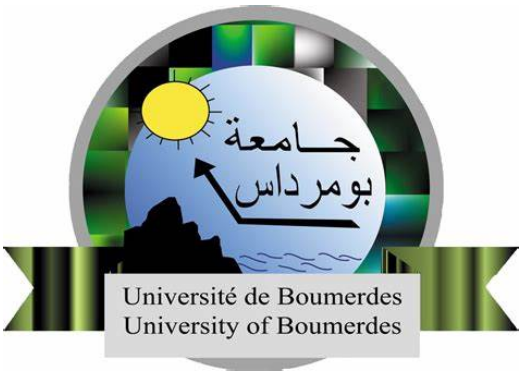
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE M'HAMED BOUGARA-BOUMERDES



Faculté des sciences

Mémoire
Présenté Pour L'Obtention Du Diplôme De Master En : Recherche Opérationnelle, Optimisation et Management Stratégique.

Présentée par : SALEM CHERIF Lina.

Méthode Branch and Bound appliquée au cas multicritère

Devant le jury :

Mme N.RAGGAS.	M.A.A.	UMBB.	Président.
Mme S.BARKA.	M.A.A.	UMBB.	Encadreur.
Mme R.FASS.	M.A.A.	UMBB.	Co-encadreur.
Mme K.KHODJA.	M.A.A.	UMBB.	Examinatrice.

2021-2022

Dédicace :

Je dédie ce projet
À la mémoire de ma mère et mon grand-père : « baba moh ».
À ma famille.
À toutes les personnes qui m'ont aidée.

Remerciement :

Je tiens à remercier toutes les personnes qui m'ont aidée lors de la rédaction de ce mémoire.
Je remercie ma promotrice Mme BARKA et ma Co-promotrice Mme FASS qui m'ont guidé et répondu à mes questions durant mes recherches, j'aimerais exprimer ma gratitude pour leur patience et disponibilité.
J'adresse mes sincère remerciement à Mme DRICI et Mr. BEZOUI de m'avoir aidé.
Je remercie ma famille pour leur encouragement.

Notations :

OM : Optimisation multi-objectif.
PL : Programmation linéaire.
PLNE : Programme linéaire en nombres entiers.
POM : Programme d'optimisation multi-objectif.
POLMNE : Programme d'optimisation linéaire multi-objectif en nombres entiers.
i.e : C'est à dire.
c-à-d : C'est à dire.

Table des matières

9 novembre 2022

Introduction

L'une des meilleures définitions du concept de recherche opérationnelle qui donne un sens direct et profond est : "L'art de mieux faire". Lorsque toutes les sciences se base de faire le travail, la recherche opérationnelle se préoccupe de trouver la meilleure stratégie de le faire [3], pour obtenir la meilleure solution du problème. La recherche opérationnelle peut être définie comme l'ensemble des méthodes et techniques rationnelles orientées vers la recherche du meilleur choix. Elle peut aider le décideur à prendre la meilleure décision lorsqu'il est confronté à un problème combinatoire, aléatoire ou concurrentiel. Elle se situe au carrefour de différentes sciences et technologies.

Pour un domaine continu, on distingue classiquement deux types d'optimisation [5] :

L'optimisation locale : recherche une solution qui est la meilleure localement, c'est- à -dire que dans son voisinage aucune solution n'est meilleure qu'elle. Cette solution est appelée un optimum local.
L'optimisation globale : recherche la meilleure solution du domaine entier, c'est-à-dire que dans tout le domaine il n'existe aucune solution meilleure qu'elle, tout en respectant les contraintes. Cette solution est appelée l'optimum global.

L'intérêt de l'optimisation globale par rapport à l'optimisation locale est épatent. Elle garantit en effet que personne ne peut avoir une solution meilleure que celle trouvée. Or, pour une entreprise, cette information a son importance, car la différence entre la solution globale et une solution locale est bien souvent significative. Mais l'intérêt n'est pas que compétitif. Dans de nombreux problèmes, l'optimum global est la seule solution mathématique correspondant à une réalité physique.

Ces dernières années de nombreux travaux ont été effectués sur les différentes approches (tel que les algorithmes révolutionnaires, les méthodes méta-heuristiques et les méthodes déterministes globales). Il existe plusieurs méthodes Pour la résolution des problèmes d'optimisations globales avec contraintes. Pour choisir la meilleure méthode à utiliser, il est nécessaire de mettre en considération :
La taille du problème.
Le temps disponible pour résoudre le problème.
Les propriétés de fonction objectif et les contraintes. (continuité, différentiable, linéarité, convexité....).

Quand on a une seule fonction objectif, on parle donc de l'optimisation mono-objectif, mais quand on a plusieurs fonctions objectifs à optimiser, nous parlons alors de l'optimisation Multi-objectifs, dans ce cas la meilleure solution est la solution qu'elle est meilleure sur tous les objectifs, c.-à-d. la solution idéale qu'il n'est pas réalisable dans la plupart des problèmes, alors le décideur doit donner ses préférences sur certains objectifs (il favorise certains objectifs i.e. il donne plus d'importance à certains objectifs que d'autres). Avec le développement des technologies de calcule, l'OM devient une branche très intéressante pour les chercheurs, elle trouve

son application dans de nombreux domaines : Pharmacie, finance, environnement, robotique, planification, etc [3].

La résolution d'un problème multi-objectif nous donne un ensemble de solutions qui s'appelle : ''ensemble de solutions efficaces". La notion de solutions efficaces remonte à Vilfredo Pareto [3]. Pour trouver ces solutions, l'étape primordiale que le mathématicien doit la faire est de choisir une méthode efficace.

Dans ce mémoire on s'intéresse à l'étude de certaines méthodes d'optimisation globale plus exactement la méthode Branch and Bround et comment l'utiliser pour résoudre les problèmes d'optimisation multi-objective en nombre entiers.

Nous avons organisé ce mémoire comme suit :

Chapitre 01 : traite la programmation linéaire mono-objective avec sa formulation générale, ses différentes formes, la définition du cas discret et continue. Nous avons cité deux méthodes de résolution : simplexe et duale du simplexe avec leurs définitions et algorithmes.

Chapitre 02 : est dédié à la méthode Branch and bound et la définition de : la procédure de la séparation, procédure d'évaluation, borne supérieure, borne inferieure et les stratégies du parcours. On termine avec deux exemples d'applications pour deux différents problèmes avec ses arborescentes.

Chapitre 03 : présente un rappel sur des notions fondamentales de l'optimisation multi-objectif. On commence par sa formulation générale, les définitions des concepts et leurs propriétés, la représentation graphique, nous avons aussi présenté la classification de méthodes et citer le nom de quelques méthodes avec une petite explication d'une entre eux.

Chapitre 04 : est consacré pour définir un problème d'optimisation multi objectif en nombre entier, donner sa formule générale, définir la méthode choisis, expliquer son principe, donner son algorithme, et on termine avec un exemple d'application.

Objectif de ce mémoire :

L'une des stratégies utilisée pour la résolution d'un problème d'optimisation multi objectif est l'agrégation, qui consiste à transformer le programme multi objectif initiale en un problème d'optimisation mono-objectif par l'ajout de paramètres et de contraintes, mais le résultat qu'on va le trouvé est une seule solution efficace, alors qu'on veut trouver toutes les solutions efficaces.

Dans ce travail on va présenter une méthode qui nous donne l'ensemble de toutes les solutions efficaces et pas seulement une, grâce à la contribution de la méthode Branch and Bound et la méthode des coupes au problème d'optimisation multi-objectif en nombres entiers.

Chapitre 1

Programmation linéaire mono-objectif

Introduction :

La programmation mathématique peut se définir comme une technique permettant de résoudre des problèmes et trouver son optimum grâce à un ensemble de méthodes ou processus mathématique. Elle englobe plusieurs types, par exemple :

⋆ La programmation dynamique.
⋆ La programmation non linéaire.
⋆ La programmation linéaire en nombres entiers.
⋆ La programmation linéaire qu'on va l'expliquer dans ce chapitre.

1.1 La programmation linéaire mono-objectif

Définition :

La programmation linéaire est un cas particulier de problèmes d'optimisation sous contrainte. Elle permit la résolution des problèmes d'optimisation dans la plupart des domaines (elle est très souvent utilisée pour résoudre des problèmes combinatoires). Utilisée directement ou comme partie d'un autre algorithme, des problèmes de tailles très variées. Il est nécessaire de passer par l'apprentissage de la formulation et la résolution des problèmes de PL.

La programmation linéaire mono-objectif est une technique utilisée pour déterminer la valeur optimale « minimum ou maximum » d'une seule fonction objectif avec un ensemble de contraintes, la fonction objectif et les contraintes sont linéaires.

1.2 Formulation d'un programme linéaire mono-objectif

La forme générale d'un programme linéaire mono-objectif est la suivante :

$$(\mathrm{P}):\begin{cases} \sum_{j=1}^{n} c_j x_j = F(max) & (fonction\ objectif) \\ & s.c \\ \sum_{j=1}^{n} a_{ij} x_j \leq b_i & (contraintes\ d'inégalité) \\ \sum_{j=1}^{n} a_{ij} x_j \geq b_i & (contraintes\ d'inégalité) \\ \sum_{j=1}^{n} a_{ij} x_j = b_i & (contraintes\ d'égalité) \\ x_j \geq 0 & (contrainte\ de\ non-négativité) \end{cases} \quad (1.1)$$

Avec les ensembles I^-, I^+ et I^0 sont disjoints, $I{=}I^- \cup I^+ \cup I^0 = \{1, ..., m\}$;
c_j ; a_{ij} et b_i (i=1,..., m ; j=1,..., m) sont des constantes supposées connues.
(P) est un programme linéaire mono-objectif à n variables et m contraintes [6].

1.3 Les différentes formes d'un programme linéaire mono-objectif

Il existe plusieurs formes, on peut citer :

La forme mixte :

On peut dire que le programme linéaire est écrit dans la forme mixte, juste dans l'existence de tous type de contraintes ($(\geq, =, \leq)$), et les variables sont positives.
Exemple :
La forme de (P) que nous avons défini dans (1.1) est : la forme mixte.

La forme canonique :

On peut dire que le programme linéaire est écrit dans la forme canonique, juste dans l'existence de deux conditions :
1) Les contraintes du PL doivent être :
⋆Dans le cas de maximisation : $\leq$
⋆Dans le cas de minimisation : $\geq$
2) Toutes les variables de décisions respectent la condition de la $non - négativité$.

La forme standard :

La forme standard c'est la forme qui contient que des contraintes de type =.
Toutes les variables de décisions respectent la condition de la non-négativité.

Remarque :

Tout programme linéaire peut s'exprimer sous forme standard, en ajoutant certaines variables appelées variables d'écart à toutes les contraintes sauf les contraintes de non-négativité (par exemple, une inéquation $a.x \leq b$ devient l'équation $a.x + s = b$, s ≥ 0).
Dans la suite, nous nous intéressons uniquement aux programmes linéaires exprimés sous forme standard.

1.4 L'optimisation linéaire disréte et continue

Face à la résolution d'un problème d'optimisation linéaire, il est important de bien identifier à quelle catégorie ce problème appartient. Il existe des PL discrets et des PL continus.

⋆ Le cas discret :
Dans ce cas, les variables de décision sont discrètes, le plus souvent sous la forme d'entiers ou de binaires. Cette branche mathématique traite des ensembles dénombrables, ces ensembles peuvent être fini ou infini.

⋆ Le cas continu :
Dans ce cas, les variables peuvent prendre n'importe quelle valeur, se sont des réels. Les problèmes d'optimisation linéaire continue sont généralement plus simples à résoudre.

Remarque :

Un problème d'optimisation mêlant variables continues et variables discrètes est dit mixte.

1.5 Notions de base

Soient :
A : m.n matrice.
b : m vecteur colonne.
c : n vecteur ligne.

On considère le PL (P) suivant écrit sous forme standard :

$$(\mathrm{P}) : \begin{cases} A.x = b; \;\; x \geq 0. \\ c.x = F(max). \end{cases}$$

Tel que le système linéaire A.x = b soit de plein rang (i.e. rg A = m)

Définition :

On appelle base du PL(P) ou du système A.x = b un ensemble J $\subset \{1, 2, \ldots, n\}$ d'indices de colonnes tel que A^J soit carrée régulière.

Définition :

Un PL (P) est dit écrit sous forme canonique par rapport à une base J, si les deux conditions suivantes sont vérifiées :
$\star A^J$ a une permutation prés des colonnes, la matrice identité.
$\star C^J = 0_{R^{|J|}}$.

Définition :

Le m vecteur ligne $\pi = C^J.(A^{J^{-1}})$ est appelé vecteur multiplicateur relatif à la base J. Le n vecteur ligne $\tilde{C} = (0_{R^{|J|}}; C^{\bar{J}} - \pi.A^{\bar{J}})$ est appelé vecteur coût relatif à la base J, ou coût réduit.

Définition :

Soit J une base de (P), la matrice A^J est la matrice de base associé à J. Les indices de J (resp.les colonnes de J, resp. variables x_j) sont de base si $j \in J$ et hors base si $j \notin J$.

1.6 Solutions d'un programme linéaire mono-objectif

Une solution est dite réalisable si elle satisfait toutes les contraintes de problème c.-à-d. $x=(x_1, \ldots, x_2)^T$ est une solution réalisable de (P) si et seulement si $A.x = b$ et $x \geq 0$.

Une solution optimale est une solution réalisable pour laquelle la fonction objectif atteint sa valeur maximale F^*.

Définition :

Une base J de (P) est dite réalisable si la solution de base associée vérifie $x_j \geq 0$.

Théorème :

Soit J une base réalisable de (P), si le vecteur cout relatif à J est négatif ou nul alors la solution de base associé à J est une solution optimale de (P).

1.7 Résolution d'un problème d'optimisation linéaire mono-objectif

Après avoir modélisé un problème pratique par un programme linéaire, l'étape qui va suivre sera certainement celle de la résolution de ce problème mathématique. La méthode graphique est l'une des premières méthodes utilisées à ce sujet, mais dans ce chapitre on s'intéresse à la résolution analytique.
Il existe plusieurs méthodes pour résoudre les PL comme : simplexe et duale du simplexe.

1.7.1 La méthode du simplexe

L'une des méthodes la plus utilisée pour résoudre les PL est la méthode du simplexe [3].

Définition :

La méthode du simplexe est une méthode exacte, itérative et à base de tableaux, développée en 1947 par G.Dantzig. Elle est la base de la plupart des algorithmes de résolution des programmes linéaires grâce à son efficacité, ses performances et également à sa capacité à fournir des solutions de base [6].

Avant l'algorithme, on doit définir l'application col :

1.7.2 Définition de l'application col

On appelle application col, l'application définie comme suit :

$$\text{col} : \{1, ..., m\} \longrightarrow J$$

$$\text{i} \longmapsto col(i).$$

telle que : $\text{A}^{col(i)} = e_i$.
Col permet de retrouver l'identité à partir de A^J.

Algorithme du simplexe

Début

1) Initialisation.

(P) : $\begin{cases} A.x = b;\ \ x \geq 0. \\ c.x = F(max) - \alpha. \end{cases}$

(P) est écrit sous forme canonique par rapport à une base réalisable J.

col=$\{1, ..., m\}$ *donnée*; solution de base initiale : $\begin{cases} x^*_{col(i)} = b_i, \forall i = 1, ..., m \\ \\ x^*_j = 0, \forall j \neq col(i) \end{cases}$

Evaluation initiale F(x^*) = α.

2)*Choisir* s $\in K$ tel que :

$$K = \left\{k, c_k = max_{j \in \bar{J}}.c_j\right\}$$

2.1) Si $c_s \leq 0$; terminer ; J base optimale de (P).
2.2) Sinon aller à l'étape 3.

3) Soit I = $\{i, A^s_i > 0\}$
3.1)*Si* I = $\emptyset$ terminer ; (P) n'a pas de solution optimale ; F(max) = $+\infty$.
3.2)*Si* I $\neq \emptyset$ aller à l'étape 4.
Soit L = $\left\{l, \frac{b_l}{A^s_l} = min_{i \in I} \frac{b_i}{A^s_i}\right\}$.

4) Choisir r $\in L$.
Effectuer le pivotage au tour de l'élément A^s_r.
$\acute{J} = (J \cup \{s\})/col(r)$ nouvelle base.
$\acute{M}$ = Pivotage(m+1, n+1, r, s, M).
$A'_s = \frac{b_r}{A^s_r}, x'_{col(i)} = b_i - A'_s.x'_s$.

$x'_s = 0, \forall j \notin J \cup \{s\}$.
$\alpha' = \alpha^* + c_s.\frac{b_r}{A^s_r}$.
$poser J = \acute{J}$, col(r) = s.
$x^* = x'$, $\alpha^* = \alpha'$.

$M = \acute{M}$.
Aller en (2).

Fin algorithme.

1.7.3 La méthode du duale du simplexe

Dans la méthode duale du simplexe, on ne travaille pas explicitement avec le problème dual mais bien avec le problème primal [7]. Il cherche à le résoudre implicitement par la méthode du simplexe.

Rappel :

Soit le programme linéaire (P) suivant :

(P) : $\begin{cases} A.x = b; \ \ x \geq 0. \\ c.x = F(max). \end{cases}$ (P) est écrit sous forme canonique par rapport à une base

(P) est primal (resp. dual) réalisable si b ≥ 0. ($resp.c \leq 0.$).

Théoréme :

La base J est optimale si et seulement si (P) est primal et dual réalisable.
La base J est dit primale (resp. duale) réalisable si (P) est primal (resp. dual) réalisable.

$\star$ Soit (P) écrit sous forme canonique par rapport à une base J duale réalisable.

(P) : $\begin{cases} A.x = b; \ \ x \geq 0. \\ c.x = F(max). \end{cases}$ $\Rightarrow c^J = 0;\ c^{\bar{J}} \leq 0;\ \mathrm{A}^J = [I].$

Propriétés :

Si $\forall i = 1, ..., m;\ \mathrm{b}_i \geq 0$; alors J est optimale.
Soit $\mathrm{r}_k \in \{1, ..., m\}$, $\mathrm{b}_i < 0$ et soit K=$\left\{k, A_r^k < 0\right\}$: Si K $= \emptyset \Rightarrow (P)$ n'admet pas de solutions réalisables car : $\sum_{j \in \bar{J}} A_r^j.x_j = b_r < 0$.
Soit r *telque* $\mathrm{b}_r < 0$ et supposons que K $= \left\{k, A_r^k < 0\right\} \neq \emptyset$.
Soit K$^* = \left\{k^*, \frac{c^{k^*}}{A_r^k} = min_{k \in K} \left\{\frac{c^k}{A_r^k}\right\}\right\}$; soit $s \in K$;
alors la base $\acute{\mathrm{J}} = \mathrm{J} \cup \{s\} \,/\, \{col(r)\}$ obtenue aprés pivotage autour de A_r^s est duale réalisable tq : $\mathrm{b}_r > 0$.

Définition de l'application col :

On appelle application col, l'application définie comme suit :

col : $\{1, ..., m\} \longrightarrow J$

$\mathrm{i} \longmapsto col(i).$

telle que : $\mathrm{A}^{col(i)} = e_i$.
Col permet de retrouver l'identité à partir de A^J.

Algorithme dual du simplexe

Début

1) (P) est écrit sous forme canonique par rapport à une base réalisable initiale :
$x^*_{col(i)} = b_i$, i= 1,..., m ;
$x^*_j = 0$, j $\in \bar{J}$.

2) Choix de la ligne pivot, choisir r tel que $b_r < 0$.
2.1) S'il n'existe pas, terminer.
2.2) Sinon aller en (3).

3) Déterminer K = $\left\{k, A^k_r < 0\right\}$.
3.1) Si K = $\emptyset$, terminer (D(P) = $\emptyset$) pas de solutions réalisables.
3.2) Sinon ; aller en (4).

4) Choix de la colonne pivot.
$K^* = \left\{k^*, \frac{c^{k^*}}{A^{k^*}_r} = min_{k\in K}\left\{\frac{c^k}{A^k_r}\right\}\right\}$. Choisi s $\in K^*$.
Effectuer le pivotage,

$$\acute{J} = J \cup \{s\} / \{col(r)\}.$$

Aller en 2.

Fin algorithme.

Conclusion :

Dans ce chapitre nous avons présenté les définitions et les méthodes que nous trouvons nécessaires pour la compréhension de la suite de ce mémoire.
Nous avons commencé par la définition d'un PL et la formule générale d'un PL mono objectif, ensuite nous avons défini les concepts de base et nous avons présenté deux méthodes de résolution avec leurs algorithmes.

Chapitre 2

La méthode de Branch and Bound

Introduction :

Chaque problème posséde des méthodes mieux adaptées que d'autres, dans ce chapître nous allons présenter une méthode de résolution appelé « Séparation et Evaluation », Branch and Bound en anglais, ses techniques et deux exemples d'applications.
Le premier exemple developpé du cette procédure est l'algorithme proposé par LAND et DOIG en 1960, très vite cet algorithme est devenu l'outil le plus utilisé pour résoudre les problèmes d'optimisation NP-difficile [3].

Avant définir cette méthode, nous commençons par des notions principales :

2.1 Définition de problème d'optimisation combinatoire

Un problème d'optimisation combinatoire consiste à trouver la meilleure solution dans un ensemble discret dit ensemble des solutions réalisables.

2.2 Définition d'un programme relaxé de (P)

Le programme linéaire (P') obtenu à partir de (P) en relachant les contraintes d'integrité est appelé :" programme linéaire relaxé ".

Exemple :

Soit (P) un programme d'optimisation combinatoire et (P') son programme linéaire relaxé :

$$(\mathrm{P}) : \begin{cases} A.x = b; \;\; x \in N. \\ c.x = F(max). \end{cases} \quad ; \quad (P') : \begin{cases} A.x = b; \;\; x \geq 0. \\ c.x = F(max). \end{cases}$$

2.3 Définition de la méthode Branch and Bound

Les méthodes par séparation et évaluation sont des méthodes de résolution exactes de problèmes d'optimisation combinatoire introduite par LAND et DOIG.

L'algorithme Branch and Bound éffectue une rechérche récursive déscendante [3], elle commence par le problème relaxé du problème originale et elle se base sur l'énumération des solutions d'une maniére intelligente en utilisant des propriétés, cette énumération des solutions consiste

à construire un arbre Branch and Bound, les nœuds sont des sous ensembles du problème et les branches sont les nouvelles contraintes à respécter, On a trois catégories des nœuds dans l'arbre de recherche au cours de déroulement : le nœud courant, les nœuds actifs, les nœuds inactifs [5]. La taille de l'arbre depend de la stratégie utilisée pour la construire.

Cette méthode se base sur une borne supérieure (cas maximisation) ou une borne inférieure (cas minimisation) sur cértaines solutions pour les éliminer ou les maintenir comme des solutions potentiels. Nous allons les voir dans des exemples par la suite.
La borne supérieure et la borne inférieure sont des éléments essentiels de l'algorithme Branch and Bound.

L'utilité de ces bornes :

⋆ Diminuer le temps de calcul.
⋆ Augmenter la rapidité de la méthode Branch and Bound.
⋆ Diminuer la taille de l'arbre.
⋆ Stopper l'exploration d'un sous ensemble de solution qui ne sont pas candidates à l'optimalité.

Cette méthode est basée sur trois axes principaux [5] :

⋆ La procédure de séparation (branchement).
⋆ La procédure d'évaluation.
⋆ La stratégie de parcours.

2.3.1 La procédure de séparation

Avant de commencer cette procédure on doit résoudre le problème relaxé du problème original, au départ l'arbre est construit d'un sommet racine. Le principe de séparation est appliqué de manière récursive des sous-ensembles tant qu'ils contiennent une solution mieux que la présente.
La séparation consiste à diviser le problème en un certain nombre de sous-ensembles, qui ont chacun leur ensemble de solutions réalisables [1]. On choisit une variable de séparation x_i non entiére de la solution courante x^* et on coupe en deux son domaine de valeurs possibles, x^* n'est plus condidate à une étude ultérieure.

Le processus de séparation d'un sous ensemble P_i s'arrête (minimisation) [5] :

Quand P_i n'admet pas de solution réalisable.
Quand P_i admet une solution dont ses composantes sont entiéres.
Quand la borne inférieur de P_i est la meilleure solution trouvée jusqu'à maintenant pour le problème initial.

2.3.2 La procédure d'évaluation

La procédure d'évaluation consiste à analyser un sous problème, cette analyse vise à déterminer une borne inférieur ou une borne supérieur de la valeur optimale de la fonction objectif de sous problème [5].

Le but de l'évaluation est prouvé que cet ensemble contient une solution intéressante pour la résolution du problème ou non.

Le nœud de l'arborescence est écarté de la recherche s'il est éliminé [5] :

Par optimalité.
Par borne.
Par infaisabilité.

On dit alors que le noeud (le sommet) est saturé.

2.3.3 La stratégie du parcours

Une stratégie de recherche selecte un nœud parmi les nœuds possibles selon une priorité définie, les politiques de parcours appliquées sont différentes selon la nature du problème, on peut citer les stratégies suivantes [5] :

a) Profondeur d'abord (depth first) :

Cette stratégie est souvent utilisée, elle consiste à une exploration en profondeur de l'arbre de recherche, les variables sont fixés une à une itérativement jusqu'à l'écarte du sommet dernier, après on remonte plus haut dans la plus proche décision prise et ainsi de suite.

b) Largeur d'abord :

Le parcours en largeur est rarement utilisé, il correspond à un parcours par niveau de nœud de l'arbre (i.e. nœud situé à la même distance du nœud racine).

c) Meilleur d'abord :

Ce type de parcours consiste à choisir parmi les nœuds restants à traiter, celui qui a la meilleure valeur optimale de la fonction objectif.

Remarque :

On peut utiliser et mélanger deux stratégies de parcours ou plus au cours de la résolution.

2.4 Pourquoi la méthode Branch and Bound ?

L'algorithme Branch and Bound est plus efficace grâce à différentes techniques utilisées, on peut citer quelques techniques comme [5] :

a) Choix d'une stratégie de recherche :

Le bon choix de la stratégie du parcours rend l'algorithme Branch and Bound plus efficace en terme d'espace mémoire et de la rapidité de trouver la solution optimale. Ce choix dépend

de la taille du problème et les ressources informatiques disponibles.

b) Choix de la borne inférieure ou supérieure :

Le bon choix de ces bornes nous aide à éliminer un grand nombre de nœuds le plus rapidement possible.

c) Règles de dominance :

Ils sont des conditions posées sur un nœud pour pouvoir l'éliminer. Cette technique est très utilisée pour réduire l'arbre de recherche.

2.5 Algorithme de branch and bound

Début :

X^* : la solution du (P) → Problème linéaire en nombres entiers.
$X^{opt(i)}$: la solution du $i^{éme}(P')$ → $i^{éme}$ problème linéaire relaxé. i ∈ N.
W^* : la valeur optimale de la fonction objectif du (P) → Problème linéaire en nombres entiers.
$W^{opt(i)}$: la valeur de la fonction objectif du $i^{éme}(P')$ → $i^{éme}$ problème linéaire relaxé. i ∈ N.

1) Résolution du premier problème relaxé (P') par la méthode du simplexe.
Si $X^{opt(1)}$ est entier ; fin.
Sinon, aller à 2).

2) Initialisation :
Soit $W^{opt(1)}$ obtenu dans (P') une borne supérieure pour un probléme de maximisation respectivement (borne inférieure pour un Problème de minimisation) pour (P_i).
Soit P_1 le sommet initial de l'arborescence et son ensemble ($S_1 = S$).
$W_1 = W^{opt(1)}$.

3) Séparation ($K^{iéme}$ itération) :
Choisir une variable non entiére x_{ne}, créer deux branches (deux sommets fils n_{i+1} et n_{i+2}), on obtient deux sous Problèmes sous la forme :
P_{i+1} : P_i+ la contrainte $x_{ne} \leq \lfloor x_{ne} \rfloor$.
P_{i+2} : P_i+ la contrainte $x_{ne} \geq \lfloor x_{ne} \rfloor + 1$.
Avec $\lfloor x_{ne} \rfloor$: la partie entiére inférieure de x_{ne}.

4) Résolution des sous problèmes :
Résoudre chaque sous problème en utilisant le simplexe ou le dual du simplexe.

5) Evaluation :
Examiner chaque sous-ensemble :
On peut tailler un sommet si :
La solution est non réalisable.
$W^{opt(1)} \leq W$ pour un problème de maximisation ($W^{opt(1)} \geq W$ pour un problème de minimisation avec W la solution du sous-problème).
La solution est non entiére et son W inférieur ou égale à une solution entiére pour un problème de maximisation(supérieure ou égale pour un problème de minimisation).
Il est inutile de séparer si :
La solution est entiére.

6) test :
S'il y a plus un sous-ensemble à séparer, alors :
On compare tous les W des solutions entiéres et on prend la plus grande entre elles soit W^* pour un problème de maximisation (la plus petite pour un problème de minimisation).Elle sera la valeur de la fontion objectif de la solution optimale X^* de notre problème (P).
Sinon retour à 3).

Fin algorithme. [8]

Remarque :

$W^* \leq W^{opt(1)}$ pour un Problème de maximisation ($W^* \geq W^{opt(1)}$ pour un Problème de minimisation).

2.6 Exemples d'application

Exemple 01 :

(Problème de minimisation).

Soit (P) un PLNE tel que :

$$(P) : \begin{cases} x_1 - 2x_2 = W(min). \\ s.c. \\ -4x_1 + 6x_2 \leq 9. \\ x_1 + x_2 \leq 4. \\ x_1 \in N;\ x_2 \in N. \end{cases}$$

La résolution du problème relaxé du (P) avec la méthode du simplexe nous donne la solution suivante : $X^{opt(1)} = (1.5, 2.5)$ avec $W_1 = W^{opt(1)} = -3.5$.

Remarque :

(P') est le Problème relaxé du (P) ;
(P') est représenté par le sommet (P_1) dans l'arborescence.

On remarque que cette solution $x^{opt(1)}$ est non entière, on choisit une composante non entière de cette solution et on partitionne notre problème en deux sous problèmes en ajoutant deux contraintes : $X_{ne}^{opt(1)} \leq \lfloor X_{ne}^{opt(1)} \rfloor$ et $X_{ne}^{opt(1)} \geq \lfloor X_{ne}^{opt(1)} \rfloor + 1$;

$où \lfloor X_{ne}^{opt(1)} \rfloor$ désigne la partie entiére inférieur de la composante non entiére de la solution $X^{opt(1)}$.

On est dans le cas de minimisation donc la valeur de la fonction objectif de la solution optimale est la borne inférieur(LB) de (P), c-à-d. $W_{LB} = W^{opt(1)} = -3.5$
En rajoutant les contraintes $x_2 \leq 2$ et $x_2 \geq 3$; on obtient les deux sous problémes P_2 et P_3 :

(P_2) :	(P_3) :
$x_1 + x_2 = W(min)$	$x_1 + x_2 = W(min)$
s.c.	s.c.
$4x_1 + 6x_2 \leq 9.$	$4x_1 + 6x_2 \leq 9$
$x_1 + x_2 \leq 4.$	$x_1 + x_2 \leq 4$
$x_2 \leq 2.$	$x_2 \geq 3$
$x_1 \geq 0$ et $x_2 \geq 0.$	$x_1 \geq 0$ et $x_2 \geq 0.$

On remarque que (P_3) n'est pas réalisable.

La solution optimale du (P_2) est : $X^{opt(2)} = (0.75, 2)$ et $W^{opt(2)} = -3.25$.
On continue la séparation car cette solution optimale est non entiére.
On partitionne selon la première variable en ajoutant deux contraintes : $x_1 \leq 0$ et $x_1 \geq 1$,on obtient (P_4) et (P_5) :

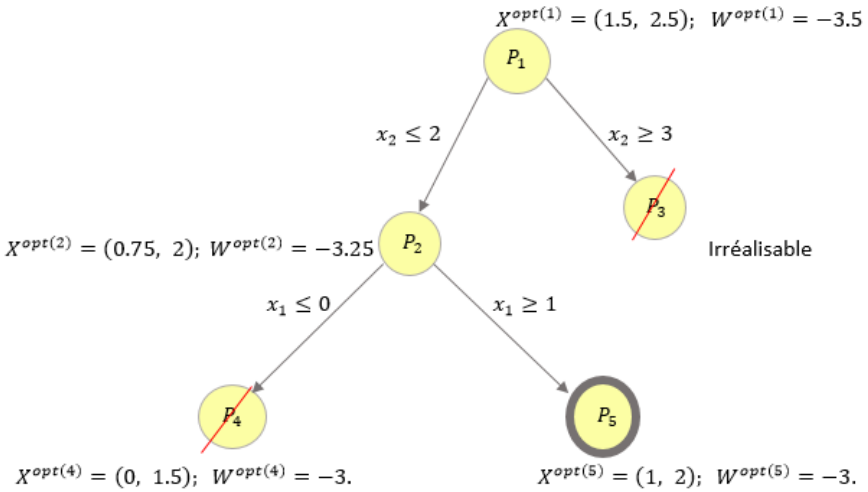
(P_4) :	(P_5) :
$x_1 + x_2 = W(min)$	$x_1 + x_2 = W(min)$
s.c.	s.c.
$4x_1 + 6x_2 \leq 9.$	$4x_1 + 6x_2 \leq 9$
$x_1 + x_2 \leq 4.$	$x_1 + x_2 \leq 4$
$x_2 \leq 2.$	$x_2 \geq 3$
$x_1 \leq 0$	$x_1 \geq 1$
$x_1 \geq 0$ et $x_2 \geq 0.$	$x_1 \geq 0$ et $x_2 \geq 0.$

La solution optimale de (P_5) nous donne $X^{opt(5)} = (1, 2)$ et $W^{opt(5)} = -3$, elle est entiére donc on arrete la séparation.

La solution optimale de (P_4) est : $X^{opt(4)} = (0, 1.5)$ et $W^{opt(4)} = -3$.
On arrête cette branche car la borne inferieure trouvée -3 est supérieur ou égale la meilleure solution trouvée jusqu'à maintenant -3.

La solution optimale du (P) PLNE est donc : $X^* = (1, 2)$ et $W^* = -3$.

l'arborescence correspondante à cet exemple est :



Exemple 02 :

(problème de maximisation) « sac à dos » :
On suppose que l'arbre construit est binaire.

Remarque :

Dans le cas où l'arbre construit est binaire ; les contraintes supplémentaires ajoutées durant la séparation est : $x_{ne} = 0$; $x_{ne} = 1$.

Soit (P) :

$$(P) : \begin{cases} 6x_1 + 5x_2 + 4x_3 \leq 9. \\ 10x_1 + 8x_2 + 5x_3 = W(max). \\ x_i \in \{0, 1\}\,;\, i \in \{1, 2, 3\}\,. \end{cases}$$

Soit (P') son programme relaxé :

$$(P') : \begin{cases} 6x_1 + 5x_2 + 4x_3 \leq 9. \\ 10x_1 + 8x_2 + 5x_3 = W(max). \\ x_i \in [0, 1];\, i \in \{1, 2, 3\}\,. \end{cases}$$

En utilisant l'algorithme glouton sur le programme relaxé (P') qui est représenté par le noeud P_1 dans l'arborescence, on obtient la solution suivante : $X^{opt(1)} = (1,\ 0.6,\ 0)$ *et* $W^{opt(1)} = 14.8$
On remarque que la solution obtenue n'est pas entière, donc on applique la procédure séparation et évaluation.
On fait la création de deux sous problèmes en ajoutant deux contraintes : $x_2 = 0$ et $x_2 = 1$.

(P_2) : La solution optimale *est* : $X^{opt(2)} = (1, 0, 0.75)$ et $W^{opt(2)} = 13.75$, la solution trouvée est non entiére on continue la séparation.

(P_4) : La solution optimale est : $X^{opt(4)} = (1, 0, 0)$ et $W^{opt(4)} = 10$, arrét de cette branche car on a obtenue une solution entiére. Le noeud P_4 est saturé.

(P_5) : La solution optimale est : $x^{opt(5)} = (5/6, 0, 1)$ et $W^{opt(5)} = 13.33$ la solution est non entiére on continue la séparation.

(P_6) : La solution optimale est : $X^{opt(6)} = (0, 0, 1)$ et $W^{opt(6)} = 5$, arrêt de cette branche, car la solution est entiére. Le noeud P_6 est saturé.

(P_7) : Irréalisable.

(P_3) : La solution optimale est : $X^{opt(3)} = (2/3, 1, 0)$ et $W^{opt(3)} = 14.66$, la solution trouvée est non entière, on continue la séparation.

(P_8) : La solution optimale est : $X^{opt(8)} = (0, 1, 1)$ et $W^{opt(8)} = 13$, arret de cette branche car la solution est entiére. Le noeud P_8 est saturé.

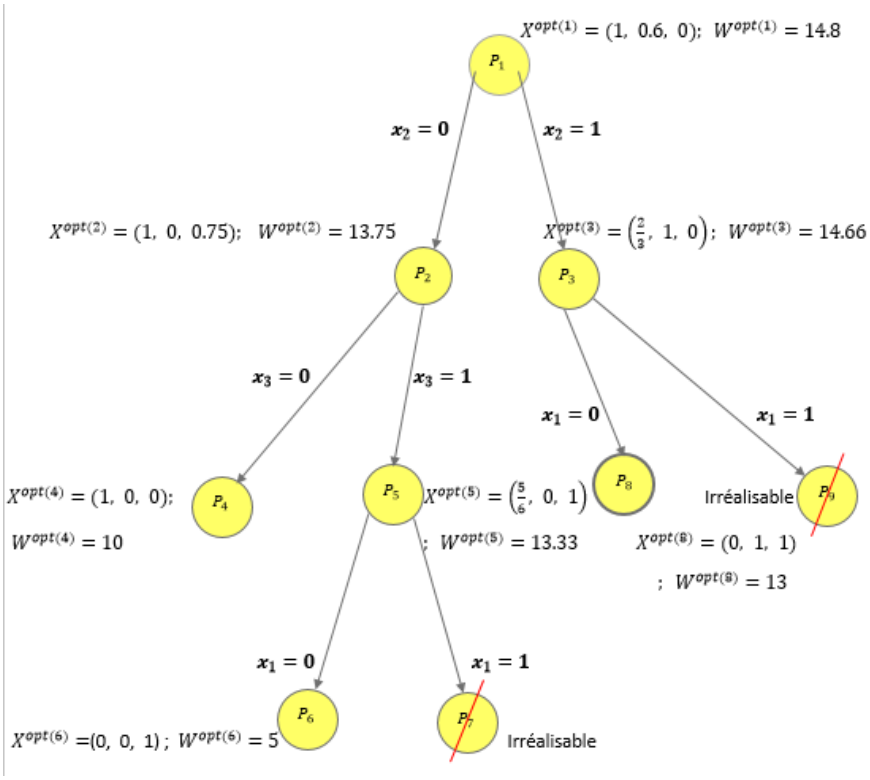
(P_9) : Irréalisable.

nous choisissons la meilleure solution parmi les solutions des noeuds saturés.

les noeuds saturés sont : P_4, P_6, P_8.

La meilleure solution trouvée est : $X^* = (0, 1, 1)$ et $W^* = 13$, la solution optimale du nœud (P_8).

L'arborescence correspondante à cet exemple est :



Conclusion :

Dans ce chapitre nous avons expliqué la méthode Branch and Bound que nous allons utiliser ses techniques pour réaliser notre objectif de ce mémoire. Nous avons commencé par la définition de : la méthode et ses axes principaux. Ensuite nous avons donné son algorithme et nous avons fini par deux exemples d'applications différents avec leurs arbre arborescente.

Chapitre 3

L'optimisation multi-objectif

Introduction :

Dans la vie nous sommes amenées à prendre des décisions dans plusieurs domaines, ces problèmes se traduisent à des problèmes d'optimisations. L'optimisation est la tâche qui nous permet de trouver une ou plusieurs solutions qui minimisent (ou maximisent) un ou plusieurs objectifs précis tout en respectant les contraintes. L'optimisation multi objectifs considère plusieurs objectifs (critères) souvent contradictoires et incompatibles à optimiser simultanément [3].

La résolution d'un problème d'optimisation multi objectif nous donne pas une solution optimale mais un ensemble de solutions, appelées solutions Pareto optimale (efficaces) et une parmi eux sera choisis d'après l'information de préférence du décideur. L'OM nous permet de trouver une solution qui puisse donner les valeurs de toutes les fonctions objectifs acceptables au décideur. Dans ce chapitre on fait un rappel de quelques notions de l'optimisation multi objectif.

3.1 Formulation d'un problème d'optimisation multi-objectif

Définition :

Un problème d'optimisation multi objectif consiste à chercher une solution qui satisfait les contraintes et optimise un vecteur ayant comme élément les fonctions objectifs souvent contradictoires.

La formulation générale d'un problème d'optimisation multi objectif est : (cas de maximisation) :

$$(P) : \begin{cases} (f_1(x),\ f_2(x), ...,\ f_p(x)) = F(max). \\ \\ x \in S; S \subseteq R^n \end{cases} \tag{3.1}$$

où
p $\geq$ 2 représente le nombre d'objectifs à maximiser, x représente un vecteur de variables de décision, S est l'ensemble de solutions réalisables associées à des contraintes, F(x) est le vecteur des objectifs à optimiser.f_k fonction à valeurs réelles du vecteur de décision (k=1, 2,. . ., p).

3.2 Concept de dominance et d'efficacité

3.2.1 Dominance (cas maximisation)

La comparaison entre deux solutions dans l'optimisation mono-objective est facile mais ce n'est pas le cas pour l'optimisation multi-objectif qui utilise la dominance. Le concept de dominance est utilisé pour signifier qu'une solution est meilleure qu'une autre [3].

Définition :

Soient f^1, f^2 deux vecteurs dans R^n, on dit que [3][1] :
f^1 domine f^2 si toutes les composantes de f^1 sont supérieures où égales à celles de f^2, avec au moins une inégalité stricte,$c-\grave{a}-d\ f_i^1 \geq f_i^2, \forall\ i \in \{1,\ldots,p\}\ et\ \exists\ i_0 \in \{1,\ldots,p\}\ telque : f_{i_0}^1 > f_{i_0}^2$.
et on note :

$$f^1 \succeq f^2$$

3.2.2 Dominance stricte (dominance forte)

Définition [3] :

f^1 domine strictement f^2, si toutes les composantes de f^1 sont supérieurs strictement à celles de f^2, c'est à dire $f_i^1 > f_i^2, \forall\ i \in \{1,\ldots,p\}$. Et on note : $f^1 \succ f^2$.

Propriété :

La relation binaire de dominance $\prec$:
N'est pas reflexive (une solution ne se domine pas elle-méme).
N'est pas symétrique.
N'est pas anti-symétrique.
Est transitive.[2]

3.2.3 Efficacité (cas maximisation)

Comme le concept précedent, La notion d'optimalité a un autre sens dans l'optimisation multi-objectif [3].

Définition :

Une solution est dite éfficace (Pareto optimale) si son vecteur critére n'est pas dominée, c-à-d, $\mathrm{x}^* \in X$ est efficace $\Longleftrightarrow$ il n'existe aucun $\mathrm{x} \in X$ tel que $\mathrm{F(x)} \succeq F(x^*)$

3.3 Solutions spéciales

On va présenter des points qui sont souvent utilisées dans l'optimisation multi-objectif (cas maximisation) [2].

3.3.1 Vecteur objectif idéale

Le vecteur idéal F* du probléme est le vecteur de l'espace des critéres dont chaque composante f_k^* est la solution du problème d'optimisation de la fonction f_k sous les contraintes du problème. En réalité cette situation n'est pas réalisable.

Définition :

Le vecteur idéal F* est le vecteur qui optimise chacune des fonctions objectifs individuellement.

$$f_k^* = max_{x \in S} \ \ (f_k(x), k = 1, ..., p \ )$$

3.3.2 Vecteur objectif Nadir

Pour réduire l'espace de recherche on utilise le concept de pire solution éfficace, appelé '' Point Nadir".

Définition :

Le point Nadir est le vecteur critére défini comme suit :

$$\eta_k = min_{x \in E} f_k(x).$$

Où : E est l'ensemble des solutions éfficaces de (3.1).

3.4 Représentation graphique

(Cas maximisation [9] :)

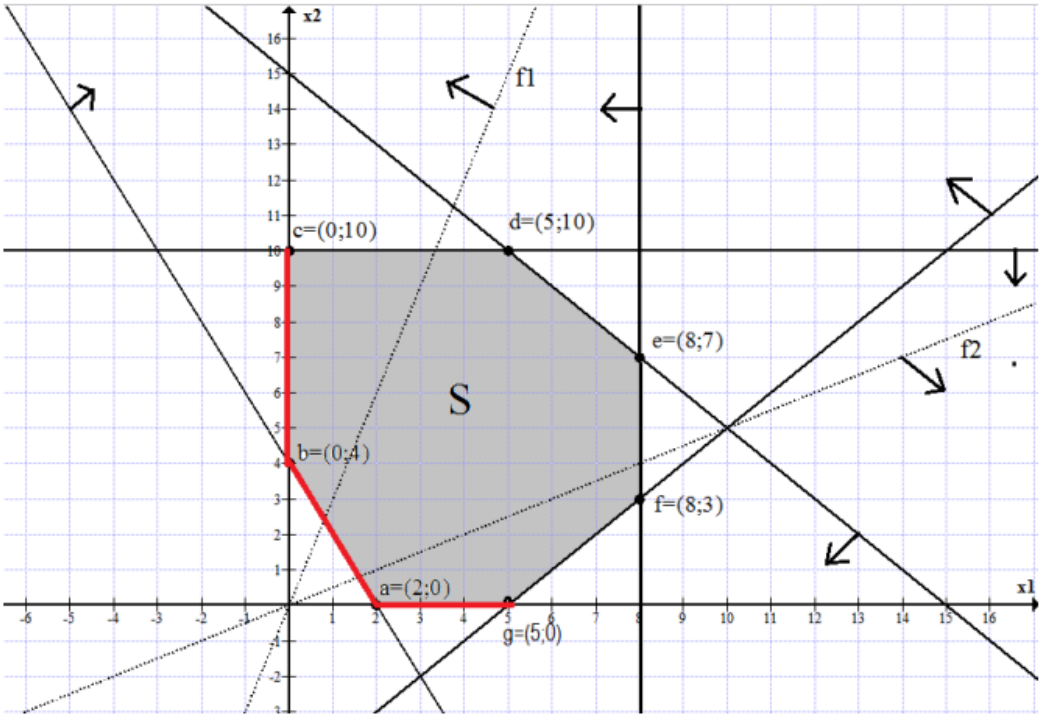


FIGURE 3.1 – Les points efficaces "en rouge" sur l'espace de décision.

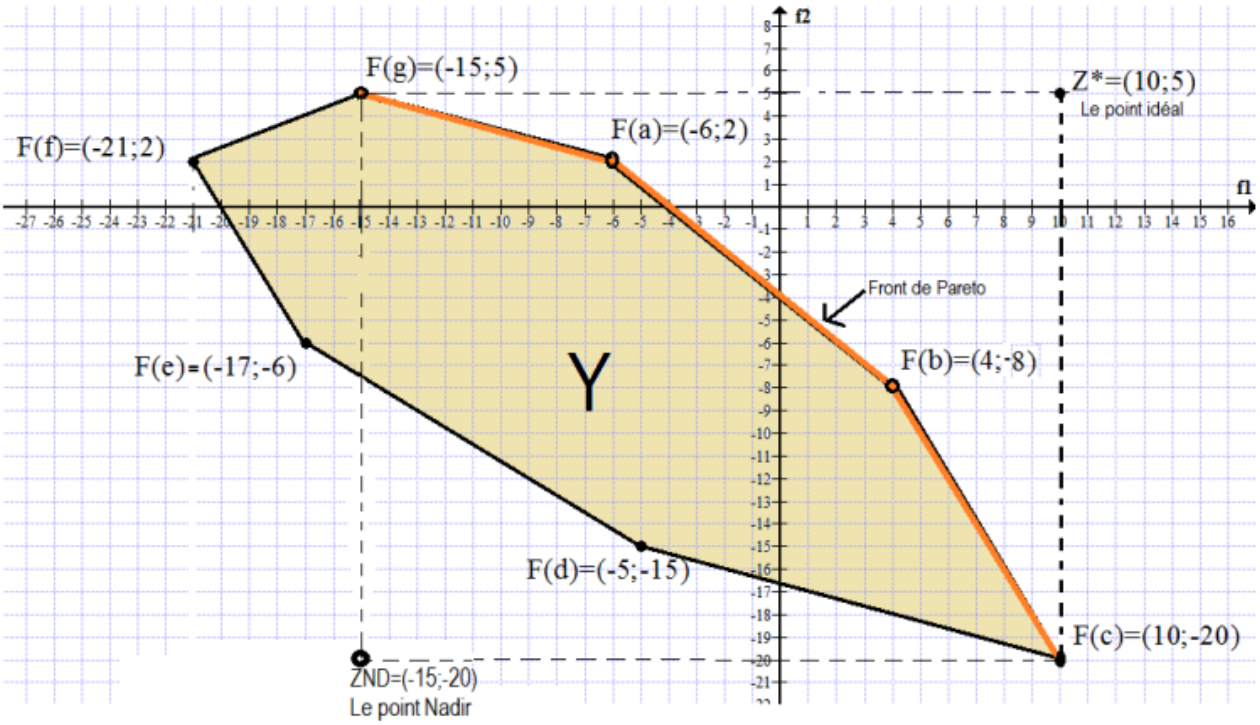


FIGURE 3.2 – Le point idéal, le point Nadir, le front de pareto sur l'espace de critére.

Le front de Pareto représente les points non dominés.

3.5 Classification de méthodes

Il existe des méthodes nombreuses, on peut les classer selon l'intéraction avec le décideur [3][2].

Les méthodes à préférence à priori $\left\{ Décideur \rightarrow Recherche \right\}$:

Dans ces méthodes le décideur définit le compromis qu'il souhaite, avant le lancement de la méthode d'optimisation. La plupart des méthodes de cette famille sont des méthodes par agrégation.

Les méthodes à préférence à postériori $\left\{ Recherche \rightarrow Décideur \right\}$:

Le décideur choisit une solution de compromis après l'examination de toutes les solutions extraites par la méthode d'optimisation. La qualité de la décision dépend du choix de la méthode de résolution.

Les méthodes à préférence progréssive $\left\{ Décideur \leftrightarrow Recherche \right\}$:

Dans ces méthodes le décideur peut modifier certaines variables ou contraintes au cours de l'optimisation pour dériger le processus de cette dérniére. Il affine son choix de compromis au méme temps de déroulement de l'optimisation.

Remarque :

Il éxiste des méthodes d'optimisation multi-objectif qui n'entrent pas dans une famille précise.

3.6 Méthodes de résolution de problèmes de programmation multi-objectif

Face à des différents problèmes d'optimisation, un grand nombre de méthodes ont été développées pour obtenir une solution de qualité acceptable dans un temps de calculs raisonnable.

Il éxiste plusieurs méthodes de résolution comme : La méthode $\epsilon-$ contrainte [2][3], la méthode de scalarisation de Benson [3], la méthode du simplexe (elle n'a rien à voir avec la méthode du simplexe du PL), la méthode de scalarisation pondérée de chebyshev, la méthode de keeney-Raiffa [2], la méthode de programmation de but [2], la méthode léxicographique, la méthode de Géoffrion [2] et la méthode de pondération des fonctions objectifs qu'on va l'éxpliquer par la suite.

La méthode de pondération des fonctions objectifs :

Cette méthode est connu aussi sous le nom ''Approche naive '', elle consiste à effectuer une somme pondérée des objectifs selon des coefficients, souvent donné par le décideur qui expriment les degrés d'importance des critéres pour le décideur afin de ramener notre probléme multi-critére à un problème mono-critére. Dans le cas où les coéfficients ne sont pas définit par le décideur [3], nous les remplaçons par des $\lambda_i \geq 0, i = 1; ...;$ p $avec \sum_{i=1}^{p} \lambda_i = 1$.

Le probléme 3.1 devient :

$$\begin{cases} max f_{éq}(x) = \sum_{i=1}^{p} \lambda_i f_i(x) \\ s.c \\ x \in S \end{cases}$$

Cette méthode est la plus utilisée pour résoudre les POMs. Elle se caractérise par la forte dépendance entre les résultats obtenus et les poids utilisés [3].

Conclusion :

Dans ce chapitre nous avons fait un rappel sur quelques définitions et notions de l'optimisation multi-objectif, nous avons donné la formulation d'un problème d'optimisation multi-objectif, ensuite les concepts et les notions essentiels et à la fin nous avons cité quelques méthodes de résolution de POM.

Chapitre 4

Une méthode de résolution d'un problème d'optimisation linéaire multi-objectif en nombres entiers

Introduction :

La programmation linéaire en nombres entiers est un domaine très riche de la programmation mathématique. Les recherches dans ce domaine sont nombreuses. Un problème de programmation linéaire en nombres entiers (PLNE) est un programme linéaire, sous des contraintes linéaires, dans lequel il y a la contrainte supplémentaire que les variables sont entières. Les problèmes d'optimisation en nombres entiers est parmi les problèmes difficiles à résoudre. On s'intéresse dans ce chapitre à une méthode qui nous donne toutes les solutions efficaces d'un POLMNE.

4.1 La formulation d'un problème d'optimisation linéaire multi objectif en nombres entiers

Soit (P) un probléme d'optimisation linéaire multi-objectif en nombres entiers :

$$(P) : \begin{cases} max f^1 = c^1.x \\ max f^2 = c^2.x \\ \quad \vdots \\ max f^r = c^r.x \\ x \in D \end{cases}$$

Où r $\geq$ 2 représente le nombre d'objectifs à maximiser, x représente un vecteur de variables de décision, D=$\{A.x = b, x\, entiers\}$ est l'ensemble de solutions réalisables associées à des contraintes.

Soit (P') son programme relaxé :

$$(P') : \begin{cases} max f^1 = c^1.x \\ max f^2 = c^2.x \\ \vdots \\ max f^r = c^r.x \\ x \in S. \end{cases}$$

Où S =$\{A.x = b, x \geq 0\}$

4.2 Méthode de résolution

4.2.1 Principe de la méthode

Cette méthode est une combinaison entre la méthode Branch and Bound et la méthode des coupes, elle est basé sur le principe de Branch and Cut.
La méthode se base sur la recherche arborescente en programmation linéaire [2], au début, il faut choisir une seule fonction objectif, nous choisissons la premiére fonction objectif f^1.

à chaque noeud actif l on doit résoudre un problème correspondant (P_l') qui est un programme mono-objectif (nous avons déja cité la fonction objectif qu'on va travailler avec ("f^1")).
(P_l') définit comme suit :

$$(P_l') : \begin{cases} max f^1 = c^1.x \\ x \in S_l \end{cases}$$

avec $S_0 = S$ et S_l est le sous ensemble de S contenant les solutions réalisables correspondant au noeud l.

Remarque :
on peut prendre n'importe quelle fonction objectif.

On initialise : l = 0 et l'ensemble des solutions efficaces du probléme (P) ; Eff= $\emptyset$.

On résoud le problème (P_l') au noeud l en utilisant l'algorithme du simplexe (eventuellement la méthode duale du simplexe).
Mettre à jours les vecteurs gradients c^q; q $\in \{1, ..., r\}$ et les nouveaux vecteurs c^q sont calculés en respectant la base correspondante du tableau du simplexe.

Le noeud l est saturé si le programme (P_l') est non réalisable.
Si le programme (P_l') admet une solution entiére x_l^*, on teste son efficacité en comparant les vecteurs critéres de la solution entiére trouvée avec les solutions de l'ensemble Eff. [2][4]

On determine B_l ensembles des indices de base de x_l^*, N_l ensemble des indices hors base de x_l^*, pour construire l'ensemble H_l qui nous aide à la construction de la coupe efficace.
La formule de H_l est la suivante [2][4] :

$$H_l = \left\{ j \in N_l / \exists q \in \{1, ..., r\}, c_j^q > 0 \right\} \cup \left\{ j \in N_l / c_q^j = 0, \forall q \in \{1, ..., r\} \right\}$$

Cette coupe efficace est considérée comme contrainte dans le programme de sommet suivant.

On utilise la coupe efficace pour eviter d'explorer les régions non efficace de l'ecpace de solutions réalisables. Pour chaque solution entiére trouvée on construit une coupe efficace tant que H_l est non vide sinon le noeud correspondant est saturé.
Cette coupe efficace est de la forme suivante [2][4] :

$$\sum_{j \in H_l} x_j \geq 1.$$

Si la solution x_l^* du programme $(P_l^{'})$ n'est pas entiére, on utilise la séparation de la méthode Branch and Bound, le sommet l est séparé en deux sommets avec des contraintes supplimentaires : $x_j \leq \lfloor \alpha \rfloor$ et $x_j \geq \lfloor \alpha \rfloor + 1$.

4.2.2 Algorithme

Début
1) (Initialisation) $S_0 = S$, l = 0, $Eff = \emptyset$.
Résoudre le programme linéaire relaxé de $(P_l^{'})$ au noeud 0 et soit x_l^* sa solution optimale ;
Si x_l^* n'est pas entiére passer à 2.1.
Sinon passer à 2.2.

2) Tant qu'il y a un sommet non saturé, faire

choisir le premier sommet l crée dans l'arbre, résoudre le programme linéaire correspond à $(P_l^{'})$.
Si $(P_l^{'})$ n'a pas de solutions réalisables, alors $(P_l^{'})$ devient saturé,
Sinon soit x^* sa solution optimale, si x^* n'est pas entiére passer à 2.1 ; sinon passer à 2.2.

2.1) Choisir une coordonnée x_j de x_l^* telle que $x_j = \alpha$ avec α fractionnaire, et séparer le sommet actuel l en deux sommets l_1 et l_2 ($l_1 \geq l + 1$ et $l_2 \geq l + 2$), ajouter la contrainte : $x_j \leq \lfloor \alpha \rfloor$ au premier sommet et la contrainte $x_j \geq \lfloor \alpha \rfloor + 1$ au deuxiéme sommet et passer à 2.

2.2) Si $F(x_l^*) = ((f^1(x_l^*), f^2(x_l^*); ..., f^r(x_l^*))$ n'est pas dominé par
$F(y) = (f^1(y), f^2(y), ..., f^r(y)), \forall y \in Eff, alors\ Eff = Eff \cup \{x_l^*\}$.
S'il existe $y \in Eff$ telle que F(y) est dominé par $F(x_l^*)$, alors $Eff = Eff/\{y\} \cup \{x_l^*\}$.

Déterminer les ensembles B_l, N_l et H_l.
Si $H_l = \emptyset$ alors le sommet l correspondanr est saturé, passer à 2.
Sinon ajouter la contrainte :

$$\sum_{j \in H_l} x_j \geq 1.$$

pour obtenir l'ensemble S_{l+1}, résoudre le programme corréspondant en utilisant la méthode duale du simplexe, soit x^* la solution optimale trouvée. l = l+1.

Fin algorithme.

Cette méthode a plusieurs avantages :

Efficacité de calcul remarquable.
Les coupes supplémentaires réduisent le temps grâce à l'inclusion de contraintes de serrage non triviales.
L'inclusion de contraintes de serrage s'avère être une clé fondamentale pour accélérer les performances de calcul sans consacrer d'efforts inutiles à des techniques de solution plus compliquées.
Supprimer les solutions non efficaces sans les calculer.

4.3 Exemple d'application :

Soit (P) le programme linéaire multi-objectif en nombre entiers :

$$(P) : \begin{cases} x_1 + x_2 = f_1(max) \\ 3x_1 - 2x_2 = f_2(max) \\ -x_1 + 2x_2 = f_3(max) \\ \\ s.c. \\ \\ x_1 + x_2 \leq 7 \\ 2x_1 \leq 11 \\ 2x_2 \leq 7 \\ x_1, x_2 \ entiers. \end{cases}$$

Soit (P') le programme relaxé écrit sous forme standard de (P) :

$$(P') : \begin{cases} x_1 + x_2 = f_1(max) \\ 3x_1 - 2x_2 = f_2(max) \\ -x_1 + 2x_2 = f_3(max) \\ \\ s.c. \\ \\ x_1 + x_2 + x_3 = 7 \\ 2x_1 + x_4 = 11 \\ 2x_2 + x_5 = 7 \\ x_1 \geq 0, x_2 \geq 0. \end{cases}$$

On résout le problème avec la méthode du simplexe :
Dans tous ce qui suit, on effectue le pivotage par rapport à la premiére fonction objectif.

Création du premier tableau :

t_1	x_1	x_2	x_3	x_4	x_5	b
x_3	1	1	1	0	0	7
x_4	2	0	0	1	0	11
x_5	0	2	0	0	1	7
$-f_1$	1	2	0	0	0	0
$-f_2$	3	-2	0	0	0	0
$-f_3$	-1	2	0	0	0	0

Effectuer le pivotage :

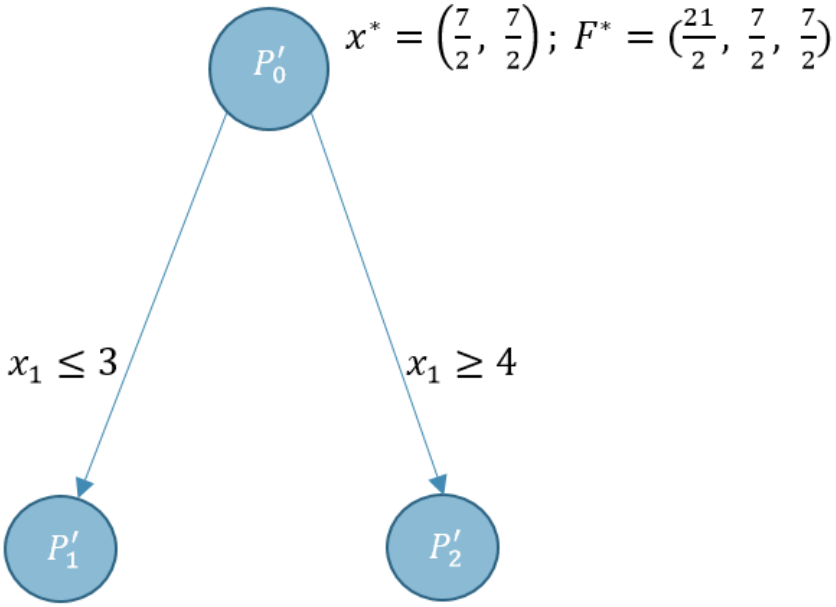
t_2	x_1	x_2	x_3	x_4	x_5	b
x_3	1	0	1	0	-1/2	7/2
x_4	2	0	0	1	0	11
x_2	0	1	0	0	1/2	7/2
$-f_1$	1	0	0	0	-1	-7
$-f_2$	3	0	0	0	1	7
$-f_3$	-1	0	0	0	-1	-7

Effectuer le pivotage encore une fois :

t_f	x_1	x_2	x_3	x_4	x_5	b
x_1	1	0	1	0	-1/2	7/2
x_4	0	0	-2	1	1	4
x_2	0	1	0	0	1/2	7/2
$-f_1$	0	0	-1	0	-1/2	-21/2
$-f_2$	0	0	-3	0	5/2	-7/2
$-f_3$	0	0	1	0	-3/2	-7/2

Le tableau est optimal et la solution optimale est :
$x^* = (7/2, 7/2)$; $F^* = (21/2, 7/2, 7/2)$

On remarque que la solution optimale n'est pas entière, on applique le principe de séparation.



On résout le programme (P_1') :
le tableau initial :

t_1	x_1	x_2	x_3	x_4	x_5	x_6	b
x_3	1	1	1	0	0	0	7
x_4	2	0	0	1	0	0	11
x_5	0	2	0	0	1	0	7
x_6	1	0	0	0	0	1	3
$-f_1$	1	2	0	0	0	0	0
$-f_2$	3	-2	0	0	0	0	0
$-f_3$	-1	2	0	0	0	0	0

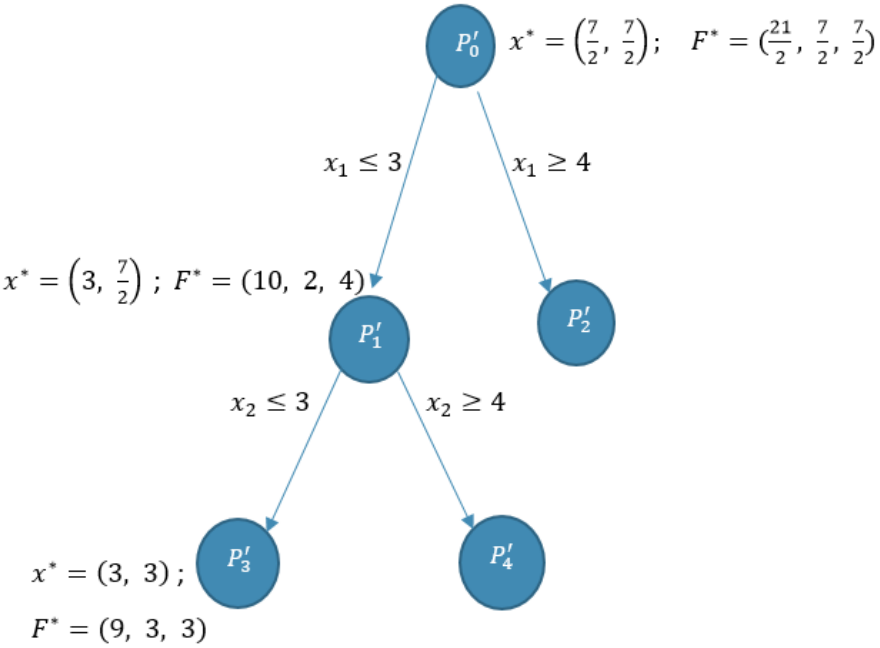
Aprés le pivotage on obtient :

t_2	x_1	x_2	x_3	x_4	x_5	x_6	b
x_3	1	0	1	0	-1/2	0	7/2
x_4	2	0	0	1	0	0	11
x_2	0	1	0	0	1/2	0	7/2
x_6	1	0	0	0	0	1	3
$-f_1$	1	0	0	0	-1	0	-7
$-f_2$	3	0	0	0	1	0	7
$-f_3$	-1	0	0	0	-1	0	-7

le pivotage encore une fois :

t_f	x_1	x_2	x_3	x_4	x_5	x_6	b
x_3	0	0	1	0	-1/2	-1	1/2
x_4	0	0	0	1	0	-2	5
x_2	0	1	0	0	1/2	0	7/2
x_1	1	0	0	0	0	1	3
$-f_1$	0	0	0	0	-1	-1	-10
$-f_2$	0	0	0	0	1	-3	-2
$-f_3$	0	0	0	0	-1	1	-4

Le tableau est optimal.
La solution optimale est : $x^* = (3, 7/2)$; $F^* = (10, 2, 4)$.
La solution optimale n'est pas entière, on applique le principe de séparation.



On résout le programme (P_3'), aprés les calculs on aura le tableau final suivant :

t_f	x_1	x_2	x_3	x_4	x_5	x_6	x_7	b
x_3	0	0	1	0	0	-1	-1	1
x_4	0	0	0	1	0	-2	0	5
x_5	0	0	0	0	1	0	-2	1
x_1	1	0	0	0	0	1	0	3
x_2	0	1	0	0	0	0	1	3
$-f_1$	0	0	0	0	0	-1	-2	-9
$-f_2$	0	0	0	0	0	-3	2	-3
$-f_3$	0	0	0	0	0	1	-2	-3

La recherche d'une coupe efficace :

$$B_l = \{1, 2, 3, 4, 5\}.$$
$$N_l = \{6, 7\}.$$

$$H_l = \{6, 7\}$$

donc la coupe efficace existe, elle s'écrit sous la forme suivante :

$$x_6 + x_7 \geq 1.$$

On prend le dernier tableau obtenu et on ajoute la coupe efficace comme une contrainte et on résout avec le duale du simplexe le problème standard correspondant qu'on va l'obtenu.

t_1	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	b
x_3	0	0	1	0	0	-1	-1	0	1
x_4	0	0	0	1	0	-2	0	0	5
x_5	0	0	0	0	1	0	-2	0	1
x_1	1	0	0	0	0	1	0	0	3
x_2	0	1	0	0	0	0	1	0	3
x_8	0	0	0	0	0	-1	-1	1	-1
$-f_1$	0	0	0	0	0	-1	-2	0	-9
$-f_2$	0	0	0	0	0	-3	2	0	-3
$-f_3$	0	0	0	0	0	1	-2	0	-3

Après l'application du duale du simplexe on obtient :

t_f	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	b
x_3	0	0	1	0	0	-1	0	-1	2
x_4	0	0	0	1	0	-2	2	-2	7
x_5	0	0	0	0	1	0	-2	0	1
x_1	1	0	0	0	0	1	-1	1	2
x_2	0	1	0	0	0	0	1	0	3
x_6	0	0	0	0	0	-1	1	-1	1
$-f_1$	0	0	0	0	0	-1	-1	-1	-8
$-f_2$	0	0	0	0	0	-3	5	-3	0
$-f_3$	0	0	0	0	0	1	-3	1	-4

Le tableau est optimal.
La solution optimale est : $x^* = (2, 3)$ et $F^* = (8, 0, 4)$
La solution est entière on fait le test de dominance sur les solutions efficaces trouvées jusqu'à maintenant et on cherche une coupe efficace :

Test : On a : $9 \geq 8, 3 \geq 0, 3 \geq 4$; donc la nouvelle solution est non dominée et les solutions précédentes sont non dominées aussi, alors Eff = Eff $\cup \{(2; 3)\}$

La recherche d'une coupe efficace :
On a $H_l = \{7, 8\}$, donc la coupe efficace existe et sa formule est : $x_7 + x_8 \geq 1$.

De la même manière on continue toutes les itérations restantes pour obtenir l'ensemble de toutes les solutions efficaces suivant :
Les solutions efficaces sont :

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

Conclusion :

Dans ce chapitre nous avons abordé la partie principale de ce mémoire qui est une méthode intéressante de résolution de problème d'optimisation multi-objectif en nombres entiers, au début nous avons défini la méthode et donné la formule d'un POLMNE, ensuite nous avons exposé le principe de la méthode et son algorithme. Nous avons fini par un exemple d'application.

Conclusion générale

Dans ce mémoire, nous avons présenté comment utiliser la méthode Branch and Bound pour résoudre un problème d'optimisation linéaire multi objectif en nombres entiers.
Au premier chapitre nous avons fait un rappel sur les notions, les définitions et les concepts de l'optimisation mono-objectif avec deux méthodes de résolution et leurs algorithmes correspondants.
Au deuxième chapitre nous avons défini la méthode Branch and Bound et ces axes principaux et ces différentes techniques, ainsi que deux exemples d'applications.
Au troisième chapitre : nous avons abordé l'optimisation multi objectif, ses notions de base et quelques méthodes de résolution.
Nous avons terminé avec le quatrième chapitre où nous avons présenté et montré l'éfficacité d'une méthode de résolution des problèmes d'optimisation linéaire multi objectif en nombres entiers, au début nous avons définit le principe de la méthode, ensuite nous avons donné son algorithme et nous avons terminé avec un exemple d'application.

Bibliographie

Bibliographie

[1] Salima Amrouche. *Optimisation linéaire stockastique multi-objectifs en nombres entiers*. PhD thesis, Alger, 2005.

[2] Madani Bezoui. *Programmation multiobjectif quadratique en variables mixtes*. PhD thesis, Alger, 2011.

[3] Madani Bezoui. *Contribution de la programmation multiobjectif dans l'optimisation des portefeuilles*. PhD thesis, Faculté de Mathématiques, 2019.

[4] Mohamed El-Amine Chergui, Mustapha Moulaï, and Fatma Zohra Ouaïl. Solving the multiple objective integer linear programming problem. In *International Conference on Modelling, Computation and Optimization in Information Systems and Management Sciences*, pages 69–76. Springer, 2008.

[5] Houssem Louakhche and Islem Hamlat. *Optimisation globale avec applications*. PhD thesis, UMMTO, 2019.

[6] Catherine Mancel. *modélisation et résolution de problèmes d'optimisation combinatoire issus d'applications spatiales*. PhD thesis, INSA de Toulouse, 2004.

[7] JF SCHEID. Méthode du simplexe dual (revisitée). 2020.

[8] Thanina Zidane and Kahina Hamdad. *Optimisation d'un problème de programmation linéaire en nombres entiers par la méthode Branch and Bound*. PhD thesis, UMMTO, 2017.

[9] RAMDANI Zoubir. Cours d'optimisation multiobjectifs.

[1] [2] [3] [4] [5] [6] [7] [8] [9]