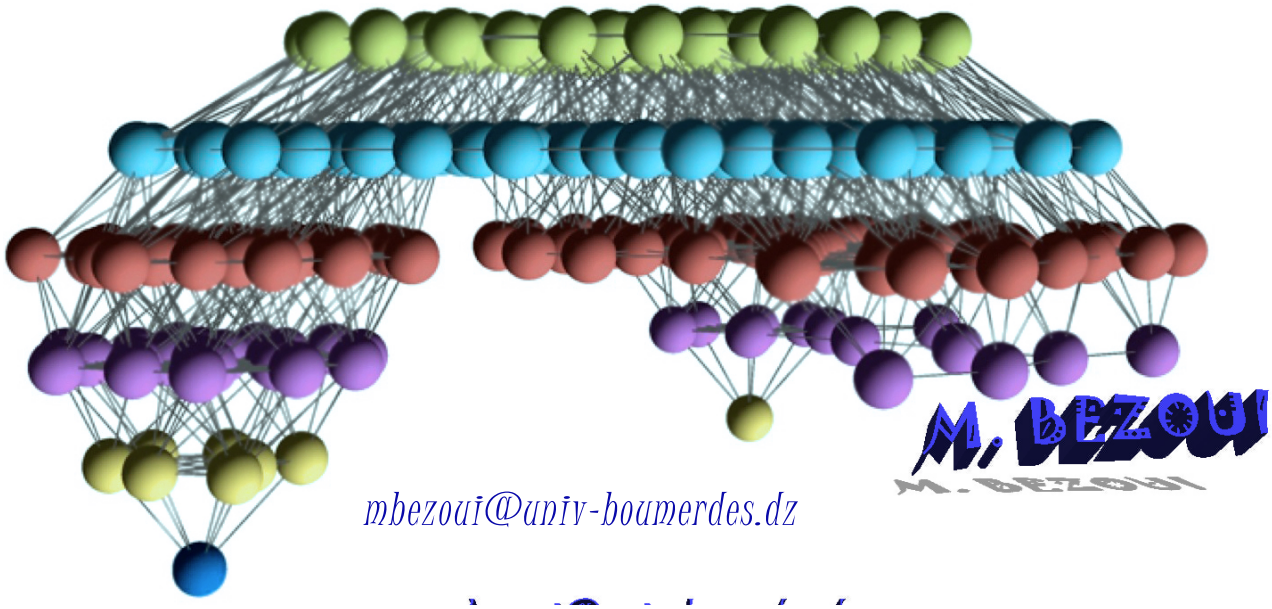
algorithmAlgorithme

OPTIMISATION COMBINATOIRE



mbezoui@univ-boumerdes.dz

Madani BEZOUI

mbezoui@univ-boumerdes.dz

https://sites.google.com/site/profbezoui/

Université de Boumerdes
University of Boumerdes

Année Universitaire 2015-2016

Table des matières

Première partie

Résolution de problèmes de programmation linéaire en nombres entiers PLNE

1. Généralités sur les PLNE

La programmation en nombres entiers concerne les programmes d'optimisation sous contraintes pour lesquels les variables doivent prendre des valeurs entières. Elle est un domaine très riche de la programmation mathématique.Les recherches dans ce domaine sont nombreuses et les commencements de vraie étaient avec **Gomory** en 1958.
Dans beaucoup de problèmes d'optimisation une solution à valeurs entières est exigée.par exemple dans le problème de la production,on l'on cherche le nombre optimal de pièces à fabriquer où ce nombre est pour des raisons pratiques,souvent entier.

1.1 Forme générale d'un programme linéaire en nombres entiers "PLNE"

La forme générale d'un programme linéaire en nombres entiers se présente comme suit :

$$\begin{cases} \max ou \min(Z) = \Sigma_{j=1}^{n} c_j x_j. \\ \forall i = \overline{1,m} : \Sigma_{j=1}^{n} a_{ij} x_j \leq, = ou \geq b_i. \\ \forall j = \overline{1,n} : x_j \geq 0. \\ x_j \quad entier, j = \overline{1,k} \quad (k \leq n) \end{cases}$$

Remarque 1.

Dans le cas d'un programme linéaire avec des variables binaires,on aura,à la suite des contraintes fonctionnelles.
$x_j = 0 \; ou \; 1, j = \overline{1,n}$

Relaxation

soit la forme générale du PLNE suivante :

$$(PLNE)\begin{cases} \max(\min) Z = CX \\ s.c \\ AX(\geq,\leq,=)B \\ \forall x_j \in X, \qquad x_j \geq 0 \qquad (j=\overline{1,n}) \\ x_j \in Z^+ (j=\overline{1,k}) \qquad avec \qquad k \leq n. \end{cases}$$

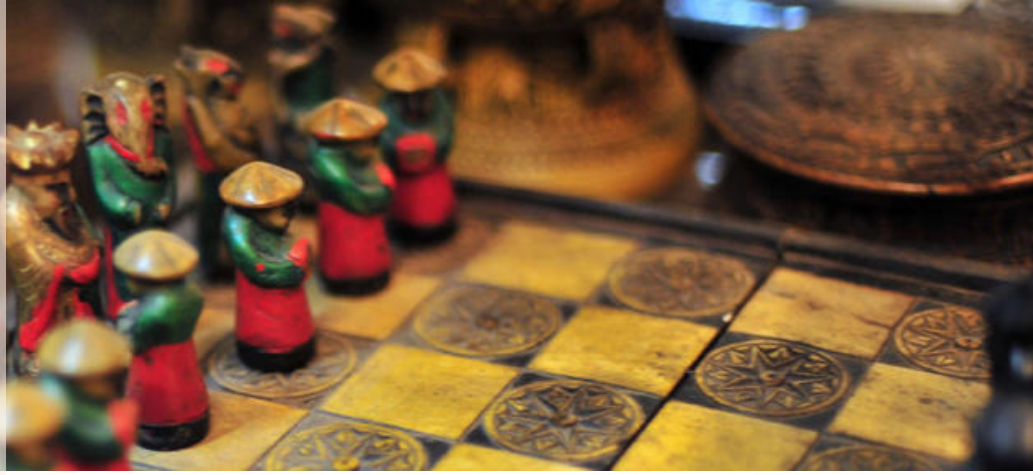
Notez bien que si on remplace les derniers restrictions par $x_j \geq 0$ pour $j=\overline{1,k}$ avec $(k \leq n)$ on aura un programme linéaire qui sera une relaxation du problème original .
se fait est très important,ça implique qu'on peut résoudre la relaxation par des méthodes qu'on a déjà considérés.

Définition 1.

La relaxation linéaire (ou continue)R_p de P est obtenue par relâchement (enlèvement) des contraintes d'intégralité.

Définition 2.

Soit $F(P)$ la région des solutions possibles de P :
on a $F(P) \subseteq F(R_p)$.



2. Méthode de séparation évaluation

L'algorithme de séparation et évaluation, plus connu sous son appellation anglaise Branch and Bound (BB) (Land et Doig 1960), repose sur une méthode arborescente de recherche d'une solution optimale par séparations et évaluations, en représentant les états solutions par un arbre d'états, avec des noeuds, et des feuilles.

Le branch-and-bound est basé sur trois axes principaux :

— L'évaluation,
— La séparation
— La stratégie de parcours

2.1 Evaluation

L'évaluation permet de réduire l'espace de recherche en éliminant quelques sous ensembles qui ne contiennent pas la solution optimale.

L'objectif est d'essayer d'évaluer l'intérêt de l'exploration d'un sous-ensemble de l'arborescence. Le branch-and-bound utilise une élimination de branches dans l'arborescence de recherche de la manière suivante : la recherche d'une solution de coût minimal, consiste à mémoriser la solution de plus bas coût rencontré pendant l'exploration, et à comparer le coût de chaque noeud parcouru à celui de la meilleure solution. Si le coût du noeud considéré est supérieur au meilleur coût, on arrête l'exploration de la branche et toutes les solutions de cette branche seront nécessairement de coût plus élevé que la meilleure solution déjà trouvée.

2.2 La séparation

La séparation consiste à diviser le problème en sous-problèmes. Ainsi, en résolvant tous les sous-problèmes et en gardant la meilleure solution trouvée, on est assuré d'avoir résolu le problème initial. Cela revient à construire un arbre permettant d'énumérer toutes les solutions. L'ensemble de neouds de l'arbre qu'il reste encore à parcourir comme étant susceptibles de contenir une solution optimale, c'est-à-dire encore à diviser, est appelé ensemble des neouds actifs.

2.3 La stratégie de parcours

2.3.1 La largeur d'abord

Cette stratégie favorise les sommets les plus proches de la racine en faisant moins de séparations du problème initial. Elle est moins efficace que les deux autres stratégies présentées,

2.3.2 La profondeur d'abord

Cette stratégie avantage les sommets les plus éloignés de la racine (de profondeur la plus élevée) en appliquant plus de séparations au problème initial. Cette voie mène rapidement à une solution optimale en économisant la mémoire.

2.3.3 Le meilleur d'abord

Cette stratégie consiste à explorer des sous problèmes possédant la meilleure borne. Elle permet aussi d'éviter l'exploration de tous les sous-problèmes qui possèdent une mauvaise évaluation par rapport à la valeur optimale.

Algorithme 1 : Algorithme de Branch and Bound

$k \leftarrow 0$;
Résoudre le problème relaxé P^k;
Soit x^k la solution trouvée ; Soit $Z^k = c^T * x$: la valeur de la fonction objectif en ce point;
si $x^0 \in Z^n$ **alors**
    STOP;
    tant que *Il existe une branche non exploré* **faire**
        Choisir la stratégie d'exploration;
        ;
        Procédure de séparation :;
        Choisir une composante $x_i \in x^0 Z$;
        il est d'usage de choisir la composante de x^0 contenant la plus grande partie fractionnaire;
        $k \leftarrow k+1$; Ajouter la contrainte $x_i \leq \lfloor x_i \rfloor$ au problème P^k;
        $k \leftarrow k+1$; Ajouter la contrainte $x_i \geq \lceil x_i \rceil$ au problème P^kv;
    fin
fin

Exemple d'application "Branch and Bound"

soit le problème initial suivant :

$$(P)\begin{cases} \max(Z) = 2x_1 + 3x_2 \\ s.c \\ 2x_1 + x_2 \leq 10 \\ x_1 + 4x_2 \leq 20 \\ x_i \in Z^+; (i = \overline{1,2}) \end{cases}$$

et soit (Pr) le problème relaxé(continu) :

$$(Pr)\begin{cases} \max(Z) = 2x_1 + 3x_2 \\ s.c \\ 2x_1 + x_2 \leq 10 \\ x_1 + 4x_2 \leq 20 \\ x_i \geq 0; (i = \overline{1,2}) \end{cases}$$

Solution de base

	x_1	x_2	x_3	x_4	$\overline{B}$
x_3	2	1	1	0	10
x_4	1	4	0	1	20
-Z	2	3	0	0	0

En résolvant le programme linéaire avec la méthode du simplexe, on avait obtenu le tableau optimal suivant :
Tableau optimal au (Pr)

	x_1	x_2	x_3	x_4	$\overline{B}$
x_1	1	0	4/7	-1/7	20/7
x_2	0	1	-1/7	2/7	30/7
-Z	0	0	-5/7	-4/7	-130/7

La solution optimal est : x_1^*=20/7=2.85,x_2^*=30/7,Z^*=130/7
cette solution n'est pas entière donc on applique Branch and Bound avec $2 \leq x_1^* = 2.85 \leq 3$
La séparation
En intégrant la contrainte $x_1 \leq 2$ à la solution optimale précédente nous obtenons le sous problème 1 (SP_1)

$$(SP_1)\begin{cases} \max(Z) = 2x_1 + 3x_2 \\ s.c \\ 2x_1 + x_2 \leq 10 \\ x_1 + 4x_2 \leq 20 \\ x_1 \leq 2 \\ x_i \geq 0; (i = \overline{1,2}) \end{cases}$$

En intégrant la contrainte $x_1 \geq 3$ à la solution optimale précédente nous obtenons le sous problème 2 (SP_2)

$$(SP_2)\begin{cases} \max(Z) = 2x_1 + 3x_2 \\ s.c \\ 2x_1 + x_2 \leq 10 \\ x_1 + 4x_2 \leq 20 \\ x_1 \geq 3 \\ x_i \geq 0; (i = \overline{1,2}) \end{cases}$$

L'évaluation :
• Résolvons le sous problème (SP_1).
On doit ajouter la contrainte $x_1 \leq 2$ au (Pr),sous forme d'équation ,on a $x_1 + x_5$=2
On obtient alors le tableau suivant en ajoutant cette équation (contrainte) au tableau précédent.

	x_1	x_2	x_3	x_4	x_5	$\overline{B}$
x_1	1	0	4/7	-1/7	0	20/7
x_2	0	1	-1/7	2/7	0	30/7
x_5	1	0	0	0	1	2
-Z	0	0	-5/7	-4/7	0	-130/7

On doit toute fois modifier ce tableau puisque x_1 est une variable dans la base et il doit lui correspondre un vecteur unitaire dans le tableau.
Multipliant la 1^{re} ligne par (-1) et additionnant à la 3^{me} ligne, on obtient le tableau suivant :

	x_1	x_2	x_3	x_4	x_5	$\overline{B}$
x_1	1	0	4/7	-1/7	0	20/7
x_2	0	1	-1/7	2/7	0	30/7
x_5	0	0	-4/7	1/7	1	-6/7
-Z	0	0	-5/7	-4/7	0	-130/7

On a (x_{B_3}=-6/7 < 0) alors on applique l'algorithme dual du simplexe,on trouve :
variable sortant :x_5 (x_{B_3}=-6/7 < 0)
variable entrante :x_3 $[\min -Z_j/\mu_{3j}, \mu_{3j} < 0]$=[(-5/7)/(-4/7)]=5/4
le pivot est -4/7(μ_{33}) $= -4/7$
On obtient le tableau optimal suivant effectuant l'opération de pivotement.

	x_1	x_2	x_3	x_4	x_5	$\overline{B}$
x_1	1	0	0	0	1	2
x_2	0	1	0	1/4	-1/4	63/14
x_3	0	0	1	-1/4	-7/4	3/2
-Z	0	0	0	-3/4	-5/4	-245/14

La solution optimale au sous problème (SP_1) est :
$x_1 = 2$, x_2=63/14=4+1/2, Z_1=17+1/2

•On doit maintenant procéder de la même façon pour déterminer la solution optimale du sous problème (SP_2).
Il faut ajouter la contrainte $x_1 \geq 3 \Leftrightarrow$ -$x_1 \leq -3$ et sous forme d'équation on a : -x_1+x_5=-3.
Ajoutons cette dernière équation au tableau optimal du (Pr), on obtient :

	x_1	x_2	x_3	x_4	x_5	$\overline{B}$
x_1	1	0	4/7	-1/7	0	20/7
x_2	0	1	-1/7	2/7	0	30/7
x_5	-1	0	0	0	1	-3
-Z	0	0	-5/7	-4/7	0	-130/7

Additionnant la 1^{re} ligne à la 3^{me} (pour obtenir le vecteur unité sous x_1), on obtient le tableau suivant :

	x_1	x_2	x_3	x_4	x_5	$\overline{B}$
x_1	1	0	4/7	-1/7	0	20/7
x_2	0	1	-1/7	2/7	0	30/7
x_5	0	0	4/7	-1/7	1	-1/7
-Z	0	0	-5/7	-4/7	0	-130/7

Appliquant l'algorithme dual du simplexe , on obtient :
variable sortante : x_5
variable entrante : x_4
pivot=-1/7
Le tableau optimal est alors le suivant :

	x_1	x_2	x_3	x_4	x_5	$\overline{B}$
x_1	1	0	0	0	-1	3
x_2	0	1	1	0	2	4
x_4	0	0	-4	1	-7	1
-Z	0	0	-3	0	-4	-18

La solution optimale au sous problème (SP_2) est :
x_1=3, x_2=4, Z_2= 18
• La solution optimale entière Z_2 est la solution recherchée car Z_2=18 $>$ Z_1=17+1/2 ;
le sous problème (SP_1) ne sera pas subdivisé en deux nouveaux sous problèmes.

3. Méthodes de coupes

3.1 Coupe de Gomory

Soit (P) un programme linéaire en nombres entiers écrit sous forme canonique par rapport à une base J.

$$(P)\begin{cases} \max(Z) = CX \\ AX = b \\ x_j \in N \qquad j = \overline{1,n} \end{cases}$$

on sait que si les trois conditions suivantes :

$$C \leq 0 \tag{3.1}$$

$$b \geq 0 \tag{3.2}$$

$$b \qquad entier \tag{3.3}$$

sont satisfaites alors $\bar{x}$, solution de base relative à la base J est solution optimale de (P). si (3.1) et (3.2) sont vérifiés $\bar{x}$ est solution optimale du programme linéaire (P') obtenu à partir de (P) en relâchant les contraintes d'intégrité sur les variables.

Définition 3.

Etant donné le programme linéaire en nombres entiers (P), on dit que l'inéquation ax$\leq \alpha$ est valide si elle est satisfaite par toute solution réalisable de (P), une coupe est une inéquation valide qui n'est pas satisfaite pour tout point de D' domaine des solutions réalisables de la relaxation (P') de (P).

Définition 4.

Si α est un scalaire quelconque, on désigne par :

— $\lfloor \alpha \rfloor$ le plus grand entier inferieur à α

— $\lceil \alpha \rceil$ le plus petit entier superieur à α

— $\langle \alpha \rangle$ =α- $\lfloor \alpha \rfloor$

$\langle \alpha \rangle$ est appelée la partie fractionnaire de α et $\lfloor \alpha \rfloor$ sa partie entière inferieure

Pour toute valeur du scalaire α,l'inéquation :

$$\sum_{j \notin J} (\lfloor \alpha \rfloor A_i^j - \lfloor \alpha A_i^j \rfloor) x_j \geq \lfloor \alpha \rfloor b_i - \lfloor \alpha b_i \rfloor \tag{3.4}$$

est valide, pour certaines valeurs de α (et en particulier α=1), c'est une coupe.

Multiplions chaque terme de la i^{eme} contrainte de (P) par le scalaire α. il vient :

$$\alpha x_r + \sum_{j \notin J} \alpha A_i^j x_j = \alpha b_i \qquad (r = col(i))$$

et décomposons chaque coéfficient en la somme de sa partie entière et de sa partie fractionnaire :

$$\lfloor \alpha \rfloor x_r + \sum_{j \notin J} \lfloor \alpha A_i^j \rfloor x_j + \langle \alpha \rangle x_r + \sum_{j \notin J} \langle \alpha A_i^j \rangle x_j = \lfloor \alpha b_i \rfloor + \langle \alpha b_i \rangle$$

compte tenu de ce que,par définition : $0 \leq \langle \alpha \rangle < 1$ et $x_j \geq 0$ ceci implique :

$$\lfloor \alpha \rfloor x_r + \sum_{j \notin J} \lfloor \alpha A_i^j \rfloor x_j \leq \lfloor \alpha b_i \rfloor + \langle \alpha b_i \rangle$$

Or le membre de gauche est entier et $\langle \alpha b_i \rangle < 1$, par conséquent cette dernière inéquation implique :

$$\lfloor \alpha \rfloor x_r + \sum_{j \notin J} \lfloor \alpha A_i^j \rfloor x_j \leq \lfloor \alpha b_i \rfloor$$

et en soustrayant cette dernière inéquation de la i^{eme} contrainte de (P) multipliée au préalable par $\lfloor \alpha \rfloor$, on obtient (3.4) qui est donc bien une inéquation valide. Supposons les inéquations (3.1) et (3.2) soient vérifiées mais que, pour certain indice i, b_i ne soit pas entier(i.e.$\langle b_i \rangle > 0$). posons α=1 dans (3.4), on a :

$$\sum_{j \notin J} \langle A_i^j \rangle x_j \geq \langle b_i \rangle \tag{3.5}$$

qui est une coupe puisque (3.5) n'est pas satisfait pour la solution réalisable de base de (P').

3.2 Principe des méthodes de coupe

- Introduire de nouvelles contraintes linéaires au problème pour réduire le domaine réalisable du problème relaxé sans pour autant éliminer de points du domaine réalisable du problème en nombres entiers.
- La procédure consiste à résoudre une suite de problèmes relaxés jusqu'a ce qu'une solution optimale en nombre entier soit obtenue.
- Un problème de la suite est obtenu du précédent en lui ajoutant une contrainte linéaire (coupe) supplémentaire.

et voici l'algorithme qu'on suit :

Algorithme 2 : Coupes de Gomory

début;
(0) A et b étant entiers, résoudre le problème par une méthode de programmation linéaire en oubliant les contraintes d'intégrité. Aller à**(1)**;
(1);
si *la solution optimale est entière* **alors**
    Terminer, et cette solution est aussi la solution optimale du problème posé;
fin
sinon
    Aller à **(2)**;
fin
(2) Déterminer $\langle b_r \rangle = \max \langle b_i \rangle$, ajouter au système d'équations obtenu à la dernière itération la coupe;

$$-\sum_{j=1}^{p} \langle a_{rj} \rangle x_j + x_{p+1} = -\langle b_r \rangle$$

où p est le nombre de variables avant d'ajouter cette coupe;
Aller à**(3)**;
(3) Appliquer l'algorithme dual du simplexe à partir du tableau optimal précédent et retourner à **(1)**;