Table des matières

1) Introduction

La programmation en nombres entiers regroupe l'ensemble des techniques permettant de résoudre des programmes linéaires dont les solutions doivent être entières. Formellement, le programme linéaire en nombres entiers s'exprime comme suit:

$$(P)\begin{cases} Max\ Z = c\ x \\ A\ x \leq b \\ x \in Z \end{cases}$$

La matrice A d'ordre m x n, le vecteur colonne b d'ordre m et le vecteur ligne c d'ordre n sont supposés être des entiers. Le vecteur colonne x d'ordre n est entier, c'est-à-dire, les variables du système A x ≤ b sont soumises à être entières. La relaxation de cette contrainte d'intégrité donne le programme linéaire (Q) associé comme suit :

$$(Q)\begin{cases} Max\ Z = c\ x \\ A\ x \leq b \\ x \geq 0 \end{cases}$$

La résolution de (P) ne peut pas se faire par la résolution de (Q) puis d'arrondir ensuite les solutions non entières. En effet, le simple arrondi des variables non entières à des valeurs entières peut rendre la solution non réalisable ou simplement non optimale. Considérons l'exemple suivant :

$$\text{Max } Z = 3\ x_1 + 4\ x_2$$
$$2\ x_1 + x_2 \leq 6$$
$$2\ x_1 + 3\ x_2 \leq 9$$
$$x_1, x_2 \in N$$

La résolution graphique du programme linéaire associé (Q) donne :

Soit D le polyèdre délimité par l'ensemble des contraintes de (P) illustré par la figure suivante : Les points extrêmes sont O, A, B et C et la solution optimale est atteinte au point B qui n'est pas entier. L'arrondi de cette solution donne (2, 2) qui n'est pas réalisable, (2, 1) qui est réalisable mais n'est pas optimale.

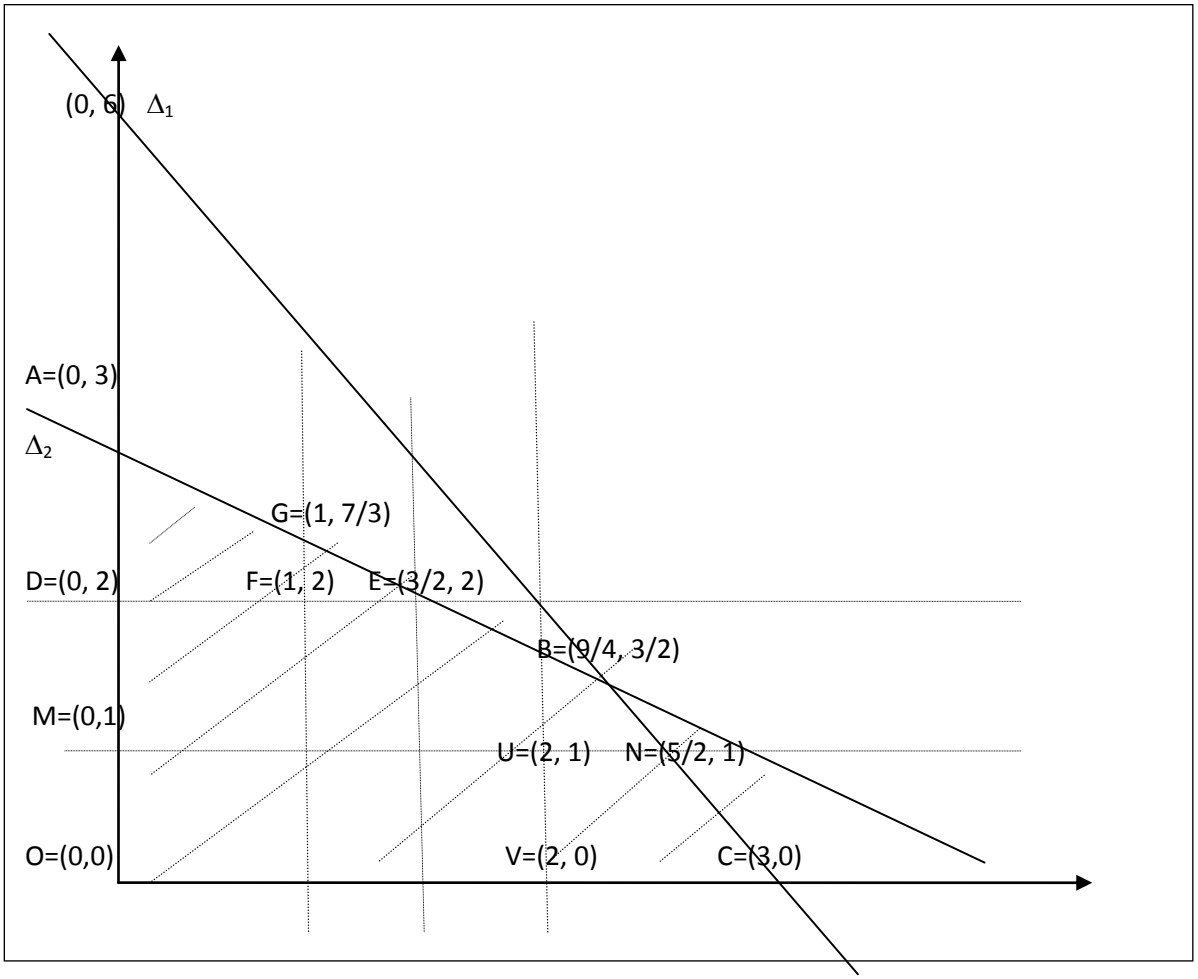

Figure 1 : Polyèdre D délimitant l'espace des solutions de (P)

Dans ce chapitre nous allons présenter deux approches différentes pour résoudre (P) à savoir : La méthode de séparation et d'évaluation (Brand and Bound) et la méthode des coupes.

2) Quelques exemples de modélisation en PLNE

Beaucoup de problèmes pratiques se modélisent comme des programmes linéaires en nombres entiers. En voici quelque un :

2.1) Le problème du sac-à-dos

Le problème du sac à dos où on doit remplir un sac à dos avec unités de n éléments où chaque élément j possède un poids p_j et une valeur nutritive c_j de telle façon à ne pas dépasser son poids total p et en maximisant la valeur nutritive.

En notant xj le nombre d'unités de l'élément j à prendre dans le sac à dos, ce problème se modélise comme suit :

$$\begin{cases} Max\ Z = \sum_{j=1}^{n} c_j\ x_j \\ \sum_{j=1}^{n} p_j\ x_j \leq p \\ x \in Z^+ \end{cases}$$

Il existe une variante où les variables de décision sont binaires. Dans ce cas, il s'agit de prendre ou de ne pas prendre l'élément j dans le sac à dos.

2.2) Planification de la production

Dans un problème de planning de production de n produits, le coût de production d'un produit j est constitué d'un coût fixe Kj, indépendant du volume de production de ce produit et d'un coût variable cj par unité du produit.

Si on désigne par xj le niveau de production du produit j, le coût total de production du produit j est : $c_j(x_j) = \begin{cases} K_j + c_j x_j & si\ x_j > 0 \\ 0 & sinon \end{cases}$

La fonction objectif est : Min $Z = \sum_{j=1}^{n} c_j(x_j)$

On notera que la fonction objectif Z n'est pas continue au point origine. Pour palier ce problème, nous introduisons une variable auxiliaire y_j définie comme suit :

$$y_j = \begin{cases} 0 & si\ \ x_j = 0 \\ 1 & si\ \ x_j > 0 \end{cases}$$

Ces deux conditions peuvent être exprimées comme une contrainte linéaire suivante :

$x_j \leq M\ y_j$ où M est un nombre très grand. Le problème peut être formulé comme suit :

$$\begin{cases} Min\ Z = \sum_{j=1}^{n} c_j\ x_j + K_j\ y_j \\ 0 \leq x_j \leq M\ y_j \\ y_j = 0\ ou\ 1\ \ j = 1, \dots, n \end{cases}$$

Remarques :

Le problème d'origine n'était pas un problème de programmation en nombres entiers, il a été transformé pour régler le problème de la discontinuité au point origine et est devenu un problème de linéaire mixte.

Un programme linéaire est dit mixte si une partie des variables est entière et une autre ne l'est pas.

2.3) Problème de dichotomie

Supposons que dans une situation donnée, il est requit qu'uniquement k contraintes parmi les m contraintes du problème soient actives. Supposer que les m contraintes sont sous forme la forme suivante: $g_i(x_1, \ldots, x_n) \leq b_i$ avec i=1,2, …, m.

On définit la variable y_i associée à la $i^{ème}$ contrainte comme suit :

$$y_i = \begin{cases} 1 & \text{si la ième contrainte est inactive} \\ 0 & \text{si la ième contrainte est active} \end{cases}$$

Pour assurer qu'uniquement k parmi m contraintes soient actives, il suffira de considérer

$$\begin{cases} g_i(x_1, \ldots, x_n) \leq b_i + My_i \quad i = 1, 2, \ldots, m \\ y_1 + y_2 + \ldots + y_m = m - k \end{cases}$$

Avec M un nombre très grand.

En effet, pour les i (i=1, …, k) contraintes actives, nous avons yi = 0 et par conséquent on aura: $g_i(x_1, \ldots, x_n) \leq b_i$. Pour les (m-k) autres contraintes nous avons $g_i(x_1, \ldots, x_n) \leq b_i + M$ qui deviennent redondantes.

Ce type de situation se produit souvent lorsque le second membre d'une contrainte prend une ou plusieurs valeurs, c'est-à-dire,

$$g(x_1, \ldots, x_n) \leq b_1, b_2, \ldots, b_r \qquad (*)$$

On introduit une variable Booléenne y_k, définie comme suit :

$$y_k = \begin{cases} 1 & \text{si } b_k \text{ est le second membre} \\ 0 & \text{sinon} \end{cases}$$

La contrainte (*) est transformée comme suit :

$$g(x_1 \ldots, x_n) \leq \sum_{k=1}^{r} b_k \; y_k$$

$$\sum_{k=1}^{r} y_k = 1$$

3) Lemme

Soient x_e, x_r les solutions optimales respectivement des programmes (P) et (Q) et Z_e, Z_r les valeurs de leurs fonctions objectifs respectivement, alors nous avons : $Z_e \leq Z_r$, c'est-à-dire, que la valeur de la solution optimale du programme (Q) dont les variables ne sont pas astreintes est une borne supérieure à la valeur de la solution optimale entière.

En effet, supposons le contraire, c'est-à-dire, pour un problème donné, nous avons $Z_e > Z_r$. Puisque x_e est aussi solution du programme Q car il vérifie $A\,x_e \leq b$, il s'ensuit que x_r n'est pas optimale car $c\,x_e > c\,x_r$. D'où la contradiction avec le fait que x_r est optimale.

Remarque : Même si le domaine de solution de (Q) est convexe et borné, le nombre de solution du programme (P) peut être tellement grand qu'il serait couteux de choisir la meilleure solution par l'énumération explicite de toutes les solutions

4) Résolution de (P) par Branch-And-Bound

Cette approche a été proposée par Little et Al. pour résoudre le problème de voyageur de commerce puis repris par d'autres auteurs pour résoudre d'autres problèmes en introduisant plusieurs variantes. Le principe de cette approche est « diviser pour mieux régner ». En d'autres termes, il s'agit de partitionner, à chaque itération, le polyèdre convexe représentant l'ensemble des solutions réalisables du problème en trois sous ensembles et à éliminer celui qui ne peut pas contenir de solutions entières (principe de séparation). Chaque sommet généré dans l'arborescence des solutions est évalué par la résolution du programme linéaire associé (principe d'évaluation). Le choix du sommet sur lequel la séparation doit se faire est fait en utilisant une stratégie de sélection. Dès qu'une solution réalisable entière est atteinte alors le sommet correspondant sera élagué et un retour arrière est nécessaire et la valeur de la fonction objectif trouvée sera une borne inférieure à toutes les solutions réalisables à rechercher.

4.1) Principe de séparation

Soit x_m la solution optimale du programme (Q) et x_i une variable non entière. Pour se fixer les idées, supposons que $\alpha < xi < \alpha+1$ où α est nombre entier. La séparation de (P) consiste à scinder ce programme en deux sous programmes :

$$(P_1)\ \begin{cases} Programme\ (P) \\ x_i \leq \alpha \end{cases} \qquad (P_2)\ \begin{cases} Programme\ (P) \\ x_i \geq \alpha + 1 \end{cases}$$

Si nous désignons par R_0 le domaine ne contenant pas de solution entière, R_1 le domaine (P_1) et R_2 celui de (P_2), la séparation consiste à éliminer R_0 et à chercher des solutions entières dans R_1 et R_2.

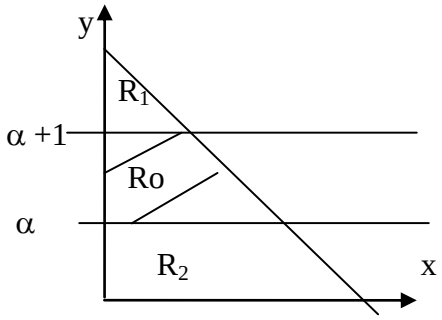

Figure 2 : Principe de séparation

Dans ce qui suit, nous allons adopter les notations suivantes :

$P_{(0, 0)}$ et $Q_{(0, 0)}$ le programme P et Q respectivement,

$P_{(k, m)}$ le programme linéaire en nombres entiers numéro m du niveau k (relativement à l'arborescences des solutions) et $Q_{(k , m)}$ le programme linéaire qui lui est associé,

$x_{(k, m)}$ la solution optimale de $Q_{(k, m)}$,

La séparation de $P_{(k, m)}$ donne lieu à deux sommets $P_{(k+1, 2m)}$ et $P_{(k+1, 2m+1)}$

4.2) Principe d'évaluation

Chaque nœud généré par la séparation à l'étape k, à savoir, (P_k) est évalué en résolvant le programme linéaire associé. Deux cas se présentent:

1er cas: La solution obtenue est entière, elle est une borne inférieure au programme linéaire (P) et en même temps un majorant à toutes les solutions de la branche issue de ce sommet (P_k). On élague (coupe) ce nœud et on fait un backtraking (retour arrière) aux sommets non encore explorés.

2eme cas: La solution obtenue n'est pas entière et le sommet (P_k) correspondant est candidat à la séparation.

4.3) Stratégies d'exploration

L'exploration des sommets de l'arborescence des solutions obéit à une stratégie donnée. Dans la littérature, nous distinguons plusieurs stratégies dont voici quelque unes :

Stratégie 'Profondeur d'abord'

Cette stratégie fait la séparation d'un sommet tant que celui-ci peut se faire. En d'autre termes, le nœud est séparé en profondeur jusqu'à atteindre la feuille de l'arborescence des solutions. Un retour-arrière est nécessaire dans le cas où il existe des sommets non élagués (coupés). A partir d'un nœud $Q_{(k, m)}$ on explore $Q_{(k+1, m)}$, $Q_{(k+2, m)}$, etc.

Stratégie 'Largeur d'abord'

Cette stratégie résout d'abord tous les problèmes d'un même niveau puis sépare les sommets séparables pour ensuite résoudre les problèmes du niveau suivant. A partir d'un nœud $Q_{(k, m)}$ on explore $Q_{(k, m+1)}$, $Q_{(k, m+2)}$, etc.

Stratégie 'Meilleur d'abord'

Dans le cas des deux stratégies précédentes, l'ordre d'exploration des sommets est prédéfini au départ, par contre dans cette stratégie, il ne l'est pas. On explore le sommet ayant la meilleure valeur de la fonction objectif.

A partir du sommet $Q_{(k, m)}$ on explore $Q_{(i,j)}$ tel que $Z_{(i j)} > Z_{(u, v)}$ pour tout sommet (u, v) non encore exploré.

Il existe d'autres stratégies telles que celle basée sur la notion de pénalité.

4.4) Algorithme de Branch and Bound

(0) Choisir une stratégie d'exploration. Poser k=0, m=0 et $Z_e = -\infty$

(1) Résoudre $Q_{(k, m)}$ et soit $x_{(k,m)}$ sa solution optimale.

Si $x_{(k,m)}$ est entière. On coupe le nœud et si $Z_{(k, m)} > Z_e$ on pose $Z_e = Z_{(k, m)}$ (

Si $x_{(k, m)}$ n'est entière

Si $Z_{(k, m)} \leq Z_e$. On coupe le nœud

Sinon $Z_{(k, m)} > Z_e$ Aller en 2)

2) Soit x_i une variable non entière de $x_{(k,m)}$ et a est nombre entier tel que $a < x_i < a+1$.

Séparer $P_{(k,m)}$ en $P_{(k+1, 2m)}$ et $P_{(k+1, 2m+1)}$ où :

$P_{(k+1, 2m)} = \{ P_{(k,m)}, x_i \leq a\}$, $P_{(k+1, 2m+1)} = \{P_{(k,m)}, x_i \geq a+1\}$.

Résoudre $Q_{(k+1, 2m)}$ et $Q_{(k+1, 2m+1)}$.

3) Tester si tous les nœuds sont coupés.

Si oui. Terminer.

Sinon, choisir le nœud à explorer mettre à jour les paramètres m et k et aller en 1)

4.5) Exemple illustratif

Reprenons le programme linéaire en nombres entiers $P_{(0,0)}$) précédent:

$$\text{Max } Z = 3x_1 + 4x_2$$
$$2x_1 + x_2 \leq 6$$
$$2x_1 + 3x_2 \leq 9$$
$$x_1, x_2 \in N$$

Son programme linéaire associé $Q_{(0,0)}$ est :

$$\text{Max } Z = 3x_1 + 4x_2$$
$$2x_1 + x_2 \leq 6$$
$$2x_1 + 3x_2 \leq 9$$
$$x_1, x_2 \geq 0$$

La solution optimale de $Q_{(0,0)}$ est atteinte au point B=(9/4, 3/2) qui n'est pas entière. On pose $Z_e = -M$ (M très grand positif).

Dans cet exemple, le lecteur est prié de se référer à la figure 3.1 pour la construction des différents polyèdres obtenus après introduction de contraintes supplémentaires. Il est bien évident que cet exemple est à titre illustratif et que dans la réalité nous devons résoudre des programmes linéaires en utilisant des tableurs appropriés comme le Cplex ou autre.

1) Résolution de (P) par la stratégie « Profondeur d'abord »

On pose k=0 et m=0 et Z_e=-M (M un nombre positif grand). On choisit de séparer sur la variable x_2= 3/2 (choix arbitraire, on raffinera ce choix plus tard). On a $1< x_2< 2$
On sépare $P_{(0,0)}$ en $P_{(1,0)}$ et $P_{(1,1)}$ avec $P_{(1,0)} = \{P_{(0,0)}, x_2 \geq 2\}$ et $P_{(1,1)} = \{P_{(0,0)}, x_2 \leq 1\}$
On résout $Q_{(1,0)}$, on obtient le domaine délimité par DAE où D=(0, 2) A=(0,3) et E=(3/2, 2). La solution optimale est atteinte en E avec x_1= 3/2, x_2=2 et Z_E = 12.5. Cette solution n'est pas entière. On explore alors $P_{(1,0)}$ en séparant sur la variable non entière x_1 car $1 < x_1 < 2$. Le nœud $P_{(1,0)}$ est séparé en $P_{(2,0)}$ et $P_{(2,1)}$ avec $P_{(2,0)}=\{P_{(1,0)}, x_1 \leq 1\}$ et $P_{(2,1)} = \{P_{(1,0)}, x_1 \geq 2\}$. On résout $Q_{(2,0)}$ qui donne le polytope DAFG où D=(0, 2) A=(0, 3) F=(1, 2) et G=(1, 7/3). La solution optimale est atteinte en G avec x_1=1 et x_2=7/3 et Z_G=12.33 qui n'est pas entière. On explore le nœud $P_{(2,0)}$ en séparant sur la variable x_2 avec $2< x_2< 3$. On définit deux sommets $P_{(3,0)}$ et $P_{(3,1)}$ où $P_{(3,0)}=\{P_{(2,0)}, x_2 \leq 2\}$ et $P_{(3,1)} = \{P_{(2,0)}, x_2 \geq 3\}$. On résout $Q_{(3,0)}$, le polytope est réduit à un segment [D, F]. La solution optimale est atteinte au point F avec x_1=1 et x_2=2

et $Z_F = 11 > Z_e$. Cette solution est entière, on coupe le nœud $P_{(3,0)}$ et on pose $Z_e = 11$, nouvelle borne inférieure au programme (P).

Nous faisons un retour arrière au niveau 2 et on explore $P_{(2,0)}$ et on résout $Q_{(3, 1)}$ qui est réduit au point (0, 2). Cette solution étant entière et on coupe le nœud $P_{(3, 0)}$. Comme $Z_{(3, 1)} = 12 > Z_e = 11$, elle constitue une nouvelle borne inférieure et on pose $Z_e=12$ au programme (P).

Nous faisons un retour arrière au niveau 1. On résout $Q_{(2, 1)}$ dont le polyèdre correspondant est vide. On coupe ce sommet car il ne peut pas mener vers une solution de base réalisable entière.

Nous remontons au niveau 0 et nous résolvons $Q_{(1, 1)}$) où le polyèdre correspondant est OMNC avec O=(0, 0) M=(0, 1) N=(5/2, 1) et C=(3, 0). La solution optimale est atteinte au point N avec $x_1=5/2$, $x_2=1$ et $Z_N = 11.5 < Z_e=12$. On coupe ce sommet car on ne peut pas espérer obtenir une meilleur solution que celle déjà trouvée avec une valeur de la fonction objectif égale à 12. Par conséquent, $x_1^*=0$, $x_2^*=3$ et $Z^*=12$.

En résumé, la stratégie « profondeur d'abord » a donné l'exploration des nœuds suivants (voir Figure 3.2) :

Branche 0 : Résolution des programmes $Q_{(0, 0)}$, $Q_{(1, 0)}$, $Q_{(2, 0)}$ et $Q_{(3, 0)}$.

Branche 1 : $Q_{(3, 1)}$

Branche 2 : $Q_{(2, 1)}$

Branche 3 : $Q_{(1, 1)}$

2) Résolution de (P) par la stratégie « Largeur d'abord »

En adoptant la même démarche, cette stratégie conduit à l'exploration des nœuds suivants (voir Figure 3.2) :

Niveau 0 : Résolution de $Q_{(0, 0)}$ qui donne $x_1=9/4$ et $x_2=3/2$

Niveau 1 : Résolution de $Q_{(1, 0)}$ qui donne $x_1=3/2$, $x_2=2$, $Z=12.5$ et $Q_{(1, 1)}$ qui donne $x_1=5/2$, $x_2=1$ et $Z=11.5$.

Niveau 2 : Résolution de $Q_{(2, 0)}$, $Q_{(2, 1)}$, $Q_{(2, 2)}$ et $Q_{(2, 3)}$. Seul le nœud $P_{(2, 2)}$ est retenu.

En effet, on sépare le nœud $P_{(1, 0)}$ en $P_{(2, 0)}$ et $P_{(2, 1)}$ puis on sépare $P_{(1, 1)}$) en $P_{(2, 2)}$ et $P_{(2, 3)}$ avec : $P_{(2, 2)}=\{P_{(1, 1)}, x_1 \leq 2\}$ et $P_{(2, 3)}=\{P_{(1, 1)}, x_2 \geq 3\}$. La résolution $Q_{(2, 2)}$ donne le domaine OMUV avec M=(0, 1) U=(2, 1) et V=(2, 0). La solution optimale est atteinte en U avec $x_1=2$, $y_1=1$ et Z=10. On coupe cette branche et on définit une nouvelle borne inférieure $Z_e=10$.

La résolution de $Q_{(2, 3)}$ se résume au point (3, 0) et la solution trouvée est $x_1=3$, $x_2=0$ et $Z=9 <Ze$. On coupe cette branche et on conserve l'ancienne borne inférieure.

Niveau 3 : Résolution de $Q_{(3, 0)}$ et $Q_{(3, 1)}$

La résolution de Q(3, 0) et Q(3, 1) conduisent à des solutions entières. Le nœud P(3, 0) offre une meilleure borne inférieure que celle trouvée en P(2, 2). Le sommet P(3, 1) est optimale car il conduit à une solution entière de meilleure valeur de la fonction objectif.

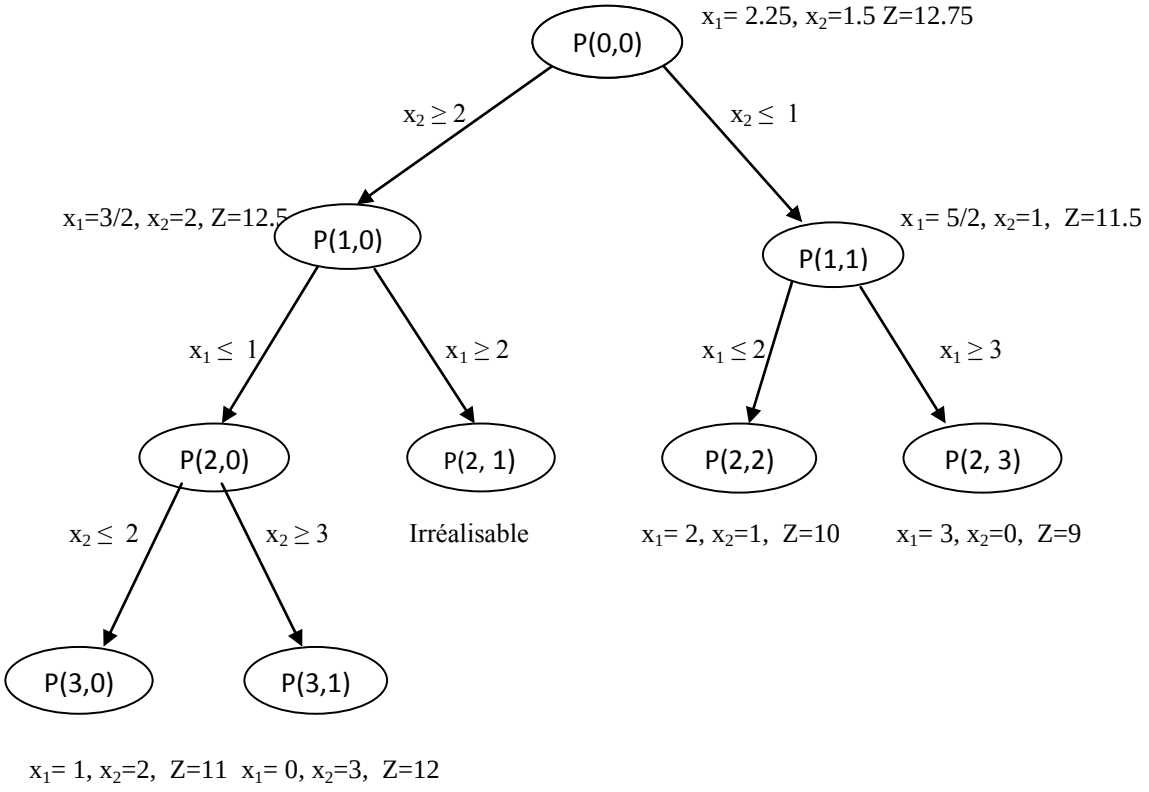

Figure 3.2 : Arborescence des solutions de (P)

3) <u>Résolution par la stratégie « meilleure d'abord » :</u>

Au niveau O, nous résolvons $Q_{(0,0)}$ et nous séparons $P_{(0,0)}$ puis nous résolvons $Q_{(1,0)}$ et Q(11). Nous séparons P(1, 0) car Q(1, 0) possède une meilleure valeur de la fonction objectif. Nous résolvons ensuite $Q_{(2,0)}$. Nous séparons $P_{(2,0)}$ en $P_{(3,0)}$ et $P_{(3, 1)}$. Nous gardons la solution offerte par $P_{(3, 1)}$. Nous arrêtons la résolution.

En résumé, cette stratégie explore les nœuds de l'arborescence représentée dans la figure 3.2 comme suit :

Niveau 0 : $Q_{(0, 0)}$

Niveau1: $Q_{(1, 0)}$ et $Q_{(1, 1)}$

Branche 2: $Q_{(2, 0)}$) et $Q_{(2, 1)}$

Branche 3: $Q_{(3, 0)}$ et $Q_{(3, 1)}$

4.6) Amélioration de l'approche

L'orque le programme à résoudre est de grande dimension, il est très important de chercher des bornes inférieure et supérieure permettant de réduire rapidement l'arborescence des solutions et le temps d'exécution des programmes car non seulement ce dernier devient exorbitant, l'espace mémoire devient insuffisante pour terminer la résolution. Si nous n'avons pas trouvé une solution réalisable entière, tous les efforts consentis pour la résolution sont vaines.

Dans l'exemple précèdent la résolution des programme $Q_{(3,0)}$ avant $Q_{(1,1)}$ a permis d'éviter la résolution de $Q_{(2,2)}$ et $Q_{(2,3)}$. Il apparait donc évident que l'ordre d'exécution des programmes est capitale pour la résolution d'un problème de programmation linéaire en nombres entiers.

Si plusieurs variables sont candidates à la séparation, une façon de gérer cette situation est de choisir celle qui intervient dans le plus grand nombre de contraintes. Cependant, la démarche suivante est plus solide qui consiste à chercher la meilleure borne inférieure avant d'entamer le processus de résolution.

La bonne inferieure qui constitue un minorant pour la solution optimale d'un problème de programmation linéaire en nombres entiers sert à élaguer des nœuds de l'arborescence assez rapidement et d'éviter ainsi l'exploration d'un nombre important de branches.

Une manière d'opérer consiste à prospecter le voisinage de la solution optimale initiale du programme linéaire associé et de considérer n'importe qu'elle solution entière réalisable. La meilleure solution voisine peut être retenue comme une borne inferieure.

Dans l'exemple précèdent, la solution optimale (2.25, 1.5), les solutions voisines sont (2,1) (2,2), (3,1), (3,2). La solution réalisable entière est (2, 1) avec Z=10. On peut démarrer l'approche avec Z_e=10. La meilleure solution de départ est celle qui est obtenue après résolution d'un programme linéaire à variables bivalents.

Nous réécrivons le programme (Q) comme suit :

$$\begin{cases} Max\ Z = \sum_{j=1}^{n} c_j x_j \\ \sum_{j=1}^{n} a_{ij} x_j \leq b_i \quad i = 1, \dots, m \\ x_j \geq 0 \quad j = 1, \dots, n \end{cases}$$

Soit $x^* = (x_1^*, \dots, x_n^*)$ la solution optimale de (Q). Posons $x_j = [x_j^*] + y_j$ où [.] désigne la partie entière de la solution non entière et y_j est une variable booléenne. Le vecteur x est entier. Si de plus, il est réalisable alors la valeur de sa fonction objectif constituera une borne inférieure au problème initial. La meilleure borne est celle qui optimise le programme suivant :

$$\begin{array}{c} Max\ Z = \sum_{j=1}^{n} c_j y_j \\ \sum_{j=1}^{n} a_{ij} y_j \leq b_i - \sum_{j=1}^{n} a_{ij} [x_j *] \quad i = 1, \dots, m \\ y_j = 1, 0 \quad j = 1, \dots, n \end{array} \qquad \text{(Pb)}$$

En effet, en remplaçant x_j^* dans le programme (P) et en utilisant le fait que les quantités $c_j [x_j^*]$ et $a_{ij} [x_j^*]$ sont des constantes, on retrouve (Pb).

La recherche de borne supérieure est utile pour améliorer les stratégies d'exploration décrites précédemment. En effet, si le sommet (P) est séparé en (P_1) et (P_2), on cherche une borne supérieure à P_1 et une autre à P_2. La priorité sera accordée au nœud qui possède la plus grande borne. Comme cette borne est un majorant à toutes les solutions sur sa branche et si elle est inférieure à la meilleure borne inférieure Z_e, elle sera alors supprimée.

Soit (P) le nœud à séparer selon la variable de base x_k non entière. Posons $x_k = [x_k] + f_k$ où $0 < f_k < 1$ et considérons $P_1 = \{P, x_k \leq [x_k]\}$. Soit *B* l'ensemble des indices des variables de base et *HB* ceux des variables hors base. La $k^{ième}$ ligne du tableau optimal du programme linéaire (Q) correspondant à (P) s'exprime comme suit: $\sum_{j \in HB} a_{kj} x_j + x_k = [x_k] + f_k$.

D'où : $x_k = -\sum_{j \in HB} a_{kj} x_j + [x_k] + f_k \leq [xk]$ (d'après la construction de P_1).

Il s'ensuit : $-\sum_{j \in HB} a_{kj} x_j \leq -f_k$.

En introduisant une variable d'écart S, on obtient : $-\sum_{j \in HB} a_{kj} x_j + S = -f_k$

On ajoute cette contrainte au tableau final du simplexe, on obtient une solution de base non réalisable mais optimale ($f_k < 0$). Le changement de base par l'algorithme dual du simplexe rapproche la faisabilité de la solution tout en la gardant optimale. La solution optimale

obtenue après une seule itération sera prise comme une borne supérieure à la solution optimale réalisable du nœud (P_1).

On procédera de la même manière pour le nœud P_2 ={P, $x_k \geq [x_k] + 1$ } et on obtiendra :

$$\sum_{j \in HB} a_{kj} x_j \; + S = \; f_k - 1$$

On définit la pénalité inférieure que l'on désignera $P_{x\text{-inf}}$ la diminution sur la valeur optimale de la fonction objectif P occasionnée par une seule itération de l'algorithme dual du simplexe après ajout de la contrainte $x_k \leq [x_k]$.

De même, on définit la pénalité supérieure que l'on désignera $P_{x\text{-sup}}$ la diminution sur la valeur optimale de la fonction objectif P occasionnée par une seule itération de l'algorithme dual du simplexe après ajout de la contrainte $x_k \geq [x_k] + 1$.

Une fois ces deux pénalités calculées, la règle consiste à se brancher en priorité sur le nœud qui génère une pénalité la plus faible car elle occasionne une plus faible diminution de la fonction objectif (dans le cas d'une maximisation).

En utilisant l'algorithme dual du simplexe en appliquant le principe de la variable sortante x_s puis entrante x_e, on obtient :

$$P_{x\text{-inf}} = f_k * \frac{\Delta_{se}}{a_{se}}$$

$$P_{x\text{-sup}} = (1 - f_k) * \frac{\Delta_{se}}{a_{se}}$$

Où Δ_{se} est le coefficient marginal de la fonction objectif après ajout de la contrainte supplémentaire et a_{se} est l'élément du tableau du simplexe situé à l'intersection entre la variable sortante et la variable entrante.

Reprenons l'exemple précédent et après ajout des variables d'écarts on obtient le programme linéaire (Q) associé à (P):

$$\text{Max } Z = 3\,x_1 + 4\,x_2$$

$$2\,x_1 + x_2 + e_1 = 6$$

$$2\,x_1 + 3\,x_2 + e_2 = 9$$

$$x_1, x_2 \geq 0$$

Après résolution par l'algorithme du simplexe, on obtient le dernier tableau du simplexe de (Q) :

Base	b	x_1	x_2	e_1	e_2
x_1	2,25	1	0	0,75	-0,25
x_2	1,5	0	1	0,5	0,5
Z	-12 ,75	0	0	0,25	-1,25

Les variables d'origine sont non entières. Supposons que l'on sépare sur la variable x_2= 1,5=1+0,5, d'où $[x_2]$=1 et f_2 = 0,5.

Pour le calcul de $Px_2 - Inf$, on ajoute la contrainte supplémentaire $-\sum_{j \in HB} a_{2j} x_j + S =$ *-0,5* et obtient le tableau suivant :

Base	b	x_1	x_2	e_1	e_2	S
x_1	2,25	1	0	0,75	-0,25	0
x_2	1,5	0	1	-0,5	0,5	0
S	-0,5	0	0	0,5	- 0,5	1
Z	-12 ,75	0	0	- 0,25	-1,25	0

La variable sortante étant S et la variable entrante est e_2. D'où :

$P_{x\text{-inf}} = f_k * \frac{\Delta_{se}}{a_{se}} = 0,5 * (\frac{1,25}{0,5}) = 1,25$

Après une itération, on obtient :

Base	b	x_1	x_2	e_1	e_2	S
x_1	2,50	1	0	0,5	0	-0,5
x_2	1	0	1	0	0	1
e_2	1	0	0	-1	1	-2
Z	-11 ,5	0	0	-1,5	0	-2.5

Pour le calcul de $Px_2 - Sup$, on ajoute la contrainte supplémentaire $\sum_{j \in HB} a_{2j} x_j + S =$ *-0,5* et on obtient le tableau suivant :

Base	b	x_1	x_2	e_1	e_2	S
x_1	2,25	1	0	0,75	-0,25	0
x_2	1,5	0	1	-0,5	0,5	0
S	-0,5	0	0	-0,5	0,5	1
Z	-12 ,75	0	0	- 0,25	-1,25	0

La variable sortante est S et la variable sortante est e_1. D'où :

$P_{x2\text{-sup}} = (1 - f_k) * \frac{\Delta_{se}}{a_{se}} = 0.5\ (\frac{0.25}{0.5}) = 0.25$

On se branche en priorité sur le nœud ayant la plus faible pénalité, à savoir, (P_2) dans cet exemple.

5) Méthode des coupes de Gomory (troncatures)

5.1. Notion de coupe de Gomory

Dans l'approche Branch-And-Bound que nous avons développé dans la section précédente, la recherche des solutions est faite en éclatant le polyèdre convexe délimité par les contraintes en trois parties distinctes et à éliminer celle qui ne contient pas de solutions entières et à explorer les deux autres. L'approche par les coupes de Gomory est complémentent différente dans le sens où elle coupe le polyèdre et se rapproche de la solution optimale entière d'itération en itération.

Considérons le tableau optimal du simplexe associé au programme linéaire (Q) et soit x_k une variable base non entière de valeur x_k*. La ligne associée à cette variable dans le tableau du simplexe est : $x_k + \sum_{j \in HB} a_{kj} x_j = x_k^*$ *(1)*

Or, $x_k^* = [x_k^*] + f_k$ où $0 < f_k < 1$

$a_{kj}^* = [a_{kj}^*] + f_{kj}$ où $0 < f_{kj} < 1$

(1) peut être ré exprimé comme suit :

$$x_k + \sum_{j \in HB} [a_{kj}] x_j + \sum_{j \in HB} f_{kj}\, x_j = [x_k^*] + f_k$$

Où encore : $x_k - [x_k^*] + \sum_{j \in HB} [a_{kj}] x_j + = f_k - \sum_{j \in HB} f_{kj}\, x_j$

La partie de gauche est entière, il sera de même pour celle de droite et comme $f_k > 0$ et $f_{kj} > 0$ il s'ensuit que $f_k - \sum_{j \in HB} f_{kj}\, x_j \leq f_k < 1$

Par conséquent, $f_k - \sum_{j \in HB} f_{kj}\, x_j \leq 0$. En introduisant une variable d'écart on obtient :

$$f_k - \sum_{j \in HB} f_{kj}\, x_j + S = 0$$

On définit ainsi une nouvelle contrainte, appelée, coupe de Gomory que l'on rajoute au tableau optimal précédent. On résout le nouveau programme linéaire par l'algorithme du simplexe.

5.2) Algorithme de coupes

Une fois le principe de cette approche introduit, nous présentons le détail de l'algorithme :

1) Résoudre le programme linéaire associé et choisir une variable de x_k non entière du dernier tableau de simplexe.
2) On réécrit chaque coefficient de la ligne correspondante comme la somme d'un entier et d'un nombre fractionnel positif inférieur à 1.

3) On forme la coupe constituée des parties fractionnelles
4) On résout le nouveau programme par l'algorithme dual du simplexe. Si la solution est entière, terminer sinon nous reprenons (1).

Remarque :

A chaque itération la dimension du tableau du simplexe devient grande et par conséquent nous augmentons la complexité du problème. L'autre inconvénient est que la solution du problème reste irréalisable jusqu'à l'arrêt de l'algorithme.

5.3 Exemple illustratif

Soit le programme linéaire en nombres entiers (P)
Max $z = 7x_1 + 9x_2$
$-x_1 + 3\,x_2 \leq 6$

$7x_1 + x_2 \leq 35$

$x_1, x_2 \in N$

Soit Π le polytope délimitant l'espace des solutions du programmes linéaire associé (Q) représenté par la Figure 3.3 ci-dessous. Le dernier tableau du simplexe associé au programme linéaire (Q) est

Base	b	x_1	x_2	e_1	e_2
x_2	7/2	0	1	7/22	1/22
x_1	9/2	1	0	-1/22	3/22
Z	-63	0	0	-28/11	-15/11

Nous remarquons que : $x^*_2 = 3 + 1/2$ et $x^*_1 = 4 + 1/2$
Les parties fractionnelles sont égales $f_1 = f_2 = ½$. On choisit au hasard la variable x_2 et on a :

$x_2 + (0+7/22)\, e_1 + (0+1/22) e_2 = (3+1/2)$

La coupe de Gomory est :

$e_3 - 7/22\, e_1 - 1/22\, e_2 = -1/2$, où encore :

Nous l'introduisons dans le dernier tableau du simplexe et on obtient:

Base	bi	x_1	x_2	e_1	e_2	e_3
x_2	7/2	0	1	7/22	1/22	0
x_1	1/2	1	0	-1/22	3/22	0
e_3	-1/2	0	0	-7/22	-1/22	1
Z	-63	0	0	-28/11	-15/11	0

La variable e_3 sort de la base et e_1 entre en base. Après changement de la nouvelle base, on obtient le nouveau tableau suivant :

Base	bi	x_1	x_2	e_1	e_2	e_3
x_2	3	0	1	0	0	1
x_1	4+ 4/7	1	0	0	1/7	-1/7
e_1	1 +4/7	0	0	1	1/7	-22/7
Z	-59	0	0	0	-1	-8

La variable x_1 n'est pas entière. On construit une coupe sur cette variable :

$x_1 + (0+1/7)\ e_2 + (0-1/7)e_3=(4+4/7)$. On doit transformer le coefficient de e_3 pour rendre la partie fractionnelle positive. Nous avons :

$x_1 + (0+1/7)\ e_2 + (-1+6/7)e_3=(4+4/7)$. La coupe de Gomory est :

$e_4 - 1/7\ e_2 - 6/7e_3= - 4/7$

En ajoutant cette coupe au tableau suivant, on obtient :

Base	bi	x_1	x_2	e_1	e_2	e_3	e_4
x_2	3	0	1	0	0	1	0
x_1	4+ 4/7	1	0	0	1/7	-1/7	0
e_1	1 +4/7	0	0	1	1/7	-22/7	0
e_4	-4/7	0	0	0	-1/7	-6/7	1
Z	-59	0	0	0	-1	-8	0

La variable e_2 entre en base en remplacement de la variable e_4. On obtient le nouveau tableau suivant :

Base	bi	x_1	x_2	e_1	e_2	e_3	e_4
x_2	3	0	1	0	0	1	0
x_1	4	1	0	0	0	-1	1
e_1	1	0	0	1	0	-4	1
e_2	4	0	0	0	1	6	-7
Z	-55	0	0	0	0	-2	-7

Cette solution est entière et optimale à savoir $x_1^*=4$, $x_2^*=3$ et $Z^*=55$

Nous pouvons interpréter géométrique l'introduction de ces coupes dans le polyèdre convexe délimitant le domaine convexe des contraintes de ce problème en les ré exprimant en termes de variables d'origines x_1 et x_2.

$e_3 - 7/22\ e_1 - 1/22\ e_2= e_3 – 7/22\ (\ 6 + x_1 - 3\ x_2) - 1/22(35-7\ x_1-x_2)=-1/2$

Où encore : $e_3+ x_2= 3$, ce qui est équivalent à $x_2 \leq 3$ ……. (1).

De la même manière, on peut ré exprimer la seconde coupe sous la forme :

$e_4 - 1/7\ e_2 - 6/7e_3= e_4 – 1/7\ (\ 35 – 7\ x_1 - x_2) – 6/7(3- x_2)= - 4/7$

ce qui entraine $e_4 + x_1 + x_2 = 7$, où encore $x_1 + x_2 \leq 7$ ……(2)

Examinons ce qui se passe lorsque nous introduisons ces deux contraintes dans le domaine délimitant l'ensemble des contraintes du programme (P)

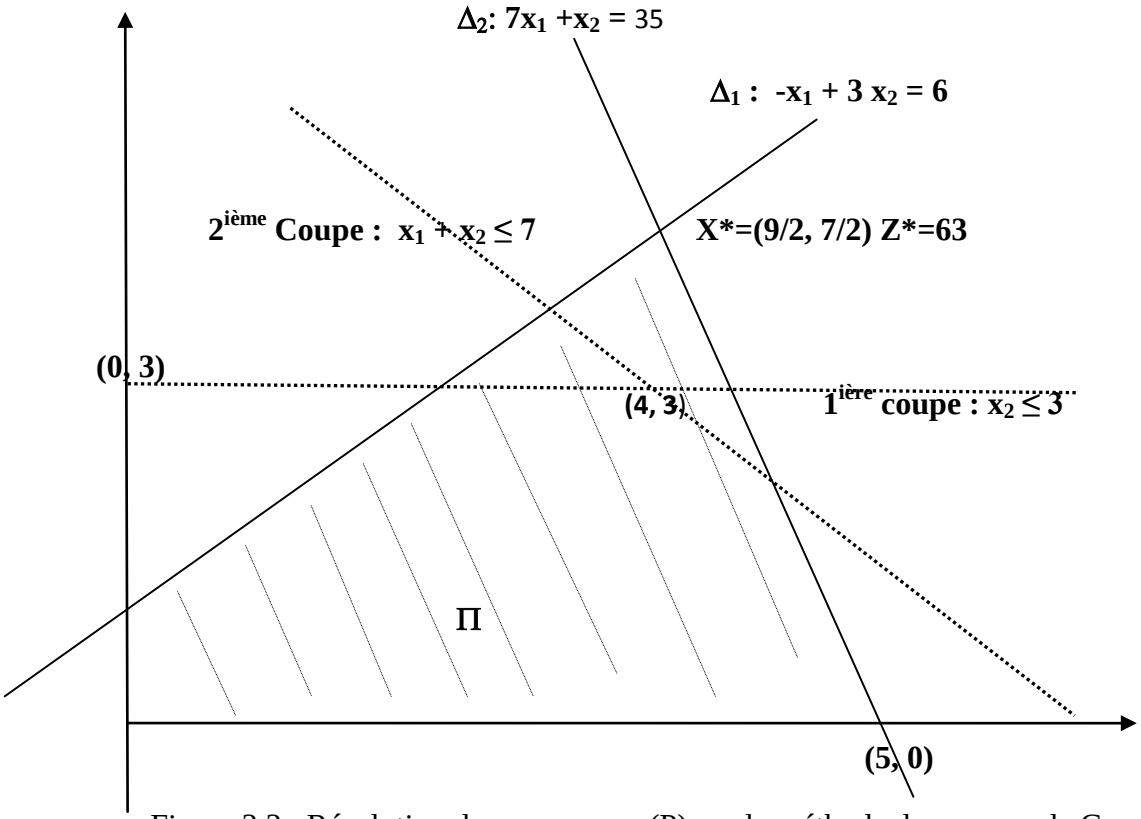

Figure 3.3 : Résolution du programme (P) par la méthode des coupes de Gomory

5.4) Amélioration de la méthode

Une variable non entière peut générée plusieurs coupes différentes et il est intéressant de choisir la meilleure. Pour cela, un critère de choix doit être fixé au préalable pour renforcer le processus de résolution par cette approche.

Une coupe C est dite meilleure qu'une autre coupe C' si elle coupe profondément le polyèdre convexe représentant l'espace des solutions.

En d'autres termes, une coupe C est dite meilleure qu'une autre coupe si et seulement si:

$$\begin{cases} f_{kj} \leq f'_{kj} \ \forall \ j \in HB \text{ sauf pour un } j' \in HB \text{ cette inégalité est stricte} \\ f_k \geq f'_k \end{cases}$$

Où encore : $\frac{f_k}{\sum_{j \in HB} f_{kj}} > \frac{f'_k}{\sum_{j \in HB} f'_{kj}}$

La génération des différentes coupes de Gomory peut être généralisée en λ-coupes où λ est un entier. En effet, reprenons l'égalité (1) précédente et multiplions ses deux membres par un entier λ quelconque :

$$\lambda x_k + \sum_{j \in HB} \lambda([a_{kj}] + f_{kj}).x_j = \lambda [x_k^*] + \lambda f_k \quad (2)$$

Posons : $\lambda f_{kj} = [\lambda f_{kj}] + f_{\lambda fkj}$

$\lambda f_k = [\lambda f_k] + f_{\lambda fk}$

En remplaçant λf_{kj} et λf_k per leurs expressions respectives, nous obtenons :

$$\lambda x_k + \sum_{j \in HB} \lambda[a_{kj}] x_j + \sum_{j \in HB} [\lambda f_{kj}].x_j - \lambda[x^*_k] - [\lambda f_k] = - \sum_{j \in HB} f_{\lambda f_{kj}} x_j + f_{\lambda f_k}$$

La partie de gauche doit être entière et il sera de même pour la partie de droite et on obtient :

$$- \sum_{j \in HB} f_{\lambda f_{kj}} x_j + f_{\lambda f_k} \leq f_{\lambda f_k} < 1$$

D'où : $- \sum_{j \in HB} f_{\lambda f_{kj}} x_j + f_{\lambda f_k} \leq 0$

Où encore : $- \sum_{j \in HB} f_{\lambda f_{kj}} x_j + S = - f_{\lambda f_k}$

Le nombre de coupes de Gomory que l'on peut générer sur une ligne est égal à (D-1) où D est le déterminant de la partie de base B de la matrice (Ax ≤ b) associée à la solution optimale de (Q). (Résultat à démontrer à titre d'exercice). En d'autres termes, on a : $0 \leq \lambda \leq (D-1)$.

La meilleure coupe est celle qui vérifie : $Max_i\ r_i$ où $r_i = \frac{f_{\lambda_i f_k}}{\sum_{j \in HB} f_{\lambda_i f_{kj}}}$

Un autre critère pour le choix de la meilleure coupe est celle qui possède la plus faible des pénalités (même sens que celle que nous avons introduite dans la section précédente).

Dans l'exemple précédent D=22. Il s'agira donc de construire 21 coupes de Gomory différentes.

λ=1 on aura la coupe: $e_3 - 7/22\ e_1 - 1/22\ e_2 = - ½$ (1ière coupe trouvée) $r_1 = 11/8$

λ=2 on aura la coupe : $e_3 - 14/22\ e_1 - 2/22\ e_2 = 0$ et $r_2 = 0$

λ=3 on aura la coupe : e_3 - 21/22 e_1 - 3/22 e_2= -1/2 et r_3= 11/16

λ=4 on aura la coupe : e_3 - 6/22 e_1 - 4/22 e_2= 0 et r_4= 0

λ=5 on aura la coupe : e_3 - 13/22 e_1 - 5/22 e_2= -1/2 et r_5= 11/18

λ=6 on aura la coupe : e_3 - 20/22 e_1 - 6/22 e_2= 0 et r_6= 0

λ=7 on aura la coupe : e_3 - 5/22 e_1 – 7/22 e_2= -1/2 et r_7= 11/12

λ=8 on aura la coupe : e_3 - 12/22 e_1 - 8/22 e_2= 0 et r_8= 0

λ=9 on aura la coupe : e_3 - 19/22 e_1 – 9/22 e_2= -1/2 et r_9= 11/28

On notera qu'il est inutile de construire les coupes dont λ est paire car son r_i correspondant est forcément nul.

λ=11 on aura la coupe : e_3 - 11/22 e_1 – 11/22 e_2= -1/2 et r_9= ½

λ=13 on aura la coupe : e_3 - 3/22 e_1 – 13/22 e_2= -1/2 et r_9=11/16

λ=15 on aura la coupe : e_3 - 17/22 e_1 – 15/22 e_2= -1/2 et r_9= 11/52

λ=17 on aura la coupe : e_3 - 9/22 e_1 – 17/22 e_2= -1/2 et r_9= 11/26

λ=19 on aura la coupe : e_3 - 10/22 e_1 – 19/22 e_2= -1/2 et r_9= 11/29

λ=21 on aura la coupe : e_3 - 15/22 e_1 – 21/22 e_2= -1/2 et r_9= 11/36

La meilleure coupe de Gomory correspond à la valeur de λ= 1.

Remarque : Il existe d'autres approches de résolution d'un PLNE comme le **Branch and Cut** qui est une méthode d'optimisation combinatoire pour résoudre des problèmes d'optimisation linéaire en nombres entiers. Cette méthode est une hydridation entre l'approche de Branch and Bound et la méthode des coupes de Gomory.

Le principe est de résoudre le programme linéaire associé (Q) à l'aide de l'algorithme du simplexe. Lorsqu'une solution optimale est trouvée, et que l'une des variables n'est pas entière, on utilise la méthode des coupes de Gomory pour trouver une contrainte linéaire satisfaite par toutes les valeurs entières de la solution mais violée par la valeur fractionnaire. Si une telle contrainte est trouvée, alors elle est ajoutée au programme linéaire de sorte que la résolution de ce programme donne une solution avec moins de valeurs non entières. On répète ce procédé jusqu'à ce qu'une solution entière soit trouvée (qui est alors optimale) ou jusqu'à ce qu'aucune coupe ne puisse être trouvée.

À ce moment, la partie séparation et évaluation de l'algorithme commence. Le problème est scindé en deux sous-problèmes, l'un en rajoutant la contrainte que la variable est supérieure ou égale à la partie entière par excès de la solution intermédiaire, et l'autre en rajoutant la contrainte que la variable est inférieure ou égale à sa partie entière usuelle (par défaut). Ces

deux nouveaux programmes linéaires sont résolus avec l'algorithme du simplexe et on itère la procédure présentée précédemment.