^i \overline{x_i}$, $\forall i = \overline{1,r}$.
- $z_{ij} = c^i \overline{x_j}\ \forall i = \overline{1,r}$, $\forall j = \overline{1,r}$ avec $i \neq j$.

**Remarque 3.2.** *Si pour un critère j, il existe plusieurs solutions optimales, la matrice des gains n'est pas unique. En effet, la colonne j de la matrice des gains dépen-*

*dra de la solution $\overline{x_j}$ choisie. Dans ce cas, les coordonnées du point idéal ne changent pas d'une solution à une autre.*

**Définition 3.2.** (Point nadir) On définit le point nadir noté $\eta \in \mathbb{R}^r$ :

$$\eta_i = \underset{i=\overline{1,r}}{Min} \left\{ c^i x | x \ \ efficace \right\}$$

#### 3.2.2.6 Solutions supportées et non supportées

L'ensemble des solutions efficaces pour les problèmes de programmation linéaire multiobjectif en variables continues :

$$(MOLP) \begin{cases} "Max" \quad z^i = c^i x \quad i = \overline{1,r} \\ x \in S \end{cases}$$

où $S = \{x \in \mathbb{R}^n | Ax = b, x \geq 0\}$ est bien caractérisé. En effet, pour toute solution efficace $\overline{x}$ du problème $(MOLP)$, $\exists \overline{\lambda} > 0$ tel que $\overline{x}$ soit optimale pour le problème paramétrique :

$$(P_\lambda) \begin{cases} Max \quad \sum_{i=1}^{r} \lambda_i c^i x \\ x \in S \end{cases}$$

$\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_r) \in \mathbb{R}^r$ avec, $\lambda_i \geq 0 \ \forall i = \overline{1,r}$ et $\sum_{i=1}^{r} \lambda_i = 1$, et réciproquement (c'est à dire : si pour une certaine valeur de $\lambda$, $\overline{x}$ est une solution optimale de $(P_\lambda)$, alors $\overline{x}$ est une solution efficace du problème $(MOLP)$).

**Théorème 3.1** (Geoffrion [33])**.** *Étant donné le problème $(P_\lambda)$, alors $\overline{x}$ est une solution efficace pour $(MOLP)$ si et seulement si $\overline{x}$ est une solution optimale du problème paramétrique $(P_\lambda)$.*

Dans le cas des problèmes $(MOILP)$, la difficulté principale rencontrée différemment des problème $(MOLP)$, est l'existence de solutions qui ne sont pas optimales pour $(P_\lambda)$ (en remplaçant $S$ par $D$), et ce en raison de la non convexité du domaine des solutions réalisables $D$, ces solutions sont dites *solutions efficaces non supportées* notées ***SENS***. Celles qui sont optimales pour $(P_\lambda)$, sont appelées solutions efficaces supportées notées ***SES***.

#### 3.2.2.7 Cônes

**Définition 3.3 (Cône).** Soit $u \in U \subset \mathbb{R}^n$, $U \neq \emptyset$. Alors, $U$ est un *cône* si et seulement si $\alpha u \in U$ pour tout scalaire $\alpha \geq 0$. L'origine $0 \in \mathbb{R}^n$ est contenu dans chaque cône.

Excepté de l'ensemble singleton qui contient uniquement l'origine, tous les cônes sont non bornés. Comme exemple, le demi espace fermé $\{x \in \mathbb{R}^n | c^t x \leq 0\}$ est un cône mais le demi espace ouvert $\{x \in \mathbb{R}^n | c^t x > 0\}$ n'est pas un cône car il ne contient pas l'origine.

**Définition 3.4 (Générateurs).** Considérons $\{u^1, u^2, \ldots, u^k\}$, un ensemble de $k$ vecteurs de $\mathbb{R}^n$ et l'ensemble $U$ tel que :

$$U = \left\{ u \in \mathbb{R}^n | u = \sum_{i=1}^{k} \alpha_i u^i, \, \alpha_i > 0 \right\}$$

U est l'ensemble de toutes les combinaisons linéaires à coefficients non négatifs des $u^i$, $i = \overline{1,k}$ et est le cône convexe engendré par l'ensemble $\{u^1, u^2, \ldots, u^k\}$. Les vecteurs $u^i$, $i = \overline{1,k}$ sont appelés les *générateurs* de $U$.

Le seul cône pour lequel l'ensemble des générateurs est unique est $\{0 \in \mathbb{R}^n\}$.
Soit $\{u^1, u^2, \ldots, u^k\}$ un ensemble de générateurs pour le cône convexe $U$ et soit $u^l \in \{u^1, u^2, \ldots, u^k\}$. Alors, $u^l$ est *générateur non essentiel* si $U$ peut être généré par $\{u^1, u^2, \ldots, u^k\} \setminus \{u^l\}$. Un générateur non essentiel est celui qui peut être exprimé comme combinaison linéaire d'autres générateurs, il est dit *essentiel* sinon.

**Définition 3.5 (Dimension d'un cône).** La *dimension* d'un cône $U \subset \mathbb{R}^n$ est donné par le nombre de vecteurs linéairement indépendants de $U$.

Par exemple, la dimension du cône singleton $\{0 \in \mathbb{R}^n\}$ est 0, et la dimension d'un cône convexe généré par un ensemble de $k$ vecteurs linéairement indépendants est $k$. On peut déterminer la dimension d'un cône en calculant le rang de la matrice dont les lignes (ou les colonnes) sont les générateurs de ce cône.

**Définition 3.6 (Cône polaire).** Soit $U \subset \mathbb{R}^n$ un cône. Alors, le *cône polaire* non négatif de $U$ (noté $U^{\geq}$) est le cône convexe :

$$U^{\geq} = \{ y \in \mathbb{R}^n | y^t u \geq 0 \, \text{pour tout} \, u \in U \}$$

C'est-à-dire, tous les vecteurs de $U^{\geq}$ font un angle inférieur ou égal à $90^0$ avec chaque vecteur de $U$.

**Définition 3.7 (Cône polaire semi positif).** Soit $U \subset \mathbb{R}^n$ un cône généré par l'ensemble $\{u^1, u^2, \ldots, u^k\}$. Alors, le *cône polaire semi positif* de $U$ noté $U^>$ est le cône convexe :

$$U^> = \left\{ y \in \mathbb{R}^n | y^t u^i \geq 0 \text{ pour tout } i \text{ et } y^t u^i > 0 \text{ pour au moins un } i \right\} \cup \{0 \in \mathbb{R}^n\}$$

Notons qu'un vecteur $y \in U^>$ doit avoir un produit vectorielle positif avec au moins l'un des $u^i$, $i = \overline{1,k}$. L'origine $\{0 \in \mathbb{R}^n\}$ est incluse car sinon $U^>$ ne serait pas un cône.

#### 3.2.2.8 Détection graphique de l'efficacité

Soit le problème de programmation linéaire multiobjectif en nombres entiers suivant :

$$(P) \begin{cases} \text{"}Max\text{"} \quad z^i = c^i x \quad i = \overline{1,r} \\ x \in D \end{cases}$$

Où : $D = \{x \in \mathbb{Z}^n | Ax = b,\ x \geq 0\}$ avec : $A \in \mathbb{Z}^{m \times n}$, $b \in \mathbb{Z}^m$ et $C = (c^i)_{i=\overline{1,r}} \in \mathbb{Z}^{r \times n}$.
Pour tester l'efficacité en un point $x^* \in D$, Ralph E. Steuer [5] a introduit le concept d'*ensemble dominant* qui est principalement basé sur la notion du cône étudiée précédemment.

**Définition 3.8 (Ensemble dominant).** Soit $x^* \in D$ et $C^>$ le cône polaire semi positif du cône $C$ généré par les gradients des $r$ fonctions objectifs i.e.,

$$C^> = \left\{ y \in \mathbb{R}^n | y^t c^i \geq 0 \text{ pour tout } i \text{ et } y^t (c^i)^t > 0 \text{ pour au moins un } i \right\} \cup \{0 \in \mathbb{R}^n\}$$

On définit l'ensemble dominant noté $ED_{x^*}$ , comme étant la somme des ensembles $\{x^*\}$ et $C^>$ :

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

C'est à dire :

$$ED_{x^*} = \{x \in \mathbb{R}^n | x = x^* + y,\ Cy \geq 0,\ Cy \neq 0\}$$

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

**Théorème 3.2** ([5]). *Soit $ED_{x^*}$ l'ensemble dominant en $x^* \in D$. Alors $x^*$ est efficace si et seulement si : $ED_{x^*} \cap D = \{x^*\}$.*

***Démonstration.*** $\Rightarrow$ / Supposons que $ED_{x^*} \cap D \neq \{x^*\}$. Alors, il existe $\overline{x} \in ED_{x^*} \cap D, \overline{x} \neq x^*$. Puisque $\overline{x} \in ED_{x^*}$, alors $\overline{x} = x^* + y$ où $y \in C^>$. Comme $Cy \geq 0, Cy \neq 0$ alors $C\overline{x} \geq Cx^*, C\overline{x} \neq Cx^*$. Ceci contredit le fait que $x^*$ est efficace. Alors si $x^*$ est efficace, $ED_{x^*} \cap D = \{x^*\}$.
$\Leftarrow$ / Si $ED_{x^*} \cap D = \{x^*\}$, ceci implique que si le vecteur critère de $\overline{x}$ domine le vecteur critère de $x^*$, $\overline{x} \notin D$ alors le vecteur critère de $x^*$ est non dominé, et par conséquent, $x^*$ est efficace. ■

Le théorème (1.2) fournit un test permettant de détecter les points efficaces qui peuvent être visualisés géométriquement :

- Si l'intersection de l'ensemble dominant avec la région réalisable contient seulement $x^*$, alors $x^*$ est efficace.
- S'il existe d'autres points appartenant à l'intersection de ces deux ensembles, alors $x^*$ n'est pas efficace.

### 3.2.3 Illustration des définitions

Considérons le programme linéaire biobjectif en nombres entiers suivant :

$$(P) \begin{cases} "Max" \quad (x_1, -x_1 + x_2) \\ x_1 + 2x_2 \leq 10 \\ x_1 + x_2 \leq 6 \\ x_1 \leq 4 \\ x_1, x_2 \in \mathbb{Z}_+ \end{cases}$$

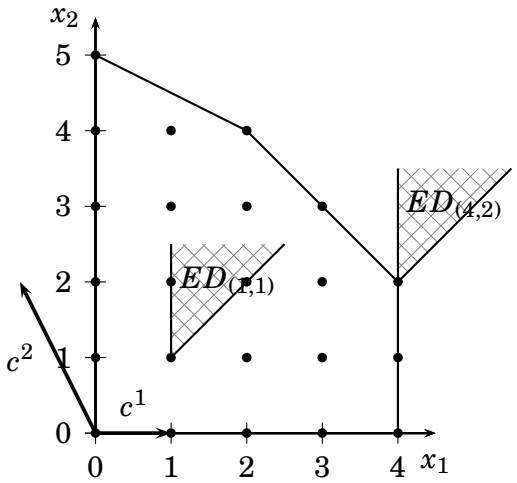


FIGURE 3.1 – Espace des décisions

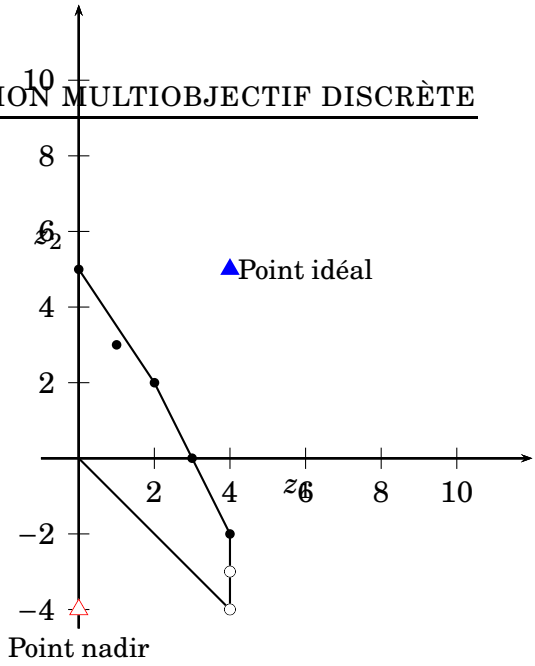


FIGURE 3.2 – Espace des critères

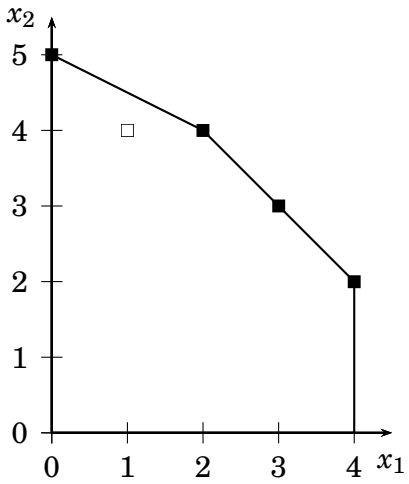


FIGURE 3.3 – Solutions supportées et non supportées

- Les solutions efficaces du programme $(P)$ sont :

$$\{(4,2),(3,3),(2,4),(1,4),(0,5)\}$$

. Par exemple le point $x' = (4,2)$ est efficace car $ED_{x'} \cap D = \{x'\}$, tandis que le point $x^* = (1,1)$ n'est pas efficace car $ED_{x^*} \cap D \neq \{x^*\}$.

- Deux solutions faiblement non dominées sont détectées, c'est les points $(4,-4)$ et $(4,-3)$.
- Les solutions efficaces supportées sont : $\{(4,2),(3,3),(2,4),(0,5)\}$. Une solution efficace non supportée est détectée, c'est le point $(1,4)$.
- Le point idéal est le point : $(4,5)$.

- La matrice des gains ici est unique : $\begin{pmatrix} 4 & 0 \\ -4 & 5 \end{pmatrix}$
- Le point nadir est le point de coordonnées : $\eta = (0, -4)$

## 3.3 Tour d'horizon des méthodes de résolutions d'un programme MOILP

Les premières méthodes de génération des solutions efficiaces de MOILP ont été proposées dans les années 1970/1980. Bitran ([36], [37]), Kiziltan et Yucaoglu [38], et Deckro et Winkofsky [39] ont proposé des algorithmes d'énumération implicite pour les problèmes MOILP avec des variables binaires seulement.

La méthode de Bitran ([36], [37]) utilise un processus constructif, dans lequel de nouvelles solutions non dominantes sont successivement générées et ajoutées à l'ensemble des solutions non dominées.

Ce type de méthodes de génération peut être arrêté avant de calculer toutes les solutions non-dominées, en retournant un sous-ensemble de solutions non-dominées. D'un autre côté, les méthodes de Kiziltan et Yucaoglu [38] et Deckro et Winkofsky [39] génèrent des solutions potentiellement non dominées et l'ensemble non-dominé n'est connu qu'à la fin du processus. Ce type de méthodes opère, dans les phases intermédiaires du processus, avec des solutions potentiellement non-dominées, c'est-à-dire des solutions qui ne sont dominées par aucune solution déjà connue.

Klein et Hannan [40] ont développé un processus constructif pour les problèmes MOILP avec des variables entières. La technique proposée par les auteurs peut être utilisée aussi bien pour identifier l'ensemble de toutes les solutions non dominées que pour en caractériser une partie seulement. Elle consiste à résoudre progressivement une séquence de programmes linéaires mono-objectifs en nombres entiers avec des contraintes ajoutées à chaque étape. Les contraintes supplémentaires éliminent les solutions efficaces déjà trouvées, et font en sorte que les nouvelles solutions générées soient efficaces. Cette méthode a été à la base des développements ultérieurs des méthodes de génération de toutes les solutions non dominées de MOILP.

Sylva et Crema [41] ont présenté une variante de l'algorithme de Klein et Hannan [40]. Son principe repose sur la résolution d'une succession de programmes linéaires en nombres entiers optimisant à chaque étape une combinaison positive des critères. Un ensemble de contraintes est rajouté à chaque fois assurant la détection d'une nouvelle solution efficace. A la fin, la méthode fournit l'ensemble de toutes les solutions non dominées du problème de programmation linéaire discrète à objectifs multiples.

Lokman et Koksalan [42] ont proposé une amélioration de cette approche, en diminuant le nombre de variables binaires de $hr$ à $h(r-1)$ et les contraintes additionnelles de $h(r+1)$ à $hr$ pour trouver la $(h+1)$e nouvelle solution non dominée. $r$ étant le nombre de critères. Cependant, la taille du modèle continue de croître et entraîne des difficultés de calcul lorsque nombre de solutions non dominées est important.

Les difficultés à trouver toutes les solutions non dominées diminuent sensiblement pour les problèmes bi-objectifs pour lesquels la conception d'un algorithme qui balaie l'ensemble de la frontière non dominée est beaucoup plus facile. Une approche similaire à celle de Klein et Hannan [40] mais limitée aux problèmes MOILP avec deux fonctions objectifs était proposée par Chalmet et al. [43]. Dans chaque itération, une somme pondérée des fonctions objectif est optimisée avec $p = 2$ contraintes supplémentaires. Aucune variable binaire supplémentaire n'est requise et les bornes inférieures peuvent être facilement déterminées pour chaque nouveau calcul.

Mavrotas et Diakoulaki ([44], [45]) ont proposé une méthode de génération de toutes les solutions non dominées du problème bi-objectifs mixte en 0-1. La technique consiste à énumérer implicitement toutes les valeurs possibles des variables 0-1, en utilisant un algorithme branch and bound, afin de générer des solutions potentiellement non dominées. Les solutions dominées sont successivement éliminées par des comparaisons par paire pour éliminer les solutions dominées. Une méthode améliorée de celle ci a été proposée par Vincent et al. [46].

Cette méthode décrite pour générer l'ensemble des solutions non dominées du problème multiobjectif en nombres entiers, est une extension pour le cas tri-objectif

de la méthode $e-contrainte$ [34] classique dédiée au cas bi-objectifs. L'approche vise à générer toutes les solutions non dominées du problème MOILP et consiste à optimiser le problème sur l'un des objectifs en bornant tous les autres objectifs. L'idée est que pour résoudre un problème bi-objectif, on le ramène à un problème mono-objectif. Pour résoudre un problème tri-objectif, on détermine d'abord les solutions non dominées d'un problème bi-objectif correspondant, et de façon générale, pour résoudre un problème avec $r$ objectifs, on doit le ramener à un problème à $r-1$ objectifs en utilisant chaque fois la transformation $e-contrainte$ [34]. Une extension de cet algorithme pour plus de trois fonctions objectifs est également proposée mais, comme dans d'autres algorithmes générateurs, son intérêt pour $r > 3$ est principalement théorique en raison de la charge de calcul résultant du processus récursif.

Kirlik et Sayın [47] ont également présenté un algorithme pour générer toutes les solutions non dominées du problème MOILP basés sur la scalarisation $e-contrainte$ [34].

Indépendamment des méthodes reposant sur la méthode $e-contrainte$ [34], Chergui et al [48] ont proposé une méthode pour résoudre MOILP dans l'espace des critères. Le principe de cette méthode est basé sur le branch & bound couplé à des coupes efficaces dans l'espace des critères ; c'est-à-dire des contraintes faisant intervenir les critères.

Toutes les méthodes abordées jusqu'ici étaient des méthodes opérant dans l'espace des critères. Il faut noter que peu de travaux ont été proposées pour la résolution du programme MOILP dans l'espace des décisions , en l'occurrence méthode de Abbas et Moulai [49], Abbsas et Chaabane et Chergui et al. [50]. Dans Abbas et Moulai [49], l'algorithme développé utilise les coupes fractionnaires de Gomory, range les solutions réalisable du problème MOILP dans l'ordre décroissant des valeurs de l'un des critères (ranking), sans omettre les solutions entières alternatives. Elle peut être vue comme une alternative à celle de Gupta et Malhotra [51] où les auteurs ont proposé un autre test d'arrêt permettant à l'algorithme de trouver toutes les solutions efficaces. Dans Abbas et Chaabane [50], les auteurs proposent la détermination de toutes les solutions efficaces du problème (MOILP) sans en

manquer aucune. Une solution initiale est déterminée en une première étape en résolvant un problème monoobjectif optimisant l'un des critères, puis une séquence de coupes type Gomory sont appliquées après avoir exploré une des arêtes adjacentes. La méthode est une forme modifiée de la méthode de Gupta et Malhotra [51] ; le test d'arrêt est modifié pour produire toutes les solutions efficaces de la région d'admissibilité.

Dans Chergui *et al.* [17] l'idée repose sur le balayage du domaine des solutions réalisables afin de détecter des solutions entières potentiellement efficaces, tout en se donnant la possibilité d'éliminer des parties du domaine réalisable ne contenant pas de solutions entières efficaces lorsque cela est possible, par adjonction de coupes dites efficaces.

## 3.4 L'optimisation sur l'ensemble efficient d'un programme MOILP

Alors que le problème de l'optimisation multiobjectif linéaire en nombres entiers (MOILP) a reçu un intérêt particulier ces dernières années. Le problème de l'optimisation sur l'ensemble efficient d'un MOILP demeure vacant et difficile à investiguer. En effet, parmi toutes les solutions efficaces, dont le nombre peut être très grand, le décideur n'est intéressé que par celle qui optimise un nouveau critère (généralement différent de ceux considérés dans le problème original). Ce problème avec les variables continues a été étudié par plusieurs auteurs. En fait, l'un des pionniers dans ce domaine est Philip [52] qui a étudié ce problème et a suggéré de résoudre deux procédures. Dans la première, il supposait que la fonction à optimiser était une agrégation strictement positive des critères, où la restriction "*ensemble efficient*" est remplacée par "*espace de décision*" et devient un problème de programmation linéaire ordinaire. Dans la seconde, l'optimisation est faite selon une fonction linéaire quelconque et une technique de coupes planes est utilisée pour surmonter la difficulté de la non-convexité de l'ensemble efficient et supprime des points efficaces inintéressants. Plus tard, Benson ([53], [54], [55]), Ecker & Song [56] et d'autres ont également étudié ce problème et ont proposé différentes approches. Il faut noter

que, dans ce cas, l'ensemble efficient a deux propriétés intéressantes : il est connexe et contient des points extrêmes du polyèdre ([57]).

L'étude de Yamamoto [58] propose une classification des algorithmes existants pour l'optimisation sur l'ensemble efficient. Cette classification contient sept classes : algorithmes de recherche de sommets adjacents, algorithmes de recherche de sommets non adjacents, algorithmes de recherche de facettes, algorithme de recherche branch & bound, algorithmes basés sur la relaxation lagrangienne, approche duale, algorithmes de Dichotomie. L'article de Yamamoto [58] rapporte des informations sur les algorithmes considérés concernant les exigences de calcul correspondantes.

Néanmoins, le cas où les variables de décision sont entières n'a pas reçu autant d'intérêt que le cas continu. Ceci est dû en fait, non seulement à la structure inconnue du domaine des solutions réalisables (non convexité), mais aussi à l'aspect discret des variables de décision.

Nonobstant son importance dans le monde réel, une recherche bibliographique nous a permis de conclure que seulement quatre méthodes ont été proposées pour traiter ce sujet. D'abord par Nguyen et *al.* [57] qui ont proposé des algorithmes pour deux cas particuliers. Dans le premier cas, il a réduit le problème multiobjectif à un problème biobjectif et dans l'autre, la fonction à optimiser est une combinaison non-négative des critères. Deuxièmement, Abbas & *al.* [8], ont introduit pour résoudre ce problème des coupes planes dans l'espace de décisions basé sur le travail de Ecker & Song [56] dans le cas continu. Alors que Jorge [10] a défini une suite de problèmes de programmation linéaire en nombres entiers à objectif unique de plus en plus contraints qui élimine les points inacceptables. Contrairement aux méthodes précédentes, celle de Jorge [10] a été implémentée et l'auteur montre les résultats obtenus sur différentes instances de problèmes générées aléatoirement. En 2017, Boland *et al.* [59] ont proposé un nouvel algorithme pour optimiser une fonction linéaire sur l'ensemble des solutions efficaces d'un programme MOILP. L'algorithme repose sur un nouvel algorithme dédié à l'énumération des points non-dominés d'un MOILP. Cet algorithme est le résultat de l'emploi d'un nouveau schéma de décomposition de l'espace des critères qui limite le nombre de sous-espaces créés et celui d'ensembles de contraintes disjonctives requis pour définir le programme en nombres

entiers à objectif unique à chaque étape. Une étude expérimentale est reportée montrant l'efficacité de l'algorithme.

### 3.4.1 Formulation du problème et quelques résultats

Dans cette sous section, nous nous intéressons au problème $(OI)$ suivant :

$$(OI) \quad \max\{\varphi(x) = dx | x \in X_E\} \tag{3.1}$$

Où : $X_E$ est l'ensemble des points efficients du problème de programmation linéaire multiobjectif en nombres entiers suivant :

$$(P) \begin{cases} "Max" \quad z^i = c^i x \quad i = \overline{1,r} \\ x \in D \end{cases}$$

Où : $D = \{x \in \mathbb{Z}^n | Ax = b, x \geq 0\}$ est supposé borné et non vide.

L'intérêt de s'attaquer au problème $(OI)$ est motivée par de nombreux facteurs. Premièrement, en termes de temps de calcul, il peut être plus facile de résoudre $(OI)$ directement que de résoudre $(P)$ et ensuite d'obtenir une solution efficace optimale. Deuxièmement, les décideurs peuvent être confus par la taille importante de l'ensemble des solutions efficaces (la cardinalité de $X_E$ peut être très grande) et peut ne pas être en mesure d'en choisir une solution préférée.

Dans toutes les méthodes proposées aussi bien pour le cas continu que discret, le test proposé par Benson [60] afin de juger de l'efficacité d'une solution réalisable à une étape donnée est utilisée.

**Théorème 3.3** (Benson1978)**.** *Soit $x^*$ une solution réalisable de la région $D$. $x^* \in X_E$ si et seulement si la valeur optimale de la fonction objectif $\Theta$ est nulle dans le problème de programmation linéaire mixte suivant :*

$$(P_{x^*}) \begin{cases} Max \;\; \Theta = \sum_{i=1}^{r} s_i \\ s.t. \;\; Cx = Is + Cx_l^*, \\ Ax \leq b, \; x \in D, \;\; s_i \in \mathbb{R}^+ \forall i \end{cases} \tag{3.2}$$

*Où $C$ est une matrice $r \times n$, $I$ est la matrice identité $(r \times r)$ et $s = (s_i)_{i=1,\cdots,r}$.*
*Si la valeur maximale de $(P_{x^*})$ est finie non nulle, la solution $y^*$ obtenue est efficace.*

***Démonstration.*** Soit $x^* \in X_E$. Comme $s_i \geq 0$ pour tout $i = 1, \cdots, r$ , alors $Cx \geq Cx*$ dans $P_{x^*}$. Supposons que $\max\Theta \neq 0$, alors il existe $i$ tel que $s_i > 0$, $Is \neq 0$ ce qui implique $Cx > Cx^*$. Ainsi il existe $x \in D$ tel que $Cx \geq Cx^*$ et $Cx \neq Cx^*$, ce qui est en contradiction avec l'hypothèse selon laquelle $x^*$ est efficace.

Soit $\max\Theta = 0$ ; supposons que $x^*$ est non efficace. Il existe un point $x \in D$ tel que $Cx \geq Cx^*$ et $Cx \neq Cx^*$. Par conséquent, il existe $x \in S$ tel que $Cx - Cx^* \geq 0$ et $Cx - Cx^* \neq 0$, d'où $Is > 0$ ; et donc, il existe $i$ tel que $s_i > 0$, ce qui est en contradiction avec l'hypothèse $\max\Theta = 0$. ■

### 3.4.2 Méthode Abbas& Chaaabane ([8], [9])

Le problème principal étudié est décrit par :

$$(PLI_E) \quad \max\{\phi(x) = dx | x \in E(P)\} \tag{3.3}$$

où $d$ est un vecteur ligne de dimension $n$ qui a pour $j$ème composante le nombre entier $d_j$ .

$E(P)$ est l'ensemble des solutions efficaces du problème de programmation multiobjectf en nombres entiers suivant :

$$(P(D)) \quad \text{"max"}\left\{Z_i(x) = c^i x | x \in D\right\} \tag{3.4}$$

où $D = S \bigcap \mathbb{Z}^n$, $S = \{x \in \mathbb{R}^n | Ax = b, x \geq 0\}$, $A \in \mathbb{Z}^{m \times n}$, $b \in \mathbb{Z}^m$, $p \geq 2$ ; $c^1, \cdots, c^p \in \mathbb{Z}^n$ sont des vecteurs lignes, $\mathbb{Z}$ est l'ensemble des entiers relatifs. On suppose que $D$ est non vide et que $S$ est un polyèdre convexe et borné.

Soit le problème relaxé :

$$(P_R) \quad \max\{\phi(x) = dx | x \in D\} \tag{3.5}$$

et le problème $(P_i(D))$, $i = 1, \cdots, p$

$$(P_i(D)) \quad \text{"max"}\left\{Z_i(x) = c^i x | x \in D\right\} \tag{3.6}$$

**Étape 1.** Résoudre le problème relaxé $(P_R)$. Soit $x^*$ une solution optimale. Cette solution est testée pour l'efficacité en résolvant le problème $(P(x^*))$ mentionné dans le théorème (Benson).

Si elle est efficace, alors elle est aussi une solution de $(PLI_E)$ et l'algorithme se termine. Autrement, on aller à l'étape 2.

**Étape 2.** Soit $\phi_{opt} = -\infty$. On résout le problème $(P_1(D))$. (On peut alternativement considérer n'importe quel problème $(P_i(D)\,;\, i = 2, \cdots, p$ au lieu de $(P_1(D))$.

1. Si $J_1 = \left\{ j \in N_1 | z^1_{1,j} - c^1_j = 0 \right\} = \emptyset$, alors la solution optimale trouvée, $x_1$, est unique et elle est efficace. Poser $\phi_o pt = dx_1$, $x_{opt} = x_1$, et aller à l'étape 3.
2. Si $J_1 \neq \emptyset$, alors la solution optimale $x_1$ du problème $(P_1(D))$ peut ne pas être unique ; tester l'efficacité de $x_1$ ; si $x_1$ n'est pas efficace aller à l'étape 3 ; sinon, soit $\phi_o pt = dx_1$ et $x_{opt} = x_1$, et aller aussi à l'étape 3.

**Étape 3** Poser $k = 1$ et exécuter les sous-étapes suivantes :

3.1 Construire l'ensemble $\Gamma_k = \left\{ j \in N_k | z^k_{1,j} - c^1_j \geq 0 \; et \; \phi^k_j - d_j \leq 0 \right\}$

- Si $\Gamma_k = \emptyset$, ajouter la coupe de Dantzig $\sum_{j \in N_k} x_j \geq 1$ et aller à l'étape 3.3.
- Autrement, soit $\gamma = \Gamma_k$. Aller à (a).

(a) Si $\gamma = \emptyset$, alors soit $j_k \in \Gamma_k$, appliquer la coupe de type I $\sum_{j \in N_k \; j_k} x_j \geq 1$ et aller à la sous étape 3.3. Autrement, sélectionner $j_k \in \gamma$ et calculer $\theta^0_{j_k}$, la partie entière de $\min_{i \in I_k} \left\{ \dfrac{x_{k,i}}{y_{k,ij_k}} | y_{k,ij_k} > 0 \right\}$.

$I_k$ étant l'ensemble des indices correspondants aux variables de base.

- Si $\theta^0_{j_k} = 0$, alors il n'y a aucune solution réalisable entière sur l'arête $E_{j_k}$ , faire $\gamma := \gamma \; j_k$ et aller à (a).
- Autrement, si $\theta^0_{j_k} \geq 1$, alors aller à (b).

(b) Si $x_k$ est efficace et $dx_k \geq \phi_{opt}$, calculer la valeur du paramètre $\beta_k$ définie dans l'équation par :

$$\beta_k = d_{j_k} - \sum_{i \in I_k} d_i \times y_{k,ij_k}$$

Si cette valeur n'est pas égale à zéro, alors aller à (c), autrement, faire $\gamma := \gamma \; j_k$ et aller à (a).

Si $x_k$ n'est pas efficace ou $dx_k < \phi_{opt}$, alors aller à (c) (l'arête $E_{j_k}$ doit être explorée quelle que soit la valeur de $\beta_k$).

(c) Explorer l'arête $E_{j_k}$ , en recherchant les solutions réalisables de $(P_1(D_k))$ correspondant à $\theta$ et tester pour l'efficacité à partir de $\theta = \theta^0_{j_k}$ jusqu'à $\theta = 1$ ($\theta$ est un nombre entier positif). Dès qu'une solution efficace $x'_k$ vérifiant $dx'_k > \phi_{opt}$ est trouvée pour une valeur de $\theta$, remplacer $x_{opt}$ par $x'_k$ et $\phi_{opt}$ par $dx'_k$ et aller à la sous-étape 3.3. S'il n'y a aucune solution efficace entière sur l'arête, alors faire $\gamma := \gamma \setminus \{j_k\}$ et aller à (a).

3.2 Soit $k := k + 1$. La nouvelle région tronquée $D_k$ est obtenue comme sous ensemble de $D_{k-1}$ en appliquant la coupe $dx \geq \phi_{opt}$ (une coupe de type II) puis en utilisant la méthode dual simplexe et la méthode des coupes de Gomory autant de fois que nécessaire pour trouver une nouvelle solution optimale $x_k$. Faire $x_{opt} := x_k$ et $\phi_{opt} := dx_k$ et aller à la sous étape 3.1.

3.3 Soit $k := k + 1$. La nouvelle région tronquée $D_k$ est obtenue comme sous ensemble de $D_{k-1}$ (ou $D$ si $k = 1$) en appliquant la coupe de Dantzig ou la coupe de type I puis en utilisant la méthode dual simplexe et la méthode des coupes de Gomory autant de fois que nécessaire pour trouver une nouvelle solution optimale $x_k$ ; soit $\phi_k = dx_k$.

Si la solution $x_k$ est efficace et $dx_k > \phi_o pt$, faire $x_{opt} := x_k$ et $\phi_o pt := dx_k$ et aller à la sous étape 3.1. Sinon, aller aussi à la sous étape 3, sans mise à jour de $x_{opt}$ et $\phi_{opt}$.

**Fin.** La procédure prend fin, ou bien à la première étape si la solution $x_0$ est efficace, ou lorsque l'impossibilité des opérations de pivot indique que la région courante ne contient aucun point entier réalisable. La solution optimale est alors $x_{opt}$ et sa valeur sur le critère $\phi$ est $\phi_{opt}$.

### 3.4.3 Méthode Jorge [10]

Soit à résoudre le problème :

$$(Q) \quad \max\{v^t x | x \in XE\} \tag{3.7}$$

où : $v$ est une fonction réelle à valeurs dans $\mathbb{R}$. $XE$ est l'ensemble des solution efficaces du problème de programmation multiobjectif en nombres entiers suivant :

$$(P) \quad \max\{Cx | x \in S\} \tag{3.8}$$

où : $S = \{x \in \mathbb{Z}_+^n | Ax \leq b\}$ est l'ensemble des solutions réalisables, avec $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ et $C$ une $p \times n$-matrice définissant un nombre $p \geq 2$ de fonctions objectifs. $z(XE)$ sera désormais utilisé pour désigner l'ensemble de toutes les solutions non dominées de $(P)$ à travers la fonction d'objectif $z = Cx$.

Le problème $E$ défini comme suit :

$$(RE) \quad \max\{v^t x | x \in S\} \tag{3.9}$$

peut être considéré comme une relaxation du problème $Q$.

## L'algorithme

L'algorithme proposé est prouvé pour fournir une solution optimale globale de $(Q)$ sans avoir à calculer l'ensemble de toutes les solutions efficaces du problème sous-jacent $(P)$.

la procédure commence à résoudre le problème relaxé $RE$. Évidemment, seulement dans un nombre réduit de cas spéciaux la solution de $RE$ fournirait la solution optimale de $Q$. Si ce n'est pas le cas, une nouvelle solution efficace dominant la précédente est obtenue. Ensuite, dans les itérations suivantes, des relaxations progressivement plus contraintes de $Q$ sont résolues afin de fournir de nouvelles solutions qui ne sont pas dominées par les points examinés précédemment, jusqu'à ce que la solution optimale soit finalement trouvée.

Étape 0 (Initialisation) Soit $v^{lower} = -\infty$, $v^{upper} = +\infty$ et $l = 1$.

Résoudre le problème relaxé $RE \equiv \max\{v^t x | x \in S\}$. Si $RE$ est irréalisable $\Rightarrow$ **STOP**, $Q$ est irréalisable. Sinon, soit $x^1$ la solution optimale de $RE$.

Étape 1 Si $x^l \in XE \Rightarrow$ **STOP**. $x^{inc} = x^l$ est une solution optimale pour $Q$.

Sinon, fixer $v^{upper} = v^t x^l$ et aller à l'étape 2.

Étape 2 Trouver $\hat{x}^l \in XE$ dont le vecteur critère domine $Cx^l$ et soit $\bar{x}^l$ une solution optimale du problème :

$$T_l \equiv \max \left\{ v^t x | Cx = C\hat{x}^l,\ x \in S \right\} \tag{3.10}$$

Si $v^t \bar{x}^l > v^{lower}$, poser $v^{lower} = v^t \bar{x}^l$ et $x^{inc} = \bar{x}^l$

Si $v^{lower} = v^{upper} \Rightarrow$ **STOP**. $x^{inc}$ est une solution optimale pour $Q$.

Étape 3 Soit $RE_l \equiv \max\{v^t x | x \in S - \cup_{r=1}^{l} D_r\}$, où : $D_r = \{x \in \mathbb{Z}^n | C\bar{x}^l \geq Cx\}$

Si $RE_l$ est irréalisable $\Rightarrow$ **STOP**. $x^{inc}$ est une solution optimale pour $Q$. Sinon, soit $x^{l+1}$ une solution optimale de $RE_l$.

Si $v^t x^{l+1} \leq v^{lower} \Rightarrow$ **STOP**. $x^{inc}$ est une solution optimale pour $Q$.

Sinon, poser $l = l + 1$ et aller à l'étape 1.

La caractéristique principale de l'algorithme réside dans la résolution, à chaque itération l, d'un programme linéaire en nombres entiers $RE_l$ qui impose de nouvelles contraintes sur l'ensemble réalisable du problème résolu à l'itération précédente, $RE_{l-1}$. De telles contraintes, connues dans la littérature sous le nom de contraintes de coin (Klein et Hannan [40]), écartent non seulement toutes les solutions efficaces précédemment générées, mais aussi toutes les autres solutions réalisables de (P) dont les vecteurs ont été dominés par (ou identiques à) l'un des points non dominés déjà trouvés.

En outre, il convient de noter que ces contraintes de coin sont ajoutées chaque fois qu'un nouveau point non-dominé est trouvé. Le nombre de problèmes de programmation linéaire discrète devant être résolus est donné par le nombre de solutions non dominées rencontrées.

## 3.5 Conclusion

Dans ce chapitre, nous avons défini quelques concepts importants de l'optimisation multiobjectif discrète et utiles à la compréhension des méthodes exactes exposées dans le reste de cette thèse. Nous avons aussi présenté un état de l'art sur l'optimisation d'une fonction linéaire sur l'ensemble efficient d'un programme MOILP. Nous présentons notre contribution sur ce même problème dans le chapitre qui suit.

CHAPITRE 4

# UNE MÉTHODE D'OPTIMISATION D'UN CRITÈRE LINÉAIRE SUR L'ENSEMBLE EFFICIENT D'UN MOILP

## 4.1 Introduction

Dans la présente étude, nous proposons une méthode basée sur la technique "branch & bound" pour la résolution du problème de l'optimisation d'un critère linéaire sur l'ensemble efficient d'un problème MOILP [14]. En utilisant le concept de coupe efficace, notre approche réussit à trouver la solution optimale en évitant de visiter toutes les solutions efficaces. En outre, notre méthode a été mise en œuvre en utilisant Matlab2013a et une étude comparative avec la méthode proposée par Jorge [10] est rapportée à la fin de ce chapitre.

## 4.2 Formulation mathématique

Nous supposons que r est un entier, $r \geq 2$ tel que $c^i,\ i = 1, \cdots, r$, sont des vecteurs lignes de $\mathbb{R}^n$. Soit $C$ une $r \times n$- matrice formée par les vecteurs $c^i;\ i = 1, \cdots, r$, et soit $S$ un polyèdre non vide, compact dans $\mathbb{R}^n$. Également, $S$ est défini par $\{x \in \mathbb{R}^n | Ax \leq b, x \geq 0\}$, où $A$ est une $m \times n$-matrice d'éléments entiers ; $b \in \mathbb{Z}^m$ et $D$ est l'ensemble des solutions entières dans $S$. Alors, le problème de programmation multiobjectif linéaire

en nombres entiers est décrit comme suit :

$$(P)\begin{cases} Max \quad Z_i(x) = c^i x;\ i = 1,\cdots,r \\ s.t.\ \ x \in D = S \cap \mathbb{Z}^n \end{cases} \tag{4.1}$$

On note par $X_E$ l'ensemble de toutes les solutions efficaces issu de la résolution de $(P)$.

Fondamentalement, l'ensemble des solutions efficaces du programme $(P)$ peut être très grand et la tâche d'en choisir une qui correspond aux préférences du décideur est très difficile. Ceci induit un nouveau problème de recherche d'une solution optimale efficace selon le problème de programmation mathématique suivant :

$$(OI)\begin{cases} Max \quad \varphi(x) \\ s.t.\ \ x \in X_E \end{cases} \tag{4.2}$$

où $\varphi : \mathbb{R}^n \longrightarrow \mathbb{R}$ est une fonction linéaire continue quelconque.

Considérons le programme linéaire à l'étape $l$, $l \geq 0$ de la méthode proposée :

$$(OI_l)\begin{cases} Max \quad \varphi(x) \\ s.t.\ \ x \in S_l \end{cases} \tag{4.3}$$

On note par $x_{opt}$ la meilleure solution efficace de $(OI)$ trouvée jusqu'à l'étape $l$ et $\varphi_{opt}$ la valeur de $\varphi$ correspondante. $S_0 = S$, et $S_{l+1}$ sont obtenus à partir de $S_l$ en rajoutant la coupe efficace définie ci-après. Pour ce faire, soit $x_l^*$ la première solution entière trouvée en résolvant $(OI_l)$ en utilisant, éventuellement, le processus de séparation de la méthode "branch & bound".

Au point $x_l^*$, nous utilisons les notations suivantes :

- Par $B_l$ $(N_l)$, nous entendons l'ensemble des indices de base (respectivement hors base) de $x_l^*$ ;
- Soit $\bar{c}_j^i$ la $j^{ème}$ composante du vecteur coût réduit $c^i$, $i = 1,\cdots,r$, au niveau du tableau du simplexe ;
- Le concept de coupes est crucial pour résoudre $(OI)$. Pour ce faire, deux types de coupes sont construits. En fait, pour construire la **coupe de type 1**, nous définissons l'ensemble $H_l$ comme :

$$H_l = \left\{ j \in N_l | \exists i = 1,\cdots,r;\ \bar{c}_j^i > 0 \right\} \cup \left\{ j \in N_l | \bar{c}_j^i = 0\ \forall i = 1,\cdots,r \right\} \tag{4.4}$$

et la coupe efficace

$$\sum_{j \in H_l} x_j \geq 1 \tag{4.5}$$

qui a la propriété de supprimer des solutions non efficaces sans avoir à les énumérer. D'autre part, la **coupe de type 2** est construite selon l'inégalité suivante :

$$\varphi(x) \geq \varphi_{opt} \tag{4.6}$$

pour éliminer des points qui ne sont pas optimaux.

- Nous définissons les deux ensembles au nœud $l$ de type 1 (en une solution entière) :

$$S_{l+1}^1 = \left\{ x \in S_l | \sum_{j \in H_l} x_j \geq 1 \right\} \tag{4.7}$$

- De plus, l'ensemble suivant est considéré au nœud $l$ de type 2 (en un point entier) :

$$S_{l+1}^2 = \{ x \in S_l | \varphi(x) \geq \varphi_{opt} \} \tag{4.8}$$

- $S_{l+1} = S_{l+1}^1 \cup S_{l+1}^2$
- $Eff_l$ est l'ensemble des solutions potentiellement efficaces de $(P)$ obtenues jusqu'à l'étape $l$. Toutes les solutions dans $Eff_l$ sont des solutions entières telles qu'aucune ne domine l'autre dans l'espace des critères. L'ensemble $Eff_l$ est mis à jour, testant chaque fois qu'une solution entière $x_l^*$ est atteinte, que $Z(x_l^*)$ soit dominé ou non. Si $Z(x_l^*)$ n'est dominé par aucun vecteur $Z(x)$, $x \in Eff_{l-1}$, alors $Eff_l = Eff_{l-1} \cup \{x_l^*\}$ et en supprimant aussi les solutions de $Eff_{l-1}$ dont le vecteur critère est dominé par $Z(x_l^*)$.

Nous devons donc tester l'efficacité des solutions via deux options. Le premier consiste à considérer l'ensemble $Eff_l$ initialement vide et à chaque étape $l$ est mis à jour. Le second concerne la résolution du programme linéaire mixte suivant :

Étant donné un point $x_l^* \in D$, soit $(P_{x_l^*})$ le programme linéaire :

$$(P_{x_l^*}) \begin{cases} Max \ \ e^t s \\ s.t. \ \ Cx = Is + Cx_l^*, \\ Ax \leq b, \ x \in D, \ s \geq 0 \end{cases} \tag{4.9}$$

où $e$ est un vecteur colonne de 1 et $I$ est une matrice d'identité $(r \times r)$.
$x_l^*$ est efficace si et seulement si $(P_{x_l^*})$ a une valeur maximale zéro. Sinon (la valeur maximale de $(P_{x_l^*})$ est finie non nulle), la solution $y_l$ obtenue est efficace.

**Remarque 4.1.** *L'ensemble $Eff_l$ est aussi mis à jour comme précédemment, chaque fois qu'une nouvelle solution $y_l$ de $(P_{x_l^*})$ est trouvée.*

## 4.3 Méthodologie

L'algorithme proposé génère la solution optimale de OI sans avoir à énumérer $X_E$, l'ensemble efficient de $(P)$. Basé sur la technique "branch & bound", la méthode est renforcée par le recours aux coupes efficaces et aux nouveaux tests d'arrêt permettant une recherche intelligente de la solution optimale. Nous commençons par résoudre le programme $OI_l$ défini par le programme 4.3 en utilisant la méthode du simplexe à l'étape $l$ de l'algorithme (éventuellement la méthode duale du simplexe). Ensuite, pour voir comment les vecteurs critères évoluent de base en base, $r$ lignes sont ajoutées au tableau du simplexe et les coûts réduits sont calculés par rapport à la base correspondante. Si la solution obtenue n'est pas entière, nous imposons alors des restrictions d'intégrité sur les variables du programme jusqu'à obtenir des solutions entières. Une fois qu'une solution entière $x_l^*$ est trouvée, de nouvelles coupes sont établies et ajoutées au tableau du simplexe actuel qui permettent de réduire considérablement la zone de recherche (contenant des solutions non efficaces et non intéressantes pour OI). Nous considérons désormais deux types de nœuds, ceux relatifs au processus de branchement (type 1) et les autres aux coupes efficaces (type 2). Ainsi, un nœud de type 2 est élagué si aucune amélioration des critères ne peut être faite le long du domaine restant ou si une solution efficace est atteinte à un stade $l$. Un nœud de type 1 est sondé si $\varphi_{opt}$ la meilleure valeur de $\varphi$ obtenue jusqu'à l'étape $l$ soit supérieure ou égale à la valeur de $\varphi$ sur ce nœud (même si la solution sur le nœud actuel n'est pas entière).

L'algorithme, dénommé $LMOILP$ est résumé comme suit :

***Étape 1 (initialisation)*** : initialiser l'indice des programmes à résoudre $l = 0$ et la valeur initiale de la fonction objectif $\varphi_{opt} = -\infty$ pour laquelle ne correspond aucune

solution optimale encore ($x_{opt}$ inconnue au départ), $Eff_l$ l'ensemble des solutions potentiellement efficaces de ($P$), $Eff_0 = \emptyset$.

***Étape 2 (étape générale)*** : Tant qu'il existe un nœud non sondé dans l'arborescence de recherche, choisir le nœud $l$ le plus récemment créé (recherche en profondeur d'abord), résoudre le programme ($OI_l$) correspondant en utilisant l'algorithme du simplexe ou l'algorithme duale, tout dépend du signe du vecteur $b$ du programme en question. Aller à à l'étape 3.1.

***Étape 3 (tests)*** :

3.1 ***test de faisabilité :*** Si ($OI_l$) est irréalisable, stop et le nœud $l$ est sondé, sinon, soit $x_l^*$ la solution trouvée, si $\varphi_{opt} \geq \varphi(x_l^*)$, le nœud $l$ est sondé, sinon aller à l'étape 3.2 ;

3.2 ***test d'intégrité :*** Si $x_l^*$ est entière, alors mettre à jour $Eff_l$ et aller à l'étape 3.3, sinon aller à l'étape 4 ;

3.3 ***test d'efficacité :*** Si $x_l^*$ n'est pas gardée dans $Eff_l$, $x_l^*$ est non efficace, aller à à l'étape 5, sinon, résoudre ($P_{x_l^*}$). Si $x_l^*$ est non efficace alors, mettre à jour éventuellement $\varphi_{opt} = \varphi(x_l^*)$ et $x_{opt} = x_l^*$ ; le nœud $l$ est élagué puisque aucune amélioration de $\varphi$ ne peut être faite désormais, sinon, soit $y_l$ la solution de ($P_{x_l^*}$), mettre à jour si nécessaire $\varphi_{opt}$, $x_{opt}$ et l'ensemble $Eff_l$ aussi, aller à l'étape 5.

***Étape 4 (branchement)*** : Choisir une coordonnée $x_j$ de $x_l^*$ telle que $x_j = \alpha_j$, avec $\alpha_j \notin \mathbb{N}$ . Alors, éclater le programme ($OI_l$) en deux sous programmes, en ajoutant la contrainte $x_j \leq \lfloor \alpha_j \rfloor$ pour obtenir ($OI_{l_1}$), $x_j \geq \lfloor \alpha_j \rfloor + 1$ et construire $S_{l+1}^2$ pour avoir ($OI_{l_2}$) tels que $l_1 > l+1$, $l_2 > l+1$ et $l_1 \neq l_2$, aller à l'étape 2. En effet, comme l'arborescence est traité en profondeur d'abord, nous rajoutons la coupe $\varphi(x) \geq \varphi_{opt}$ dans la branche $l_2$.

***Étape 5 (coupe efficace)*** : Construire l'ensemble $H_l$. Si $H_l = \emptyset$ ; le nœud $l$ est sondé puisque aucune solution efficace n'existe dans le domaine courent , sinon, construire l'ensemble $S_{l+1}^1$ (ajouter la coupe efficace), aller à l'étape 2.

## 4.4 Résultats fondamentaux

Afin de justifier les différentes étapes de l'algorithme proposé, les résultats suivants sont établis. On note $D_l$ l'ensemble $D_l = S_l \cap \mathbb{Z}^n$.

**Théorème 4.1.** *Supposons que $H_l \neq \varnothing$ au point entier courant $x_l^*$. Si $x \neq x_l^*$ est une solution optimale pour OI dans le domaine $S_l$, alors $x \in S_{l+1}$.*

***Démonstration.*** Soit $x \neq x_l^*$ une solution entière dans le domaine $S_l$ telle que $x \notin S_{l+1}$, alors $x \notin S_{l+1}^1$ et $x \notin S_{l+1}^2$.

- Si $x \notin S_{l+1}^1$, alors $x \in \left\{ x \in S_l | \sum_{j \in N_l \setminus H_l} x_j \geq 1 \right\}$. Par conséquent, les coordonnées de $x$ vérifient l'inégalité suivante : $\sum_{j \in H_l} x_j < 1$ et $\sum_{j \in N_l \setminus H_l} x_j \geq 1$. Il s'en suit que $x_j = 0$ pour tout $j \in H_l$, et $x_j \geq 1$ pour au moins un indice $j \in N_l \setminus H_l$. En utilisant le tableau du simplexe en $x_l^*$, l'inégalité suivante est supportée par tous les critères $i \in \{1, \cdots, r\}$ :

  $c^i x = c^i x_l^* + \sum_{j \in N_l} \widehat{c}_j^i x_j$

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

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

  Donc, $c^i x \leq c^i x_l^*$ pour tous les critères $i \in \{1, \cdots, r\}$, avec $c^i x < c^i x_l^*$ pour au moins un critère comme $\widehat{c}_j^i \leq 0$ pour tout $j \in N_l \setminus H_l$.

  Nous concluons que la solution $x$ n'est pas efficace et alors, toutes les solutions efficaces entières appartiennent au domaine $S_{l+1}^1$, ... (*).

- Si $x \notin S_{l+1}^2$, $x$ n'est pas optimale, contradiction, ...(**).

De (*) et (**), on en déduit que $x \in S_{l+1}$. ■

**Théorème 4.2.** *Soit $x_l^*$ la solution entière courante du programme $(OI_l)$. Si $x_l^*$ est efficace du programme $(P)$, alors optimale pour (OI) sur $D_l$.*

***Démonstration.*** Supposons que $x_l^*$ n'est pas optimale pour $(OI)$, alors, $\exists x \in D_l$, $x \neq x_l^*$ tel que $\varphi_{opt} \geq \varphi(x_l^*)$. Ainsi $\varphi(x) \geq \varphi(x_l^*)$. D'autre part, au tableau du simplexe courant, l'expression de $\varphi$ peut être écrite comme :

$\varphi(x) = \varphi(x_l^*) + \sum_{j \in N_l} \widehat{\varphi}_j x_j$
$\Rightarrow \varphi(x_l^*) + \sum_{j \in N_l} \widehat{\varphi}_j x_j > \varphi(x_l^*)$
$\Rightarrow \sum_{j \in N_l} \widehat{\varphi}_j x_j > 0$
Ce qui contredit le fait que $\widehat{\varphi}_j \leq 0,\ \forall j \in N_l$.

■

**Proposition 4.1.** *Si $H_l = \emptyset$, alors $\forall x \in D_{l+1}$ , x n'est pas efficace.*

***Démonstration.*** $H_l = \emptyset$, alors $\forall i \in \{1, \cdots, r\}$, $\forall j \in N_l$, nous avons $\hat{c}_j^i \leq 0$ et $\exists i_0 \in \{1, \cdots, r\}$ tel que $\hat{c}_j^{i_0} < 0\ \forall j \in N_l$. Donc, $x_l^*$ domine tous les points $x$, $x \neq x_l^*$ du domaine $D_{l+1}$. ■

**Proposition 4.2.** *Si $\varphi_{opt} \geq \varphi(x_l^*)$, alors $\nexists x \in D_l$ telle que $\varphi(x) > \varphi_{opt}$.*

***Démonstration.*** Il est évident que toutes les solutions $x$ pour lesquelles $\varphi(x) < \varphi_{opt}$ ne sont pas intéressantes (mêmes efficaces), car l'existence d'une solution efficace donnant déjà la meilleure valeur de $\varphi$. ■

**Théorème 4.3.** *L'algorithme se termine en un nombre fini d'étapes et renvoie la solution optimale pour (OI).*

***Démonstration.*** L'ensemble $S$ des solutions réalisables du programme $(P)$ étant compact, il contient un nombre fini de solutions entières. A chaque étape $l$ de l'algorithme, si une solution entière $x_l^*$ est atteinte, nous procédons à son élimination ainsi qu'à un sous-ensemble de solutions entières non intéressantes en prenant en compte du théorème 4.1 ci-dessus (ajout de coupes). D'autre part, quatre tests de saturation sont utilisés sans omission de la solution optimale de $(OI)$. Premièrement, lorsque l'ensemble $H_l$ est vide, la solution correspondante $x_l^*$ constitue un point idéal local et le nœud actuel peut être élagué car aucun critère ne peut être amélioré. Deuxièmement, si à une étape $l$, la solution entière courante $x_l^*$ est efficace, le nœud correspondant est sondé puisque $x_l^*$ est optimal pour $(OI)$ sur $D_l$. Troisièmement, si $\varphi_{opt}$ (valeur de la meilleure solution efficace trouvée pour $(OI)$ est supérieure à celle de la solution optimale sur $D_l$, le nœud $l$ est également sondé. Enfin, le cas trivial lorsque le domaine réduit devient irréalisable. Par conséquent,

l'algorithme converge vers une solution optimale pour $(OI)$ en nombre fini d'étapes.

■

## 4.5 Illustration numérique

Nous considérons le problème d'optimisation d'un critère linéaire sur l'ensemble efficient entier, traité par Jesus M. Jorge dans [10] :

$$(OI)\begin{cases} Max \quad -x_1 - 2x_2 \\ x \in X_E \end{cases} \tag{4.10}$$

où $X_E$ est l'ensemble efficient du programme :

$$(P)\begin{cases} Max \;\; x_1 - 2x_2 \\ Max \;\; -x_1 + 4x_2 \\ s.t. \;\; -2x_1 + x_2 \leq 0 \\ \qquad x_1 \leq 3 \\ \qquad x_2 \leq 2 \\ x_1, x_2 \geq 0, \; entiers \end{cases} \tag{4.11}$$

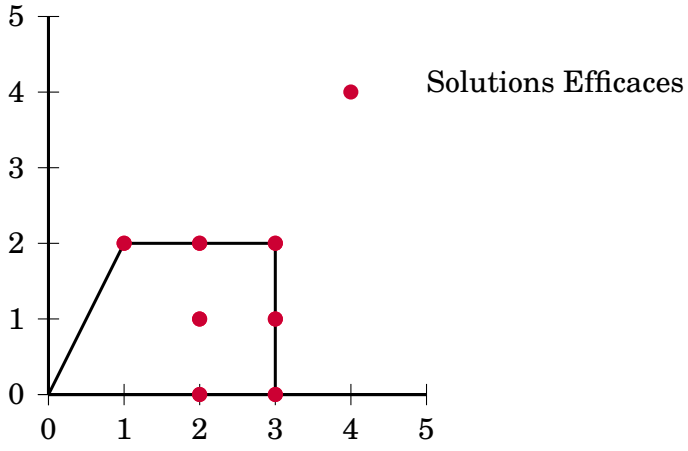


FIGURE 4.1 – Domaine réalisable

Etape 1 : initialiser la valeur de $\varphi_{opt}$ à $-\infty$ et $l$ à 0. Après résolution du programme $(OI_0)$, la solution optimale ainsi obtenue est $x_0^* = (0,0)$ qui n'est pas efficace, mais la solution optimale obtenue en résolvant $(Px_0^*)$ qui est $(3,1)$ est efficace.

Mettre à jour $\varphi_{opt} = -5$ et $x_{opt} = (3,1)$. $H_0 = \{1,2\} \neq \emptyset$.

| | $x_1$ | $x_2$ | $b_i$ |
|---|---|---|---|
| $x_3$ | $-2$ | 1 | 0 |
| $x_4$ | 1 | 0 | 3 |
| $x_5$ | 0 | 1 | 2 |
| $-\varphi$ | $-1$ | $-2$ | 0 |
| $-c^1$ | 1 | $-2$ | 0 |
| $-c^2$ | $-1$ | 4 | 0 |

Appliquer la coupe efficace $x_1 + x_2 \geq 1$ et utiliser la méthode duale du simplexe pour obtenir le tableau suivant :

| | $x_6$ | $x_2$ | $b_i$ |
|---|---|---|---|
| $x_3$ | $-2$ | 3 | 2 |
| $x_4$ | 1 | $-1$ | 2 |
| $x_5$ | 0 | 1 | 2 |
| $x_1$ | $-1$ | 1 | 1 |
| $-\varphi$ | $-1$ | $-1$ | $-1$ |
| $-c^1$ | 1 | $-3$ | 1 |
| $-c^2$ | $-1$ | 5 | $-1$ |

La solution $x_1^* = (1,0)$ est obtenue mais n'est pas efficace (test d'efficacité) et la solution entière obtenue de la résolution de $(Px_1^*)$, $(3,1)$ est efficace. $H_1 = \{6,2\} \neq \emptyset$,

appliquer la coupe efficace $x_2 + x_6 \geq 1$ et la méthode duale du simplexe :

| | $x_7$ | $x_3$ | $b_i$ |
|---|---|---|---|
| $x_6$ | $-\frac{3}{5}$ | $-\frac{1}{5}$ | $\frac{1}{5}$ |
| $x_4$ | $\frac{1}{5}$ | $\frac{2}{5}$ | $\frac{13}{5}$ |
| $x_5$ | $\frac{2}{5}$ | $-\frac{1}{5}$ | $\frac{6}{5}$ |
| $x_1$ | $-\frac{1}{5}$ | $-\frac{2}{5}$ | $\frac{2}{5}$ |
| $x_2$ | $-\frac{2}{5}$ | $\frac{1}{5}$ | $\frac{4}{5}$ |
| $-\varphi$ | $-1$ | $0$ | $-2$ |
| $-c^1$ | $-\frac{3}{5}$ | $\frac{4}{5}$ | $-\frac{6}{5}$ |
| $-c^2$ | $\frac{7}{5}$ | $-\frac{6}{5}$ | $\frac{14}{5}$ |

La solution optimale est $x_2^* = (\frac{2}{5}, \frac{4}{5})$, nous utilisons la technique de branchement : $x_1 \leq 0$ ou $x_1 \geq 1$ :

- Pour $x_1 \leq 0$, le programme $(OI)$ devient irréalisable ;
- La solution optimale obtenue $x_3^* = (1, 1/2)$ n'est pas entière :

| | $x_8$ | $x_3$ | $b_i$ |
|---|---|---|---|
| $x_6$ | $-\frac{1}{2}$ | $-\frac{1}{2}$ | $\frac{1}{2}$ |
| $x_4$ | $1$ | $0$ | $2$ |
| $x_5$ | $-\frac{1}{2}$ | $\frac{1}{2}$ | $\frac{3}{2}$ |
| $x_1$ | $-1$ | $0$ | $1$ |
| $x_2$ | $\frac{1}{2}$ | $-\frac{1}{2}$ | $\frac{1}{2}$ |
| $x_3$ | $-\frac{5}{2}$ | $\frac{1}{2}$ | $\frac{3}{2}$ |
| $-\varphi$ | $0$ | $-1$ | $-2$ |
| $-c^1$ | $2$ | $-1$ | $0$ |
| $-c^2$ | $-3$ | $2$ | $1$ |

Deux sous problèmes sont crées (processus de branchement) :

Pour $x_2 \leq 0$, le tableau suivant est obtenu :

| | $x_7$ | $x_9$ | $b_i$ |
|---|---|---|---|
| $x_6$ | $-1$ | $-1$ | 1 |
| $x_4$ | 1 | 2 | 1 |
| $x_5$ | 0 | $-1$ | 2 |
| $x_1$ | $-1$ | $-2$ | 2 |
| $x_2$ | 0 | 1 | 0 |
| $x_3$ | $-2$ | $-5$ | 4 |
| $x_8$ | $-1$ | $-2$ | 1 |
| $-\varphi$ | $-1$ | 0 | $-2$ |
| $-c^1$ | 1 | 4 | 2 |
| $-c^2$ | $-1$ | $-6$ | $-2$ |

La solution optimale est $x_4^* = (2,0)$ qui n'est pas efficace (test d'efficacité), alors $\varphi_{opt} = -2$, $x_{opt} = (2,0)$. Pour $x_2 \geq 1$, nous avons :

| | $x_9$ | $x_8$ | $b_i$ |
|---|---|---|---|
| $x_6$ | $-1$ | $-1$ | 1 |
| $x_4$ | 0 | 1 | 2 |
| $x_5$ | 1 | 0 | 1 |
| $x_1$ | 0 | $-1$ | 1 |
| $x_2$ | $-1$ | 0 | 1 |
| $x_3$ | 1 | $-2$ | 1 |
| $x_7$ | 2 | $-1$ | 1 |
| $-\varphi$ | $-2$ | $-1$ | $-3$ |
| $-c^1$ | $-2$ | 1 | $-1$ |
| $-c^2$ | 4 | $-1$ | 3 |

La solution optimale est $x_5^* = (1,1)$ et le nœud est sondé puisque $\varphi_{opt} = -2$, $\varphi(x_5^*) = -3$ alors $\varphi_{opt} > \varphi(x_5^*)$. D'où, la solution optimale de $(OI)$ est : $x_{opt} = (2,0)$ et $\varphi_{opt} = -2$.

## 4.6 Expérimentation et étude comparative

La méthode proposée a été codée à l'aide de MATLAB R2013a et exécutée sur un ordinateur personnel avec un processeur Core i7 de 2,7 GHz et 4 Go de mémoire. Notons que les codes relatifs aux deux méthodes ont été entièrement réalisés par nous même (toutes les fonctions ont été programmées) et qu'aucun package n'a été utilisé. De plus, pour tester l'efficacité de notre algorithme, la méthode décrite dans [10] (méthode de Jorge) a également été programmée en utilisant le même environnement afin de comparer les performances des deux méthodes.
Pour ce faire, nous avons choisi la troisième classe de problèmes de tests fournie par Gokhan Kirlik & Serpil Sayin sur le site `//home.ku.edu.tr/~moolibrary/`. Ces instances de test concernent la programmation multiobjectif linéaire en nombres entiers (MOILP) avec $m$ contraintes, $m \in \{10, 15, 20\}$, $n$ variables, $n = 2m$ et $r$ fonctions objectif, $r \in \{3, 4, 5\}$. Cependant, les coefficients de la fonction $\varphi$ sont générés aléatoirement dans [1,100]. Au total, 90 instances ont été résolues à raison de 10 instances pour chaque triplet $(r, n, m)$.

Afin de comparer les performances des deux méthodes, nous avons choisi le temps d'exécution en secondes, le nombre de solutions efficaces rencontrées noté $|Eff|$ et le nombre de programmes en nombres entiers résolus noté $|ILP|$ pour être les métriques d'évaluation. En effet, les tableaux 1 et 2 résument les résultats obtenus en utilisant les deux méthodes sur plusieurs instances identiques.

TABLE 4.1 – Résutats pour $r = 3$

| | Méthode LMOILP | | | | | | | Méthode Jorge | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| (n,m) | $\|ILP\|$ | $\|Eff\|$ | | | $CPU(s)$ | | | $\|ILP\|$ | $\|Eff\|$ | | | $CPU(s)$ | | |
| | moy | moy | min | max | moy | min | max | moy | moy | min | max | moy | min | max |
| (20,10) | 1 | 7 | 3 | 21 | 51,86 | 0,77 | 378,7 | 9,3 | 4,9 | 2 | 14 | 415,78 | 9,79 | 1766,9 |
| (30,15) | 1 | 11,9 | 4 | 22 | 267,42 | 29,04 | 1251,3 | 18,3 | 9,6 | 2 | 23 | 1654,73 | 100,22 | 6654,53 |
| (40,20) | 1 | 7,1 | 1 | 15 | 508,37 | 0,08 | 2500 | 10,6 | 5,7 | 1 | 15 | 1558,22 | 0,11 | 5010,9 |

Nous pouvons voir que notre méthode surpasse la méthode de Jorge [10] sur la plupart des instances considérées dans cette étude, en terme de temps CPU avec un minimum de déviation de CPU $0s$ et un écart maximum de CPU $16294,96s$. L'écart minimum égal à $0s$ est dû à certains cas où une seule solution efficace est

TABLE 4.2 – Résultats pour $r = 4$

| | Méthode LMOILP | | | | | | | Méthode Jorge | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| (n,m) | $\|ILP\|$ | $\|Eff\|$ | | | $CPU(s)$ | | | $\|ILP\|$ | $\|Eff\|$ | | | $CPU(s)$ | | |
| | moy | moy | min | max | moy | min | max | moy | moy | min | max | moy | min | max |
| (20,10) | 1 | 5,8 | 1 | 10 | 3,32 | 0,01 | 15,72 | 6,9 | 3,8 | 1 | 8 | 168,19 | 0,01 | 832,12 |
| (30,15) | 1 | 10,2 | 1 | 24 | 592,11 | 0,02 | 2594,54 | 11,2 | 6,1 | 1 | 18 | 3406,17 | 0,02 | 18889,5 |
| (40,20) | 1 | 13,7 | 1 | 28 | 479,49 | 0,09 | 1327,51 | 8,64 | 4,9 | 1 | 8 | 1442,85 | 0,12 | 7251,3 |

TABLE 4.3 – Résultats pour $r = 5$

| | Méthode LMOILP | | | | | | | Méthode Jorge | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| (n,m) | $\|ILP\|$ | $\|Eff\|$ | | | $CPU(s)$ | | | $\|ILP\|$ | $\|Eff\|$ | | | $CPU(s)$ | | |
| | moy | moy | min | max | moy | min | max | moy | moy | min | max | moy | min | max |
| (20,10) | 1 | 4,7 | 1 | 10 | 1,92 | 0,01 | 10,85 | 4,8 | 2,7 | 1 | 6 | 27,86 | 0,01 | 193,03 |
| (30,15) | 1 | 10,73 | 1 | 35 | 181,36 | 0,02 | 1672,46 | 8,2 | 5,6 | 1 | 14 | 1802,35 | 0,01 | 6198,29 |
| (40,20) | 1 | 11,3 | 1 | 19 | 140,52 | 0,04 | 464,8 | 8,2 | 6,5 | 1 | 19 | 541,28 | 0,04 | 1740,56 |

trouvée pour les deux méthodes. En outre, la méthodologie proposée est construite de sorte qu'un seul programme entier est résolu (en une seule exécution). Alors que la méthode de Jorge [10] est influencée par le nombre de solutions efficaces trouvées, c'est-à-dire derrière chaque solution efficace trouvée, il existe un programme linéaire en nombres entiers résolu augmenté d'un certain nombre de variables et de contraintes.

Nous constatons que le nombre de solutions efficaces rencontrées pour la méthode de Jorge [10] est meilleur que dans le cas de la méthodologie proposée avec un écart minimum $(-1)$ et un écart maximum de 21 . En moyenne, nous n'avons que des solutions efficaces d'environ 4 comme déviation. Et nous confirmons, à travers cette étude, que notre méthode n'est pas influencée par le nombre de solutions efficaces trouvées.

En plus de cela, notre méthode a l'avantage d'être appliquée à des problèmes avec des coefficients réels des fonctions objectif tandis que la méthode de Jorge [10] ne peut être appliquée que pour des coefficients de fonctions objectifs entiers comme décrit par l'auteur. De plus, notre méthode peut être généralisée pour résoudre le problème de l'optimisation d'une fonction hyperbolique sur l'ensemble efficient d'un problème de programmation linéaire fractionnaire multiobjectif.

Pour une meilleure représentation , le temps d'exécution pour les deux méthodes était représenté par un histogramme en pourcentage Jorge/Méthode LMOILP %

pour $CPU$ et $|Eff|$ et Méthode-Proposée/Jorge % pour $|ILP|$. La moyenne est prise pour chaque triplet $(r,n,m)$.

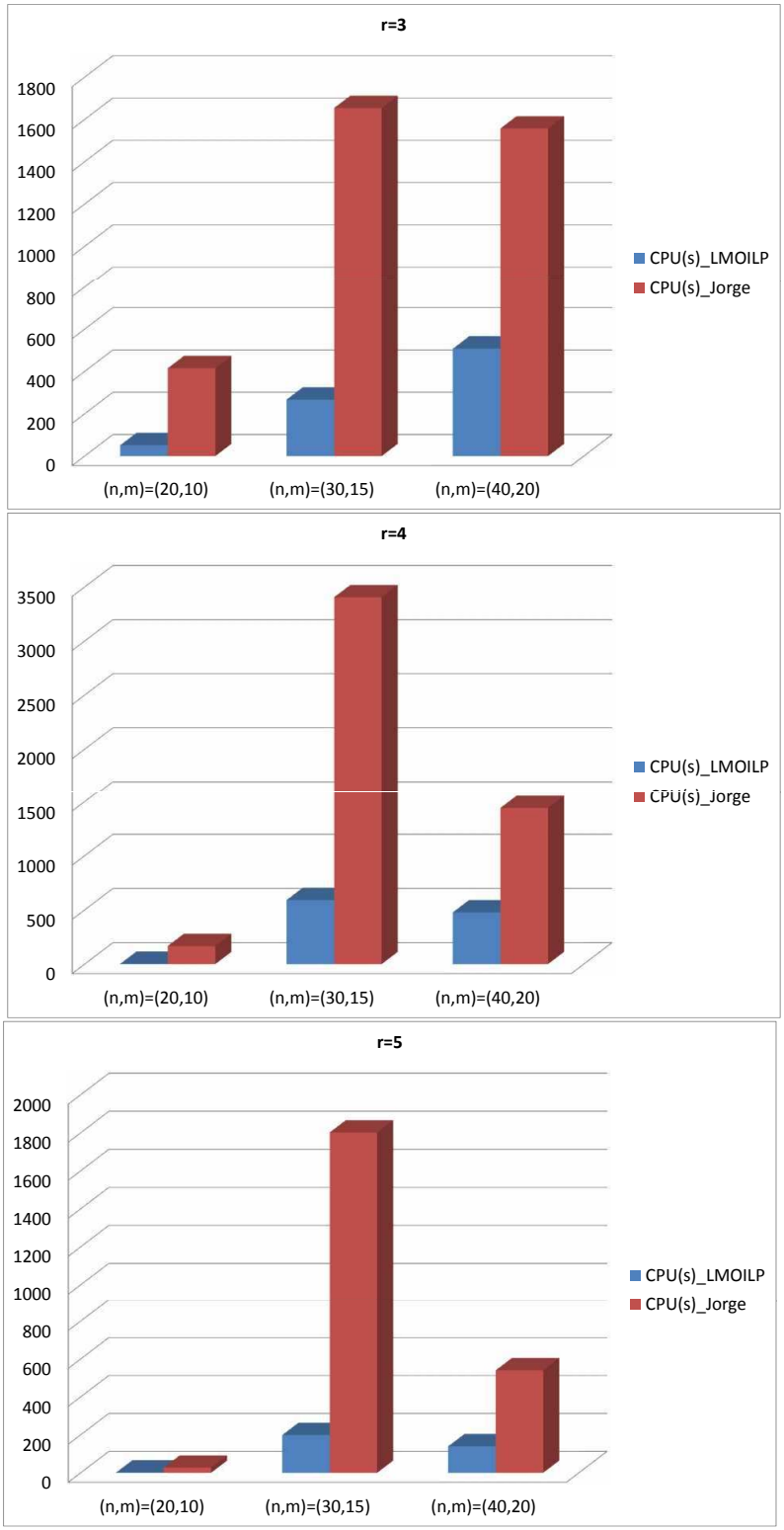


FIGURE 4.2 – Comparaison de CPU(s) pour chaque méthode

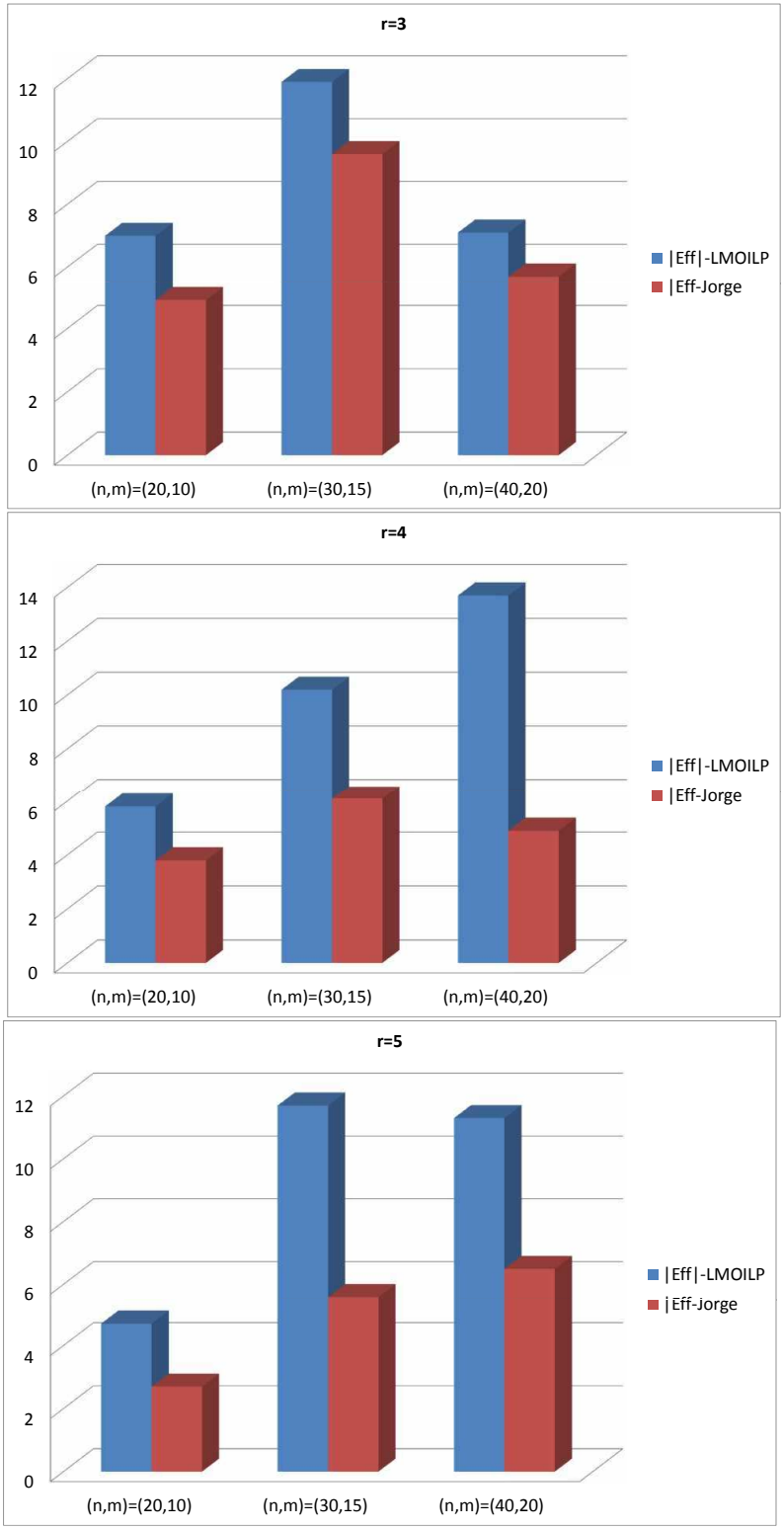


FIGURE 4.3 – Comparaison de |Eff| pour chaque méthode

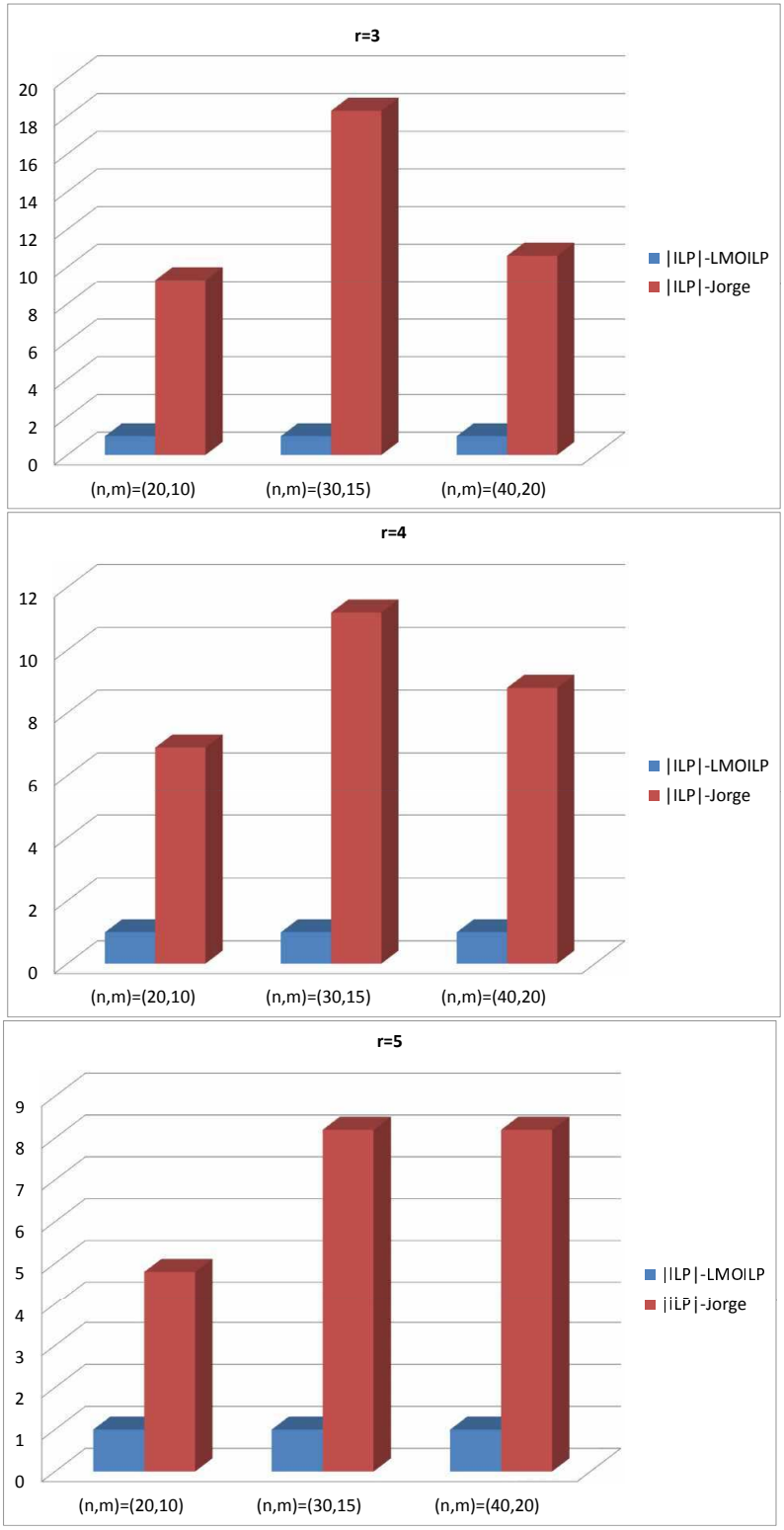


FIGURE 4.4 – Comparaison de |ILP| pour chaque méthode

## 4.7 Conclusion

Dans cet article, nous avons proposé une méthode basée sur l'algorithme "branch & bound" pour optimiser une fonction linéaire sur l'ensemble efficient d'un problème MOILP. Deux types de coupes sont utilisés, les coupes de *type 1* consacrées à éviter la recherche dans les domaines ne contenant pas de solutions efficaces et celles de *type 2* éliminant les domaines ne contenant pas de solutions optimales. La lecture des résultats de l'expérimentation montre que la méthode proposée surpasse la méthode de Jorge [10] presque à 100%, en outre, elle a l'avantage de pouvoir être appliquée même si les coefficients des fonctions objectif sont réels. Un autre avantage réside dans le fait que notre méthode peut être étendue à d'autres problèmes d'optimisation globaux avec des fonctions objectif non linéaires, notamment pour les problèmes où les fonctions objectif sont hyperboliques.

CHAPITRE 5

# UNE MÉTHODE BRANCH-AND-CUT POUR LA RÉSOLUTION DU PROBLÈME MULTIOBJECTIF QUADRATIQUE EN NOMBRES ENTIERS

## 5.1 Introduction

En optimisation, il est y a eu consensus que la programmation linéaire est un outil puissant pour résoudre de nombreuses applications du monde réel. Néanmoins, ces modèles ne sont pas toujours de forme linéaire, ils constituent une classe particulière de modèles plus généraux, les modèles non linéaires. De plus, dans la plupart des problèmes du monde réel, il ne s'agit pas d'optimiser seulement un seul critère mais plutôt d'optimiser simultanément plusieurs critères et qui sont généralement conflictuels.

Dans cette étude, nous traitons de l'optimisation multiobjectif non linéaire et contribuons à la littérature en instaurant une nouvelle méthode exacte pour la résolution du problème de la programmation quadratique multiobjectif en nombres entiers.

Nous avons d'abord mené une étude bibliographique approfondie, à la recherche de travaux liés à notre sujet, mais en vain. Nous trouvons le problème d'affectation quadratique multiobjectif qui a été étudié en utilisant des métaheuristiques, voir

par exemple les travaux dans [61] et [62], et aucune méthode exacte n'existe jusqu'à maintenant pour résoudre le problème de programmation multiobjectif quadratique en nombres entiers MOIQP. En outre, le problème de la gestion de portefeuille [15] est une application de l'optimisation quadratique avec des variables continues. Un nombre important d'articles ont été proposés dans la littérature traitant cette thématique. Nous citons en particulier dans un ordre chronologique, [63], [64], [65] et [66].

Cependant, et considérons le cas d'un seul objectif, le problème d'affectation quadratique et le problème de sac à dos quadratique ont reçu beaucoup plus d'attention, voir par exemple [67] et [68]. De plus, deux articles récents traitant de l'optimisation quadratique fractionnaire en nombres entiers dans les version monobjectif et multiobjectif et sont proposés respectivement dans [69] et dans [70]. Dans [69], les auteurs proposent de résoudre le problème de programmation fractionnaire quadratique monobjectif en nombres entiers avec des variables bornée. Ils ont développé une méthode de classement des solutions entières réalisables en établissant une fonction linéaire ou fractionnaire linéaire, qui agissent comme une borne inférieure sur les valeurs de la fonction objectif sur tout l'ensemble réalisable. Ceci est rendu possible en utilisant le principe de coupes planes. Seuls les exemples numériques sont présentés.

Cependant dans [70], un algorithme pour résoudre le problème de programmation quadratique fractionnaire biobjectif en nombres entiers est présenté. Les auteurs proposent de le résoudre en utilisant la méthode $\epsilon$-contrainte et une approche de classement des solutions entières réalisables pour trouver tous les points non-dominés. Une fonction linéaire ou fractionnaire linéaire agissant comme une borne inférieure sur les valeurs de la première fonction objectif du problème biobjectif sur tout l'ensemble réalisable est utilisée le long du processus de résolution. Des exemples numériques sont également présentés à l'appui des résultats théoriques.

De toutes ces considérations et de tous ces détails, nous avons pris notre motivation pour résoudre le programme mathématique MOIQP. En effet, l'algorithme proposé est nouveau et général et peut être appliqué par des changements appropriés pour résoudre des problèmes pratiques comme le problème d'affectation qua-

dratique multiobjectif, le problème de sac-à-dos quadratique multiobjectif et le problème de la gestion de portefeuille avec plus de deux fonctions quadratiques.

Dans cette contribution, une généralisation de notre méthode dans le cas linéaire [17] est décrite pour générer l'ensemble de toutes les solutions efficaces du programme MOIQP [16]. En utilisant le principe de séparation et évaluation, notre approche ne nécessite qu'une seule optimisation linéaire par itération en appliquant la méthode du simplexe et explore le domaine réalisable en recherchant intelligemment des solutions potentiellement efficaces. Ceci est rendu possible en utilisant des coupes planes construites à partir des directions d'améliorations des fonctions quadratiques fournis par les gradients correspondants, ce qui permet de neutraliser considérablement des solutions non efficaces sans avoir à les calculer. La gestion de toutes ces opérations est effectuée dans une arborescence structurée où le processus de séparation génère des solutions potentiellement efficaces. D'autre part, à chaque nœud de la recherche arborescente, un processus d'évaluation est activé en calculant des bornes plus restrictives afin de réduire l'espace de recherche. L'ensemble des vecteurs critères potentiellement non dominés générés progressivement par l'algorithme est utilisé comme borne supérieure et le point idéal local comme borne inférieure.

Le présent chapitre est organisé de la manière suivante. Dans la section 1, nous formulons mathématiquement le problème et donnons quelques définitions. Les résultats théoriques sont rassemblés dans la section 2. Le principe de la méthode et l'algorithme sont énoncés dans la section 3. Les résultats de calcul sont donnés dans la section 4. Dans la section 5, une illustration est détaillé pour comprendre les différentes étapes de l'algorithme. Enfin, des remarques et quelques perspectives sont données à la section 6.

## 5.2 Définitions et notations

Nous considérons le problème de programmation multiobjectif quadratique en nombres entiers suivant :

$$(P)\begin{cases} Min \quad f_i(x) = \dfrac{1}{2}x^t Q_i x + c_i^t x,\ i = 1,\cdots,r \\ x \in D \end{cases} \tag{5.1}$$

où $Q_i, i = 1,\ldots,r$ sont des $n \times n$ matrices semi-définies positives, $c_i, i = 1,\ldots,r$ sont des $n$-vecteurs réels et $D = S \cap \mathbb{Z}^n$ avec $S = \{x \in \mathbb{R}^n_+ \mid Ax \leq b\}$ est le domaine relaxé, supposé être compact du problème relaxé $(P)$ et $\mathbb{Z}^n$ est l'ensemble des vecteurs entiers. Avec $A$ étant une $m \times n$-matrice et $b$ un $m$-vecteur, nous supposons aussi que $A$ et $b$ sont des entiers, et que les fonctions $f_i : \mathbb{R}^n \to \mathbb{R}^r$, $i = 1,\ldots,r$ sont différentiables en tout point de $S$.

Afin de décrire notre procédure pour générer les solutions efficaces du programme $(P)$, les notations suivantes sont utilisées tout au long du document : Soit $x_l^*$ une solution entière de $(P)$ obtenu à l'étape $l$ de la procédure de résolution de $(P)$ en appliquant la méthode du simplexe.

- L'assemble de toutes les solutions efficaces est noté $E$ et l'ensemble des vecteurs critères non-dominés $v \in V$ est noté $SND$ ;

- Nous pouvons écrire $f_i$, $i = 1,\cdots,r$, au voisinage de $x_l^*$ comme suit :

  $$f_i(x) = f_i(x_l^*) + \nabla f_i(x_l^*)(x - x_l^*) + (x - x_l^*)\epsilon_i(x - x_l^*)$$

  où $\epsilon_i : \mathbb{R}^n \to\!\!\to \mathbb{R}$, avec $\lim\limits_{x \to x_l^*} \epsilon_i(x - x_l^*) = 0, \quad \forall i = 1,\cdots,r$ ; représente le reste de l'approximation de $f_i(x)$ par $f_i(x_l^*) + \nabla f_i(x_l^*)(x - x_l^*)$, $i = 1,\cdots,r$ ; au voisinage de la solution $x_l^*$ ;

- L'expression linéaire $\nabla f_i(x_l^*)(x - x_l^*)$ permet de détecter les directions d'amélioration des critères le long des arêtes du domaine réalisable provenant du point $x_l^*$ ;
- Dans le tableau du simplexe, par $B_l$ et $N_l$, nous désignons les ensembles des indices des variables de base et hors-base respectifs ;

- Dans l'équation $x_k = \hat{b}_{p(k)} - \sum_{j=1}^{n} \hat{a}_{p(k)j} x_j$ pour tout indice $k \in B_l$, $p(k)$ indique la position de la variable de base $x_k$ d'une solution réalisable $x$ , $\hat{a}_{p(k)j}$ et $\hat{b}_{p(k)}$ désigne les valeurs actualisées, dans le tableau du simplexe, des éléments de la matrices $A$ et le vecteur $b$, respectivement ;
- Les coordonnées du point idéal $id^l$ au nœud $l$ sont calculées par l'optimisation individuelle de chacun des critères quadratiques sur $S_l$, comme suit : $id_i^l = Min_{x \in S_l} f_i(x)$ , $i = 1, \cdots, r$ ; tels que $id^l = (id_1^l, \cdots, id_r^l)$ ;
- L'ensemble $H_l$ ci-après désigne les directions d'améliorations des critères $f_i$, $i = 1, \cdots, r$ :

$$\begin{aligned} H_l = &\left\{ j \in N_l | \exists i \in \{1, \ldots, r\}; \left( \alpha_j - \sum_{k \in B_l \cap \{\overline{1,n}\}} \nabla f_i(x_l^*)_k \hat{a}_{p(k)j} \right) < 0 \right\} \cup \\ &\left\{ j \in N_l | \forall i \in \{1, \ldots, r\}; \left( \alpha_j - \sum_{k \in B_l \cap \{\overline{1,n}\}} \nabla f_i(x_l^*)_k \hat{a}_{p(k)j} \right) = 0 \right\} \end{aligned} \tag{5.2}$$

  où $\nabla f_i(x_l^*)_k$ est la $k$-ème coordonnée du vecteur gradient du critère $i$, en $x_l^*$, et $\alpha_j$ est défini par :

$$\alpha_j = \begin{cases} \nabla f_i(x_l^*)_j \text{ if } j \in N_l \cap \{\overline{1,n}\} \\ 0 \qquad \text{if } j \in N_l \setminus \{\overline{1,n}\} \end{cases}$$

- La relation suivante en $x_l^*$ défini une coupe efficace dont l'adjonction en $x_l^*$ a pour effet de tronquer des régions contenant des solutions non efficaces :

$$\sum_{j \in H_l} x_j \geq 1 \tag{5.3}$$

- L'ensemble suivant décrit le domaine des solutions réalisables à l'étape $l+1$ lorsque $H_l \neq \emptyset$ :

$$S_{l+1} = \left\{ x \in S_l | \sum_{j \in H_l} x_j \geq 1 \right\}, S_0 = S \tag{5.4}$$

**Définition 5.1.** Une solution entière $x \in S_l$ est appelée *potentiellement efficace* si à l'étape $l$ de la procédure, le vecteur critère correspondant n'est dominé par aucun vecteur critère d'une autre solution entière $y \in S_l$ générée jusque là.

## 5.3 Résultats théoriques

Dans cette section, nous donnons les résultats théoriques justifiant les différentes étapes de la méthode proposée. Pour ce faire, nous introduisons l'ensemble

suivant :

$$G_{l+1} = \left\{ x \in S_l | \sum_{j \in N_l \setminus H_l} x_j \geq 1 \right\} \cap \mathbb{Z}^n,\ l \geq 0$$

Nous considérons aussi l'ensemble $D_l = S_l \cap \mathbb{Z}^n$ et $x_l^* \in D_l$ la solution optimale entière obtenu à partir du tableau du simplexe à l'étape $l$. Les résultats suivants garantissent qu'aucune solution efficace ne sera omise quel que soit le stade $l$ tel que $D_{l+1} \subset D_l$.

**Lemme 5.1.** $D_l \setminus \{x_l^*\} = D_{l+1} \cup G_{l+1}$

***Démonstration.*** Soit $x \in D_l$, $x \neq x_l^*$. Alors $x$ est dans le domaine fermé généré par la coupe de Dantzig [71] $\sum_{j \in N_l} x_j \geq 1$. Comme les ensembles $H_l$ et $N_l \setminus H_l$ définissent une partition de l'ensemble $N_l$, la coupe de Dantzig peut être réécrite de a façon suivante :

$$\sum_{j \in H_l} x_j + \sum_{j \in N_l \setminus H_l} x_j \geq 1$$

. Si la solution $x$ satisfait les inégalités $\sum_{j \in H_l} x_j \geq 1$, alors $x \in D_{l+1}$. Sinon, $\sum_{j \in N_l \setminus H_l} x_j \geq 1$ et donc $x \in G_{l+1}$. Par conséquent, $x \in D_{l+1} \cup G_{l+1}$, et $D_l \setminus \{x_l^*\} \subseteq D_{l+1} \cup G_{l+1}$.
D'un autre côté, il est clair que $D_{l+1} \cup G_{l+1} \subseteq D_l \setminus \{x_l^*\}$ et nous pouvons conclure que l'égalité est vraie. ■

**Théorème 5.1.** *Soit $x \neq x_l^*$ une solution efficace dans le domaine $D_l$, alors $x$ est dans le domaine $D_{l+1}$.*

***Démonstration.*** Par contraposée, soit $x \in D_l$, $x \neq x_l^*$. Supposons que $x \notin D_{l+1}$, alors $\sum_{j \in H_l} x_j < 1$ ce qui signifie que $x_j = 0 \ \ \forall j \in H_l$, $x$ étant entière.
D'un autre côté, lorsque l'on ne considère que les variables originelles du problème ($P$), c'est-à-dire les $n$ premières composantes des solutions $x_l^*$ et $x$, nous pouvons calculer ce qui suit :

$$\nabla f_i(x_l^*)(x - x_l^*) = -\nabla f_i(x_l^*)x_l^* + \nabla f_i(x_l^*)x \ \ \forall i = 1, \cdots, r$$

Également, on a du tableau du simplexe en $x_l^*$, la $k-$è$me$ coordonnée de la solution $x$ peut être réécrite comme :

$$x_k = \widehat{b}_{p(k)} - \sum_{j \in N_l} \widehat{a}_{p(k)j} x_j$$

pour tout indice $k \in B_l$ où $p(k)$ indique la position de la variable de base $x_k$ de $x$ et $\hat{a}_{p(k)j}$ les valeurs actualisées des éléments de la matrice des contraintes par la méthode du simplexe. Maintenant, nous pouvons écrire ce qui suit :

$$\nabla f_i(x_l^*)x = \sum_{k \in B_l \cap \{\overline{1,n}\}} \nabla f_i(x_l^*)_k (\hat{b}_{p(k)} - \sum_{j \in N_l} \hat{a}_{p(k)j} x_j) + \sum_{j \in N_l \cap \{\overline{1,n}\}} \nabla f_i(x_l^*)_j x_j$$
$$\forall i = 1, \cdots, r$$

$$\nabla f_i(x_l^*)x = \sum_{k \in B_l \cap \{\overline{1,n}\}} \nabla f_i(x_l^*)_k \hat{b}_{p(k)} - \sum_{j \in N_l} \left[ \sum_{k \in B_l \cap \{\overline{1,n}\}} \left(\nabla f_i(x_l^*)\right)_k \hat{a}_{p(k)j} \right] x_j +$$
$$\sum_{j \in N_l \cap \{\overline{1,n}\}} \nabla f_i(x_l^*)_j x_j \quad \forall i = 1, \cdots, r$$

D'autre part, nous avons :

$$\nabla f_i(x_l^*)x_l^* = \sum_{k \in B_l \cap \{\overline{1,n}\}} \nabla f_i(x_l^*)_k \hat{b}_{p(k)}$$

Ainsi, nous obtenons :

$$\nabla f_i(x_l^*)(x - x_l^*) = \sum_{j \in N_l \cap \{\overline{1,n}\}} \left[ \nabla f_i(x_l^*)_j - \sum_{k \in B_l \cap \{\overline{1,n}\}} \nabla f_i(x_l^*)_k \hat{a}_{p(k)j} \right] x_j +$$
$$\sum_{j \in N_l \setminus \{\overline{1,n}\}} \left[ - \sum_{k \in B_l \cap \{\overline{1,n}\}} \nabla f_i(x_l^*)_k \hat{a}_{p(k)j} \right] x_j \quad \forall i = 1, \cdots, r$$

Donc :

$$\nabla f_i(x_l^*)(x - x_l^*) = \sum_{j \in N_l} \left[ \alpha_j - \sum_{k \in B_l \cap \{\overline{1,n}\}} \nabla f_i(x_l^*)_k \hat{a}_{p(k)j} \right] x_j \quad \forall i = 1, \cdots, r$$

où :

$$\alpha_j = \begin{cases} \nabla f_i(x_l^*)_j \text{ if } j \in N_l \cap \{\overline{1,n}\} \\ 0 \qquad \text{if } j \in N_l \setminus \{\overline{1,n}\} \end{cases}$$

En utilisant la définition de l'ensemble $H_l$ comme sous ensemble de $N_l$, nous pouvons écrire :

$$\begin{aligned} \nabla f_i(x_l^*)(x - x_l^*) = & \sum_{j \in H_l} \left( \alpha_j - \sum_{k \in B_l \cap \{\overline{1,n}\}} \nabla f_i(x_l^*)_k \hat{a}_{p(k)j} \right) x_j + \\ & \sum_{j \in N_l \setminus H_l} \left( \alpha_j - \sum_{k \in B_l \cap \{\overline{1,n}\}} \nabla f_i(x_l^*)_k \hat{a}_{p(k)j} \right) x_j \quad \forall i = 1, \cdots, r \end{aligned} \tag{5.5}$$

Mais comme il a été supposé $x_j = 0 \ \forall j \in H_l$, alors cette dernière expression est réduite à :

$$\nabla f_i(x_l^*)(x - x_l^*) = \sum_{j \in N_l \setminus H_l} \left( \alpha_j - \sum_{k \in B_l \cap \{\overline{1,n}\}} \nabla f_i(x_l^*)_k \hat{a}_{p(k)j} \right) x_j \quad \forall i = 1, \cdots, r \tag{5.6}$$

Selon la définition de l'ensemble $H_l$, pour tout indice $j$ tel que $j \in N_l \setminus H_l$, nous avons l'inégalité suivante :

$$\alpha_j - \sum_{k \in B_l \cap \{\overline{1,n}\}} \nabla f_i(x_l^*)_k \hat{a}_{p(k)j} \geq 0 \quad \forall i = 1, \cdots, r$$

avec au moins une inégalité stricte, d'où :

$$\nabla f_i(x_l^*)(x - x_l^*) \geq 0 \quad \forall i = 1, \cdots, r \tag{5.7}$$

avec au moins une inégalité stricte. Alors, nous avons l'inégalité suivante :

$$f_i(x_l^*) + \nabla f_i(x_l^*)(x - x_l^*) \geq f_i(x_l^*) \quad \forall i = 1, \cdots, r \tag{5.8}$$

avec au moins une inégalité stricte. Des formules 5.7 et 5.8, nous concluons que :

$$f_i(x) \geq f_i(x_l^*) \ \ \forall i = 1, \cdots, r \tag{5.9}$$

avec au moins une inégalité stricte. Par conséquent, $f(x)$ est dominé par $f(x_l^*)$ signifiant que la solution $x$ n'est pas une efficace et par conséquent, toutes les solutions efficaces du domaine $D_l$ se trouvent dans le domaine $D_{l+1}$ ■

**Corollaire 5.1.** *Si $H_l = \emptyset$, alors le domaine $D_l \setminus \{x_l^*\}$ ne contient aucune solution efficace.*

***Démonstration.*** Partons de la définition de l'ensemble $H_l$, il ressort qu'aucune amélioration n'est possible en $x_l^*$ lorsque $H_l = \emptyset$. En effet, toutes les fonctions objectif sont croissantes dans les directions admissibles $N_l$. D'où, $f(x_l^*)$ domine tous les vecteurs critères $f(x), \forall x \in D_l \setminus \{x_l^*\}$ (voir la preuve du théorème 5.1, équation 5.6). ■

**Théorème 5.2.** *L'algorithme proposé génère toutes les solutions efficaces du problème de programmation multiobjectif quadratique en nombres entiers $(P)$ et converge en un nombre fini d'étapes.*

***Démonstration.*** L'ensemble des solutions réalisables $D$ étant compact, il contient un nombre fini de solutions entières. A chaque étape $l$, l'algorithme détermine une

solution entière $x_l^*$ lorsqu'elle existe. En prenant en considération le lemme et le théorème précédents, on élimine au moins la solution $x_l^*$ quand la coupe efficace est rajoutée. D'autre part, lorsque l'ensemble $H_l$ est vide, la solution $x_l^*$ représente un point idéal relativement au domaine courant $D_l$ et signifiant donc, que ce dernier ne contient pas de solutions efficaces. ■

## 5.4 Principe de la méthode

La recherche des solutions entières de $S_l$ ne nécessite pas l'optimisation des fonctions $f_i$ sur $S_l$, et peut donc se faire en optimisant une fonction linéaire quelconque sur $S_l$, notamment lorsqu'au moins l'un des termes linéaires des fonctions objectif n'est pas nul, une agrégation des termes linéaires des critères. En effet , cette fonction est utilisée afin de scanner l'espace des décisions à la recherche de solutions potentiellement efficaces, qui contribuent, à la fin, à former l'ensemble efficace.

A l'étape $l$, $l \geq 0$ nous devons résoudre le programme linéaire suivant :

$$(P_l) \begin{cases} Min \quad z(x) = \sum_{i=1}^{r} \sum_{j=1}^{n} c_{ij} x_j, \; x \in S_l \end{cases}$$

Le principe de la méthode consiste en la recherche d'une solution entière $x^*$ potentiellement efficace pour le problème $(P)$ à travers la résolution du programme linéaire $(P_l)$, $l \geq 0$, et éventuellement, la procédure de séparation de la méthode "branch & bound". Nous utilisons également des bornes sur les fonctions objectif au nœud $l$ de l'arbre de recherche, en fixant les bornes inférieures aux coordonnées du point idéal et les bornes supérieures aux vecteurs de critères potentiellement non-dominés trouvés. En effet, si le point idéal est dominé par l'un des vecteurs critères potentiellement non-dominés déjà trouvés, le nœud correspondant est sondé pour éviter l'exploration des domaines ne contenant que des solutions non efficaces. Puis, il est bien connu de la programmation mathématique qu'une fonction différentiable diminue dans la direction opposée à son vecteur gradient pour un problème de minimisation. Une fois ces directions détectées, l'ensemble $H_l$ est construit à partir du calcul de l'expression $\nabla^t f_i(x^*)x$, $\forall x \in S_l$, en fonction des variables hors base de

$x^*$ et permettra la reconnaissance des directions possibles d'amélioration des fonctions $f_i$. L'ensemble $H_l$ de toutes les directions d'amélioration des critères est alors construit et l'adjonction de la coupe efficace a pour effet de supprimer la solution entière courante $x^*$ en même temps qu'un sous ensemble de solutions entières non efficaces. Le processus réitère à la recherche d'une autre solution entière potentiellement efficace pour $(P)$. Nous montrons également que l'introduction d'une telle coupe donne tout l'ensemble efficace en un nombre fini d'étapes.

Le pseudo-code de la méthode proposée est donné dans l'algorithme 1. Nous rappelons que $E$ est l'ensemble des solutions efficaces (output) et $SND$ l'ensemble des vecteurs critères non-dominés. L'algorithme commence par définir l'ensemble $E$ comme l'ensemble de toutes les solutions potentiellement efficaces connues a priori, par exemple, nous prenons les solutions optimales issues de l'optimisation individuelle des critères sur l'ensemble $D$ ou bien générées par une heuristique, et $SND$, l'ensemble des vecteurs critères potentiellement non-dominés correspondant. L'indice d'itération $l$ est initialisé à 0 et le domaine relaxé initial $S_0$ à l'ensemble $S$. Ensuite, itérativement, suit la résolution des problèmes $(P_l)$ et la recherche de solutions réalisables entières est organisée dans une structure en arbre. Chaque fois qu'une solution réalisable entière $x_l^*$ est trouvé à un nœud $l$, les coordonnées du point idéal $(id^l)$ sont calculées à travers la résolution individuelle des programmes quadratiques, les ensembles $SND$ et $E$ sont respectivement mis à jour et la coupe efficace 5.3 est ajoutée pour supprimer $x_l^*$ ainsi qu'un sous-ensemble de solutions non efficaces du domaine $S_l$. Tous les programmes obtenus en ajoutant de nouvelles coupes (coupe efficace ou séparation) sont ré-optimisés en utilisant la méthode duale du simplexe. L'algorithme s'arrête dans trois cas : si l'ensemble $S_l$ devient irréalisable, ou si à un stade donné $l$, $id^l$ est dominé par le vecteur critère d'au moins un vecteur de $SND$ ou aucune amélioration des critères n'est possible selon la définition de $H_l$.

**Données :** $r$ fonction quadratiques $f = (f_1, \cdots, f_r)$, et une matrice $m \times n$-$A$, et un $m$-vecteur $b$.

**Résultat :** $E$ et $SND$ : ensembles des solutions efficaces et l'ensemble des vecteurs non-dominés de MOIQP respectivement.

***Étape 1.*** (***Initialisation***)

$l \leftarrow 0$ créer le premier nœud, (nœud 0) par $(P_0) \leftarrow (P)$ ;

$E \leftarrow \{premières\ solutions\ potenntiellemeny\ efficaces\}$ ;

$SND \leftarrow \{premiers\ vecteurs\ critères\ non-dominés\}$.

***Étape 2.*** (***Étape générale***)

Soit $O$ l'ensemble des nœuds non stérilisés dans l'arborescence :

**tant que** *$O$ est non vide* **faire**

Choisir le nœud ayant a le plus grand indice $l$ dans $O$ et résoudre le programme correspondant $(P_l)$ en utilisant la méthode du simplexe ou duale du simplexe (initialement, pour résoudre $(P_0)$, seulement la méthode du simplexe est utilisée) :

**si** *$(P_l)$ est irréalisable* **alors**

nœud correspondant $l$ est saturé

**sinon**

soit $\overline{x}_l$ une solution optimale pour $(P_l)$ :

**si** *$\overline{x}_l$ est non entière* **alors**

aller à l'étape 2*a*;

**sinon**

mettre $x_l^* \leftarrow \overline{x}_l$ comme la solution optimale trouvé et aller à l'étape 2*b* ;

***Étape 2a.*** (***Processus de branchement***)

Soit $\overline{x}_{lj}$ une coordonnée fractionnaire $\overline{x}_l$ , créer à partir du nœud $l$ deux nœuds $l_1$ et $l_2$, $l_1 \geq l+1$, $l_2 \geq l+1$, $l_1 \neq l_2$, Actualiser $O$ ;

Ajouter la contrainte $x_j \leq \lfloor \overline{x}_{lj} \rfloor$ dans le premier nœud pour avoir $(P_{l1})$, et la contrainte $x_j \geq \lceil \overline{x}_{lj} \rceil$ dans le second pour avoir $(P_{l2})$, aller à l'étape 2.

***Étape 2b.*** (***Generation de coupe***)

**si** *$f(x_l^*)$ est non dominé par $f(y)$ pour toutes les solutions $y \in E$* **alors**

$E := E \cup \{x_l^*\}$ et $SND := SND \cup \{f(x_l^*)\}$

**sinon**

**si** *il existe $y_1 \in E$ tel que $f(y_1)$ est dominé par $f(x_l^*)$* **alors**

$E := E \setminus \{y_1\} \cup \{x_l^*\}$ et $SND := SND \setminus \{f(y_1)\} \cup \{f(x_l^*)\}$

**sinon**

Calculer le point idéal local $id^l$, déterminer les ensembles $B_l$, $N_l$ et construire $H_l$ défini par (3) :

**si** *$id^l$ est dominé par au moins des vecteurs potentiellement non dominés trouvés jusqu'ici ou bien $H_l = \emptyset$* **alors**

noeud correspondant est saturé,aller à l'étape 2

**sinon**

Ajouter la coupe $\sum_{j \in H_l} x_j \geq 1$ à $(P_l)$, $l = l+1$, mettre à jour $O$ et aller à l'étape 2

**Algorithme 1 :** *Ensemble Efficace Quadratique (EQUA)*

## 5.5 Exemple numérique

Considérons le problème $(P)$ donné par :

$$(P)\begin{cases} Min \quad f_1(x) = \frac{1}{2}x^t Q_1 x + c_1^t x \\ Min \quad f_2(x) = \frac{1}{2}x^t Q_2 x + c_2^t x \\ 8x_1 + 17x_2 + 14x_3 + 19x_4 \leq 69 \\ 9x_1 + 7x_2 + x_3 + 21x_4 \leq 88 \\ (x_1, x_2, x_3, x_4) \in \mathbb{N}^4 \end{cases}$$

où :

- $Q_1 = \begin{pmatrix} 21 & 39 & 13 & 16 \\ 39 & 75 & 23 & 28 \\ 13 & 23 & 12 & 13 \\ 16 & 28 & 13 & 15 \end{pmatrix}$ and $Q_2 = \begin{pmatrix} 35 & 19 & 33 & 35 \\ 19 & 18 & 25 & 30 \\ 33 & 25 & 46 & 40 \\ 35 & 30 & 40 & 52 \end{pmatrix}$
- $c_1^t = (-77, 604, -495, -155)$ and $c_2^t = (186, 575, -479, -746)$.

**Étape 1. Aller à Initialisation** :

- $l$=0
- $(P_0) \leftarrow (P)$, $S_0 \leftarrow S$
- $E = \{(1,0,4,0),(0,0,2,2)\}$, $SND = \{(-1898.5, -1212.5),(-1194, -2094)\}$ où (1,0,4,0) est la solution obtenue en optimisant $f_1$ sur $S$ et (0,0,2,2) est la solution obtenue en optimisant $f_2$ sur $S$.

Résoudre $(P_0)$, par rapport à la fonction linéaire

$$\sum_{i=1}^{2}\sum_{j=1}^{4} c_{ij}x_j = 109x_1 + 1179x_2 - 974x_3 - 901x_4$$

ce qui signifie résoudre le problème suivant :

$$(P_0)\begin{cases} Min \quad z(x) = 109x_1 + 1179x_2 - 974x_3 - 901x_4 \\ x \in S_0 \end{cases}$$

La méthode du simplexe est appelée pour résoudre le programme linéaire ($P_0$) et renvoie le tableau du simplexe optimal suivant :

| $B_0$ | $b$ | $x_1$ | $x_2$ | $x_5$ | $x_4$ |
|---|---|---|---|---|---|
| $x_2$ | 69/14 | 4/7 | 17/14 | 1/14 | 19/14 |
| $x_6$ | 1163/14 | 59/7 | 81/14 | −1/14 | 275/14 |
| $-z$ | 33603/7 | 4659/7 | 16532/7 | 487/7 | 2946/7 |

La solution optimale de ($P_0$) est $\bar{x}_0 = (0, 0, 69/14, 0)$, n'est pas entière, le processus de branchement est déclenché.**Aller à l' Étape 2a**.

**Étape 2a.** Créer deux nœuds 1 et 2 alors $O = \{1, 2\}$ :

- **Nœuds 2** : Ajouter au dernier tableau du simplexe la coupe $x_3 \leq 4$, et en appliquant la méthode duale du simplexe, la solution optimale entière obtenue $\bar{x}_2 = (0, 0, 4, 13/19)$ n'est pas entière.

  **Aller à l' Étape 2a**.

  **Étape 2a.** Séparer par rapport à $x_4$ et créer deux nœuds 3 et 4, $O = \{1, 3, 4\}$ :

  - **Nœuds 4** : $x_4 \leq 0$, appliquer la méthode duale du simplexe pour avoir la solution entière $x_4^* = (0, 0, 4, 0)$ comme il est montré dans le tableau suivant :

| $B_4$ | $b$ | $x_1$ | $x_2$ | $x_8$ | $x_7$ |
|---|---|---|---|---|---|
| $x_3$ | 4 | 0 | 0 | 0 | 1 |
| $x_6$ | 84 | 9 | 7 | -21 | -1 |
| $x_4$ | 0 | 0 | 0 | 1 | 0 |
| $x_5$ | 13 | 8 | 17 | -19 | -14 |
| $-z$ | 3896 | 109 | 1179 | 901 | 974 |
| $\left(\alpha_j - \sum_{k \in B_4 \cap \{\overline{1,n}\}} \nabla f_1(x_2^*)_k \hat{a}_{p(k)j}\right)$ | 1884 | -25 | 696 | 447 | 103 |
| $\left(\alpha_j - \sum_{k \in B_4 \cap \{\overline{1,n}\}} \nabla f_2(x_2^*)_k \hat{a}_{p(k)j}\right)$ | 1548 | 318 | 675 | 295 | 586 |

  **Aller à l' Étape 2b**.

  **Étape 2b.** Mettre à jour les ensembles

  $E = \{(1, 0, 4, 0), (0, 0, 2, 2), (0, 0, 4, 0)\}$ et

$SND = \{(-1898.5, -1212.5), (-1194, -2094), (-1884, -1548)\}$, $B_4 = \{3,6,4,5\}$ et $N_4 = \{1,2,7,8\}$. Calculer le point idéal local $id^4$ par l'optimisation individuelle de chacun des critères quadratiques sur $S_4$, nous obtenons $id^4 = (-1898.9, -1548)$ qui n'est dominée par aucun des vecteurs critères potentiellement non dominés dans $SND$. Construis donc l'ensemble $H_4$ selon **expression 2**, $H_4 = \{1\}$, appliquer la coupe efficace $x_1 \geq 1$ pour avoir $(P_5)$ et en utilisant la méthode duale du simplexe, la solution entière $x_5^* = (1,0,4,0)$ est atteinte.

**Aller à l' Étape 2b**.

**Étape 2b.** $E$ et $SND$ restent inchangés, $B_5 = \{3,6,4,5,1\}$ and $N_5 = \{9,2,7,8\}$. Calculer le point idéale $id^5$ sur $S_5$, qui est égale à $(-1998.9, -1212.5)$ et il n'est dominé par aucun vecteur de $SND$. Construire $H_5 = \{9\}$ et ajouter la coupe efficace $x_9 \geq 1$ pour obtenir $(P_6)$. La méthode du simplexe fournit la solution : $\bar{x}_6 = (2,0,53/14,0)$ qui est non entière, **Aller à l' Étape 2a**.

**Étape 2a.** Séparer selon la variable $x_3$ pour créer deux nœuds, donc $O = \{1,3,7,8\}$ :

- **Nœud 8** : en ajoutant la coupe $x_3 \leq 3$pour obtenir la solution optimale entière $x_8^* = (2,0,3,0)$.

  **Aller à l' Étape 2b**.

  **Étape 2b.** Aucun changement pour $E$ et $SND$ car $x_8^*$ n'est pas efficace. $B_8 = \{3,6,7,5,1,9,4\}$, $N_8 = \{2,11,8,10\}$. Calculer les coordonnés du point idéal local $id^8 = (-1465, -590)$. Dans ce cas, le point idéale $id^8$ est dominé par tous les vecteurs critères potentiellement non dominés dans $SND$, alors le nœud actuel stérilisé.

- **Nœud 7** : En ajoutant la coupe $x_3 \geq 4$, le programme obtenu $(P_7)$ est non réalisable.

- **Nœud 3** : ajouter la coupe $x_4 \geq 1$ et appliquer la méthode duale du simplexe pour obtenir la solution $\bar{x}_3 = (0,0,25/7,1)$. le processus de séparation est appliqué $x_3$ et l'ensemble $O$ est actualisé ; $O = \{1,9,10\}$

  - **Nœud 10** : la coupe $x_3 \leq 3$ est ajouté et la ré-optimisation du nou-

veau tableau donne la solution $\bar{x}_{10} = (0,0,3,27/19)$, nous séparons par rapport à $x_4$, $O = \{1,9,11,12\}$ :

- **Nœud 12** : Le rajout de la coupe $x_4 \leq 1$ fournit la solution entière $\bar{x}_{12} = x^*_{12} = (0,0,3,1)$ . **Aller à l' Étape 2b**.

  **Étape 2b.** Actualiser $E = E \cup \{(0,0,3,1)\}$ et
  $SND = SND \cup \{(-1539.5,-1830)\}$.
  $B_{12} = \{3,6,4,7,8,9\}$ et $N_{12} = \{1,2,10,9\}$. Le point idéal local est $id^{12} = (-1551,-1830)$ qui est non dominé par aucun des vecteurs critères potentiellement non dominés dans $SND$.
  Construire $H_{12} = \{1\}$. Appliquer la méthode duale du simplexe pour obtenir la solution $x^*_{13} = (1,0,3,1)$ qui est entière.
  **Aller à l' Étape 2b**.

  **Étape 2b.** $E$ et $SND$ restent les même, $B_{13} = \{3,6,4,7,8,5,1\}$, $N_{13} = \{11,2,9,10\}$. Les coordonnées du point idéal local sont calculées et $id^{13} = (-1551,-1492.5)$ est dominée par au moins un de vecteurs critères potentiellement non dominants de $SND$. Le nœud 13 est stérilisé.

- **Nœud 11** : $x_4 \geq 2$ On a $\bar{x}_{11} = (0,0,31/14,2)$, $O = \{1,9,14,15\}$

  - **Nœud 15** : $x_3 \leq 2$, nous continuons jusqu'à obtenir la solution $x^*_{15} = (0,0,2,2)$ qui est entière.
    **Aller à l' Étape 2b**.

    **Étape 2b.** Les ensembles $E$ et $SND$ restent inchangés.
    Calculer $id^{15} = (-1199.6,-2094)$ qui est non dominé. Construire $H_{15} = \{1\}$. Appliquer $x_1 \geq 1$, nous obtenons : $\bar{x}_{16} = (1,23/14,1,2)$, nous séparons en deux nœuds, $O = \{1,9,14,17,18\}$ :

    - **Nœud 18** : $x_2 \leq 1$, $x^*_{18} = (1,0,1,2)$, $E$ et $SND$ restent inchangés, car le point idéal $id^{18} = (-767.4,-1457.5)$ est dominé, nous stérilisons ce nœud.
    - **Nœud 17** : $x_2 \geq 2$, $(P_{17})$ irréalisable.

- **Nœud 14** : $x_3 \geq 3$ : $(P_{14})$ irréalisable.
- **Nœud 9** : $x_3 \geq 4$ : $(P_9)$ irréalisable.

- **Nœud 1** : $x_4 \geq 5$ : $(P_1)$ irréalisable.

L'ensemble $O$ étant vide, l'algorithme s'arrête et renvoie l'ensemble de toutes les solutions efficaces :

$$E = \{(1,0,4,0),(0,0,2,2),(0,0,4,0),(0,0,3,1)\}$$

et l'ensemble correspondant de vecteurs de critères non-dominés est :

$$SND = \{(-1898.,-1212.5),(-1194,-2094),(-1884,-1548),(-1539.5,-1830)\}$$

Il est à noter que parmi les 103 solutions réalisables entières de l'exemple décrit, seulement 8 sont générées. Deux outils ont été utilisés pour réduire considérablement le domaine ; le concept de point localement idéal et la coupe efficace proposée.

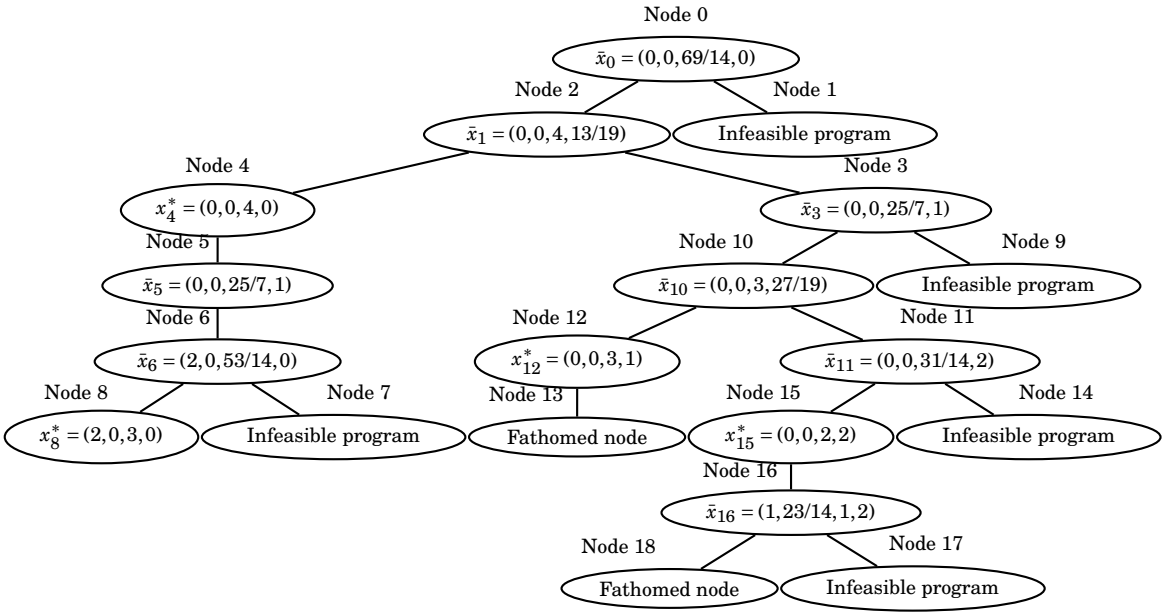


FIGURE 5.1 – L'arbre de recherche de l'exemple numérique

## 5.6 Expérimentation et résultats

Dans cette section, nous donnons quelques résultats relatifs à notre méthode qui était mise en œuvre dans le programme $Matlab\ R2012a$, en utilisant un PC intel(R)

Core(TM) i3 CPU, 2.13 GHz et 3GB RAM. Toutes les procédures ont été programmées et aucun package d'optimisation n'est utilisé. La méthode a été testée sur des instances générées aléatoirement. En effet, les $m$ contraintes ont été générées dans l'intervalle $\{20,25,60,80,100,120,140,160\}$ et $r$ fonctions objectif, $r \in \{3,5,7\}$. Les coefficients sont des entiers générées aléatoirement non corrélés et uniformément distribués dans l'intervalle $[1,30]$ pour les contraintes. Cependant, le nombre de variables $n$, est pris dans $\{30,40,50,80,100,120,140,160,180\}$. Les fonctions objectif sont générées de la manière suivante : initialement une matrice triangulaire est générée aléatoirement dans $[1,\ 5]$, puis multipliée par sa transposée pour former les matrices $Q_i$, $i = 1,\cdots,r$ dans $[1,\ 5^2 * n]$ (cet intervalle est le résultat de multiplication de $[1,\ 5] \times [1,\ 5]$ et le nombre de variables $n$). Procédons de cette manière, nous nous assurons que toutes les matrices sont semi définies positives. Les vecteurs $c_i$, $i = 1,\cdots,r$ sont générés dans $[-1000,1000]$. Pour chaque contrainte $j$, le vecteur $b_j$ est dans l'intervalle $[50,100]$. Pour chaque instance $(n,m,r)$, une batterie de 20 problèmes sont résolus et l'ensemble de toutes les solutions efficaces est entièrement formé.

TABLE 5.1 – Résultat pour les problèmes de tailles moyennes

| r | r=3 | | r=5 | | r=7 | |
|---|---|---|---|---|---|---|
| $n \times m$ | $\|E\|$ | CPU(s) | $\|E\|$ | CPU(s) | $\|E\|$ | CPU(s) |
| $30 \times 20$ | * 30.6 | 3.6 | 149 | 9.6 | 260.2 | 14.6 |
| | [13, 53] | [0.79, 5.95] | [119, 199] | [5.65, 13.71] | [152, 604] | [5.3, 48.8] |
| $30 \times 25$ | 32.3 | 4.7 | 138.3 | 12.1 | 225.9 | 17.1 |
| | [9, 71] | [1.5, 10.5] | [55, 288] | [5.7, 30.2] | [134, 387] | [8.2, 33.9] |
| $40 \times 20$ | 35.6 | 12.3 | 132.8 | 19.2 | 268.6 | 42.1 |
| | [13, 57] | [2.8, 36.2] | [51, 186] | [4.5, 30.8] | [180, 385] | [23.3, 94.5] |
| $40 \times 25$ | 38.4 | 11.3 | 129.6 | 19.3 | 259.4 | 39.1 |
| | [20, 71] | [3.9, 25.5] | [65, 235] | [9.864, 33.9] | [141, 366] | [23.8, 79.2] |
| $50 \times 20$ | 34.3 | 26.1 | 122.3 | 37.4 | 218.1 | 60.6 |
| | [14, 51] | [8.7, 59.4] | [87, 182] | [11.6, 70.7] | [129, 329] | [20.5, 113.8] |
| $50 \times 25$ | 33.7 | 27.6 | 110.9 | 32.6 | 208.1 | 58.8 |
| | [9, 53] | [8.8, 86.1] | [73, 171] | [10.6, 63.9] | [112 295] | [17.6, 135.7] |

TABLE 5.2 – Résultats pour les problèmes de grandes tailles

| r | r=3 | | r=5 | | r=7 | |
|---|---|---|---|---|---|---|
| $n \times m$ | $\|E\|$ | CPU(s) | $\|E\|$ | CPU(s) | $\|E\|$ | CPU(s) |
| $80 \times 60$ | 15.8<br>[4, 22] | 226.4<br>[40.2, 405] | 32.1<br>[21, 51] | 291.7<br>[65.8, 432.6] | 40.9<br>[15, 70] | 288.5<br>[73.1, 701.7] |
| $100 \times 80$ | 18.2<br>[15, 24] | 737.9<br>[312.6, 1624.5] | 31.6<br>[14, 48] | 581.6<br>[192.1, 1263.3 ] | 76.6<br>[73, 80] | 1111.5<br>[796.4, 1607.4] |
| $120 \times 100$ | 15.6<br>[8, 22] | 1500<br>[798.2, 2003.4] | 44.8<br>[39, 49] | 1690<br>[1342.2, 2112.4] | 77.8<br>[74, 86] | 2517.5<br>[1726.2, 3563.1] |
| $140 \times 120$ | 17.3<br>[10,24] | 4169.9<br>[1762.5,6384.9] | 44.9<br>[36 ,58] | 4331.4<br>[3155.3,6858.4] | 71.1<br>[61,79] | 6130.2<br>[3985.8,9124.8] |
| $160 \times 140$ | 14<br>[11,18] | 5865,52<br>[3907.7, 9430.8] | 32.8<br>[29,36] | 6242.17<br>[3738.36, 8395.14] | 56.75<br>[53,66] | 9165.05<br>[6970.1,11995] |
| $180 \times 160$ | 7.4<br>[6,9] | 12756<br>[4702.17 18747.09] | 12.2<br>[11,14] | 13598.07<br>[8284.17 17558.28] | 19.2<br>[11,23] | 18892.11<br>[8517.33,23840.84] |

(*) : Moyenne [Min,Max]

Les résultats de calcul sont présentés dans les tableaux 5.1 et 5.2, dans lesquels les problèmes de tailles moyennes et grande sont résolus respectivement. Pour déterminer la performance de notre algorithme, nous reportons dans les tableaux 5.1 et 5.2, la moyenne du temps CPU (en secondes) ainsi que la moyenne du nombre de solutions efficaces trouvées, noté $|E|$ pour chaque triplet $(n,m,r)$. De plus, le minimum et le maximum du temps CPU et $|E|$ sont indiqués entre parenthèses. Les figures suivantes illustrent mieux les résultats obtenus.

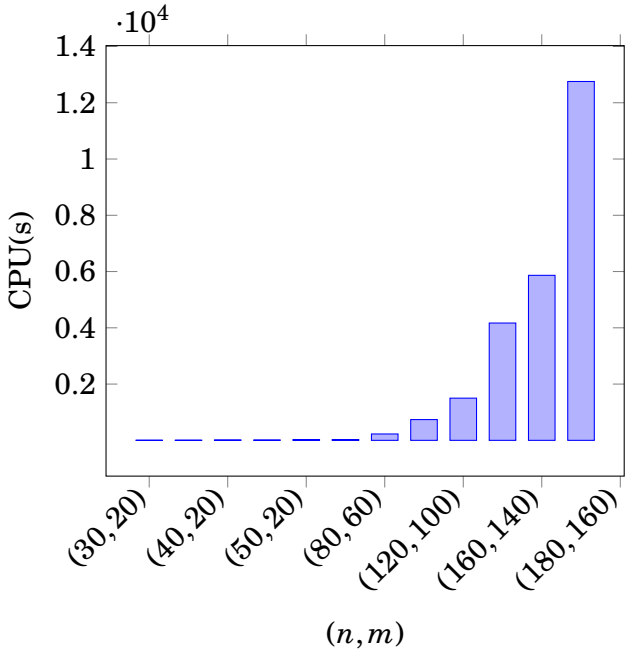


FIGURE 5.2 – Moyenne du temps de calcul pour chaque $(n,m)$ et $r = 3$

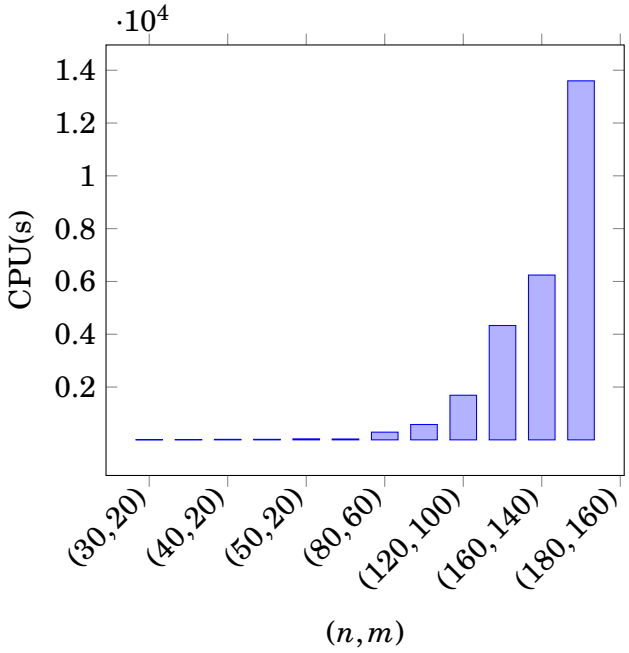


FIGURE 5.3 – Moyenne du temps de calcul pour chaque $(n,m)$ et $r = 5$

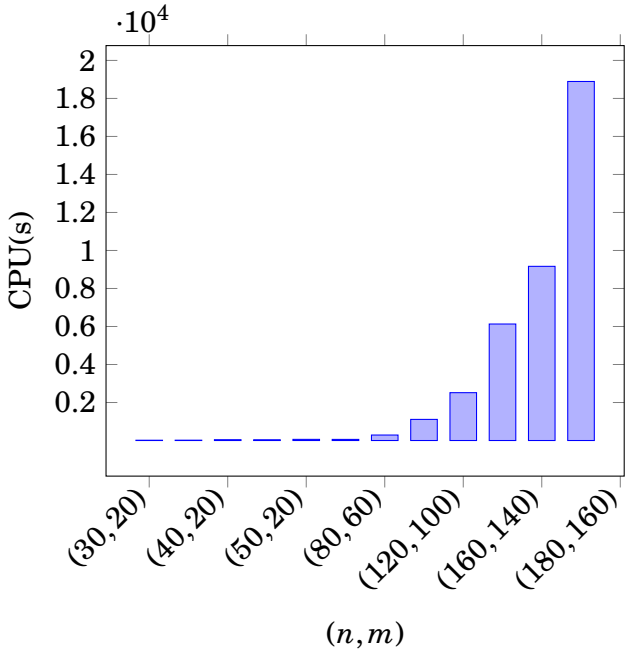


FIGURE 5.4 – Moyenne du temps de calcul pour chaque $(n,m)$ et $r = 7$

Comme le montrent les figures 5.2, 5.3 et 5.4, l'augmentation simultanée du nombre de variables et de contraintes influence notre méthode en termes de temps CPU. En effet, le temps CPU augmente exponentiellement en fonction du nombre de variables et contraintes du problème. Dans le cas contraire, l'effet d'une augmentation du nombre de contraintes tout en maintenant fixe le nombre de variables est moins perceptible car la méthode explore le domaine par des coupes efficaces et plus de contraintes donnent de petits domaines. Néanmoins, la méthodologie proposée réussit à résoudre des problèmes avec 180 variables, 160 contraintes et 7 critères en environ 4 heures en moyenne, ce qui est attrayant sur le plan informatique.

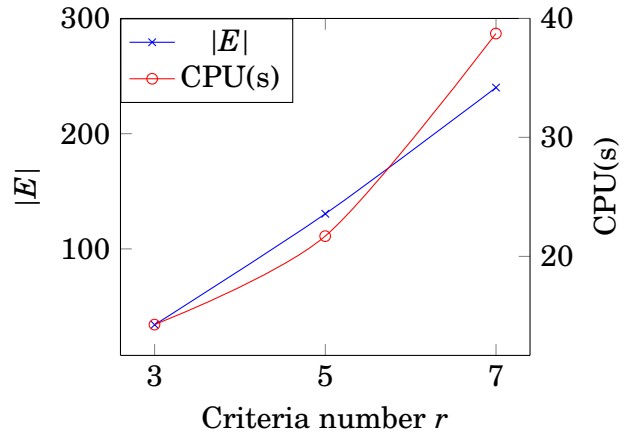


FIGURE 5.5 – Sensibilité de CPU(s) /au nombre de critères (pbs de tailles moyennes)

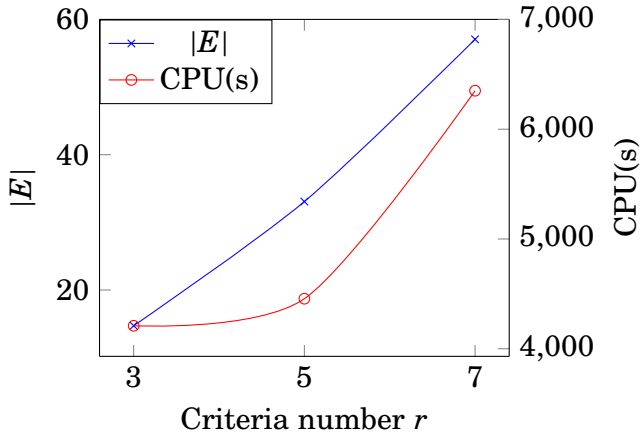


FIGURE 5.6 – Sensibilité de CPU(s) /au nombre de critères (pbs de grandes tailles)

Dans les figures 5.5 et 5.6, nous rapportons l'évolution du nombre de solutions efficaces, $|E|$ (ordonnée à gauche) et le temps CPU (à droite) en fonction du nombre de critères pour les instances de moyenne et de grandes tailles (la moyenne du temps d'exécution CPU et $|E|$ est prise pour chaque $r$). On peut remarquer que ces derniers augmentent significativement avec l'augmentation du nombre de critères, pour les mêmes valeurs de $n$ et $m$.

Notons que le nombre de solutions efficaces est intimement lié à la nature de l'instance considérée compte tenu de la difficulté du problème étudié. En effet, en réalisant une nouvelle génération des matrices $Q_i$, $i = 1, \cdots, r$, mais en considérant l'intervalle $[-5,5]$ au lieu de $[1,5]$ par exemple, on remarque que le nombre de solutions efficaces ainsi que le temps d'exécution augmentent, en particulier, pour les

instances de grandes tailles comme indiqué dans le tableau 5.3. Ces résultats s'expliquent par l'existence d'un conflit important entre les critères ce qui a conduit à générer plus de solutions efficaces et par conséquent plus de temps de calcul.

TABLE 5.3 – Autres résultats

| r | r=3 | | r=5 | | r=7 | |
|---|---|---|---|---|---|---|
| $n \times m$ | $\|E\|$ | CPU(s) | $\|E\|$ | CPU(s) | $\|E\|$ | CPU(s) |
| $30 \times 20$ | 49.7<br>[25,100] | 14.83<br>[6.36, 33.65] | 215.60<br>[55,558] | 20.45<br>[8.71, 41.91] | 505.8<br>[228, 1037] | 36.1<br>[10.50, 100.98] |
| $50 \times 25$ | 46.4<br>[18, 83] | 106.4<br>[50.51, 196.74] | 332.8<br>[141, 618] | 176.26<br>[79.54, 256.96] | 1080.9<br>[307, 2140] | 335.9<br>[123.52, 773.14] |
| $80 \times 60$ | 43<br>[16, 65] | 823.53<br>[675.18, 1105.56] | 335.28<br>[ 238, 490] | 979.16<br>[665.7, 1557] | 686.85<br>[423, 983] | 927.25<br>[727.36, 1212.45] |
| $100 \times 80$ | 50.2<br>[42, 66] | 2088.25<br>[1525.90, 3093.62] | 335.28<br>[ 205, 382] | 2287.25<br>[ 1864.2, 3668.7] | 853<br>[644, 1054] | 2947<br>[2098,3907.5] |
| $120 \times 100$ | 50.4<br>[20,72] | 5814.91<br>[4604.16, 8087.74] | 325.67<br>[ 246, 413] | 7182.23<br>[ 6726.28, 8029.42] | 853.2<br>[745, 1060] | 8694.05<br>[5663.18, 10331.65] |

## 5.7 Conclusion

Dans cette étude, une nouvelle méthode exacte combinant le principe bien connu de branch and bound lié à la programmation linéaire discrète avec une coupe efficace est décrite pour générer l'ensemble de toutes les solutions efficaces du problème de programmation quadratique multiobjectif en nombres entiers. Cette méthode peut être considérée comme une approche générale pour la résolution des problèmes MOIQP et peut ainsi résoudre des problèmes lorsque les variables de décision sont binaires. Le défi était le fait qu'il est classé comme NP-difficile et qu'aucune méthode n'a été proposée jusqu'à présent pour résoudre cette classe de problèmes. Plusieurs points forts sont rassemblés dans un outil de calcul intéressant : l'optimisation d'une fonction linéaire quelconque sur le domaine des solutions réalisables au lieu d'optimiser les fonctions quadratiques, l'utilisation de bornes inférieures et supérieures et les coupes efficaces afin d'élaguer les branches inutiles de l'arbre de recherche dans des zones ne contenant pas de solutions efficaces. Les résultats expérimentaux montrent que les instances ayant jusqu'à 50 variables, 25 contraintes et 7 critères sont résolues en une minute en moyenne. Néanmoins, le

temps CPU augmente avec la taille du problème compte tenu de la difficulté du problème MOIQP.

Certaines instances du monde réel traitant le problème de la gestion de portefeuille ainsi que les problèmes avec des critères différentiables non linéaires semblent être des cas appropriés et pertinents à résoudre en utilisant notre algorithme. De plus, la méthode peut être généralisée pour résoudre des problèmes de programmation non-linéaire multiobjectif mixtes à travers des changements appropriés.

# CONCLUSION GÉNÉRALE

Dans ce présent travail, nous avons passé en revu dans les premiers chapitres, les concepts fondamentaux de la programmation linéaire en nombres entiers. Ces éléments représentent les outils de base nécessaires au développement des notions et concepts de la programmation multiobjectif qui constitue le sujet central de cette thèse.

C'est ainsi que nous nous sommes focalisé sur la mise en œuvre de deux méthodes exactes. Nous avons déployées, dans une première étape, une méthode qui s'inscrit dans l'optique de l'optimisation globale, elle concerne la recherche de la meilleure solution parmi les solutions efficaces d'un problème d'optimisation multiobjectif linéaire en nombres entiers. Ceci a été rendu possible grâce au couplage du principe du "branch & bound" avec des coupes efficaces. Plusieurs tests ont été mis en place permettant ainsi de réduire l'espace de recherche. L'étude expérimentale a montré que notre méthode est meilleure que celle proposée par Jorge[10] en considérant les mêmes conditions de programmation.

Nous avons aussi abordés l'étude du problème de la programmation multiobjectif quadratique en nombres entiers MOIQP. Nous avons réussi à généraliser la méthode donnant l'ensemble efficace complet au problème MOILP. Une expérience est établie mettant en exergue les résultats obtenus sur des instances générées aléatoirement et un exemple illustratif est développé montrant le déroulement de l'algorithme. Dans ce cas, aucune comparaison n'est faite en l'absence dans la littérature, de méthodes générales dédiées au problème.

Parmi les travaux qui peuvent présenter des perspectives et qu'on souhaite aborder pour l'avenir on y trouve :

- La généralisation de la méthode proposée pour la résolution de MOIQP au cas

mixte ;

- L'adaptation de la méthode proposée pour la résolution de MOIQP à la résolution du problème de la gestion de portefeuille avec plus de deux critères quadratique, le problème d'affectation quadratique multiobjectif et celui du sac-à-dos quadratique multiobjectif ;
- La généralisation de la méthode proposée pour la résolution du problème de l'optimisation d'un critère linéaire sur l'ensemble efficient d'un MOILP au cas de l'optimisation d'un critère hyperbolique sur l'ensemble efficient d'un problème de d'optimisation multiobjectif fractionnaire linéaire en nombres entiers.

# BIBLIOGRAPHIE


[1] S. S. Rao, *Engineering Optimization : Theory and Practice, 4th ed.*
W, 2009.

[2] M. Simonnard, *Programmation linéaire : technique du calcul économique,.*
Paris, 1972.

[3] M. Minoux, *Programmation Lineaire. Theorie et Algorithmes. Tome 1.*
Dunod, 1983.

[4] M. Sakarovitch, *Optimisation combinatoire*.
Hermann, 1984.

[5] R. E. Steuer, *Multiple Criteria Optimization : Theory, Computation, and Application (Wiley Series in Probability and Statistics)*.
Wiley, 1986.

[6] C. H. Antunes, M. J. Alves, and J. Clímaco, *Multiobjective Linear and Integer Programming*.
Springer International Publishing, 2016.

[7] M. Moulaï, *Optimisation multicritère fractionnaire linéaire en nombres entiers*.
PhD thesis, Usthb, Algiers, Oct 2002.

[8] M. Abbas and D. Chaabane, "Optimizing a linear function over an integer efficient set," *European Journal of Operational Research*, vol. 174, pp. 1140–1161, oct 2006.

[9] C. Djamel, *Contribution à l'optimisation multicritère en variables discrètes*.
PhD thesis, Facult´e Polytechnque de Mons, 2007.

[10] J. M. Jorge, “An algorithm for optimizing a linear function over an integer efficient set,” *European Journal of Operational Research*, vol. 195, pp. 98–103, may 2009.

[11] B. Roy, “Des critères multiples en recherche opérationnelle : pourquoi?,” in *Operational research ’87 (Buenos Aires, 1987)*, North-Holland, 1988.

[12] F. Y. Edgeworth, “Mathematical psychics,” *P. Keagan, London, England.*, 1881.

[13] V. Pareto, “Manuale di economia politica,” *Societa Editrice Libraria, Milano, Italy*, 1906.

[14] F. Z. Ouaïl, M. E.-A. Chergui, and M. Moulaï, “An exact method for optimizing a linear function over an integer efficient set,” *WSEAS TRANSACTIONS on CIRCUITS and SYSTEMS*, vol. 16, pp. 141–148, 2017.

[15] H. Markowitz, “Portfolio selection,” *The Journal of Finance*, vol. 7, pp. 77–91, mar 1952.

[16] F. Z. Ouaïl and M. E.-A. Chergui, “A branch-and-cut technique to solve multiobjective integer quadratic programming problems,” *Annals of Operations Research*, oct 2017.

[17] M. E.-A. Chergui, M. Moulaï, and F. Z. Ouaïl, “Solving the multiple objective integer linear programming problem,” pp. 69–76, Springer Berlin Heidelberg, 2008.

[18] H. W. Kuhn and A. W. Tucker, “Nonlinear programming,” in *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability* (C. University of California Press, Berkeley, ed.), pp. 481–492, 1951.

[19] V. G. Karmanov, *Mathematical programming*.
Mir Publishers, 1989.

[20] S. M. Sinha, *Mathematical Programming : Theory and Methods*.
Elsevier Science, 2006.

[21] D. Fredon, *Mathematiques Resume du cours en fiches PCSI-PTSI-PC-PSI-PT (French Edition)*.
DUNOD, 2010.

[22] R. G. Bland, "New finite pivoting rules for the simplex method," *Mathematics of Operations Research*, vol. 2, pp. 103–107, may 1977.

[23] U. Faigle, W. Kern, and G. Still, *Algorithmic Principles of Mathematical Programming*.
Springer Netherlands, 2002.

[24] V. Chvátal, "Edmonds polytopes and a hierarchy of combinatorial problems," *Discrete Mathematics*, vol. 4, pp. 305–337, apr 1973.

[25] R. E. Gomory, "Outline of an algorithm for integer solutions to linear programs," *Bulletin of the American Mathematical Society*, vol. 64, pp. 275–279, sep 1958.

[26] E. Balas, S. Ceria, and G. Cornuéjols, "A lift-and-project cutting plane algorithm for mixed 0-1 programs," *Mathematical Programming*, vol. 58, pp. 295–324, jan 1993.

[27] M. E.-A. Chergui, *Contribution à la programmation non linéaire multicritère*.
PhD thesis, USTHB-Algiers, Dec. 2010.

[28] E. Balas, *Disjunctive programming*, pp. 3–51.
Annals of Discrete Mathematics, 1979.

[29] G. Cornuéjols, "Valid inequalities for mixed integer linear programs," *Mathematical Programming*, vol. 112, pp. 3–44, jan 2007.

[30] A. H. Land and A. G. Doig, "An automatic method for solving discrete programming problems," vol. 28, pp. 427–520, nov 1960.

[31] R. J. Dakin, "A tree-search algorithm for mixed integer programming problems," *The Computer Journal*, vol. 8, pp. 250–255, mar 1965.

[32] S. Martello, *Annotated Bibliographies in Combinatorial Optimization (Wiley Interscience Series in Discrete Mathematics)*.
Wiley, 1997.

[33] A. M. Geoffrion, "Proper efficiency and the theory of vector maximization," *Journal of Mathematical Analysis and Applications*, vol. 22, pp. 618–630, jun 1968.

[34] Y. H. YV, L. L. LS, and D. W. DA, "On a bicriterion formulation of the problems of integrated system identification and system optimization," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-1, pp. 296–297, jul 1971.

[35] A. Charnes, W. W. Cooper, and R. O. Ferguson, *Optimal Estimation of Executive Compensation by Linear Programming*, vol. 1.
Institute for Operations Research and the Management Sciences (INFORMS), jan 1955.

[36] G. R. Bitran, "Linear multiple objective programs with zero–one variables," *Mathematical Programming*, vol. 13, pp. 121–139, dec 1977.

[37] G. R. Bitran, "Theory and algorithms for linear multiple objective programs with zero–one variables," *Mathematical Programming*, vol. 17, pp. 362–390, dec 1979.

[38] G. Kiziltan and E. Yucaoğlu, "An algorithm for multiobjective zero-one linear programming," *Management Science*, vol. 29, pp. 1444–1453, dec 1983.

[39] R. Deckro and E. Winkofsky, "Solving zero-one multiple objective programs through implicit enumeration," *European Journal of Operational Research*, vol. 12, pp. 362–374, apr 1983.

[40] D. Klein and E. Hannan, "An algorithm for the multiple objective integer linear programming problem," *European Journal of Operational Research*, vol. 9, pp. 378–385, apr 1982.

[41] J. Sylva and A. Crema, "A method for finding the set of non-dominated vectors for multiple objective integer linear programs," *European Journal of Operational Research*, vol. 158, pp. 46–55, oct 2004.

[42] B. Lokman and M. Köksalan, "Finding all nondominated points of multi-objective integer programs," *Journal of Global Optimization*, vol. 57, pp. 347–365, jul 2012.

[43] L. Chalmet, L. Lemonidis, and D. Elzinga, "An algorithm for the bi-criterion integer programming problem," *European Journal of Operational Research*, vol. 25, pp. 292–300, may 1986.

[44] G. Mavrotas and D. Diakoulaki, "A branch and bound algorithm for mixed zero-one multiple objective linear programming," *European Journal of Operational Research*, vol. 107, pp. 530–541, jun 1998.

[45] G. Mavrotas and D. Diakoulaki, "Multi-criteria branch and bound : A vector maximization algorithm for mixed 0-1 multiple objective linear programming," *Applied Mathematics and Computation*, vol. 171, pp. 53–71, dec 2005.

[46] T. Vincent, F. Seipp, S. Ruzika, A. Przybylski, and X. Gandibleux, "Multiple objective branch and bound for mixed 0-1 linear programming : Corrections and improvements for the biobjective case," *Computers & Operations Research*, vol. 40, pp. 498–509, jan 2013.

[47] G. Kirlik and S. Sayın, "A new algorithm for generating all nondominated solutions of multiobjective discrete optimization problems," *European Journal of Operational Research*, vol. 232, pp. 479–488, feb 2014.

[48] M. A. Mehdi, M. E.-A. Chergui, and M. Abbas, "An improved method for solving multiobjective integer linear fractional programming problem," *Advances in Decision Sciences*, vol. 2014, pp. 1–7, 2014.

[49] M. Abbas and M. Moulaï, "Solving multiple objective integer linear programming," *Journal of the Italian Operations Research Society (Ricerca Operativa)*, vol. 29, pp. 15–38, 1999.

[50] M. Abbas and D. Chaabane, "An algorithm for solving multiple objective integer linear programming problem," *RAIRO - Operations Research*, vol. 36, pp. 351–364, oct 2002.

[51] R. Gupta and R. Malhotra, "Multi-criteria integer linear programming problem," *Cahiers de CERO*, vol. 34, pp. 51–68, 1992.

[52] J. Philip, "Algorithms for the vector maximization problem," *Mathematical Programming*, vol. 2, pp. 207–229, feb 1972.

[53] H. P. Benson, "An algorithm for optimizing over the weakly-efficient set," *European Journal of Operational Research*, vol. 25, pp. 192–199, may 1986.

[54] H. P. Benson, "A bisection-extreme point search algorithm for optimizing over the efficient set in the linear dependence case," *Journal of Global Optimization*, vol. 3, no. 1, pp. 95–111, 1993.

[55] H. P. Benson and D. Lee, "Outcome-based algorithm for optimizing over the efficient set of a bicriteria linear programming problem," *Journal of Optimization Theory and Applications*, vol. 88, pp. 77–105, jan 1996.

[56] J. G. Ecker and J. H. Song, "Optimizing a linear function over an efficient set," *Journal of Optimization Theory and Applications*, vol. 83, pp. 541–563, dec 1994.

[57] N. C. Nguyen, "An algorithm for optimizing a linear function over the integer efficient set," tech. rep., Konrad-Zuse-Zentrum fur Informationstechnik Berlin„ 1992.

[58] Y. Yamamoto, "Optimization over the efficient set : overview.," *Journal of Global Optimization*, vol. 22, no. 1/4, pp. 285–317, 2002.

[59] N. Boland, H. Charkhgard, and M. Savelsbergh, "A new method for optimizing a linear function over the efficient set of a multiobjective integer program," *European Journal of Operational Research*, vol. 260, pp. 904–919, aug 2017.

[60] H. P. Benson, "Optimization over the efficient set," *Journal of Mathematical Analysis and Applications*, vol. 98, pp. 562–580, feb 1984.

[61] R. O. Day and G. B. Lamont, "Multiobjective quadratic assignment problem solved by an explicit building block search algorithm – MOMGA-IIa," in *Evolutionary Computation in Combinatorial Optimization*, pp. 91–100, Springer Berlin Heidelberg, 2005.

[62] B. Malakooti and G. I. D'souza, "Multiple objective programming for the quadratic assignment problem," *International Journal of Production Research*, vol. 25, pp. 285–300, feb 1987.

[63] R. E. Steuer, Y. Qi, and M. Hirschberger, "Suitable-portfolio investors, nondominated frontier sensitivity, and the effect of multiple objectives on standard portfolio selection," *Annals of Operations Research*, vol. 152, pp. 297–317, dec 2006.

[64] R. E. Steuer, M. Wimmer, and M. Hirschberger, "Overviewing the transition of markowitz bi-criterion portfolio selection to tri-criterion portfolio selection," *Journal of Business Economics*, vol. 83, pp. 61–85, jan 2013.

[65] S. Utz, M. Wimmer, M. Hirschberger, and R. E. Steuer, "Tri-criterion inverse portfolio optimization with application to socially responsible mutual funds," *European Journal of Operational Research*, vol. 234, pp. 491–498, apr 2014.

[66] S. Utz, M. Wimmer, and R. E. Steuer, "Tri-criterion modeling for constructing more-sustainable mutual funds," *European Journal of Operational Research*, vol. 246, pp. 331–338, oct 2015.

[67] H. Zhang, C. Beltran-Royo, and L. Ma, “Solving the quadratic assignment problem by means of general purpose mixed integer linear programming solvers,” *Annals of Operations Research*, vol. 207, pp. 261–278, feb 2012.

[68] D. Pisinger, “The quadratic knapsack problem—a survey,” *Discrete Applied Mathematics*, vol. 155, pp. 623–648, mar 2007.

[69] E. Jain, K. Dahiya, and V. Verma, “Integer quadratic fractional programming problems with bounded variables,” *Annals of Operations Research*, apr 2017.

[70] V. Sharma, K. Dahiya, and V. Verma, “A ranking algorithm for bi-objective quadratic fractional integer programming problems,” *Optimization*, vol. 66, pp. 1913–1929, jun 2017.

[71] G. B. Dantzig, “Note on solving linear programs in integers,” *Naval Research Logistics Quarterly*, vol. 6, pp. 75–76, mar 1959.