République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Abderahmane Mira de Béjaia
Faculté des Sciences Exactes
Département de Recherche Opérationnelle

Mémoire de Fin d'Études
Pour l'obtention du diplôme de Master
Option : Modélisation Mathématique et évaluation des performance des réseaux

Thème

Programmation linéaire en nombres entiers

Présenté par :
M^{r} Abderaouf TIKOUDANE

Devant le jury composé de :

Présidente :	M^{elle}	BOUCHEBAH KAHINA
Encadreur :	M^{r}	TAOUINET SMAIL
Examinatrice :	D^{r}	ANZI AICHA
Examinatrice :	M^{elle}	BOUGHANI CHAFIA

Année Universitaire 2018-2019

Remerciements

Au terme de ce travail qui marque la fin du cycle de mastre de notre formation au sein de l'université de ABDERAHMANE MIRA, il nous est opportun d'exprimer notre gratitude à tous ceux qui, de loin ou de près, ont matériellement ou moralement contribué â la réalisation de notre modeste travail. Qu'ils trouvent ici l'expression de notre considération.

a notre Dieu, qui nous a donné la vie, l'intelligence et le courage de réaliser ce travail.
A mes chers parents par leur affection et amour de nous avoir donné la vie et l'éducation. Voila aujourd'hui nous sommes comptés parmi les hommes intellectuels du monde. Qu'ils se réjouissent du fruit de leur progéniture.

Nous exprimons nos vifs remerciements, notre profonde gratitude et notre reconnaissance à notre encadreur **Mr Taouinet smail**, qui a dirigé ce travail.
Nous tenons également à remercier les membres de jury d'avoir accepté de juger notre travail.

Enfin, Nous remercions, de tout coeur, tous ceux qui ont contribué de prés ou de loin à la réalisation de ce travail.

Dédicaces

Je dedie ce modeste travail tout d'abord à mes trés chers parents qui nous ont soutenu tout le long de mon parcour d'études ;
À mes frères et sœurs ;
À toute la famille TIKOUDANE ;
À mes amies ;
À toute la famille RO ;
À toute la promotion 2018-2019 ;

TABLE DES MATIÈRES

TABLE DES FIGURES

INTRODUCTION

La Recherche Opérationnelle est une discipline des mathématiques appliquées. Cette discipline a pour rôle le traitement d'optimalité des ressources industrielles. Depuis une décennie, plusieurs domaines comme l'économie, la finance, le marketing et la planification des entreprises, font recours à la RO. Ce qui élargit peu à peu son champ d'emploi. Actuellement, la RO a été utilisée dans la gestion des systèmes de santé et d'éducation, ainsi que dans la résolution des problèmes environnementaux.

L'une des caractéristiques majeures de la RO consiste à rechercher la solution la plus adéquate au modèle qui représente le problème traité. Cette solution est appelée « la solution optimale », d'où vient le thème de la recherche d'optimalité qui occupe une place importante dans la RO, C'est un procès sus nommé « optimisation ». Lorsque il est linéaire, il s'agit d'une programmation linéaire. Ceci exige que les formules mathématiques du modèle soient des fonctions linéaires. Ainsi, la planification d'activités est impliquée dans ce processus afin d'atteindre le résultat optimal qui témoigne l'aboutissement du but spécifié parmi les alternatives réalisables.

Le développement de la programmation linéaire est considéré parmi les plus importants progrès scientifiques. Son impact depuis 1950 a été extraordinaire. Aujourd'hui, c'est un outil standard qui a permis à de nombreuses entreprises d'économiser des milliers ou des millions de dollars dans les différents pays industrialisés du monde. Cependant, une limitation clé qui empêche l'application de cet outil est de trouver le plus souvent des valeurs non entières pour les variables de décision. Alors que dans de nombreux problèmes pratiques, les variables de décision n'ont de sens que si elles ont des valeurs entières. Par exemple, il est souvent nécessaire d'affecter des personnes, des machines et des véhicules à des activités en quantités entières. Il s'agit alors d'un problème de programmation linéaire en nombres entiers. (voir,[18])

Structurellement, notre mémoire s'organise en trois chapitres, dans le premier chapitre intitulé « Les bases de la Programmation linéaire en nombre entiers »nous lançons une introduction à l'optimisation combinatoire. Dans le deuxième

chapitre intitulé « Méthode des coupes de Gomory » nous allons mettre l'accent sur l'algorithme de coupe fractionnaire de Gomory et l'algorithme primal totalement en nombres entiers. Le troisième chapitre est consacré à la méthode de Séparation et Évaluation "Branch and Bound" ,Nous allons présenter la méthode ,son principe ,ces trois procédures essentielles et les trois stratégies de parcours ,ainsi que l'algorithme général de la méthode et un exemple d'application avec une implémentation informatique en utilisant le logiciel Matlab version 2018 et sur l'logiciel LPSolve IDE.

CHAPITRE 1

LES BASES DE LA PROGRAMMATION LINÉAIRE EN NOMBRE ENTIERS

1.1 Introduction

Quotidiennement, des problèmes de complexité affrontent la majorité des décideurs. Ces problèmes se relèvent des différents domaines tels que la Recherche Opérationnelle, la conception des systèmes mécaniques, le traitement des images, etc.
Le problème d'optimisation représente un problème qui nécessite une résolution. Ce dernier se définit comme une fonction objectif que nous cherchons à minimiser ou à maximiser par rapport à des paramètres déterminés. Cette définition se complète par la donnée des contraintes qui doivent être respectés par toutes les composantes de la solution retenue, raison pour laquelle les solutions sont souvent irréalisables. Le problème d'optimisation se divise en deux types : les problèmes discrets ou combinatoires et les problèmes continus. Dans notre étude nous nous intéressons au premier à savoir l'optimisation combinatoire. Cette dernière dispose des diverses méthodes qui nous permettent de résoudre les problèmes d'optimisation combinatoire. Bien que la modélisation de ce genre de problème fût facile, sa résolution reste généralement difficile. Dans le présent chapitre, l'accent est mis d'abord sur la Problématique et les outils fondamentaux de l'optimisation combinatoire (la complexité des algorithmes, la programmation linéaire en nombre entiers et la Programmation linéaire). Ensuite, nous traitons la formulation d'un problème de programmation linéaire, sa forme canonique et sa forme standard, la méthode du simplexe et la Dualité.

1.2 Quelques notions et notations de base pour l'optimisation

• **La programmation mathématique :** La programmation mathématique est un ensemble de méthodes ou processus mathématiques dont le but est de

trouver un optimum pour un problème décisionnel donné. C'est à dire il s'agit de rechercher l'optimum d'une fonction de n variables et de m contraintes .

• **Variables artificielles :**variables supplémentaires qui permettent de transformer des contraintes d'inégalités en contraintes d'égalités.

• **Polyèdre :** Un polyèdre $P \subset \mathbb{R}^n$ est l'ensemble des solutions d'un système fini d'inégalités linéaires.

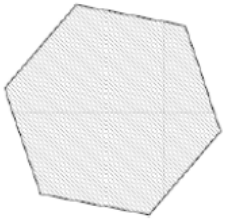
$$P = \{x \in \mathbb{R}^n / Ax \leq b\}$$

C'est l'intersection d'un nombre fini de demi-espaces
$H = \{x \in \mathbb{R}^n / ax \leq b\} = \{x \in \mathbb{R}^n / a_{i1}x_1 + ... + a_{ij}x_j + ... + a_{in}x_n \leq b_i\}$. $a \in \mathbb{R}^n, b \in \mathbb{R}^n$.

• **Polyèdre convexe :**Un polyèdre est convexe si chaque point d'un segment de droite qui joint deux points quelconques appartient au polyèdre. Dans tout polyèdre convexe, la somme du nombre de sommets et du nombre de faces est égale au nombre d'arêtes plus deux.

• **Enveloppe convexe :**
l'enveloppe convexe d'un sous-ensemble S de $\mathbb{R}^n$ est l'intersection de tous les sous-ensembles convexes de $\mathbb{R}^n$ contenant S. L'enveloppe convexe de S est par conséquent le plus petit sous-ensemble convexe de $\mathbb{R}^n$ qui contient S, on le note $conv(S)$.

• **Polytope :**
Un polytope convexe est un polyèdre convexe et borné .



polytope

1.3 La Problématique de l'optimisation combinatoire

bien que, pour fixer les idées, on introduit la problématique de l'optimisation combinatoire à partir de petits exemples.

Exemple 1 : "le problème du voyageur de commerce" : Un voyageur de commerce ayant n villes à visiter souhaite établir une tournée qui lui permette de passer une fois et une seule dans chaque ville pour finalement revenir à son point de départ, ceci en minimisant la longueur du chemin parcouru. il s'agit donc, étant donné un réseau $R = (X, U, d)$ de trouver un circuit passant une fois et une seule fois par tous les sommets (qui représentent les villes), c'est-à-dire un chemin élémentaire dont les extrémités sont confondues et ayant un nombre d'arcs égal au nombre n de villes, de longueur totale minimum.
Une manière équivalente de formuler ce problème est de ce donner une $n \times n$-matrice dont l'élément D_j^i de la $i^{éme}$ ligne et de la $j^{éme}$ colonne est égale à :

• la distance de la ville i et la ville j s'il existe un moyen d'aller directement de i à j (c'est-à-dire si $(ij) \in U$).

•∞ sinon.

Le problème consiste alors à trouver une permutation p sans cycle (non décomposable) de n objets telle que :
$\sum_{i=1}^{n} D_i^{p(i)}$ soit minimum.[14]

Exemple 2 : le problème du sac à dos : On dispose de n objets ayant chacun un poids a_j,une valeur $c_j (j = 1, 2, ..., n)$ et une capacité b. il faut effectuer une sélection (déterminer un sous- ensemble de ces objets) dont le poids totale soit inférieur ou égal à un nombre donnée et dont la valeur, somme des valeurs objets sélectionnés, soit maximum. c'est-à-dire qu'on cherche $J \subset \{1, 2, ..., n\}$ tel que : $\sum_{j \in J} c_j$ soit maximum

sous la contrainte $\sum_{j \in J} a_j \leq b$.
Associons à chaque objet $j (j = 1, 2, ..., n)$ une variable x_j qui prend la valeur 1 si le $j^{ème}$ objet fait partie de la sélection $(j \in J)$, 0 sinon. Le problème du sac à dos se formule alors de la manière suivante :

$$(k) \begin{cases} a_1x_1 + a_2x_2 + ... + a_nx_n \leq b \\ c_1x_1 + c_2x_2 + ... + c_nx_n = z(Max) \end{cases} \qquad x_j \in \{0, 1\} \quad j = 1, 2, ..., n$$

où la dernière expression signifie que « la somme $\sum_{j=1}^{n} c_j x_j$ est appelée z et on cherche à rendre la valeur de z maximum ».
Le problème traité ci- dessus doit son nom au scénario qui est souvent utilisé pour l'introduire : un campeur prépare une randonnée, les n objets sont ceux qu'il envisage d'emporter. L'objet j ayant un poids a_j et une utilité c_j, le campeur cherche à maximiser l'utilité totale de son chargement tout en limitant le

poids . Ce n'est évidement que pour des raisons historiques et folkloriques que ce modèle a conservé le nom de « sac à dos » (en anglais : « knapsack ») car il est assez mal adapté à la préparation des randonnées :
il ne tient pas compte des utilités « croisées », c'est-à-dire du fait que le dentifrice sans la brosse-à-dents et la boîte de sardines sans l'ouvre-boîtes sont de faible utilité. il suppose que le poids maximum admissible est une donnée intangible.

Dans la réalité ou bien le poids maximum doit être considéré comme une donnée « floue » avec laquelle on s'autorise à jouer ou bien on peut chercher un compromis entre la dés-utilité d'emporter un poids supplémentaire et l'utilité de disposer d'un objet de plus.
Enfin quel randonneur accepterait d'entrer dans ce formalisme ?

Tout ceci n'empêche pas le problème du sac-à-dos d'avoir de nombreuses applications pratique et constituer un modèle théorique particulièrement intéressant . il en est d'ailleurs de même d'autre modèle introduit. L'intérêt, théorique et pratique, du problème du voyageur de commerce dépasse beaucoup la préparation des tournées du V.R.P (Véhicule Routine Problème).[14]

Définition 1 : Un problème d'optimisation combinatoire est défini à partir d'un ensemble fini S et d'une application $f : S \longrightarrow \mathbb{R}$. il s'agit de déterminer $\hat{s} \in S$ où E est un ensemble fini et $S \subset \Im(E)$ où S est une collection des éléments de $\Im(E)$ qui satisfait les contraintes du problème tel que :

$$f(\hat{s}) = Min_{s \in S}[f(s)]. \qquad (1)$$

l'apparente trivialité de cette définition ne doit pas amener le lecteur à se dérouter. Reprenons l'exemple du problème du voyageur de commerce sur un réseau comportant 101 villes (on résoudre des cas à plus de 300 villes). Dire qu' « il suffit » de faire la liste des tournées et d'en choisir une de longueur minimum ne résoudre rien puisqu'il y a ici 100 ! tournées possibles, un nombre supérieur au nombre estimé d'atomes de l'univers. il faudra chercher des méthodes dont le nombre d'opérations est en rapport avec le nombre de données de problème (100 ou plutôt, ici, 100^2) et non avec le nombre de solutions potentielles. [14]

Remarque 1 : Lorsqu'on utilise le terme « problème » on se place, suivant les cas, à deux niveaux différents :

1. On peut se référer à une question générale. Ainsi les problèmes du voyageur de commerce et du sac à dos sont-ils définis dans leur généralité aux exemples 1 et 2. il s'agit alors de savoir ce qu'on peut dire avant que des valeurs numériques n'aient été données aux paramètres.

2. On peut aussi entendre le terme comme une question précise et particulière : « je considère le problème du sac à dos avec n = 3, a = (6, 5, 4), b = 9, c = (10, 8, 5) ; quelle est la cargaison optimale ? On dit qu on a ici une **instance** ou un **exemple** du problème considéré comme question générale.

Inversement,d'ailleurs, le « problème général » peut être identifié a l'ensemble des cas.En général le contexte fait ressortir clairement avec quelle acception (exemple numérique ou cas général) on utilise le terme problème et nous nous conformerons bien sur a l'usage qui veut qu on évite de préciser a quel niveau on se situe lorsque la chose est claire. Ainsi la définition 1 ci-dessus signifie qu'un problème d'optimisation combinatoire considéré comme une question générale est caractérisé par le fait que pour chacun de ses cas on a a déterminer un élément de valeur minimum dans un ensemble fini.[14]

Remarque 2 : Le problème consistant a chercher un élément maximum au lieu d'un élément minimum est de même nature puisque :

$$Max_{s\in S}[f(s)] = -Min_{s\in S}[-f(s)] \qquad (2)$$

Remarque 3 : En toute généralité il suffi, de considérer, que Implication f prend ses valeurs dans un ensemble totalement, ordonné. Très souvent f prend ses valeurs dans $\mathbb{N}$. Un cas particulier important est celui où f prend ses valeurs dans $\{0, 1\}$ (problème « de décision »).

Définition 2 : considérons le problème d'optimisation combinatoire de la définition 1. s'il existe un ensemble E (fini) et une application $c : E \longrightarrow \mathbb{R}$ tels que :

- $S \subset \Im(E)$
- $f(s) = \sum_{e\in s} c(e)$

le problème d'optimisation combinatoire est dit à « fonction objective séparée ».
Le problème de sac à dos est clairement à fonction objective séparée (il suffit de poser E = {1, 2, ..., n}). On montre que le problème de voyageur de commerce est aussi à fonction objective séparée.[14]

1.4 Outils fondamentaux de l'optimisation combinatoire

1.4.1 Complexité théorique d'un problème

Cette notion de problème combinatoire est formellement caractérisée par la théorie de la complexité qui propose une classification des problèmes en fonction de la complexité de leur résolution.[3]

On entend ici par « complexité d'un problème » une estimation du nombre d'instructions à exécuter pour résoudre les instances de ce problème, cette estimation étant un ordre de grandeur par rapport à la taille de l'instance. Il s'agit là d'une estimation dans le pire des cas dans le sens où la complexité d'un problème est définie en considérant son instance la plus difficile. Avant de passer aux classes des problèmes, il est nécessaire de se familiariser avec les termes suivants :

– Instance d'un problème d'optimisation combinatoire (OC).

– Problème de décision.

– Problème d'optimisation.

– Problème polynomial.

Définition 3 :(Instance d'un problème d'optimisation combinatoire (OC).

Soit X un problème caractérisé par l'ensemble E de ses données, E = $(e_1, e_2, ..., e_n)$. Une instance de X serait alors caractérisée par un ensemble concret $E^{'} = (e_1^{'}, e_2^{'}, ..., e_n^{'})$. Une deuxième instance serait caractérisée par un deuxième ensemble concret $E^{''} = (e_1^{''}, e_2^{''}, ..., e_n^{''})$ .et ainsi de suite. L'invariant suivant est toujours vérifié : $|E| = |E^{'}| = |E^{''}|$.[3]

Définition 2 : (Problème de décision).
C'est un problème qui possède pour solution Oui/Non. On en déduit qu'un problème décisionnel abstrait fait correspondre à toute instance du problème l'ensemble des solutions vrai/faux.[3]

Définition 4 : (Problème d'optimisation).
Un problème d'optimisation est un problème caractérisé par un quadruplet $<< I, S, f, mode >>$ où I représente les données du problème, S est le type de collection, ensemble ou le type d'éléments qu'on attend comme résultat, f est la fonction qui mesure la qualité du résultat et *mode* (max ou min) indique s'il s'agit de maximisation ou minimisation.[3]

Définition 5 : (Problème polynomial).
C'est un problème combinatoire pour lequel existe un algorithme polynomial pour le résoudre.

1.4.2 Classes de complexité

L'expérience montre que certains problèmes sont plus faciles à résoudre que d'autres, dans le sens où la meilleure solution peut être obtenue rapidement. La théorie de la complexité a été développée pour permettre à classer mathématiquement les problèmes selon leur difficulté. On définit les trois grandes classes P, NP et NP-Complet. Le lecteur intéressé par de plus amples informations pourra consulter différents ouvrages dédiés à la complexité, dont les livres de Garey et Johnson [19].

La classe P

Elle contient l'ensemble des problèmes polynomiaux, i.e., pouvant être résolus par un algorithme de complexité polynomiale. Cette classe caractérise l'ensemble des problèmes que l'on peut résoudre « efficacement ». On considère les problèmes appartenant à cette classe comme étant facile.

La classe NP

C'est la classe des problèmes de décision pour lesquels il est possible de vérifier en temps polynomial qu'une solution donnée est réalisable ; c'est-à-dire, on peut construire un algorithme polynomial qui est capable de vérifier si pour une solution donnée, la réponse au problème de décision est OUI ou NON. Aussi, tout problème de décision qui peut être résolu par un algorithme polynomial, donc appartenant à la classe P, appartient également à la classe NP, d'où $P \subseteq NP$ cependant $NP \in P$ reste une conjecture.

La classe NP-complet

Un problème de décision est dit « NP-complet » si tout problème de la classe NP peut se réduire polynomiale-ment à lui.

1.4.3 programmation linéaire en nombre entiers

Étant donnée une m x n-matrice A, un m-vecteur colonne b et un n-vecteur ligne c, on appelle « programme linéaire en nombres entiers » le problème d'optimisation :

$$(P)\begin{cases} z(Max) = cx \\ Ax \leq b \end{cases} \qquad x_j \in \mathbb{N}\ j = 1, 2, ..., n$$

ou z(Max) a la même interprétation qu'au paragraphe précédent ; x est un n-vecteur inconnu. Dans le cas particulier ou les variables $x_j \in N$ sont remplacées par $x_j \in \{0, 1\}$ on dit qu'on a un « programme linéaire en 0,1 » ou « programme linéaire en variables bivalentes ».[14]

Le problème du sac a dos peut donc être caractérisé (cf. la formulation (K)) par le fait qu'il s'agit d'un programme linéaire en variables bivalentes n'ayant qu'une contrainte. On notera que l'extension de(K) dans laquelle les variables x_j sont astreintes a être entières et non plus seulement a appartenir a { 0, 1 } est également désignée sous le nom de problème de sac a dos.[14]

1.4.4 programmation linéaire

Lorsqu'on « relâche » les contraintes d'intégrité sur les variables ($x_j \in \mathbb{N}$) dans le programme linéaire en nombres entiers (P) ci-dessus.

$$(PR)\begin{cases} Ax \leq b \\ cx = z(max) \end{cases} \qquad x \geq 0$$

le problème (PR) est appelé programme linéaire relaxé associé au programme linéaire en nombres entiers.
Par construction il existe une relation étroite entre (P) et (PR). Si par hasard la solution optimale de (PR) est entière, c'est aussi une solution optimale de (P).

La résolution du programme linéaire (PR) peut grandement aider à résoudre (P) et on verra des méthodes dans lesquelles la solution d'un programme linéaire en nombres entiers passe par la résolution de programmes linéaire relaxé associés.[14] il faut cependant noter que la méthode de solution de (P) consistant à « arrondir » aux entiers les plus proches une solution de (PR) est en général, mauvaise. Considérons par exemple le problème de sac à dos :

$$(3)\begin{cases} 6x_1 + 5x_2 + 4x_3 \leq 9 \\ 10x_1 + 8x_2 + 5x_3 = z(Max) \end{cases} \qquad x_1, x_2, x_3 \in \mathbb{N}$$

Le programme linaire relaxé a pour solution $x_1 = 3/2; x_2 = x_3 = 0$. Si on arrondit x_1 à 2 on obtient une solution inacceptable. Si on arrondit x_1 à 1, on obtient une solution bien éloignée de la solution optimale $x_1 = 0, x_2 = x_3 = 1$.
Mais la raison principale du rôle central joué par la programmation linéaire n'est pas la relation étroite de celle-ci avec la programmation linéaire en nombres entiers. L'ensemble des solutions réalisables d'un programme linéaire correspond a un polyèdre convexe de $\mathbb{R}^n$ (un tétraèdre, un cube, un diamant sont des polyèdres convexes de $\mathbb{R}^3$) et il existe toujours un sommet du polyèdre qui coïncide avec une solution optimale (si toutefois une telle solution existe), Or a un problème d'optimisation combinatoire on peut en général faire corresponds un ensemble de points d'un espace $\mathbb{R}^n$. On montre que « I'enveloppe convexe » de cet ensemble de points est un polyèdre convexe. Et par conséquent, si on réussit à déterminer les faces de ce polyèdre convexe, on se ramène a la résolution d'un programme linéaire.[14]
Or on connait un algorithme pratiquement très efficace pour résoudre les programmes linéaires : la méthode du simplexe introduite en 1947 par G. B. Dantzig. L'aspect combinatoire de la programmation linéaire n'a été dégagé que progressivement puisque (PR) se présente d'abord sous la forme d'un problème d'optimisation continue.[14]

1.5 Formulation d'un problème de programmation linéaire

1.5.1 Définition

La programmation linéaire a pour objet l'étude d'une catégorie particulière de problèmes d'optimisation : celle ou la fonction à optimiser est une fonction linéaire, appelée fonction objectif ou aussi fonction économique ; les liaisons entre les variables s'appellent les contraints (qui sont sous forme d'équations ou d'inéquations linaires).

Remarque : D'une façon générale, résoudre un problème de programmation linéaire consiste à déterminer les valeurs des variables de décision qui maximisent (on minimisent, selon le cas) une fonction économique (linéaire) soumise à un ensemble de contraintes (linéaire).

1.5.2 Forme générale d'un programme linéaire

La forme générale d'un problème de programmation linéaire est la suivante :

$$\begin{cases} Optz = \sum_{j=1}^{n} c_j x_j \qquad (1.1) \\ \begin{cases} \sum_{j=1}^{n} a_{ij} x_j \geq (\leq)\ bi; i = \overline{1,p} \qquad (1.2a) \\ \sum_{j=1}^{n} a_{ij} x_j = b_i; i = \overline{p+1,m} \qquad (1.2b) \end{cases} \\ x_j \geq 0; j = \overline{1,q} \qquad (1.3) \\ x_j \text{ de signe quelconque } j = \overline{q+1,n} \end{cases}$$

avec c_j,b_i,a_{ij} des constantes réelles et x_j les variables.

- il ya **n** variables notées x_j $(j = \overline{1,n})$
- z : représente la fonction à optimiser.
- c_j : est le coefficient de la variable x_j dans cette fonction.
- i :(i=$\overline{1,m}$) est l'indice des "m" contraintes réelles.
- a_{ij} : le coefficient de la variable x_j dans la contrainte i (appelé coefficient technologique).
- b_i : second membre de cette contrainte.

Les notations vectorielles suivantes seront indispensables.
- $x = (x_j)_{j=\overline{1,n}}$ vecteur colonne $(n \times 1)$.
- $c = (c_j)^t_{j=\overline{1,n}}$ vecteur ligne $(1 \times n)$.
- $b = (b_i)_{i=\overline{1,m}}$ vecteur colonne $(m \times 1)$.
- $A = (a_{ij}; i = \overline{1,m}, j = \overline{1,n})$ matrice $(m \times n)$.
- $a_i = (a_{i1,\cdots,a_{in}})$ vecteur ligne $(1 \times n)$ de la matrice A.
- $A_j = (a_{i1}, \cdots, a_{mj})^t$ vecteur colonne $(m \times 1)$ de la matrice A.

La forme générale d'un problème de programmation linéaire s'écrit alors :

$$\begin{cases} \text{opt z} = \text{cx} \\ a_i x \geq (\leq) b_i; i = \overline{1,p} \\ a_i x = b_i; i = \overline{p+1,m} \\ x_j \geq 0; j = \overline{1,q} \\ x_j \text{ de signe quelconque } j = \overline{q+1,n}. \end{cases}$$

Cette forme générale peut être simplifiée ; en la ramenant à des formes plus compactes mais équivalentes ; en particulier aux formes dites "canonique" et "standard " [9].

1.5.3 Forme canonique d'un programme linéaire

ll est toujours possible de ramener :
- L'optimisation à une minimisation (maximiser la fonction z est équivalant a minimiser la fonction $z' = (-z)$.

- Toute les inégalités (1.2.a) de même type (il suffit de les multiplier par (-1) le cas échéant) soit par convention, des inégalités "supérieur ou égale ($\geq$)".
Toute les variables à être non négatives, une variable x_j de signe quelconque peut être décomposée :
• Soit en deux variables non négatives :
x_j=x_j^+-x_j^- avec x_j=$x_j^+ = max(0, x_j)$ et $x_j^- = max(0, -x_j)$
ce qui présente l'inconvénient de doubler le nombre de variables. [9].

- Toute les contraintes (1.2.b) à des formes d'inégalités au prix du dédoublement suivant :

$$a_i x = b_i \Longleftrightarrow \begin{cases} a_i x \geq b_i \\ \text{et} \\ -a_i x \geq -b_i \end{cases}$$

En notation matricielle, la forme canonique est donc :

$$(pc) : \begin{cases} minz = cx \\ Ax \geq b \\ x \geq 0 \end{cases}$$

Remarque 1.2.2

Deux propriétés caractérisent la forme canonique :

• Toutes les variables $(x_j)_{j=\overline{1,n}}$ sont astreintes a être positives ou nulles.

• Toutes les autres contraintes sont des inéquations.

Remarque 1.2.3

La forme canonique nous permet de trouver le dual d'un programme linéaire facilement.

1.5.4 Forme standard d'un programme linéaire

(a) **Changement des inéquations :**

Si on a une inéquation de la forme :
$a_{i1}x_1 + a_{i2}x_2 + ... + a_{in}x_n \leq b_i$ (1.4)
alors pour la changer, il faudrait ajouter une variables x_{n+1} ; avec $x_{n+1} \geq 0$ (1.5)
on a alors :
$a_{i1}x_1 + a_{i2}x_2 + ... + a_{in}x_n + x_{n+1} = b_i$ (1.6)
x_{n+1} est appelée variable d'écart.

Théorème 1.2.1
A chaque solution $\overline{x}$ de l'équation (1.4), $\overline{x}$=$(B_1, B_2, ..., B_n)$,il correspond une solution unique $\overline{y}$,$\overline{y} = (B_1, B_2, ..., B_n, B_{n+1})$ de l'équation (1.6) et vérifiant la contrainte (1.5).

Réciproquement, à chaque solution $\overline{y}$ de (1.6) il correspond une solution unique $\overline{x}$ de (1.4)

Démonstration.

1. Soit $\overline{x} = (B_1, B_2, ..., B_n)$ une solution de (1.4).

   Alors $a_{i1}B_1 + a_{i2}B_2 + ... + a_{in}B_n \leq b_i \Longrightarrow \underbrace{b_i - (a_{i1}B_1 + a_{i2}B_2 + ...)}_{B_{n+1}} \geq 0$. On pose $B_{n+1} = b_i - (a_{i1}B_1 + a_{i2}B_2 + ... + a_{in}B_n)$, on aura : $\overline{y} = (B_1, B_2, ..., B_n, B_{n+1})$ vérifiant (1.5) et (1.6).

2. Réciproquement :
   Soit $\overline{y} = (B_1, B_2, ..., B_n, B_{n+1})$ vérifiant (1.5) et (1.6).

$$\Longrightarrow \begin{cases} B_{n+1} \geq 0 \\ a_{i1}B_1 + a_{i2}B_2 + ... + a_{in}B_n + B_{n+1} = b_i \Longrightarrow a_{i1}B_1 + a_{i2}B_2 + ... + a_{in}B_n \leq b_i. \end{cases}$$

Remarque 1.2.4
Toute inégalité linéaire de la forme $a_{i1}x_1 + a_{i2}x_2 + ... + a_{in}x_n =\leq b_i$ peut être transformée sous la forme d'une équation et d'une inéquation :

$$\begin{cases} a_{i1}x_1 + a_{i2}x_2 + ... + a_{in}x_n + x_{n+1} = b_i \\ x_{n+1} \geq 0. \end{cases}$$

(b) **Forme standard :**
Tout problème de programmation linéaire, peut se mettre sous la forme suivante :

$$\begin{cases} min\ z = c_1x_1 + c_2x_2 + ... + c_nx_n \\ a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n = b_1 \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + ... + a_{mn}x_n = b_m \\ x_j \geq 0, j = \overline{1.n}; m < n. \end{cases}$$

Si $b_i < 0$, c'est à dire $a_{i1}x_1 + a_{i2}x_2 + ... + a_{in}x_n = b_i$ avec $b_i < 0$; alors $-a_{i1}x_1 - a_{i2}x_2 - ... - a_{in}x_n = -b_i \Longrightarrow a'_{i1}x_1 + a'_{i2}x_2 + ... + a'_{in}x_n = b'_i$ avec $b'_i = -b_i$ et $a'_{ij} = -a_{ij}, \quad j = \overline{1.n}$

L'écriture matricielle de la forme standard d'un programme linéaire est donnée par :

$$(ps) : \begin{cases} minz = cx \\ Ax = b \\ x \geq 0 \end{cases}$$

Remarque 1.2.5 Deux propriétés caractérisent la forme standard :

- Toute les variables $(x_j)_j = \overline{1.n}$ sont astreintes a être positives ou nulles.
- Toute les autres contraintes sont des équations.

Remarque 1.2.6

la forme standard est utilisée par l'algorithme du simplexe.

Démonstration
La $i^{éme}$ contrainte de (ps) s'écrit :

$$a_i x = b_i$$

et cette équation est équivalente à l'ensemble des deux inéquation :

$$a_i x \leq b_i;\ -a_i x \leq -b_i \qquad (1.7)$$

d'autre part l'inéquation et équivalente à la paire équation-contrainte de signe :

$$a_i x + \epsilon_i = b_i \qquad (1.8)$$

telle que $\epsilon_i \geq 0$.
et (pc) s'écrit bien sous la forme standard :

$$(ps) : \begin{cases} minz = cx \\ Ax + U\epsilon = b \\ x \ \geq \ 0 \\ \epsilon \ \geq \ 0 \end{cases}$$

(ou U est la $m \times m$ matrice identité)
La variable ϵ_i qui permet d'écrire l'inéquation (1.7) sous la forme (1.8) est appelée " variable d'écart ", la valeur de la variable d'écart ϵ_i mesure la différence entre le terme de droite et le terme de gauche de l'inéquation (1.7).

1.6 La méthode du Simplexe

1.6.1 Caractérisation des solutions du problème

Un programme linéaire peut toujours se présenter sous forme standard :

$$Maximiserz = cx \tag{1.1}$$

$$souscontraintesAx = b \tag{1.2}$$

$$x \geq 0 \tag{1.3}$$

Dans cette formule le vecteur x contient toutes les variables y compris les variables d'écart ; il s'agit d'un vecteur colonne de type $(n \times 1)$.

$C = (c_1, c_2, ..., c_n)$: le vecteur ligne d'ordre $(1 \times n)$.

$A = (a_1, a_2, ..., a_n)$: la matrice d'ordre $(m \times n)$.

$b = (b_1, b_2, ..., b_n)^t$: le vecteur d'ordre $(m \times 1)$.

Définitions [4]

• Une partie C d'un espace vectoriel E est **convexe** si toute combinaison convexe de vecteurs de C est appartient à C .

$$\text{c-à-d : } \lambda\upsilon + (1-\lambda)\omega \in C, \forall(\upsilon, \omega) \in C^2 \text{ et } \forall\lambda \setminus 0 \leq \lambda \leq 1.$$

• **Une combinaison linéaire convexe des vecteurs** $\upsilon_1, \upsilon_2, ..., \upsilon_p$ d'un espace vectoriel $\mathbb{R}^n$ défini sur le corps $\mathbb{R}$ est un vecteur υ, de $\mathbb{R}^n$, de la forme :

$$\upsilon = \alpha_1\upsilon_1 + \alpha_1\upsilon_2 + ... + \alpha_p\upsilon_p, \text{ où } \alpha_i \geq 0, i = 1, ..., p \text{ et } \sum_{i=1}^{p} \alpha_i = 1.$$

• **Un point extrême** d'un polyèdre convexe D, c'est tout point $x \in D$ qui ne peut s'exprimé comme combinaison convexe d'autres point de D.

• **Un simplex** est un polyèdre convexe dans un espace de dimension "n" qui a exactement (n+1) points extrêmes.

• On appelle **solution** d'un problème de programmation linéaire tout vecteur x qui satisfait les contraintes (1.2).

• Une solution et appelée **solution réalisable** si elle vérifie les contraintes de non-négativité (1.3).
Dans le cas contraire, on dit que la solution **n'est pas réalisable**.

• Une solution réalisable est une **solution optimale** s'il n'existe pas d'autres solution réalisable qui fournissent une plus petite valeur de la fonction objectif (pour un problème de minimisation).

• On appelle base toute sous-matrice carrée régulière $(m \times m)$ extraite de A.
L'ensemble des contraintes (1.2) s'écrit donc comme un système de m équation à n inconnues : $Ax = b$.
Pour développer la méthode du simplexe, on peut former à partir de A, une sous

matrice B $(m \times m)$ non-singulière.cette matrice B peut être formée par n'importe quel ensemble de m colonnes linéairement indépendantes de A. La matrice B est appelée matrice de base puisqu'elle est formée de m vecteurs linéairement indépendants. Sans restreindre la généralité, on peut supposer que les colonnes de $A = [A_1, ..., A_n]$ ont été ordonnées de manière à pouvoir écrire A sous la forme $A = [B, N]$, avec B de dimension $(m \times m)$ la matrice de base,N de dimension $(m \times (n-m))$ contenant les colonnes de A qui n'appartiennent pas à B. De même on peut partitionner x en $[x_B, x_N] et [c_B, c_N]$.
Le programme linéaire (1.1)-(1.3) peut donc être reformulé de la manière suivante :

$$Minimiser\ Z = c_B x_B + c_N x_N \tag{1.4}$$

$$sous\ contraintes\ Bx_B + Nx_N = b \tag{1.5}$$

$$x_B, x_N \geq 0 \tag{1.6}$$

• **Une solution de base** (associée à la base B) C'est la solution particulière de (1.5) obtenue en faisant $x_N = 0.x_B$ est déterminé de façon unique par la résolution du système (de Cramer) :

$$Bx_B = b \Longrightarrow x_B = B^{-1}b$$

• **Une solution de base** est dite **réalisable** si : $x_B \geq 0$, c-à-d : si $B^{-1}b \geq 0$.

• **Une base réalisable** c'est une base qui correspond à une solution de base réalisable.

1.6.2 Algorithme de la méthode du simplexe

Principe de l'algorithme

Pour déterminer la (ou les) solution(s) optimale(s) d'un problème de programmation linéaire, il suffit, après avoir mis le problème sous forme standard, de s'intéresser aux solutions de base admissibles. c-a-d aux sommets du polyèdre D (s'il n'est pas vide) .

L'algorithme du simplexe utilise une procédure itérative qui convergera vers la (les) solution (s) de base admissible(s) fournissant l'optimum de la fonction objectif ou, pour défaut, mettra en évidence que l'ensemble D est vide.

L'algorithme du simplexe contiendra deux phases :

Phase 1 : Procédure d'initialisation

Déterminer une solution de base admissible c-a-d les coordonnées d'un premier sommet de D. Si cette procédure échoue cela signifie que D est vide.

Phase 2 : Procédure itérative

Calculer, à partir d'une solution de base admissible une autre solution de base admissible donnant une valeur meilleure pour la fonction objectif ; géométriquement , une itération consistera a passer d un sommet de D a un autre sommet de D.

Ce nouveau sommet sera adjacent au précédent en ce sens ils seront les deux extrémités d'une arête de D ; les coordonnées de ces deux sommets adjacents correspondront a deux solutions de base admissibles. Les ensembles d'indices de base (I) et hors base (J) ne différent que d'un seul indice. Cette caractéristique permettra de déterminer la nouvelle solution de base admissible a partir de l'ancienne.

Enfin il existera une procédure d'arrêt de l'algorithme : deux tests d'arrêt mettront en évidence, respectivement que :
• une solution optimale est déterminée.
• il n'existe pas de solution optimale finie : D est non borné et Z peut y tendre vers moins l'infini. $(-\infty)$.

1.6.3 L'algorithme primal du simplexe

Le principe de résolution nécessite un certain nombre d'étapes dont la démarche est la suivante :
On l'applique pour un problème de maximisation :

1. Écrire le système sous la forme standard.

2. Le point de départ de l'algorithme est l'existence d'une solution réalisable soit :$x = (x_B, x_N) = (x_B, 0)$ et A_B^{-1} :La matrice inversible de A_B.

3. Construire le premier tableau correspondant à la forme standard.

4. Calculer les $E_j = z_j - c_j$, tq : $Z_j = C_B^T \times A_B^{-1}$ (critère d'optimalité $E_j \leq 0$).

5. Si tous les $E_j \geq 0, \forall j \in J_N, 1 \leq j \leq (n-m)$.La solution $(x_B, 0)$ est optimale.
   Si non :
   – Si $\exists k \in J_N$ tel que :$E_k < 0$ et $A_B^{-1} a_k < 0$.
   Le (PL) n'admet pas un maximum (non borné), arrêt .

   – Sinon si $\exists k \in J_N$ tel que :$E_K < 0$ et $A_B^{-1} a_k > 0$ alors ,aller à 6.
6. Choisir la variable x_k qui entre dans la base tq :$E_k = min_{j \in J_N} E_j < 0$.

7. Choisir la variable xl qui sort de la base tel que :
   $\theta_l = min\{\frac{b_i}{a_{ik}}, a_{ik} > 0, 0 \leq i \leq m\}$, la variable sortante se trouve sur la $l^{éme}$ ligne.

8. Encadrer le pivot a_{lk} .

9. Multiplier la ligne du pivot par le rapport : $\frac{1}{a_{lk}}$.

10. Calculer les valeurs a'_{ij} des autres lignes :
$a'_{ij} = a_{ij} - (\frac{a_{ik} \times a_{lj}}{a_{lk}})$
aller à **l'étape 4**.

1.7 Méthode dual du simplexe

Pour éviter l'utilisation des variables artificielles ,quand on a un problème avec des contraintes sous forme :

$$\sum_{j=1}^{n} a_{ij} x_j \geq b_i, i \in 1, ..., m$$

On utilise l'algorithme dual du simplexe afin de réduire la taille du (PL) et le nombre d'itération .[12]

1.7.1 Principe de la dualité

Son principe est de garder le critère d'optimalité satisfait à chaque itération et de rendre positifs les b_i qui sont négatifs.

• Au programme linéaire primal :

$$(Pl) \begin{cases} max\ Z = c^T x \\ Ax \leq b \\ x \geq 0 \end{cases}$$

On associe le programme linéaire dual :

$$(Pl) \begin{cases} min\ Z' = b^T y \\ A^T y \geq c \\ y \geq 0 \end{cases}$$

• Les différentes transformations sont résumées dans le tableau suivant :[1]

Primal	**Dual**
max Z	min Z'
coefficient c de Z	second membre b
second membre b	coefficient c de Z'
$i^{éme}$ contrainte =	$y_i \in \mathbb{R}$
$i^{éme}$ contraintes $\geq$	$y_i \geq 0$
$i^{éme}$ contraintes $\leq$	$y_i \leq 0$
$x_j \in \mathbb{R}$	$j^{ème}$ contraintes =
$x_j \geq 0$	$j^{ème}$ contraintes $\leq$
$x_j \leq 0$	$j^{ème}$ contraintes $\geq$

• **Exemple :**

$$(PL)\begin{cases} maxZ = 6x_1 + 4x_2 \\ 3x_1 + 9x_2 \leq 81 \\ 4x_1 + 5x_2 \leq 55 \\ 2x_1 + x_2 \leq 20 \\ x_1, x_2 \geq 0. \end{cases}$$

On associe le programme linéaire dual :

$$(PLD)\begin{cases} minZ' = 81y_1 + 55y_2 + 20y_3 \\ 3y_1 + 4y_2 + 2y_3 \geq 6 \\ 9y_1 + 5y_2 + y_3 \geq 4 \\ y_1, y_2, y_3 \geq 0 \end{cases}$$

1.7.2 L'algorithme dual du simplexe

Le principe de résolution nécessite un certain nombre d'étapes contenues au travers de l'algorithme dual du simplexe dont la démarche est la suivante ici on l'applique pour un problème de maximisation :

1. Écrire le système sous la forme standard avec les $c_j \geq 0, \forall j = 1, ..., n$.

2. Construire le premier tableau correspondant à la forme standard.

3. Calculer les $E_j = Z_j - c_j$.

4. Choisir la variable x_r qui sort de la base :
   $b_r = \min\ b_i$ tq : $b_i < 0, i \in J_B$.
   Si b_r n'existe pas alors fin,l'optimum est atteint.
   Sinon aller à **l'étape 5**.

5. Choisir la variable x_k qui entre dans la base :
   $\frac{E_k}{a_{rk}} = max\{\frac{z_j - c_j}{a_{rj}}\}$ tq :$a_{rj} < 0, j \in J_N$.
   Le pivot doit être négatif, sinon fin,le (PL)est non borné.

6. Encadrer le pivot .

7. Multiplier la ligne du pivot par le rapport : $\frac{1}{a_{rk}}$.

8. Calculer les valeurs a'_{ij} des autres lignes :
   $a'_{ij} = a_{ij} - \left(\frac{a_{ik} * a_{rj}}{a_{rk}}\right)$.
   aller a **l'étape 3**.

1.8 Conclusion

Dans ce chapitre, nous avons introduit l'optimisation combinatoire en donnant ses caractéristiques. L'optimisation des problèmes combinatoires est une optimisation discrète dans laquelle on cherche à trouver une solution dans un espace fini qui maximise (minimise) une fonction objectif. Nous avons introduit la théorie de la complexité, car la connaissance de la complexité d'un problème nous donne directement une idée sur l'outil à utiliser pour le résoudre.

CHAPITRE 2

MÉTHODE DES COUPES DE GOMORY

2.1 Introduction

parmi les méthodes de résolution d'un problème de programmation linéaire en nombre entiers (PLNE) c'est la méthode des coupes de Gomory qu'on va les présenter dans ce chapitre en détaille.

2.2 Principe des méthodes de coupes

Nous commençons par résoudre (P.L.N.E). Si la solution optimale obtenue est un point à coordonnées entières, c'est terminé. Elle est la solution de notre problème initial à variables entières et le problème est résolu. Sinon (et c'est évidemment la situation la plus fréquemment rencontrée), on peut toujours tronquer le domaine des solutions en rajoutant une contrainte supplémentaire au problème de façon à éliminer ce point optimal sans exclure aucune solution entière. Une telle contrainte est appelée "une coupe" ou encore une : " inégalité valide ".

Beaucoup d'inégalités valides peuvent donc être générées pour éliminer une solution fractionnaire donnée.

Après avoir rajouté une coupe (ou éventuellement plusieurs), le programme linéaire augmenté des contraintes est à nouveau résolu en continu par la méthode du simplexe. Comme l'ancienne solution reste duale réalisable, il est avantageux d'utiliser pour cela l'algorithme dual du simplexe.

Si la solution optimale de ce nouveau problème est entière, c'est terminé : on a obtenu une solution optimale du problème en nombres entiers. Sinon, le raisonnement précédent peut être répété : on recherche une nouvelle coupe (éventuellement plusieurs) que l'on rajoutera à l'ensemble des contraintes ; puis le programme linéaire ainsi augmenté sera maximisé à nouveau, etc.

Si les coupes sont correctement choisies à chaque étape, le polyèdre initial se réduit jusqu'à coïncider avec l'enveloppe convexe des solutions entières, au moins au voisinage de la solution optimale. La solution continue du problème augmenté deviendra alors entière et le problème sera résolu.[3]

2.3 Coupes de Gomory [10]

L'une des coupes les plus étudiées et utilisées est la coupe de Gomory. C'est une coupe adaptée à la méthode du simplexe (et dual du simplexe).
Soit un problème de (P.L.N.E) suivant :

$$\begin{cases} max(z) = \ Cx. \\ x \in S = \{x \in \mathbb{Z}_+^n, Ax \leq b\}. \end{cases}$$

On cherche une solution optimale x^0
Ou a défait une $\epsilon-$ approximation $\overline{x}^\epsilon$ de la solution optimale x^0 c'est à dire $||\ \overline{x}^\epsilon$ - $x^0\ || < \epsilon$.

Hypothèse : On suppose que :

- Le domaine des solutions réalisable S $\neq \varnothing$.
- La valeur de la fonction objective à l'optimum est bornée c'est à dire Z $(x^0) = Cx^0 < +\infty$.

Définition 1 : Étant donné un (P.L.N.E), on dit que l'inégalité $\alpha x \leq \alpha_0$. est valide pour S si :

$$\forall \text{ x , x} \in \text{S on a : } \alpha \text{ x} \leq \alpha_0.$$

Définition 2 : une coupe est une inégalité $\alpha x \leq \alpha_0$. valide pour S, mais non valide pour S_R où S_R est le domaine relaxé de S.

2.3.1 théorème 1 (Chvâtal-Gomory)[10]

Si $\overline{b}_{i0}$ est un nombre irrationnel c'est à dire $\overline{b}_{i_0} \notin \mathbb{Z}_+$ alors l'inégalité :

$$\sum_{j \in \overline{J}} \alpha_{i_0 j} x_j \geq \alpha_{i_0}$$

est une coupe c'est à dire valide pour S et non valide pour S_R où :

$$\alpha_{i_0 j} = \overline{a}_{i_0} - \lfloor \overline{a}_{i_0} \rfloor \text{ et } \alpha_{i_0} = \overline{b}_{i_0} \text{ - } \lfloor \overline{b}_{i_0} \rfloor$$

$$\alpha_{i_0 j} = < \overline{a}_{i_0 j} > \text{ et } \alpha_{i_0} = < b_{i_0} >$$

2.3.2 Preuve 1

Supposons qu'on a un P.L.N.E définie par :

$$(P.L.N.E) \begin{cases} MaxZ = cx \\ x \in S = \{x \in \mathbb{Z}_+^n;\ Ax \leq b\} \end{cases}$$

Le problème relaxé associé s'écrit :

$$(P.R) \begin{cases} MaxZ_R = Cx \\ x \in S_R = \{x \in \mathbb{R}_+^n;\ Ax \leq b\} \end{cases}$$

Soit J une base optimale du problème relaxé (P.R), alors la forme canonique de (P.R) par rapport à la base J s'exprimer par :

$$(P.R) \Longleftrightarrow \begin{cases} MaxZ_R = \pi b + \overline{c}_{\overline{J}}.x_{\overline{J}} \\ \quad x_J + [(A_J)^{-1}.A_{\overline{J}}].x_{\overline{J}} = (A_J)^{-1}.b \\ \quad x_J \geq 0,\ x_{\overline{J}} \geq 0. \end{cases}$$

où :
– $\pi = c_J(A_J)^{-1}$ est le vecteur multiplicateur du simplex associé à la base **opt J**.
– $\overline{C}_J = C$ - $\pi.A_j$ est le vecteur, des coûts relatifs à la base J.
– $\overline{b} = (A_J)^{-1}b$.

$$(P.R) \Longleftrightarrow \begin{cases} MaxZ_R = C_J.(AJ)^{-1}.b + \overline{C_{\overline{J}}}.x_{\overline{J}} \\ x_J + [(A_J)^{-1}.A_{\overline{J}}]x_{\overline{J}} = (A_J)^{-1}.b \\ x_J \geq 0,\ x_{\overline{J}} \geq 0. \end{cases}$$

En posant

$$(P.R) \Longleftrightarrow \begin{cases} Max\ Z_R = C_J\overline{b} + \overline{C}_{\overline{J}}x_{\overline{J}} \\ x_j + [(A_j)^{-1}.A_{\overline{J}}].x_{\overline{J}} = \overline{b} \\ x_j \geq 0,\ x_{\overline{J}} \geq 0 \end{cases}$$

pour tout $j \in \overline{J}$
En posant

$$\begin{cases} \overline{a}_j = (A_J)^{-1}.a_j \\ \overline{C}_J = C_j - C_{\overline{J}}.\overline{a}_j \end{cases} \quad \text{pour tout } j \in \overline{J}$$

On aura :

$$(P.R) \Longleftrightarrow \begin{cases} Max\ ZR = c_J.\overline{b} + \sum_{j\in\overline{J}} \overline{C}_j.x_j \\ x_{i=b_i} + \sum_{i\in\overline{J}} \overline{a}_{ij}.x_j = \overline{b}_i. \\ x_i \geq 0 \quad x_j \geq 0.\ i = 1, ..., m\ j = m+1, ..., n \end{cases}$$

Ce n'est rien d'autre que le tableau optimal du simplexe correspondant à la base optimale J du problème relaxé $(P.R)$.

A la base optimale J correspond à la solution de base réalisable et optimale $\overline{x}^R = (\overline{x}^R_J, \overline{x}^R_{\overline{J}}) = (\overline{b}_{\mathbb{R}^n}, O_{\mathbb{R}^{n-m}})$

Soit $i_0 \in \{1, ..., m\}$ on a :

$$x_{i_0} + \sum_{j\in\overline{J}} \overline{a}_{i_0}.x_j = \overline{b}_{i_0} \qquad (1).$$

$\Rightarrow x_{i_0} + \sum_{j\in\overline{J}} \lfloor \overline{a}_{i_0 j} \rfloor.x_j \leq \overline{b}_{i_0}. \quad car : x_j \geq 0.$

(a) Soit $x \in S = \{x \in Z^n_+ : Ax \leq b\}$.

$\Rightarrow (x_j, x_{\overline{J}}) \in Z^n_+$, d'où :

$x_{i_0} + \sum_{j\in\overline{J}} \lfloor \overline{a}_{i_0 j} \rfloor.x_j \leq \overline{b}_{i_0}$ est entier

Si : $a \leq$ b et que a est entier alors $a \leq \lfloor b \rfloor$.

d'où :

$$x_{i_0} + \sum_{j\in\overline{J}} \lfloor \overline{a}_{i_0 j} \rfloor.x_j \leq \lfloor \overline{b}_{i_0} \rfloor \qquad (2).$$

(1) - (2) =

$$\sum_{j\in\overline{J}} (\overline{a}_{i_0\overline{j}} \lfloor a_{i_0 j} \rfloor).x_j \geq \overline{b}_{i_0} - \lfloor \overline{b}_{i_0} \rfloor.$$

$$\sum_{j\in\overline{J}} < \overline{a}_{i_0 j} > .x_j \geq < \overline{b}_{i_0} >$$

$$\sum_{j\in\overline{J}} \overline{\alpha}_{i_0 j}.x_j \geq \ \alpha_{i_0}$$

est une inégalité valide pour S.

(b) Soit $\overline{x}^R \in (S_R \backslash S) \Rightarrow \exists!\ i_0 \in \{1, ..., m\}$ tel que :

$\overline{x}^R_{i_0} > 0$ mais $\overline{x}^R_{i_0} = \overline{b}_{i_0} \notin \mathbb{Z}_+ \Rightarrow < \overline{b}_{i_0} > > 0$.

d'où : $\sum_{j \in \overline{J}} < \overline{a}_{i_0 j} >. \overline{x}^R_j = 0 < < \overline{b}_{i_0} >$.

d'où : $\sum_{j \in \overline{J}} < \overline{a}_{i_0 j} >. \overline{x}^R_j \not\geq < \overline{b}_{i_0} >$

c'est à dire : $\sum_{j \in \overline{J}} \alpha_{i_0 j} x_j \geq \alpha_{i_0}$ est non valide pour S_R .

$$[\text{ (a) et (b) }] \Longleftrightarrow \sum_{j \in \overline{J}} \alpha_{i_0 j} x_j \geq \alpha_{i_0}$$

est une coupe (c.q.f.d)

Remarque 1 : Relativement certaines transformations dans le programme linéaire relaxé (PR), à savoir son écriture sous-forme standard le théorème 1 précédent prend une forme plus intéressante.

2.3.3 Théorème 2

Si $\overline{b}_{i_0}$ est irrationnel ($\notin \mathbb{Z}^*_+$),alors l'égalité

$$\sum_{j \in \overline{J}} \alpha_{i_0 j} x_j = \alpha_{i_0} + x_{n+1}, \text{ avec } x_{n+1} \in \mathbb{R}^1_+$$

définit une coupe c'est à dire :

• égalité valide pour S^0_R où S^0 est le domaine réalisable de la forme standard du problème initiale.

$$S^0 = \{x_0 \in \mathbb{R}^1,\ x \in \mathbb{Z}^n_+ \ /\ x_0 - cx = 0,\ Ax = b\}.$$

• et non valide pour S^0_R où

$$S^0_R = \{x_0 \in \mathbb{R}^1,\ x \in \mathbb{R}^n_+ \ /\ x_0 - cx = 0,\ Ax = b\}.$$

2.3.4 Preuve 2

1. Tout programme linéaire en nombre entiers (P.L.N.E) :

$$(P.L.N.E) \begin{cases} MaxZ = cx \\ x \in S = \{x \in \mathbb{Z}^n_+ \ /\ Ax \leq b\} \end{cases}$$

pour se transformer en sa forme standard équivalente :

$$(P.L.N.E) = \begin{cases} MaxZ_0 = x_0 \\ \quad avec \\ (x_0, x) \in S^0 = \{x_0 \in \mathbb{Z}^1_+, x \in \mathbb{Z}^n_+ \ /\ x_0 - cx = 0,\ Ax = b\} \end{cases}$$

2. Toute inégalité : $\alpha x \leq \alpha_0$ valide pour S est équivalente à une égalité :

$$\alpha x + x_{n+1} = \alpha_0 \text{ valide pour } S^0 \text{ c'est à dire : définit une coupe.}$$

3. Si $\bar{b}_{i_0} \in \mathbb{Z}_+^*$, alors $\sum_{j \in \overline{J}} \alpha_{i_0 j} x_j \geq \alpha_{i_0}$ est une coupe est déjà démontrer en **Théorème 1** d'ou le résultat du **Théorème 2** (c.q.f.d)

2.3.5 Théorème 3 (λ- coupe)

Pour toute valeur du scalaire $\lambda \geq 0$ l'inégalité :

$$\sum_{j \in \overline{J}} (\lfloor \lambda \rfloor \overline{a}_{ij} - \lfloor \lambda . \overline{a}_{ij} \rfloor) . x_j \geq (\lfloor \lambda \rfloor . \bar{b}_{i_0} - \lfloor \lambda . \bar{b}_{i_0} \rfloor)$$

est : (•) valide pour S

(••) Une coupe pour certaines valeurs de λ en particulier pour $\lambda = 1$.

2.3.6 Preuve 3

• Dans la preuve du Théorème 1 on a établit l'égalité suivante :

$$x_{i_0} + \sum_{j \in \overline{J}} \overline{a}_{i_0 j} . x_j = \bar{b}_{i_0} \qquad (1)$$

• Multiplions les termes de l'égalité (1) par λ on obtient :

$$\lambda . x_{i_0} + \sum_{j \in \overline{J}} \lambda . \overline{a}_{i_0 j} . x_j = \lambda . \bar{b}_{i_0}$$

Mais $\begin{cases} \lambda = \lfloor \lambda \rfloor + < \lambda > \\ \quad et \\ \lambda . \overline{a}_{i_0 j} = \lfloor \lambda \overline{a}_{i_0 j} \rfloor + < \lambda . \overline{a}_{i_0 j} > \end{cases}$

d'où :

$$\lfloor \lambda \rfloor . x_{i_0} + \sum_{j \in J} \lfloor \lambda . \overline{a}_{i_0} \rfloor . x_j + < \lambda > x_{i_0} + \sum_{j \in J} < \lambda \overline{a}_{i_0 j} > x_j$$

$$= \lfloor \lambda . \bar{b}_{i_0} \rfloor + < \lambda . \bar{b}_{i_0} >$$

Comme

$$(< \lambda > . x_{i_0} + \sum_{j \in J} < \lambda . \overline{a}_{i_0} > x_j) \geq 0$$

On a :

$$\lfloor \lambda \rfloor . x_{i_0} + \sum_{j \in J} \lfloor \lambda . \overline{a}_{i_0 j} \rfloor . x_j \leq \lfloor \lambda . \bar{b}_{i_0} \rfloor + < \lambda . \bar{b}_{i_0} >$$

• Soit maintenant $x \in S = \{x \in \mathbb{Z}_+^n : \ Ax \leq b\}$ x se décompose selon la base optimale J en $x = (x_J, x_{\overline{J}}) \in \mathbb{Z}_+^n \Rightarrow x_J \in \mathbb{Z}_+^m, x_{\overline{J}} \in \mathbb{Z}_+^{n-m}$.

d'où :

$$\lfloor \lambda \rfloor . x_{i_0} + \sum_{j \in \overline{J}} \lfloor \lambda . \overline{a}_{i_0 j} \rfloor . x_j \text{ est un entier .}$$

et comme :

$$0 \leq < \lambda . \overline{b}_{i_0} > < 1.$$

On a :

$$\lfloor \lambda \rfloor . x_{i_0} + \sum_{j \in \overline{J}} \lfloor \lambda . \overline{a}_{i_0 j} \rfloor . x_j \leq \lfloor \lambda . \overline{b}_{i_0} \rfloor \qquad (3).$$

$a \leq b,\ a$ entier $\Rightarrow\ a \leq \lfloor b \rfloor$

$[\lfloor \lambda \rfloor .(1) - (3)]$ donne :

$$\sum_{j \in \overline{J}} (\lfloor \lambda \rfloor . \overline{a}_{i_0 j} - \lfloor \lambda . \overline{a}_{i_0 j} \rfloor) . x_j \ \geq \ (\lfloor \lambda \rfloor . \overline{b}_{i_0} - \lfloor \lambda . \overline{b}_{i_0} \rfloor). \qquad (2).$$

d'où : $\forall x, x \in S, \{x \in \mathbb{Z}_+^n : Ax \leq b\}$ on a : $x = (x_J, x_{\overline{J}})$
se décompose selon la base optimale J.

$$\sum_{j \in \overline{J}} (\lfloor \lambda \rfloor . \overline{a}_{i_0 j} - \lfloor \lambda . \overline{a}_{i_0 j} \rfloor) . x_j \ \geq \ (\lfloor \lambda \rfloor . \overline{b}_{i_0} - \lfloor \lambda . \overline{b}_{i_0} \rfloor). \qquad (2).$$

c'est à dire : l'inégalité est valide pour S .

• Si $\lambda = 1$ (2) devient :

$$\sum_{j \in \overline{J}} < \overline{a}_{i_0 j} > . x_j \geq < \overline{b}_{i_0} >$$

$$\sum_{i \in J} \alpha_{i_0 j} x_j \ \geq \alpha_{i_0}$$

qui est une coupe et ceci d'après le **théorème 1**
Remarque 2 :

Une solution $\overline{x}^R$ au problème relaxé (PR) est optimale pour le (P.L.N.E) ssi elle vérifie les trois conditions suivante :

1. $\overline{x}^R$ est primal réalisable c'est à dire : $\overline{b} = (A_J)^{-1} . b \ \geq \ 0$.
2. $\overline{x}^R$ est dual réalisable c'est à dire : $\overline{C}_{\overline{J}} \ \leq \ 0_{R^{n-m}}$.
3. $\overline{x}^R$ est entière c'est à dire : $\overline{x}^R = (\overline{x}_J^R, \overline{x}_{\overline{J}}^R) = (\overline{b}, 0) \in \mathbb{Z}_+^n$.

2.4 A- Algorithme de Gomory : [10]

• **initialisation :**

Poser $t = 1, Z_R^1(x) = x_0,\ S_R^1 = \{x_0 \in \mathbb{R},\ x \in \mathbb{R}_+^{n+1} : x_0 - cx = 0,\ Ax = b\}$.

Itération t

• **Etape 1 :** Résoudre $(PR^t) = \begin{cases} \text{Max} x_0 \\ (x_0, x) \in S_R^t \end{cases}$

• **Etape 2 :** Test d'intégrité

Si : $x_R^t \in \mathbb{Z}_+^n$, alors x_R^t est opt au P.L.N.E .

Sinon : **étape** (3).

• **Etape 3 :** Générer une coupe de Gomory

$$\sum_{j \in \overline{J}} \alpha_{i_0 j} x_j - x_{n+t} = \alpha_{i_0} \text{ avec } x_{n+t} \in \mathbb{Z}_+$$

Poser : $S_R^{t+1} = S_R^t \cap \{x_0 \in \mathbb{R},\ x \in \mathbb{R}_+^{n+t},\ \sum_{j \in \overline{J}} \alpha_{i_0 j} x_j - x_{n+t} = \alpha_{i_0}\}$.

$t \longleftarrow t + 1$ et retourner à **l'étape (1)**.

Remarque 3 : Quand on rajoute une coupe fractionnaire de Gomory la nouvelle base contenant la variable de base x_{n+t} est dual réalisable cependant non primal réalisable et c'est l'unique variable de base qui est non réalisable $x_{n+t} = -\alpha_{i_0}^t < 0$ d'où la nécessite d'utiliser l'algorithme dual du simplexe pour optimiser à nouveau.

2.4.1 Exemple 1

Résoudre le (P.L.N.E) suivant par l'algorithme dual fractionnaire de Gomory :

$$\begin{cases} \text{MaxZ=Cx} \\ Ax \leq b \\ x \in \mathbb{Z}_+^m \end{cases}$$

$$\text{où : } A = \begin{pmatrix} 2 & -1 \\ 2 & 5 \\ -1 & 2 \end{pmatrix} \quad b = \begin{pmatrix} 4 \\ 16 \\ 4 \end{pmatrix} \quad c = (4, 11) \quad x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

On pose :

$$S = \{x \in \mathbb{Z}_+^2 / 2x_1 - x_2 \leq 4, 2x_1 + 5x_2 \leq 16, -x_1 + 2x_2 \leq 4\}$$

Initialisation :

Poser $t = 1,\ Z_R^1(x) = x_0, S_R^1 = \{x_0 \in \mathbb{R}_+^n, x \in \mathbb{R}_+^n :\ x_0 - cx = 0,\ Ax = b\}$

$$\text{d'où : } S_R^1 = \{(x_0, x) \in \mathbb{R}_+^{n+4} tq :$$
$$x_0 - 4x_1 - 11x_2 = 0$$
$$2x_1 - x_2 + x_3 = 4$$
$$2x_1 + 5x_2 + x_4 = 16$$
$$-x_1 + 2x_2 + x_5 = 4 \ \}$$

Etape 1 : Résoudre $(PR^{(1)}) \equiv \begin{cases} \text{Max } x_0 \\ (x_0, x) \in S_R^1 \end{cases}$

$$\Leftrightarrow \begin{cases} \quad \text{Max } x_0 \\ x_0 - 4x_1 - 11x_2 = 0 \\ 2x_1 - x_2 + x_3 = 4 \\ 2x_1 + 5x_2 + x_4 = 16 \\ -x_1 + 2x_2 + x_5 = 4 \\ \quad (x_0, x) \in \mathbb{R}_+^{4+1} \end{cases}$$

On applique l'algorithme du simplexe au $(PR^{(1)})$ et on prend comme base de départ $J^{(0)} = \{0, 3, 4, 5\}$ c'est à dire (x_0, x_3, x_4, x_5) en effet :

- $A_{J^{(0)}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ est la matrice identité .
- $A_{J^{(0)}} = (A_{J^{(0)}})^{-1}.b = b \geq 0$ est réalisable.

d'où le premier tableau simplexe correspondant est donné par :

C_j		1	0	0	0	0	0	$X_{(0)}$
C_B	X_B	x_0	x_1	$\downarrow x_2$	x_3	x_4	x_5	
1	x_0	1	-4	-11	0	0	0	0
0	x_3	0	2	-1	1	0	0	4
0	x_4	0	2	5	0	1	0	16
0	$\leftarrow x_5$	0	-1	**(2)**	0	0	1	4
Z_j		1	-4	-11	0	0	0	$Z_0 = 0$
Cj-Zj		0	4	11	0	0	0	

La base $J^{(0)} = (x_0, x_3, x_4, x_5)$ est primal réalisable cependant elle n'est pas duale réalisable en effet $\overline{c}_j \not\leq 0$.

Soit $k = Arg(Max\{\overline{C}_j > 0\}) = Arg(Max\{4, 11\}) = 2$. la variable qui sort la base est x_5 et qui rentrer est x_2.

On détermine $I_K = I_2 = \{i \ / \ a_{ik} = a_{i2} > 0\}$.

$$I_2 = \{4, 5\}.$$

d'où : $\theta_4 = \frac{16}{5}$ et $\theta_5 = \frac{4}{2} = 2$.

Soit $r = Arg(Min_{i \in I_{k=2}}\{\theta_i\}) = Arg(Min\{3 + \frac{1}{5}, 2\}) = 5$.

d'où la variable candidate à quitter la base est $r = 5$ et le pivot est donc : $a_{52} = 2$.

C_j		1	0	0	0	0	0	$X_{(1)}$
C_B	X_B	x_0	$\downarrow x_1$	x_2	x_3	x_4	x_5	
1	x_0	1	$\frac{-19}{2}$	0	0	0	$\frac{11}{2}$	22
0	x_3	0	$\frac{3}{2}$	0	1	0	$\frac{1}{2}$	6
0	$\leftarrow x_4$	0	$\left(\frac{\mathbf{9}}{\mathbf{2}}\right)$	0	0	1	$\frac{1}{2}$	6
0	x_2	0	$\frac{-1}{2}$	1	0	0	$\frac{1}{2}$	2
Z_j		1	$\frac{-19}{2}$	0	0	0	$\frac{11}{2}$	$Z_1 =$ 22
Cj-Zj		0	$\frac{19}{2}$	0	0	0	$\frac{-11}{2}$	

La base $J^{(1)} = (x_0, x_3, x_4, x_2)$ est :

• Primal réalisable en effet $\overline{b} = \begin{pmatrix} 22 \\ 6 \\ 6 \\ 2 \end{pmatrix} \geq 0$.

• Mais Dual non réalisable car : $\overline{C} \leq 0$. elle est donc non optimale.

$$k = Arg(Max\{\overline{C}_j = c_j - z_j / \ c_j - z_j > 0\} = 1).$$

$$I_1 = \{i \ / \ a_{i1} > 0\} = \{3, 4\}.$$

$$r = Arg\ (Min\ \{\theta_i / i \in I_1\}) = Arg\ (Min\ \{\tfrac{6}{\frac{3}{2}}, \tfrac{6}{\frac{9}{2}}\}) = 4.$$

On effectuer alors une itération avec l'algorithme primal du simplexe .

C_j		1	0	0	0	0	0	$X_{(2)}$
C_B	X_B	x_0	x_1	x_2	x_3	x_4	x_5	
1	x_0	1	0	0	0	$\frac{19}{9}$	$\frac{2}{9}$	$\frac{104}{3}$
0	x_3	0	0	0	1	$\frac{-1}{3}$	$\frac{4}{3}$	4
0	x_1	0	1	0	0	$\frac{2}{9}$	$\frac{-5}{9}$	$\frac{4}{3}$
0	x_2	0	0	1	0	$\frac{1}{9}$	$\frac{2}{9}$	$\frac{8}{3}$
Z_j		1	0	0	0	$\frac{19}{9}$	$\frac{2}{9}$	$Z_2 = \frac{104}{3}$
Cj-Zj		0	0	0	0	$\frac{-19}{9}$	$\frac{-2}{9}$	

r=2 correspond à la plus grande partie fractionnaire de la solution non entière .

La base $J^{(2)} = (x_0, x_3, x_1, x_2)$ est :

$$\begin{cases} \text{Primal réalisable.} \\ \text{Dual réalisable.} \end{cases} \Leftrightarrow J^{(2)} \text{ est optimale .}$$

Etape 2 : Cependant elle n'est pas entière en effet :

$$\overline{x}_R^{J^{(2)}} = (\underbrace{34+\frac{2}{3}}_{x_0}, \underbrace{4}_{x_3}, \underbrace{1+\frac{1}{3}}_{x_1}, \underbrace{2+\frac{2}{3}}_{x_2}) \quad \text{n'est pas entière.}$$

Etape 3 : Générer une première coupe de Gomory :

$$\sum_{j \in \overline{J}^{(2)}} \alpha_{i_0 j} x_j - x_{n+m+1} = \alpha_{i_0}$$

la plus grande partie fractionnaire correspond à la variable de base no entière r=2 c'est à dire x_2 est la coupe de Gomory correspondant à r=2 est donnée pour :

$r = 2 \qquad < 1 > .x_2 + < \frac{1}{9} > .x_4 + < \frac{2}{9} > .x_5 \geq < \frac{8}{3} > .$

$$1^{er} \text{ coupe de Gomory} \quad ... \quad \tfrac{1}{9}x_4 + \tfrac{2}{9}.x_5 \geq \tfrac{2}{3}.$$

La coupe de Gomory est équivalente à :

$$\tfrac{1}{9}x_4 + \tfrac{2}{9}.x_5 - x_6 = \tfrac{2}{3} \quad \text{avec } x_6 \geq 0.$$

$$x_6 - \tfrac{1}{9}x_4 - \tfrac{2}{9}.x_5 = -\tfrac{2}{3}$$

On rajoute cette contrainte au tableau optimale précédent c'est à dire $J^{(2)}$ on obtient le tableau suivant :

C_j		1	0	0	0	0	0	0	$X^{(3)}$
$C_{J^{(3)}}$	$X_{J^{(3)}}$	x_0	x_1	x_2	x_3	x_4	x_5	x_6	
1	x_0	1	0	0	0	$\frac{19}{9}$	$\frac{2}{9}$	0	$\frac{104}{3}$
0	x_3	0	0	0	1	$\frac{-1}{3}$	$\frac{4}{3}$	0	4
0	x_1	0	1	0	0	$\frac{2}{9}$	$\frac{-5}{9}$	0	$\frac{4}{3}$
0	x_2	0	0	1	0	$\frac{1}{9}$	$\frac{2}{9}$	0	$\frac{8}{3}$
0	x_6	0	0	0	0	$\frac{-1}{9}$	$\frac{-2}{9}$	1	$\frac{-2}{3}$
Z_j		1	0	0	0	$\frac{19}{9}$	$\frac{2}{9}$	0	$Z_3 = \frac{104}{3}$
Cj-Zj		0	0	0	0	$\frac{-19}{9}$	$\frac{-2}{9}$	0	

La base $J^{(3)} = (x_0, x_3, x_1, x_2, x_6)$ est :
• Dual réalisable $\overline{C}_j < 0 \quad j \in h.b.$
• Mais non primal réalisable car $\overline{x}_6^{J(3)} = \frac{-2}{3} < 0.$
On applique l'algorithme dual du simplexe et on obtient :

C_j		1	0	0	0	0	0	0	$X^{(4)}$
$C_{J^{(4)}}$	$X_{J^{(4)}}$	x_0	x_1	x_2	x_3	x_4	x_5	x_6	
1	x_0	1	0	0	0	2	0	1	$\frac{102}{3} = 34$
0	x_3	0	0	0	1	-1	0	6	0
0	x_1	0	1	0	0	$\frac{1}{2}$	0	$-\frac{5}{2}$	3
0	x_2	0	0	1	0	0	0	1	2
0	x_5	0	0	0	0	$\frac{1}{2}$	1	$-\frac{9}{2}$	3
Z_j		1	0	0	0	2	0	1	$Z_4 =$
Cj-Zj		0	0	0	0	-2	0	-1	$\frac{102}{3} = 34$

La base $J^{(4)} = (x_0, x_3, x_1, x_2, x_5)$ est :
• Primal réalisable $\quad x_R^{J(4)} = \overline{b} = \begin{pmatrix} 34 \\ 0 \\ 3 \\ 2 \\ 3 \end{pmatrix} \geq 0.$

• Duale réalisable $\quad \overline{C}_j \leq 0 \quad j \in \text{H.b.}$

• est puisque la solution $x_R^{J(4)} = (34, 0, 3, 2, 3)$ est entière, elle est aussi optimal au (P.L.N.E) initiale.

2.4.2 Interprétation géométrique de la 1^{er} coupe de Gomory :

La première coupe de Gomory est générer par la $2^{éme}$ contrainte du tableau optimale du simplexe correspondent à la base optimale $J^{(2)} = (x_0, x_3, x_1, x_2)$ et s'exprime par :

$$\frac{1}{9}x_4 + \frac{2}{9}x_5 \geq \frac{2}{3} \qquad (1)$$

Mais x_4 et x_5 sont des variables de bases dans le premier tableau simplexe, elles s'expriment en fonction des variables hors bases de la manière suivante :

$$\begin{cases} 2x_1 + 5x_2 + x_4 = 16 & \text{pour } (x_4) \\ -x_1 + 2x_2 + x_5 = 4 & \text{pour } (x_5) \end{cases}$$

$$\begin{cases} x_4 = 16 - 2x_1 - 5x_2 \\ x_5 = 4 + x_1 - 2x_2 \end{cases} \iff \begin{cases} \frac{1}{9}x_4 = \frac{16}{9} - \frac{2}{9}x_2 - \frac{5}{9}x_2 \quad ...(a). \\ \frac{2}{9}x_5 = \frac{8}{9} + \frac{2}{9}x_1 - \frac{4}{9}x_2 \quad ...(b). \end{cases}$$

En sommant (a) et (b) on obtient :

$$\frac{2}{3} \leq \frac{1}{9}x_4 + \frac{2}{9}x_5 = \frac{8}{3} - x_2.$$

$$x_2 \leq 2.$$

c'est à dire : la 1^{er} coupe de Gomory $\frac{1}{9}x_4 + \frac{2}{9}x_5 \geq \frac{2}{3}$ est équivalente au demi-espace fermé de $\mathbb{R}^2$ définit par l'inégalité $x_2 \leq 2$ comme le montre la figure suivante :

$$\begin{cases} Max\ Z = 4x_1 + 11x_2 \\ (c_1) \quad 2x_1 - x_2 \leq 4 \\ (c_2) \quad 2x_1 + 5x_2 \leq 16 \\ (c_3) \quad -x_1 + 2x_2 \leq 4 \end{cases} \iff \begin{cases} (c_1) : \ x_2 \geq 2x_2 - 4 \\ (c_2) : \ x_2 \leq \frac{-2}{5}x_1 + \frac{16}{5} \\ (c_3) : \ x_2 \leq \frac{1}{2}x_1 + 2 \end{cases}$$

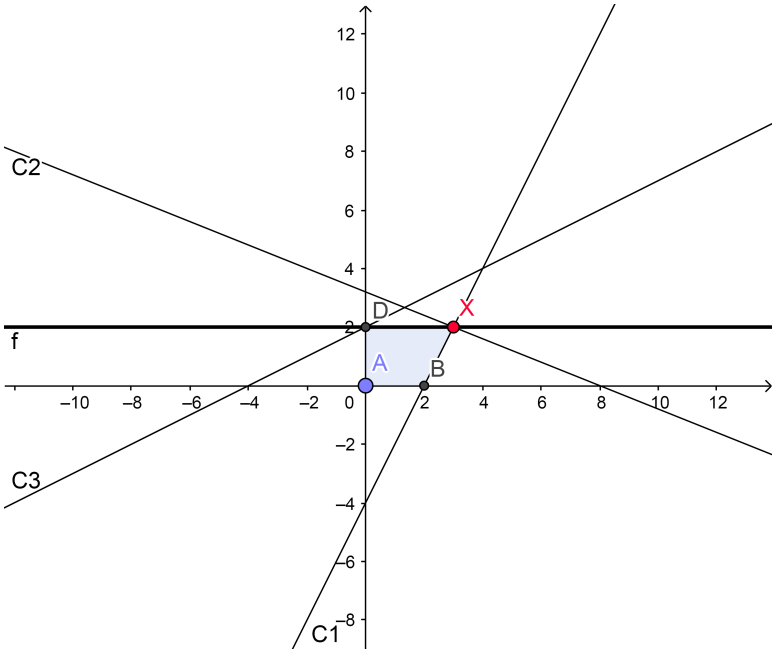


FIGURE 2.1 – l'interprétation géométrique de la 1^{er} coupe de Gomory

x=(3,1) et z(x)= 34.

Remarque sur l'algorithme dual fractionnaire de Gomory [14] :

1. on peut montrer que si :
   - on choisit toujours l'indice i_0 le plus faible pour lequel $\alpha_{i_0} =< \overline{b}_{i_0} >$.
   - on utilise la variante lexicographique de l'algorithme dual du simplexe.

• on supprime les contraintes correspondant à une coupe si la variable d'écart entre à nouveau dans la base alors : l'algorithme dual fractionnaire de Gomory est fini, voir [17].

2. on peut améliorer l'efficacité des algorithmes dual fractionnaire eu jouant sur la valeur (α, α_0) pour obtenir des coupes "profondes" qui à chaque itération permettent de supprimer un plus grand volume de $S_R/S^* o\grave{u} S^* = ENV(S)$.
3. Même améliorée par des heuristiques de recherche de coupes profondes, la méthode dual fractionnaire de Gomory présente un inconvénient majeur, en effet :
   • Les données d'une (P.L.N.E) sont entiers A,B et C .
   • Les résultats qu'on cherche $\overline{x} = \overline{b} = (A_J)^{-1}b$ sont également des entiers. Cependant, on travaille sur des réels S_R domaine relaxé ce qui ralenti considérablement les calculs et entraine la propagation des erreurs d'arrondi, ce qui rend incertain les testes d'intégrités sur les variables.
4. Le nombre imprévisible d'itération des méthodes de coupes fait que celles-ci ne sont utilisées qu'intégrées avec des procédures par séparation et évaluation.
5. puisque le nombre de coupes à engendré dans la résolution d'un (P.L.N.E) est imprévisible on arête de générer des coupes dès que ces dernières ne procurent pas de progrès notables.
6. Si la méthode dual fractionnaire permet de resserrer l'évaluation de la borne du (P.L.N.E),elle ne donne de solution réalisable entière qu'à l'optimum.
7. Un cas particulier qui peut conduire au blocage de l'algorithme dual fractionnaire est le suivant :
   • Si pour un (P.L.N.E) donnée A,b et C entiers données le nombre imprévisible de coupes générées ne procurent pas de progrès dans l'évaluation de la borne $\omega = \pi.b$ au (P.L.N.E) on est forcé de s'arrêter de générer des coupes sans autant atteindre au moins une solution réalisable entière au (P.L.N.E), et pour contourner cette difficulté pratique on utilise une méthode dite : **"La méthode primal totalement en nombres entiers"** .

2.5 B- La méthode primal totalement en nombre entiers [15] :

Cette classe d'algorithme maintient la 1^{er} et la 3^{eme} conditions c'est à dire :(la realisabilété primal de $\overline{x}^R$) et (l'intégrité de $\overline{x}_R$) puis tente d'améliorer la 2^{eme} condition à savoir la realisabilité dual de $\overline{x}^R$.
Son principe est le suivant :
• supposons qu'on commence avec un tableau simplexe totalement entiers .
• Pour un problème de maximisation la coupe est engendrée par toute variable hors-base, ayant un coût réduit positif disons x_R.

Soit

$$\frac{\overline{b}_r}{\overline{a}_{rk}} = \min_{1\leq i\leq m}\{\frac{\overline{b}_i}{\overline{a}_{ik}},\ a_{ik} > 0\}\ .$$

Puisque les données sont entières alors $\overline{a}_{ik} > 0 \Rightarrow \overline{a}_{ik} \geq 1$.
Deux cas peuvent se présenter :
1^{er} ***cas :*** $\overline{a}_{rk} = 1$ le pivot est égale à 1, et le tableau simplexe après pivotage est totalement en nombre entiers.
2^{eme} ***cas :*** $\overline{a}_{rk} > 1$, alors l'égalité suivante :

$$\sum_{j\in\overline{J}} \lfloor\frac{\overline{a}_{rj}}{\overline{a}_{rk}}\rfloor x_j + x_{i+1} = \lfloor\frac{\overline{b}_r}{\overline{a}_{rk}}\rfloor,\ x_{k+1} \in \mathbb{R}_+.$$

définit une coupe on rajoutant cette coupe au tableau simplexe initiale on obtient un tableau simplexe primal totalement en nombres entiers (car les coefficients de la coupe sont tous entiers)
Remarque :
En rajoutant la coupe précédente au tableau initial, on obtient un pivot égal à 1.

2.5.1 L'algorithme primal totalement en nombre entiers :

Etape 1 : commencer avec un tableau primal totalement en nombre entiers.

Etape 2 :
Si : la solution courante est optimale $\overline{C}_N \leq 0$ **stop**.
Sinon : aller à **l'étape (3)**.

Etape 3 : choisir une variable hors-base x_k telle que : $(\overline{C}_k > 0)$ et déterminer la variable de base r candidate à quitter la base, en évaluant :

$$\frac{\overline{b}_r}{\overline{a}_{rk}} = \min_{1\leq i\leq m}\{\frac{\overline{b}_i}{\overline{a}_{ik}},\ \overline{a}_{ik} > 0\}\ .$$

Etape 4 : Faire le test d'intégrité
Si : $\overline{a}_{rk} = 1$ effectuer un pivotage standard du simplexe et retourner à **l'étape (2)**.
Sinon : $\overline{a}_{rk} > 1$, rajouter la coupe

$$\sum_{j\in\overline{J}} \lfloor\frac{\overline{a}_{rj}}{\overline{a}_{rk}}\rfloor x_j + x_{n+1} = \lfloor\frac{\overline{b}_r}{\overline{a}_{rk}}\rfloor, \overline{x}_{n+1} \in \mathbb{Z}_+$$

et retourner en **l'étape (2)**.

Remarque :

1. Cet algorithme génère au plus une coupe par itération .
2. En effectuant un choix approprié du pivot $\overline{a}_{rk}$ qui génère la coupe :

$$\sum_{j\in\overline{J}} \lfloor\frac{\overline{a}_{rj}}{\overline{a}_{rk}}\rfloor x_j + x_{n+1} = \lfloor\frac{\overline{b}_r}{\overline{a}_{rk}}\rfloor$$

On peut obtenir un algorithme qui converge en nombre fini d'itérations, cependant pour les même raisons cités dans le paragraphe précédent **méthode dual fractionnaire**, les algorithmes de coupes totalement en nombres entiers semble être aussi lent que les algorithmes de coupes dual fractionnaires, on les utilisent pour débloquer les algorithmes dual fractionnaire seulement.

2.5.2 Théorème 4

Si : $\overline{a}_{rk} > 1$, alors l'égalité suivante :

$$\sum_{j\in\overline{J}} \lfloor \frac{\overline{a}_{rj}}{\overline{a}_{rk}} \rfloor x_j + x_{n+1} = \lfloor \frac{\overline{b}_r}{\overline{a}_{rk}} \rfloor,\ x_{n+1} \in \mathbb{R}_+.$$

définit une coupe $\begin{cases} \text{valide pour S.} \\ \text{non valide pour} S_R. \end{cases}$

2.5.3 Preuve 4

Supposons que on a un (P.L.N.E) définit par :

$$\text{(P.L.N.E)} \begin{cases} \text{Max} Z = Cx \\ x \in S = \{x \in \mathbb{Z}_+^n, Ax \leq b\} \end{cases}$$

Le problème relaxé associé s'écrit :

$$\text{(PR)} \begin{cases} \text{Max} Z = Cx \\ x \in S_R = \{x \in \mathbb{R}_+^n, Ax \leq b\} \end{cases}$$

Soit J une base optimale du problème relaxé (PR) alors la forme standard par apport à la base J s'exprime par :

$$(PR)_J \Longleftrightarrow \begin{cases} \text{Max } Z_R = C_J\overline{b} + \sum_{j\in\overline{J}} \overline{C}_j x_j \\ x_{i=B(j)} + \sum_{j\in\overline{J}} \overline{a}_{ij} x_j = \overline{b}_i \\ x_i \geq 0, x_j \geq 0, i = 1, ..., m\ j = m+1, ..., n \end{cases}$$

Soit r l'indice de la variable de base pour laquelle le coefficient $\overline{a}_{rk} > 1$, la contrainte correspondante dans $(PR)_J$ s'exprime comme suit :

$$x_r + \overline{a}_{rk}.x_k + \sum_{j\in\overline{J}\setminus\{k\}} a_{rj}x_j = \overline{b}_r$$

$$\Leftrightarrow \overline{a}_{rk}x_k + \sum_{j\in\overline{J}\setminus\{k\}} \overline{a}_{rj}x_j \leq \overline{b}_r \qquad (1)$$

Poser $\lambda = \frac{1}{\overline{a}_{rk}}$ et multiplions (1) par λ.

$$x_k + \sum_{j \in \overline{J} \setminus \{k\}} \frac{\overline{a}_{rj}}{\overline{a}_{rk}} x_j \leq \frac{\overline{b}_r}{\overline{a}_{rk}}$$

$$\Rightarrow x_k + \sum_{j \in \overline{J} \setminus \{k\}} \lfloor \frac{\overline{a}_{rj}}{\overline{a}_{rk}} \rfloor x_j \leq \frac{\overline{b}_r}{\overline{a}_{rk}} \qquad (2)$$

1. Si $\overline{x}^R \in S \Rightarrow \overline{x}^R_k$ et $\overline{x}^R_j$ sont entiers et le membre gauche de (2) est entier.

   $$a \leq b, \text{ a entier} \Rightarrow \text{a} \leq \lfloor \text{ b } \rfloor.$$

   d'où :

   $$x_k + \sum_{j \in \overline{J} \setminus \{k\}} \lfloor \frac{\overline{a}_{rj}}{\overline{a}_{rk}} \rfloor x_j \leq \lfloor \frac{\overline{b}_r}{\overline{a}_{rk}} \rfloor$$

   En rajoutant une variable d'écart entière $x_{k+1} \in \mathbb{R}_+$, on obtient :

   $$\overline{x}^R_k + \sum_{j \in \overline{J} \setminus \{k\}} \lfloor \frac{\overline{a}_{rj}}{\overline{a}_{rk}} \rfloor \overline{x}^R_j + x_{n+1} = \lfloor \frac{\overline{b}_r}{\overline{a}_{rk}} \rfloor$$

   Ce qui preuve que :

   $$x_k + \sum_{j \in \overline{J} \setminus \{k\}} \lfloor \frac{\overline{a}_{rj}}{\overline{a}_{rk}} \rfloor x_j + x_{n+1} = \lfloor \frac{\overline{b}_r}{\overline{a}_{rk}} \rfloor$$

   est une égalité valide pour S.
2. Si $\overline{x}^R \in (S_R \setminus S)$ montrons que ?

   $$\sum_{j \in \overline{J} \setminus \{k\}} \lfloor \frac{\overline{a}_{rj}}{\overline{a}_{rk}} \rfloor \overline{x}^R_j + x_{n+1} \neq \lfloor \frac{\overline{b}_r}{\overline{a}_{rk}} \rfloor$$

   où $x_{n+1} \in \mathbb{R}_+$?

d'après l'inégalité (2) du **théorème 4** on a :

$$x_k + \sum_{j \in \overline{J} \setminus \{k\}} \lfloor \frac{\overline{a}_{rj}}{\overline{a}_{rk}} \rfloor x_j + x_{n+1} = \frac{\overline{b}_r}{\overline{a}_{rk}} \quad \text{avec } x_{n+1} \in \mathbb{R}_+. \qquad (3)$$

où r est l'indice de la variable de base pour laquelle le coefficient $\overline{a}_{rk} > 1$.

2. Si $\overline{x}^R \in (S_R \setminus S)$, le premier membre de (3) n'est pas entiers.

on remarque que si r n'est pas entier alors :

$$r = \lfloor r \rfloor + < r > \Rightarrow r > \lfloor r \rfloor$$

d'où :

$$\overline{x}_k^R + \sum_{j\in\overline{J}\setminus\{k\}} \lfloor\frac{\overline{a}_{rj}}{\overline{a}_{rk}}\rfloor \overline{x}_j^R + \overline{x}_{n+1}^R > \lfloor\frac{\overline{b}_r}{\overline{a}_{rk}}\rfloor$$

$$\Longleftrightarrow \quad \sum_{j\in\overline{J}} \lfloor\frac{\overline{a}_{rj}}{\overline{a}_{rk}}\rfloor \overline{x}_j^R + \overline{x}_{n+1}^R > \lfloor\frac{\overline{b}_r}{\overline{a}_{rk}}\rfloor$$

$$\Longleftrightarrow \quad \sum_{j\in\overline{J}} \lfloor\frac{\overline{a}_{rj}}{\overline{a}_{rk}}\rfloor \overline{x}_j^R + \overline{x}_{n+1}^R \neq \lfloor\frac{\overline{b}_r}{\overline{a}_{rk}}\rfloor, \quad x_{n+1}^R \in \mathbb{R}_+.$$

c'est à dire :

$$\sum_{j\in\overline{J}} \lfloor\frac{\overline{a}_{rj}}{\overline{a}_{rk}}\rfloor x_j + x_{n+1} = \lfloor\frac{\overline{b}_r}{\overline{a}_{rk}}\rfloor, \quad x_{n+1} \in \mathbb{R}_+.$$

est non valide pour S_R

Remarque 6 :
La coupe définit par la théorème 4 élimine uniquement les solutions non entières.

2.5.4 Exemple 2

Résoudre le (P.L.N.E) suivant avec l'algorithme primal totalement en nombre entiers :

$$\begin{cases} MaxZ = Cx \\ Ax \leq b \\ x \in \mathbb{N}^n \end{cases} \qquad \text{avec :} \quad \text{C=(1,4)}, \ \begin{bmatrix} 5 & 7 \\ -1 & 3 \end{bmatrix}, \quad \text{b=}\begin{bmatrix} 21 \\ 8 \end{bmatrix},$$

c'est à dire :

$$\text{(P.L.N.E)} \begin{cases} \text{Max } Z = x_1 + 4x_2 \\ 5x_1 + 7x_2 \leq 21 \\ -x_1 + 3x_2 \leq 8 \\ x_1 \in \mathbb{N}, x_2 \in \mathbb{N} \end{cases}$$

Etape 01 : Commençons avec un tableau simplexe primal totalement en nombre entiers
mettons le (PR) sous-forme standard.

$$\text{(P.R)} \begin{cases} \text{Max}Z = x_1 + 4x_2 \\ 5x_1 + 7x_2 + x_3 = 21 \\ -x_1 + 3x_2 + x_4 = 8 \\ x_j \geq 0, \quad j = 1, ..., 4 \end{cases}$$

C_j		1	4	0	0	$X^{(0)}$
C_B	X_B	x_1	$\downarrow x_2$	x_3	x_4	
0	x_3	5	7	1	0	21
0	$\leftarrow x_4$	-1	**(3)**	0	1	8
Z_j		0	0	0	0	$Z_0 = 0$
Cj-Zj		1	4	0	0	

Etape 02 : Le tableau courant est non optimal en effet il est non dual réalisable c'est à dire $\overline{C}_j \not\leq 0$.

Etape 03 : $k = Arg(Max\{\overline{C}_j,\ \overline{C}_j > 0\})$

$= \text{Arg (Max \{ 1,4 \})} = 2.$

déterminer la variable candidate à quitter la base r ? tel que :

$$\frac{\overline{b}_r}{\overline{a}_{r2}} = Min\{\frac{\overline{b}_i}{\overline{a}_{ik}},\ \overline{a}_{ik} > 0\}$$

c'est à dire :

$$r = Arg(Min\{\frac{\overline{b}_i}{\overline{a}_{i2}},\ \overline{a}_{i2} > 0\})$$

$$r = Arg(Min\{\underbrace{\frac{21}{7}}_{x_3}, \underbrace{\frac{8}{3}}_{x_4}\}) = 4$$

$$r = Arg(Min\{3, \tfrac{8}{3}\}) = 4.$$

k=2, r=4 le pivot est donc :$\overline{a}_{42} = 3$

Etape 04 : Faire le test d'intégrité unitaire sur le pivot.

comme $\overline{a} = 3 > 1$ générer une première coupe par l'équation :

$$\sum_{j\in\overline{J}} \lfloor\frac{\overline{a}_{4j}}{\overline{a}_{42}}\rfloor x_j + x_{n+1} = \lfloor\frac{\overline{b}_4}{\overline{a}_{42}}\rfloor, \quad x_{n+1} \in \mathbb{R}_+.$$

$$\lfloor\tfrac{-1}{3}\rfloor x_1 + \lfloor\tfrac{3}{3}\rfloor x_2 + \lfloor\tfrac{0}{3}\rfloor x_3 + \lfloor\tfrac{1}{3}\rfloor x_4 + x_5 = \lfloor\tfrac{8}{3}\rfloor$$

$$-x_1 + x_2 + 0.x_3 + 0.x_4 + x_5 = 2$$

d'où la 1^{er} coupe entière :

$$-x_1 + x_2 + x_5 = 2$$

Rajouter cette contrainte au tableau primal précédent puis retourner à **l'étape (2)**.

C_j		1	4	0	0	0	$X^{(1)}$
C_B	X_B	x_1	$\downarrow x_2$	x_3	x_4	x_5	
0	x_3	5	7	1	0	0	21
0	x_4	-1	3	0	1	0	8
0	$\leftarrow x_5$	-1	**(1)**	0	0	1	2
Z_j		0	0	0	0	0	$Z_1 = 0$
Cj-Zj		1	4	0	0	0	

Etape 02 : Le tableau est non optimal en effet il n'est pas dual réalisable $\overline{C} \not\leq 0$

Etape 03 :

$$k = Arg(Max\{\overline{C}_j,\ \overline{C}_j > 0\}) = 2.$$

$$r = Arg(Min\{\tfrac{\bar{b}_i}{\overline{a}_{ir}},\ \overline{a}_{ir} > 0\}) = 5.$$

d'où le pivot est $a_{52} = 1$.

Etape 04 : Test d'intégrité unitaire du pivot est affirmative
on effectuer un pivotage standard avec l'algorithme du simplexe .

C_j		1	4	0	0	0	$X^{(2)}$
C_B	X_B	$\downarrow x_1$	x_2	x_3	x_4	x_5	
0	$\leftarrow x_3$	**(12)**	0	1	0	-7	7
0	x_4	2	0	0	1	-3	2
4	x_2	-1	1	0	0	1	2
Z_j		-4	4	0	0	4	$Z_2 = 8$
Cj-Zj		5	0	0	0	-4	

Etape 02 : Le tableau courant n'est pas optimal car il n'est pas dual réalisable.

Etape 03 : Soit $k = Arg(\max_{j\in\overline{J}}\{\overline{C}_j/\overline{C}_j > 0\}) = 1.$

$$I_1 = \{i/a_{i1} > 0\} = \{\underbrace{3}_{x_3}, \underbrace{4}_{x_4}\}$$

Soit $r = Arg(Min\{\frac{\bar{b}_i}{\overline{a}_{i1}}/\overline{a}_{i1} > 0\}) = Arg(Min\{\underbrace{\frac{7}{12}}_{x_3}, \underbrace{\frac{2}{2}}_{x_4}\}) = 3.$

Etape 04 : Le pivot est donc $a_{31} = 12 > 1$. on génère donc une coupe entière :

$\lfloor\frac{12}{12}\rfloor x_1 + \lfloor\frac{0}{12}\rfloor x_2 + \lfloor\frac{1}{12}\rfloor x_3 + \lfloor\frac{0}{12}\rfloor x_4 + \lfloor\frac{-7}{12}\rfloor x_5 + x_6 = \lfloor\frac{7}{12}\rfloor$

$x_1 * 0 + x_2 * 0 + x_3 * 0 + x_4 - x_5 + x_6 = 0.$

$x_1 - x_5 + x_6 = 0$ est la seconde coupe entière .
on rajoute cette coupe au tableau primal précédent et on retourne à **l'étape (2)**.

C_j		1	0	0	0	0	0	$X^{(3)}$
C_B	X_B	$\downarrow x_1$	x_2	x_3	x_4	x_5	x_6	
0	x_3	(12)	0	1	0	-7	0	7
0	x_4	2	0	0	1	-3	0	2
4	x_2	-1	1	0	0	1	0	2
0	$\leftarrow x_6$	**(1)**	0	0	0	-1	1	0
Z_j		-4	4	0	0	4	0	$Z_3 = 8$
Cj-Zj		5	0	0	0	-4	0	

Etape 02 : le tableau courant n'est pas optimal, il n'est pas dual réalisable ($\overline{C}_j \not\leq 0$).

Etape 03 : déterminer le pivot ?

$k = Arg(Max\{\overline{C}_j / \overline{C}_j > 0\}) = 1.$

$I_1 = \{i/\overline{a}_{i1} > 0\} = \{\underbrace{3}_{x_3}, \underbrace{4}_{x_4}, \underbrace{6}_{x_6}\}.$

Soit

$$r = Arg(Min\{\tfrac{\overline{b}_3}{\overline{a}_{31}}, \tfrac{\overline{b}_4}{\overline{a}_{41}}, \tfrac{\overline{b}_6}{\overline{a}_{61}}\})$$
$$= Arg(Min\{\tfrac{7}{12}, \tfrac{2}{2}, \tfrac{0}{1}\}) = 6.$$

Etape 04 : Le pivot est donc $a_{61} = 1$, on effectuer un pivotage standard avec l'algorithme du simplexe et on retourne à **l'étape (2)**.

C_j		1	4	0	0	0	0	$X^{(4)}$
C_B	X_B	x_1	x_2	x_3	x_4	x_5	x_6	
0	x_3	0	0	1	0	5	-12	7
0	x_4	0	0	0	1	-1	-2	2
4	x_2	0	1	0	0	0	1	2
1	x_1	1	0	0	0	-1	1	0
Z_j		1	4	0	0	-1	5	$Z_4 = 8$
Cj-Zj		0	0	0	0	1	-5	

Etape 02 : Le tableau courant n'est pas optimal car il n'est pas dual réalisable ($\overline{C} \not\leq 0$).

Etape 03 : Soit = t $k = Arg(\max_{j \in \overline{J}}\{\overline{C}_j, \overline{C}_j > 0\}) = 5$.

$I_5 = \{i/\overline{a}_{i5} > 0\} = \{3\} = 3$.

Le pivot est donc $a_{rk} = a_{35} = 5$.

Etape 04 : comme le pivot $a_{35} = 5 > 1$ on génère une coupe entière :

$$\lfloor \tfrac{0}{5} \rfloor .x_1 + \lfloor \tfrac{0}{5} \rfloor .x_2 + \lfloor \tfrac{1}{5} \rfloor .x_3 + \lfloor \tfrac{0}{5} \rfloor .x_4 + \lfloor \tfrac{5}{5} \rfloor .x_5 + \lfloor \tfrac{-12}{5} \rfloor .x_6 + x_7 = \lfloor \tfrac{7}{5} \rfloor$$

$$\Longleftrightarrow 0.x_1 + 0.x_2 + 0.x_3 + 0.x_4 + x_5 - 3x_6 + x_7 = 1.$$

$\Longleftrightarrow x_5 - 3x_6 + x_7 = 1$ est la 3^{eme} coupe entière.

On rajoute cette coupe au tableau primal réalisable précédent et on ré-optimise a nouveau.

C_j		1	4	0	0	0	0	0	$X^{(5)}$
C_B	X_B	x_1	x_2	x_3	x_4	$\downarrow x_5$	x_6	x_7	
0	x_3	0	0	1	0	5	-12	0	7
0	x_4	0	0	0	1	-1	-2	0	2
4	x_2	0	1	0	0	0	1	0	2
1	x_1	1	0	0	0	-1	1	0	0
0	$\leftarrow x_7$	0	0	0	0	**(1)**	-3	1	1
Z_j		1	4	0	0	-1	5	0	$Z_5 = 8$
Cj-Zj		0	0	0	0	1	-5	0	

Etape 02 : Le tableau courant n'est pas optimal car il n'est pas dual réalisable.

Etape 03 : déterminer le pivot ?

$k = Arg(Max\{\overline{C}_j, \overline{C}_j > 0\}) = 5$.
$I_5 = \{i/a_{i5} > 0\} = \{3, 7\}$
Soit $r = Arg(Min\{\frac{7}{5}, \frac{1}{1}\}) = 7$.
d'où le pivot est $a_{75} = 1$.

Etape 04 : Comme le pivot $a_{75} = 1$ on effectuer un pivotage standard avec l'algorithme du simplexe et on obtient le tableau suivant :

C_j		1	4	0	0	0	0	0	$X^{(6)}$
C_B	X_B	x_1	x_2	x_3	x_4	x_5	x_6	x_7	
0	x_3	0	0	1	0	0	3	-5	2
0	x_4	0	0	0	1	0	-5	1	3
4	x_2	0	1	0	0	0	1	0	2
1	x_1	1	0	0	0	0	-2	1	1
0	x_5	0	0	0	0	1	-3	1	1
Z_j		1	4	0	0	0	2	1	$Z_6 = 9$
Cj-Zj		0	0	0	0	0	-2	-1	

Etape 02 : Le tableau courant est optimal, d'où la solution
$\overline{x}^R = (1, 2, 2, 3, 1, 0, 0)$ $\overline{x}^R = (x_1, x_3, x_2, x_4, x_5, x_6, x_7)$ est une solution optimale au (P.L.N.E) initial.

Interprétation des 3 coupes entiers :

La 1^{er} coupe : $-x_1 + x_2 + x_5 = 2$

Puisque x_1 et x_2 sont des variables hors-base dans le premier tableau simplexe et que $x_5 \geq 0$ on a donc :
$-x_1 + x_2 \leq 2$ est l'interprétation de la 1^{er} coupe entière.

La $2^{éme}$ coupe : $x_1 - x_5 + x_6 = 0.$

d'après la 1^{er} coupe en déduit $x_5 = 2 + x_1 - x_2$ et on remplace dans la 2^{eme} coupe on obtient :

$$x_1 - (2 + x_1 - x_2) + x_6 = 0.$$
$$x_1 - 2 - x_1 + x_2 + x_6 = 0 \qquad x_6 \geq 0.$$
$$x_2 - 2 \leq 0 \Longleftrightarrow x_2 \leq 2$$

est l'interprétation de la seconde coupe.

La $3^{éme}$ coupe : $x_5 - 3x_6 + x_7 = 1$

d'après la 2^{eme} coupe on a : $x_6 = x_5 - x_1$ on remplace dans la 3^{eme} coupe on obtient :

$$x_5 - 3(x_5 - x_1) + x_7 = 1.$$
$$x_5 - 3x_5 + 3x_1 + x_7 = 1 \Longleftrightarrow -2x_5 + 3x_1 + x_7 = 1$$
$$\Rightarrow x_5 = 2 + x_1 - x_2$$

d'après la 1^{er} coupe d'où :

$$-2(2 + x_1 - x_2) + 3x_1 + x_7 = 1$$
$$-4 - 2x_1 + 2x_2 + 3x_1 + x_7 = 1$$
$$\Rightarrow x_1 + 2x_2 + x_7 = 5$$

Comme $x_7 \geq 0$ on a : $x_1 + 2x_2 \leq 5$ est l'interprétation géométrique de la 3^{eme} coupe entière.

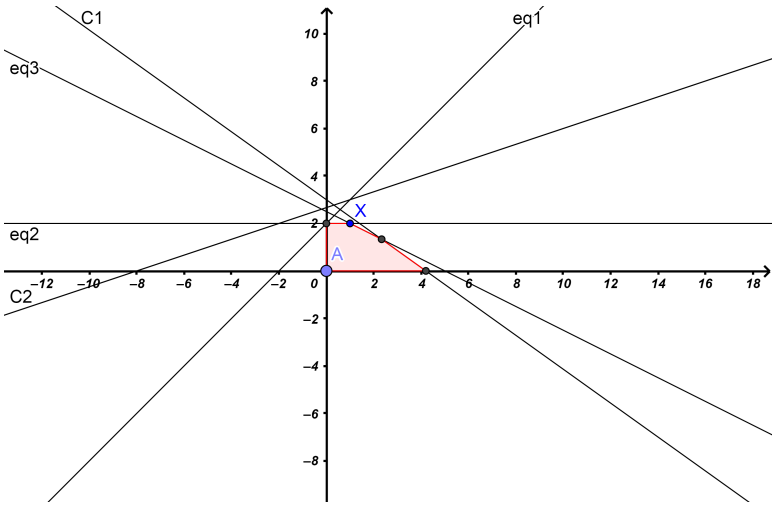


FIGURE 2.2 – La solution graphique du problème 2

$x = (1, 2)$ et $z^* = z(1, 2) = 9$.
$eq1 = -x_1 + x_2 \leq 2$.
$eq2 = x_2 \leq 2$.
$eq3 = x_1 + 2x2 \leq 5$.

2.6 Conclusion

Dans ce chapitre nous nous sommes intéressés à la résolution des programmes linéaires en nombres entiers avec la méthode de "dual fractionnaire Gomory" et la méthode "primal totalement en nombre entiers" .

CHAPITRE 3

MÉTHODE DE SÉPARATION ET ÉVALUATION

3.1 Introduction

Pour plusieurs problèmes , en particulier les problèmes combinatoires,l'espace de solutions est fini (dénombrable). Il est donc possible en principe d'énumérer toutes les solutions et ensuite de prendre la meilleure .L'inconvénient majeur de cette approche est le temps de calcul qui est en général énorme . La méthode de branch and bound "procédure par séparation et évaluation " est basée sur une méthode arborescente qui consiste à réduire par des découpages l'ensemble des solutions qui ne génèrent pas de meilleures solutions. Toutes les séparations sont permises à condition de ne perdre aucune information. La complexité algorithmique diminue alors dans la mesure où on ne calcule pas toutes les solutions du problème.[10]

3.2 méthode de séparation et évaluation

Étant donné un (P.L.N.E) définit par :

$$\begin{cases} \mathrm{z}(Max) = cx \\ Ax \leq b \\ x \in \mathbb{Z}_+^n \end{cases}$$

On définit le problème relaxé (PR) associer au problème (P) par :

$$\begin{cases} \mathrm{z}(Max) = cx \\ Ax \leq b \\ x \geq 0 \end{cases}$$

3.2.1 proposition1

Soient x_E et x_R les solutions optimales au problème (P) et (PR) respectivement et $Z_E = Z(x_E)$ et $Z_R = Z(x_R)$ sont les valeurs de leurs fonctions objectives alors on a $Z_E \leq Z_R$.

3.2.2 Preuve

Par l'absurde, supposons pour un problème donné le contraire c'est à dire $Z_E > Z_R$ puisque x_E est aussi solution au problème (PR) elle vérifie toutes les contraintes $Ax_E \leq b$, il s'en suit que x_R n'est pas optimale, d'où la contradiction.

3.3 les principes de la méthode

Elle consiste à partitionner l'ensemble de toutes les solutions du problème en deux ou plusieurs sous-ensembles en éliminant les parties ne contenant pas les solutions entière et réalisables « principe de séparation, Branching ».

En suit à chaque étape, et en choisissant une stratégie d'exploration on sélectionne un sous-ensemble prometteur qui conduit à une meilleure solution réalisable « principe d'évaluation, Bounding ».

Si : la solution trouvée est entière, on élimine ce sous-ensemble des prochaines considérations et on obtient ainsi une borne inférieure au problème (P), on coupe cette branche de l'arborescence.

Sinon on sépare à nouveau ce problème.

3.3.1 Principe de séparation

Soient x_R la solution optimale du problème relaxé (PR) et x_i une variable de base à valeur non entière de x_R telle que $[x_i] < x_i < [x_i] + 1$. la séparation de (P) selon la variable non entière x_i consiste à diviser (P) en deux programmes :

$$(P_1) = \begin{cases} (P) \\ x_i \leq [x_i] \end{cases} \quad \text{et}(P_2) = \begin{cases} (P) \\ x_i \geq [x_i] + 1 \end{cases}$$

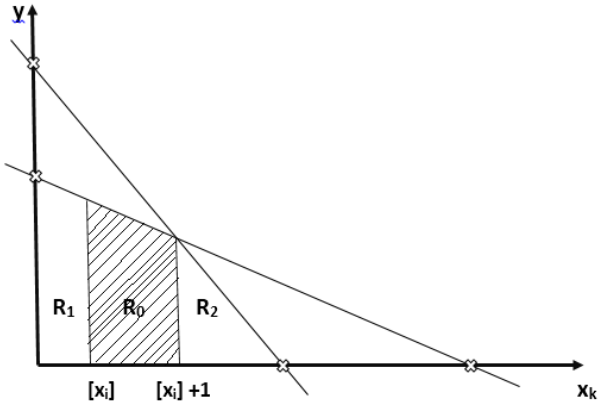


FIGURE 3.1 – Le principe de séparation

Désignons par R_0 la région ne contenant pas les solutions entières, R_1 la région des solutions réalisables de (P_1), et R_2 la région des solutions réalisables de (P_2). la séparation consiste à éliminer R_0, et à chercher des solutions entière dans les régions R_1 et R_2.

3.3.2 Principe de l'évaluation

A l'étape k, on résout le (PR_k) associé à (P_k) deux cas se présentent :

1. La solution optimale obtenue x_R^k est entière, $Z(x_R^k)$ constitue une borne inférieure à tous les programmes prédécesseurs au problème (P_k), et en même temps un majorant à tous les problèmes successeurs au problème (P_k) on coupe alors cette branche.

2. La solution optimale obtenue x_R^k n'est pas entière on sépare à nouveau le problème (P_k)

3.3.3 Définition et notations :

On désigne par :

$P(k, m)$ le programme linéaire en nombre entier du niveau k d'ordre m.
$PR(k, m)$ le programme linéaire relaxé associé à $P(k, m)$.
$P(0, 0)$ et $PR(0, 0)$ désigne (P) et (PR) respectivement.
$x(k, m)$ la solution optimale de $PR(k, m)$.
la séparation de $P(k, m)$ donne $P(k + 1, 2m)$ et $P(k + 1, 2m + 1)$.

3.3.4 Choix d'une stratégie d'exploration des nœuds

L'exploration des nœuds de l'arborescence des solutions obéit à l'une des trios stratégie suivantes :

Stratégie d'exploration par profondeur d'abord :

A Partir d'un nœud donné elle explore tous les nœuds successeurs tant que ceci est possible c'est à dire jusqu'au moment ou on coupe sa branche. a chaque étape on change de niveau.

a partir du nœud $PR(k, m)$ on explore $PR(k+1, m), PR(k+1, m+2)$ ... etc.Son avantage est de permettre de trouver une première solution entière plus rapidement possible.

Stratégie d'exploration par largeur d'abord :

A Partir d'un nœud cette stratégie explore tous les nœuds d'un même niveau, puis sépare tous les nœuds séparables pour ensuite revenir à explorer les nœuds du niveau suivant. A partir du nœud $PR(k, m)$ on explore $PR(k, m+1), PR(k, m+2)$ ... etc.
Son avantage est de permettre de garder la meilleure valeur espérer de la fonction objective.

Stratégie meilleure d'abord :

A Partir d'un nœuds donné cette stratégie explore le nœud ayant la meilleure valeur de la fonction objective. A partir du nœud $PR(k, m)$ on explore $PR(i_0, j_0)$ tel que $Z(i_0, j_0) > Z(i, j)$ avec (PR) non encore exploré $\forall(i, j)$.

3.4 Algorithme de branch and bound

Etape 01 : Étant donnée un programme $(P) = (P.L.N.E)$.

Etape 02 : Déterminer le programme relaxé (PR) associé à (P) puis le résoudre.
Si : x_R est entière terminer elle est optimale.

Etape 03 : **Sinon :** poser $k = 0$, $m = 0$ et Z_E= grand négatif, borne inférieur du (P.L.N.E).

Etape 04 : Choisir une stratégie d'exploration.

Etape 05 : Soient $x(k, m)$ une solution optimale de PR(k,m)
Si : $x(k, m)$ n'est pas réalisable on coupe sa branche, sans améliorer la borne inf.

Sinon : $x(k, m)$ est réalisable, alors voir :

Si : $x(k,m)$ est entier, on coupe sa branche, si de plus $Z(k,m) > Z_E$ on pose $Z_E = Z(k,m)$, amélioration de la borne inférieur de (P).

Sinon : $x(k,m)$ n'est pas entière, et $Z(k,m) < Z_E$.

Soit x_i une variable non entière de $x(k,m)$ et $[x_i]$ sa parti entière alors : $[x_i] < x_i < [x_i] + 1$, on sépare $P(k,m)$ en $P(k+1, 2m)$ et $P(k+1, 2m+1)$ tel que :

$$P(k+1,2m)= \begin{cases} P(k,m) \\ x_i \leq [x_i] \end{cases} \quad \text{et} \quad P(k+1,2m+1)= \begin{cases} P(k,m) \\ x_i \geq [x_i]+1 \end{cases}$$

Etape 06 : L'algorithme s'arrête lorsque toutes les branches sont coupées.

3.4.1 Remarque :

Si le $PR(k,m)$ n'admet pas de solution alors $P(k,m)$ n'a pas non plus de solution.
Un nœud $P(k,m)$ est coupé dans les trois cas suivant :

1. La solution optimale obtenue $x(k,m)$ n'est pas réalisable.

2. La solution optimale obtenue $X(k,m)$ est non entière mais $Z(k,m) \leq Z_E$ avec :
   $Z(k',m') > Z_E$ et $x(k',m')$ est entier pour $k' \leq k$ et $m' \leq m$.

3. La solution optimale obtenue $x(k,m)$ est n'est pas entière mais $Z(k,m) > Z_E$.

3.5 Résolution d'un exemple avec l'algorithme de branch and bound

$$(P) \begin{cases} Z(max) = 3x_1 + 4x_2 \\ 2x_1 + x_2 \leq 6 \\ 2x_1 + 3x_2 \leq 9 \\ x_1, x_2 \in \mathbb{Z}_+ \end{cases}$$

$$(PR) \begin{cases} Z(max) = 3x_1 + 4x_2 \\ 2x_1 + x_2 \leq 6 \\ 2x_1 + 3x_2 \leq 9 \\ x_1, x_2 \in \mathbb{R}_+ \end{cases}$$

Résoudre (PR) avec le simplexe

$$(P.E)\begin{cases} Z(max) = 3x_1 + 4x_2 \\ 2x_1 + x_2 + x_3 = 6 \\ 2x_1 + 3x_2 + x_4 = 9 \\ x_1, x_2, x_3, x_4 \in \mathbb{R}_+ \end{cases}$$

C_j		3	4	0	0	$X^{(0)}$
C_B	x_B	x_1	$\downarrow x_2$	x_3	x_4	
0	x_3	2	1	1	0	6
0	$\leftarrow x_4$	2	**(3)**	0	1	9
Z_j		0	0	0	0	$Z_0 = 0$
Cj-Zj		3	4	0	0	

C_j		3	4	0	0	$X^{(1)}$
C_B	x_B	$\downarrow x_1$	x_2	x_3	x_4	
0	$\leftarrow x_3$	$(\frac{4}{3})$	0	1	$-\frac{1}{3}$	3
4	x_2	$\frac{2}{3}$	1	0	$\frac{1}{3}$	3
Z_j		$\frac{8}{3}$	4	0	$\frac{4}{3}$	$Z_1 = 12$
Cj-Zj		$\frac{1}{3}$	0	0	$-\frac{4}{3}$	

C_j		3	4	0	0	$X^{(2)}$
C_B	x_B	x_1	x_2	x_3	x_4	
3	x_1	1	0	$\frac{3}{4}$	$-\frac{1}{4}$	$\frac{9}{4}$
4	x_2	0	1	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{3}{2}$
Z_j		3	4	$\frac{1}{4}$	$\frac{5}{4}$	$Z_2 = \frac{51}{4}$
Cj-Zj		0	0	$-\frac{1}{4}$	$-\frac{5}{4}$	

Le tableau est optimale, la solution correspondante $X_R = \begin{pmatrix} 2.25 \\ 1.50 \end{pmatrix}$ n'est pas entière.

1. On pose alors $k = 0$, $m = 0$ et $Z_E = -M$, M très grand positif.

2. On choisit la stratégie profondeur d'abord.

On choisit de séparer selon la variable x_2, on scinde donc $P(0,0)$ en $P(1,0)$ et $P(1,1)$ avec :

$$P(1,0) = \begin{cases} P(0,0) \\ x_2 \geq 2 \end{cases} \qquad \text{et} \qquad P(1,1) = \begin{cases} P(0,0) \\ x_2 \leq 1 \end{cases}$$

On résout PR(1,0) avec la méthode dual simplexe, a partir du tableau optimale précédent on rajoute la contrainte $(x_2 \geq 2)$ qui ce traduit par $(x_2 - x_5 = 2)$

$\Longleftrightarrow x_5 = x2 - 2$ mais $x_2 = \frac{3}{2} + \frac{1}{2}x_3 - \frac{1}{2}x_4$ donc $(-x_2 + x_5 = -2)$
$\Longleftrightarrow (-\frac{3}{2} - \frac{1}{2}x_3 + \frac{1}{2}x_4 + x_5 = -2)$
$\Longleftrightarrow (-\frac{1}{2}x_3 + \frac{1}{2}x_4 + x_5 = -\frac{1}{2})$

On rajoute cette contrainte au tableau optimale précédent on obtient le tableau suivant :

C_j		3	4	0	0	0	$X^{(2)}$
C_B	x_B	x_1	x_2	$\downarrow x_3$	x_4	x_5	
3	x_1	1	0	$\frac{3}{4}$	$-\frac{1}{4}$	0	$\frac{9}{4}$
4	x_2	0	1	$-\frac{1}{2}$	$\frac{1}{2}$	0	$\frac{3}{2}$
0	$\leftarrow x_5$	0	0	$-\frac{1}{2}$	$\frac{1}{2}$	1	$-\frac{1}{2}$
Z_j		3	4	$\frac{1}{4}$	$\frac{5}{4}$	0	$Z_2 = \frac{51}{4}$
Cj-Zj		0	0	$-\frac{1}{4}$	$-\frac{5}{4}$	0	

C_j		3	4	0	0	0	$PR(1,0)$ $X^{(3)}$
C_B	x_B	x_1	x_2	x_3	x_4	x_5	
3	x_1	1	0	0	$\frac{1}{2}$	$\frac{3}{2}$	$\frac{3}{2}$
4	x_2	0	1	0	0	-1	2
0	x_3	0	0	1	-1	-2	1
Z_j		3	4	0	$\frac{3}{2}$	$\frac{1}{2}$	$Z_3 = \frac{25}{2}$
Cj-Zj		0	0	0	$-\frac{3}{2}$	$-\frac{1}{2}$	

Le tableau est optimal, la solution correspondante $x(1,0) = \begin{pmatrix} \frac{3}{2} \\ 2 \end{pmatrix}$ n'est pas entière et $Z(1,0) = 12,5 > \underline{Z}_E = -M$.
On sépare donc $P(0,1)$ en $P(2,0)$ et $P(2,1)$ selon une variable non entière soit x_1 : $\quad 1 < x_1 < 2$

$$P(2,0) = \begin{cases} P(1,0) \\ x_1 \leq 1 \end{cases} \qquad P(2,1) = \begin{cases} P(1,0) \\ x_1 \geq 2 \end{cases}$$

On résout $P(2,0)$ en rajoutant la contrainte $X_1 \leq 1$ au tableau optimal de $P(1,0)$.
$(1_1 \leq 1) \Longleftrightarrow (x_1 + x_6 = 1)$ mais $(x_1 = \frac{3}{2} - \frac{1}{2}x_4 - \frac{3}{2}x_5)$ d'où $(x_1 + x_6 = 1)$.
$\Longleftrightarrow (\frac{3}{2} - \frac{1}{2}x_4 - \frac{3}{2}x_5 + x_6 = 1)$.
$\Longleftrightarrow (-\frac{1}{2}x_4 - \frac{3}{2}x_5 + x_6 = -\frac{1}{2})$.

C_j		3	4	0	0	0	0	$X^{(3)}$
C_B	x_B	x_1	x_2	x_3	x_4	$x_5 \downarrow$	x_6	
3	x_1	1	0	0	$\frac{1}{2}$	$\frac{3}{2}$	0	$\frac{3}{2}$
4	x_2	0	1	0	0	-1	0	2
0	x_3	0	0	1	-1	-2	0	1
0	$\leftarrow x_5$	0	0	0	$-\frac{1}{2}$	$(-\frac{3}{2})$	1	$-\frac{1}{2}$
Z_j		3	4	0	$\frac{3}{2}$	$\frac{1}{2}$	0	Z_3=$\frac{25}{2}$
Cj-Zj		0	0	0	$-\frac{3}{2}$	$-\frac{1}{2}$	0	

C_j		3	4	0	0	0	0	$PR(2,0)$ $X^{(4)}$
C_B	x_B	x_1	x_2	x_3	x_4	$x_5 \downarrow$	x_6	
3	x_1	1	0	0	0	0	1	1
4	x_2	0	1	0	$\frac{1}{3}$	0	$-\frac{2}{3}$	$\frac{7}{3}$
0	x_3	0	0	1	$-\frac{1}{3}$	0	$-\frac{4}{3}$	$\frac{5}{3}$
0	x_5	0	0	0	$-\frac{1}{3}$	1	$-\frac{2}{3}$	$\frac{1}{3}$
Z_j		3	4	0	$\frac{4}{3}$	0	$\frac{1}{3}$	Z_4=$\frac{37}{3}$
Cj-Zj		0	0	0	$-\frac{4}{3}$	0	$-\frac{1}{3}$	

Le tableau est optimal, la solution correspondante $x(2,0)$ x_R= $\begin{pmatrix} 1 \\ \frac{7}{3} \end{pmatrix}$ = n'est pas entière et $Z(2,0) = \frac{37}{3} > \underline{Z}_E = -M$.

On sépare donc $P(2,0)$ en $P(3,1)$ selon la variable non entière x_2 $2 < x_2 < 3$ avec :

$$P(3,0) = \begin{cases} P(2,0) \\ x_2 \leq 2 \end{cases} \quad \text{et } P(3,1) = \begin{cases} P(2,0) \\ x_2 \geq 3 \end{cases}$$

On résout $P(3,0)$ en rajoutant la contrainte $(x_2 \leq 2)$ au tableau optimal $P(2,0)$.
$x_2 \leq 2 \Longleftrightarrow (x_2 + x_7 = 2) \Longleftrightarrow (\frac{7}{3} - \frac{1}{3}x_4 + \frac{2}{3}x_6 + x_7 = 2)$
$\Longleftrightarrow (-\frac{1}{3}x_4 + \frac{2}{3}x_6 + x_7 = -\frac{1}{3})$

C_j		3	4	0	0	0	0	0	$PR(2,0)$
C_B	x_B	x_1	x_2	x_3	x_4	x_5	x_6	x_7	$X^{(5)}$
3	x_1	1	0	0	0	0	1	0	1
4	x_2	0	1	0	$\frac{1}{3}$	0	$-\frac{2}{3}$	0	$\frac{7}{3}$
0	x_3	0	0	1	$-\frac{1}{3}$	0	$-\frac{4}{3}$	0	$\frac{5}{3}$
0	x_5	0	0	0	$\frac{1}{3}$	1	$-\frac{2}{3}$	0	$\frac{1}{3}$
0	$\leftarrow x_7$	0	0	0	$-\frac{1}{3}$	0	$\frac{2}{3}$	1	$-\frac{1}{3}$
Z_j		3	4	0	$\frac{4}{3}$	0	$\frac{1}{3}$	0	$Z_5=\frac{37}{3}$
Cj-Zj		0	0	0	$-\frac{4}{3}$	0	$-\frac{1}{3}$	0	

C_j		3	4	0	0	0	0	0	$X^{(6)}$
C_B	x_B	x_1	x_2	x_3	x_4	x_5	x_6	x_7	
3	x_1	1	0	0	0	0	1	0	1
4	x_2	0	1	0	0	0	0	1	2
0	x_3	0	0	1	0	0	-2	-1	2
0	x_5	0	0	0	0	1	0	1	0
0	x_4	0	0	0	1	0	-2	-3	1
Z_j		3	4	0	0	0	3	4	Z_6=11
Cj-Zj		0	0	0	0	0	-3	-4	

Le tableau est optimal, la solution correspondante $x(3,0)=\begin{pmatrix}1\\2\end{pmatrix}$ est entière, on coupe cette branche, de plus on a $Z(3,0)=11>\underline{Z}_E=-M$, on améliore donc la borne inférieur de $\underline{Z}_E=Z(3,0)=11$. on remonte au nœud.

$$P(3,1)=\begin{cases}P(2,0)\\x_2\geq 3\end{cases}$$

On résout P(3,1) en rajoutant la contrainte $(x_2\geq 3)$ au tableau optimal de P(2,0).
$(x_1\geq 3)\Longleftrightarrow(x_2-x_7=3)\Longleftrightarrow(-x_2+x_7=-3)$ mais $x_2=\frac{7}{3}-\frac{1}{3}x_4+\frac{2}{3}x_6$
$\Longleftrightarrow -x_2=-\frac{7}{3}+\frac{1}{3}x_4-\frac{2}{3}x_6$.
On a $(-x_2+x_7=-3)\Longleftrightarrow(-\frac{7}{3}+\frac{1}{3}x_4-\frac{2}{3}x_6+x_7=-3)$
$\Longleftrightarrow(\frac{1}{3}x_4-\frac{2}{3}x_6+x_7=-\frac{2}{3})$.

C_j		3	4	0	0	0	0	0	$X^{(4)}$
C_B	x_B	x_1	x_2	x_3	x_4	x_5	$\downarrow x_6$	x_7	
3	x_1	1	0	0	0	0	1	0	1
4	x_2	0	1	0	$\frac{1}{3}$	0	$-\frac{2}{3}$	0	$\frac{7}{3}$
0	x_3	0	0	1	$-\frac{1}{3}$	0	$-\frac{4}{3}$	0	$\frac{5}{3}$
0	x_5	0	0	0	$\frac{1}{3}$	1	$-\frac{2}{3}$	0	$\frac{1}{3}$
0	$\leftarrow x_7$	0	0	0	$\frac{1}{3}$	0	$(-\frac{2}{3})$	1	$-\frac{2}{3}$
Z_j		3	4	0	$\frac{4}{3}$	0	$\frac{1}{3}$	0	Z_4=$\frac{37}{3}$
Cj-Zj		0	0	0	$-\frac{4}{3}$	0	$-\frac{1}{3}$	0	

C_j		3	4	0	0	0	0	0	$PR(3,1)$ $X^{(5)}$
C_B	x_B	x_1	x_2	x_3	x_4	x_5	x_6	x_7	
3	x_1	1	0	0	$\frac{1}{2}$	0	0	$\frac{3}{2}$	0
4	x_2	0	1	0	0	0	0	-1	3
0	x_3	0	0	1	-1	0	0	-2	3
0	x_5	0	0	0	0	1	0	-1	1
0	x_6	0	0	0	$\frac{1}{2}$	0	1	$-\frac{3}{2}$	1
Z_j		3	4	0	$\frac{3}{2}$	0	0	$\frac{1}{2}$	Z_5=12
Cj-Zj		0	0	0	$-\frac{3}{2}$	0	0	$-\frac{1}{2}$	

Le tableau est optimal,la solution correspondante $x(3,1) = \begin{pmatrix} 0 \\ 3 \end{pmatrix}$ est entière, on coupe donc le nœud $P(3,1)$. de plus on a : $Z(3,1) = 12 > 11 = \underline{Z}_E$. on améliore la borne inférieure de $P(0,0)$ c'est à dire $\underline{Z}_E = 12$, et d'après la stratégie profondeur d'abord on remonte au nœud .

$$P(2,1) = \begin{cases} P(1,0) \\ x_1 \geq 2 \end{cases}$$

il suffit de rajouter la contrainte $(x_1 \geq 2)$ au tableau optimal $PR(1,0)$.
$(x_1 \geq 2) \Longleftrightarrow (x_1 - x_6 = 2) \Longleftrightarrow (-x_1 + x_6 = -2)$ mais
$(x_1 = \frac{3}{2} - \frac{1}{2}x_4 - \frac{3}{2}x_5) \Longleftrightarrow (-x_1 = -\frac{3}{2} + \frac{1}{2}x_4 + \frac{3}{2}x_5)$ d'où
$(-x_1 + x_6 = -2) \Longleftrightarrow (-\frac{3}{2} + \frac{1}{2}x_4 + \frac{3}{2}x_5 + x_6 = -2) \Longleftrightarrow (\frac{1}{2}x_4 + \frac{3}{2}x_5 + x_6 = -\frac{1}{2})$

C_j		3	4	0	0	0	0	$PR(2,1)$ $X^{(3)}$
C_B	x_B	x_1	x_2	x_3	x_4	x_5	x_6	
3	x_1	1	0	0	$\frac{1}{2}$	$\frac{3}{2}$	0	$\frac{3}{2}$
4	x_2	0	1	0	0	-1	0	2
0	x_3	0	0	1	-1	-2	0	1
0	x_6	0	0	0	$\frac{1}{2}$	$\frac{3}{2}$	1	-$\frac{1}{2}$
Z_j		3	4	0	$\frac{3}{2}$	$\frac{1}{2}$	0	Z_3=$\frac{25}{2}$
Cj-Zj		0	0	0	$-\frac{3}{2}$	$-\frac{1}{2}$	0	

Le dual n'a pas de solution fini, donc le primal n'a pas de solution réalisable on coupe le nœud $P(2,1)$. d'après la stratégie profondeur d'abord on remonte au nœud

$$P(1,1) = \begin{cases} P(0,0) \\ x_2 \leq 1 \end{cases}$$

IL suffit de rajouter la contrainte $(x_2 \leq 1)$ au tableau optimal PR(0,0) $(x_1 \leq 1) \Longleftrightarrow (x_2 + x_5 = 1)$ mais $(x_2 = \frac{3}{2} + \frac{1}{2}x_3 - \frac{1}{2}x_4)$ donc $(x_2 + x_5 = 1) \Longleftrightarrow (\frac{3}{2} + \frac{1}{2}x_3 - \frac{1}{2}x_4 + x_5 = 1) \Longleftrightarrow (\frac{1}{2}x_3 - \frac{1}{2}x_4 + x_5 = -\frac{1}{2})$

C_j		3	4	0	0	0	$X^{(2)}$
C_B	x_B	x_1	x_2	x_3	$\downarrow x_4$	x_5	
3	x_1	1	0	$\frac{3}{4}$	$-\frac{1}{4}$	0	$\frac{9}{4}$
4	x_2	0	1	$-\frac{1}{2}$	$\frac{1}{2}$	0	$\frac{3}{2}$
0	$\leftarrow x_5$	0	0	$\frac{1}{2}$	$(-\frac{1}{2})$	1	$-\frac{1}{2}$
Z_j		3	4	$\frac{1}{4}$	$\frac{5}{4}$	0	Z_2=$\frac{51}{4}$
Cj-Zj		0	0	$-\frac{1}{4}$	$-\frac{5}{4}$	0	

C_j		3	4	0	0	0	$PR(1,1)$ $X^{(3)}$
C_B	x_B	x_1	x_2	x_3	x_4	x_5	
3	x_1	1	0	$\frac{1}{2}$	0	$-\frac{1}{2}$	$\frac{5}{2}$
4	x_2	0	1	0	0	1	1
0	$\leftarrow x_4$	0	0	-1	1	-2	1
Z_j		3	4	$\frac{3}{2}$	0	$\frac{5}{2}$	Z_3=$\frac{23}{2}$
Cj-Zj		0	0	$-\frac{3}{2}$	0	$-\frac{5}{2}$	

Le tableau est optimal et la solution correspondante $x(1,1) = X_R = \begin{pmatrix} \frac{5}{2} \\ 1 \end{pmatrix}$ est non entière, et comme $Z(1,1) = 11.5 < \underline{Z}_E$ on coupe ce nœud, sans amélioré la borne inférieur de $P(0,0).x(0,0) = \left(\frac{9}{4}\frac{3}{2}\right)$ non entière.

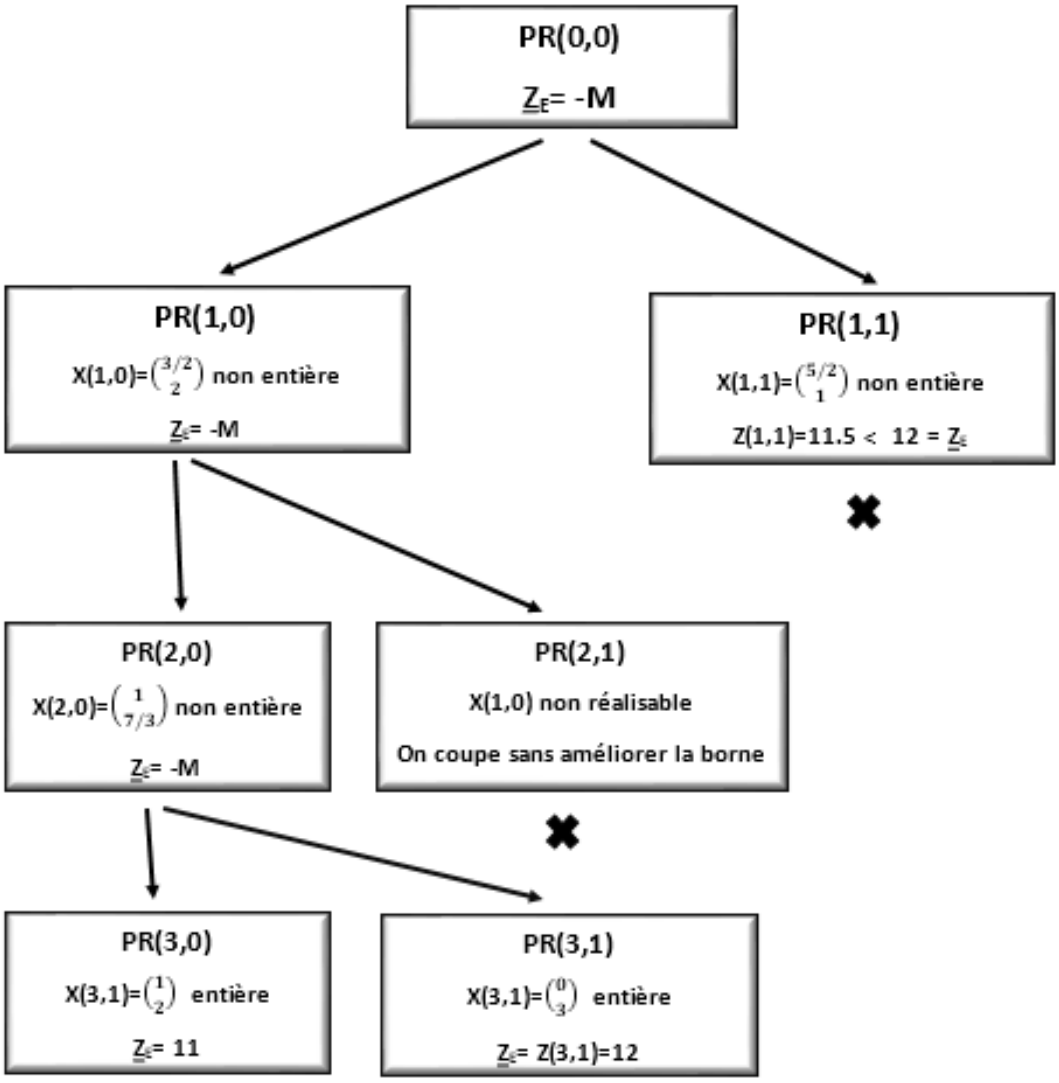


FIGURE 3.2 – la solution obtenue par la stratégie profondeur d'abord

Branche 0 : $PR(0,0)$, $PR(1,0)$,$PR(2,0)$, $PR(3,0)$.
Branche 1 :$PR(3,1)$.
Branche 2 :$PR(2,1)$.
Branche 3 :$PR(1,1)$.

Remarque :
On a utilisé la stratégie : profondeur d'abord.

les résultats obtenue par les autres stratégies sont :

• **la stratégie : meilleur d'abord :**

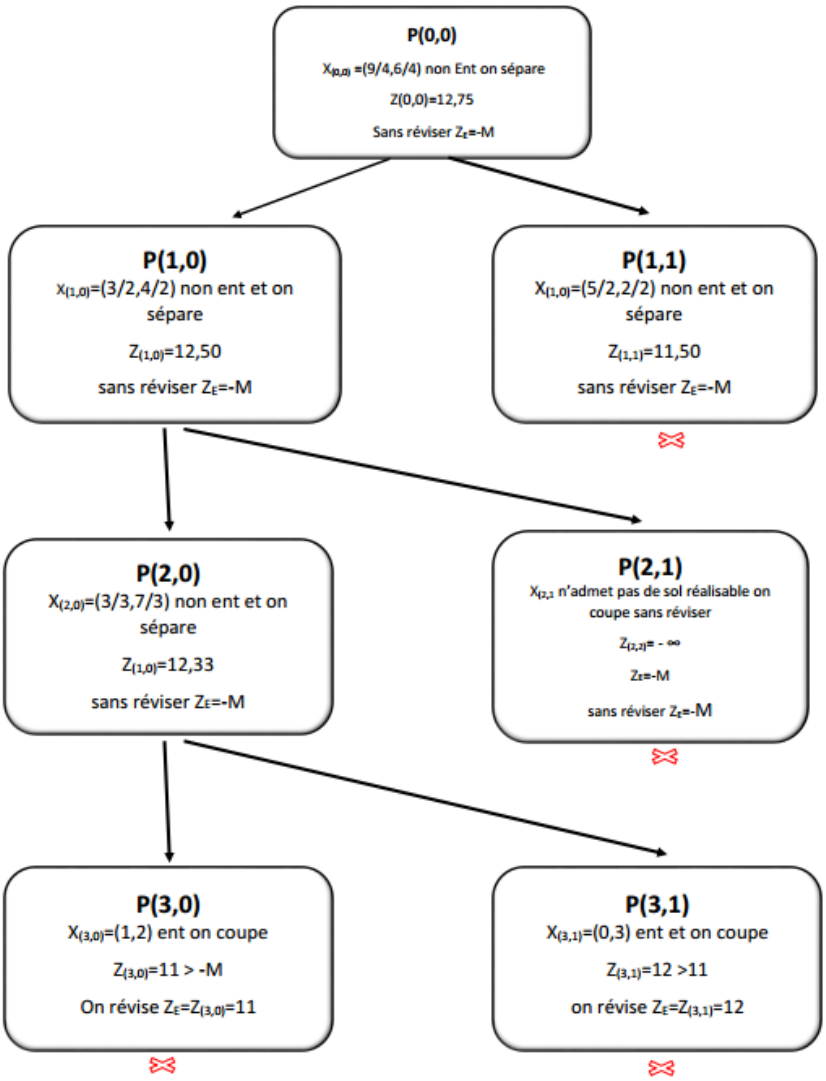


FIGURE 3.3 – la solution obtenue par la stratégie meilleur d'abord

• **la stratégie : Exploration largeur d'abord :**

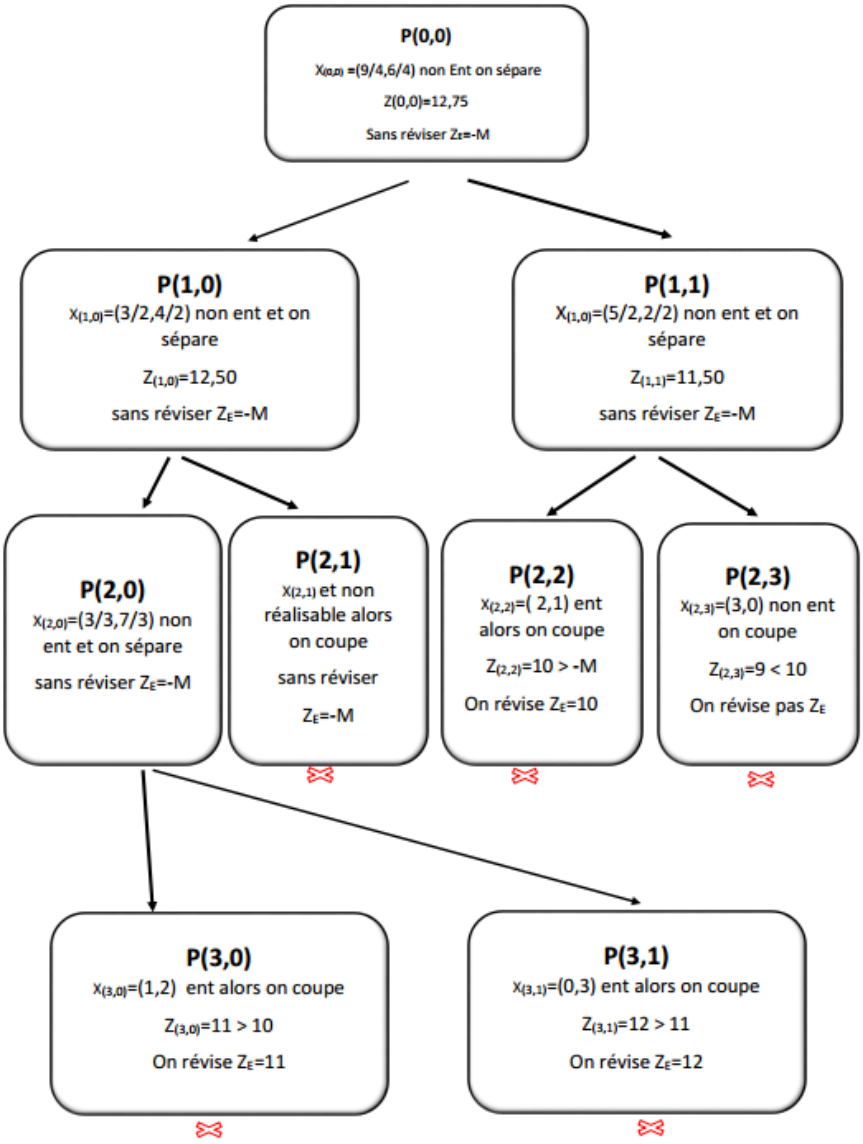


FIGURE 3.4 – la solution obtenue par la stratégie d'Exploration largeur d'abord

•**En utilisant Matlab 2018 :**

Exemple sur la coupe de Gomory :

```
f=[-3 ;-4 ;0 ;0] ;    /* la fonction objectif doit être toujours min */
Aeq=[2 1 1 0 ; 2 3 0 1] ;    /* la matrice qui représente les contraintes */
beq=[6 ;9] ;    /* le vecteur b */
lb=[0 ;0 ;0 ;0 ] ;    /* la borne inf des X ( x>=0) */
intcon=[1,2] ;    /* les indices des X */
x=intlinprog(f,intcon,[ ],[ ],Aeq,beq,lb) ;
x
```

Le résultat obtenu par ce programme est :

```
>> f=[-3;-4;0;0] ;
Aeq=[2 1 1 0 ; 2 3 0 1] ;
beq=[6;9] ;
lb=[0 ;0 ;0 ;0 ] ;
intcon=[1,2] ;
x=intlinprog(f,intcon,[ ],[ ],Aeq,beq,lb) ;
x
LP:                Optimal objective value is -12.750000.

Cut Generation:    Applied 1 Gomory cut.
                   Lower bound is -12.000000.
                   Relative gap is 0.00%.


Optimal solution found.

Intlinprog stopped at the root node because the objective value is within a gap tolerance of the optimal value,
options.AbsoluteGapTolerance = 0 (the default value). The intcon variables are integer within tolerance,
options.IntegerTolerance = 1e-05 (the default value).


x =

         0
    3.0000
    3.0000
         0
```

FIGURE 3.5 – Le résultat obtenu par programme MATLAB (coupe de Gomory)

$x1 = 0 \quad x_2 = 3 \quad$ et $z = -(-12) = 12.$

Exemple sur la coupe de branch and bound :

f = [2 10 13 17 7 5 7 3] ;
Aeq = [22 13 26 33 21 3 14 26 39 16 22 28 26 30 23 24 18 14 29 27 30 38 26 26 41 26 28 36 18 38 16 26] ;
beq = [7872 10466 11322 12058] ;
N = 8 ;
lb = zeros(N,1) ;
intcon = 1 :N ;
x=intlinprog(f,intcon,[ ],[ ],Aeq,beq,lb) ;
x
Le résultat obtenu par ce programme est :

```
LP:                Optimal objective value is 1554.047531.

Cut Generation:    Applied 8 strong CG cuts.
                   Lower bound is 1590.479592.

Branch and Bound:

   nodes      total    num int        integer       relative
explored  time (s)  solution           fval        gap (%)
   10000      0.82         0              -              -
   10947      0.90         1   2.481000e+03   3.585818e+01
   14148      1.25         2   2.073000e+03   2.319190e+01
   17131      1.56         3   1.854000e+03   1.180593e+01
   17777      1.63         3   1.854000e+03   0.000000e+00

Optimal solution found.

Intlinprog stopped because the objective value is within a gap tolerance of the optimal value,
options.AbsoluteGapTolerance = 0 (the default value). The intcon variables are integer within tolerance,
options.IntegerTolerance = 1e-05 (the default value).


x1 =

   24.0000
   15.0000
   19.0000
   11.0000
    3.0000
   99.0000
    4.0000
  226.0000
```

FIGURE 3.6 – Le résultat obtenu par programme MATLAB (branch and bound)

- **En utilisant(LP Solve IDE) :**

Le problème correspondant est le suivant :

/* Objective function */
max : 3x1 + 4x2 ;
2x1 + x2 + x3 = 6 ;
2x1 + 3x2 + x4 = 9 ;
/* Variable bounds */
x3 >= 0 ;
x4 >= 0 ;
int x1,x2 ;

Le résultat obtenu par ce programme est :

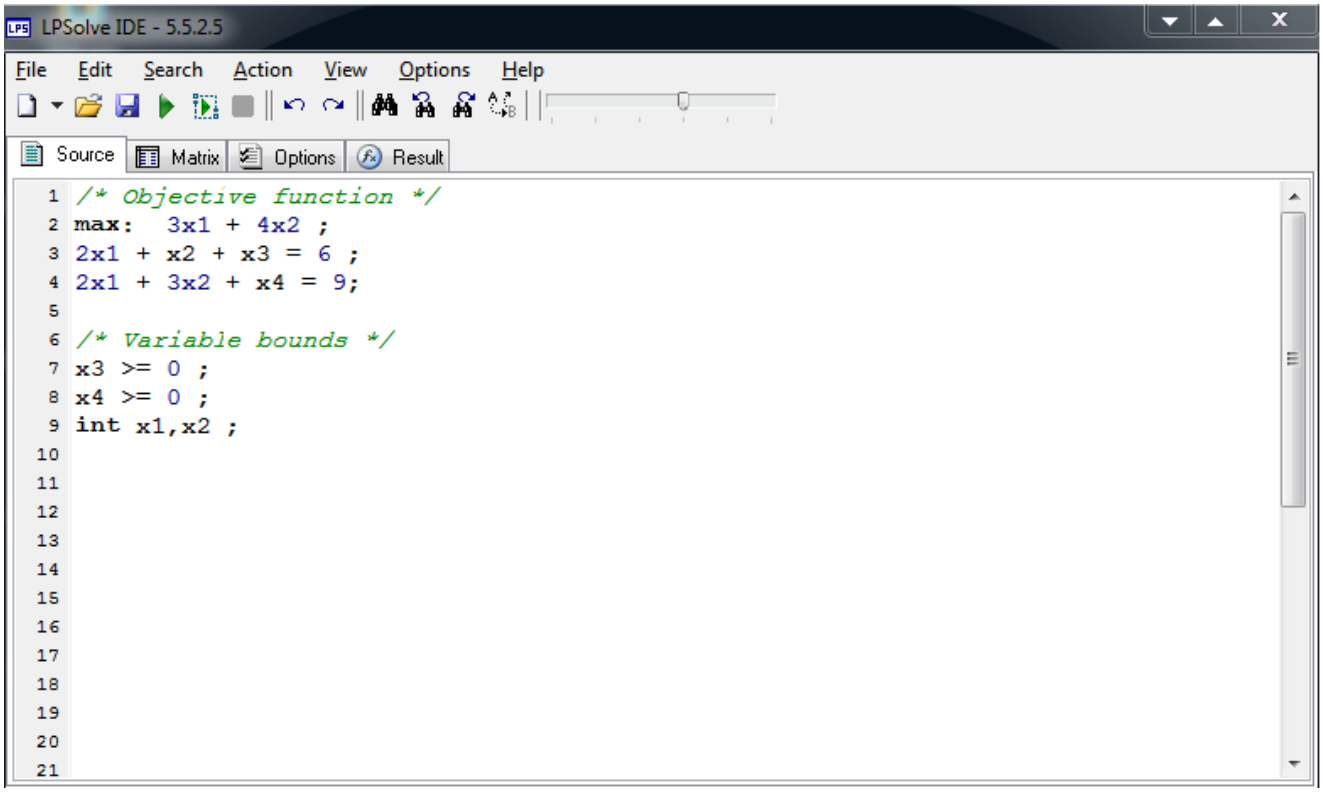


FIGURE 3.7 – les commandes sur le programme LP Solve IDE

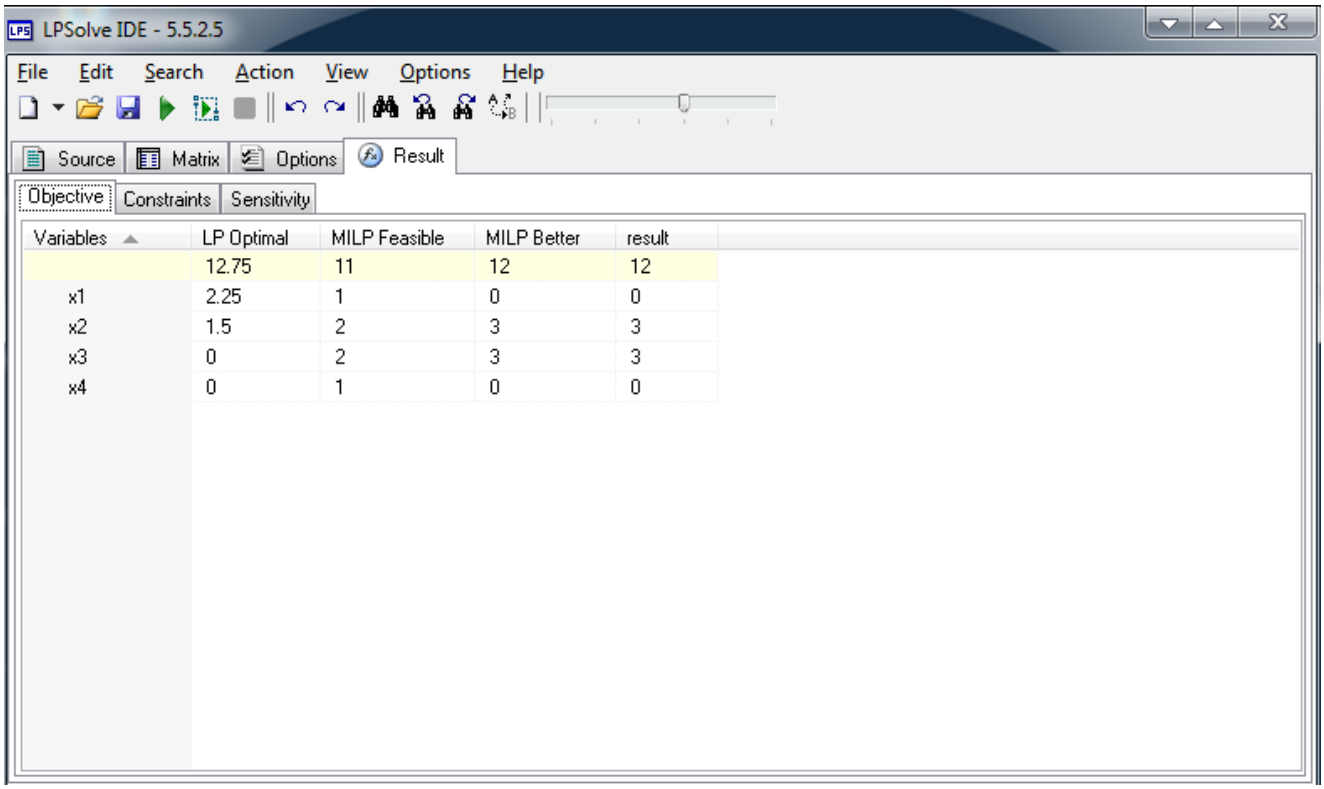


FIGURE 3.8 – Le résultat obtenu le programme LP Solve IDE

3.6 Conclusion

Dans ce chapitre nous nous sommes intéressés à la résolution des programmes linéaires en nombres entiers avec la méthode de Séparation et d'Évaluation "Branch and Bound "qui se repose sur le principe d'énumération.
Son efficacité est liée directement au principe d'élagage qui permet de diminuer l'espace de recherche des solutions.

CONCLUSION

La méthode primal totalement en nombre entiers est utilise quand la méthode dual fractionnaire de Gomory est conduite au blocage c'est à dire quand un (PLNE) donnée A,b et c entiers données, le nombre imprévisible de coupes générés ne procurent pas de progrès dans l'évaluation de la borne $\omega = \pi.b$ du (PLNE) on est forcé de s'arrêter de génère des coupes sans autant atteindre au moins une solution réalisable entière au (PLNE).

La méthode par séparation et évaluation est meilleure que la méthode dual fractionnaire de Gomory même amélioré par des heuristiques de recherche de coupes profondes la méthode dual fractionnaire de Gomory présente un inconvénient majeur en effet les données d'un (PLNE) sont entiers A,b et c. les résultats qu'on cherche $x = \overline{b} = A_J^{-1}.b$ sont également des entiers cependant on travaille sur des réels ce qui ralenti considérablement les calculs et entraine la propagation des erreurs d'arrondi ce qui rend incertain les testes d'intégrité sur les variables .

Le nombre imprévisible d'itération des méthodes de coupes fait que celles ci ne sont utilisées qu'intégrées avec des procédures par séparation et évaluation.

BIBLIOGRAPHIE

[1] AIDEN M.OUKACHA B., « les manuels de l'étudiant Recherche Opérationnelle, Programmation Linéaire ». Copyright Eurl Pages Bleues internationales, Maison d'Edition pour l'enseignement et la formation, 2005.

[2] A.SCHRIJVER. Théorie of linear and integer Programming John Wiley Sons 1986.

[3] D.AISSAOUI. Méthode hybride pour la résolution des problèmes de programmation linéaire en nombres entiers. Mémoire de master.2014

[4] D.DE WERA. Éléments de programmation linéaire avec application aux graphes. Presse polytechnique rowandes 1990.

[5] D.G.SUENBERGER. Introduction linear and Non Linear programing John Wiley Sons 1984.

[6] F.MEUNIER.Introduction à la recherche Opérationnelle.Université Paris Est, Cermics, École des Ponts Paristech .2016.

[7] G.BAILLARGEON. programmation linéaire appliquée.

[8] HAMDY A.TAHA. Integer Programming Théory. Applicatons and computation academic press 1975.

[9] J.TEGHEM Programmation linéaire. Editions Ellipse 1996.

[10] L.A.WOLSEY G.L.Nemhauser, Integer and combinatorial optimization.

[11] LAND A.H., Doig A.G., An automatic method of solving discrete programming problems, Econometrica 28 (3), 497-520, 1960.

[12] LEMKE C.E., « The dual method for solving the linear programming problem », Naval Research Logistic Quarterly 1, 36-47, 1954.

[13] M.MINOUX. Programation mathématique Théorie et algorithmes. Tome 2 (1983).

[14] M.SAKAROUVITCH. Optimisation combinatoire, programmation discrète.paris 1984.

[15] N.WU RICHARD COPPINS NESA. Linear Programing And Extensions McGraw.Hill 1981.

[16] ROSEAUX Exercices et problèmes résolus de recherche opérationnelle.paris 1998.

[17] R.S GARFINKEL G.L NEMHAUSER. Integer programing John Wiley Sons 1972.

[18] S.HILLIER, J.LIEBERMAB. Introduction to operations researche. Ninth Edition. New York, 2010

[19] M.R.GAREY, D.S.JOHNSON Computers and intractability freeman dans co.1979

Résumé

Notre travaille porte sur la programmation linéaire en nombre entiers, branche fameuse de la programmation mathématique qui s'implique dans la majorité des problèmes d'optimisation combinatoire complexes (a savoir le flot dans les réseaux et problème de recouvrement et partitionnement) le travail brassa également les coupes fractionnaire de Gomory et la méthode primal totalement en nombres entiers, ainsi que la méthode de branch and bound. une application réalisé sous le MATLAB a été appliquée avec succès au cas des coupes de Gomory ou de branch and bound.

Mots clés : Programmation linéaire, optimisation combinatoire,coupe de Gomory, branch and bound.

Abstract

Our work focuses on integer linear programming, branch of mathematical programming, which is involved in most of the complex combinatorial optimization problems (i. e. the flow in the re-evaluation of the buckets and problem of covering and partitioning) the work also brewed the fractional Gomory cuts and the primal method totally in integers, as well as the branch and bound method. an application made under MATLAB has been successfully applied to the case of Gomory cuts or branch and bound.

Keywords : Linear programming, combinatorial optimization, cutting of Gomory, branch and bound.