1 Introduction

La programmation linéaire en 0-1 ou programmation linéaire à variables booléennes joue un rôle important dans la résolution de problèmes pratiques dans divers domaines de la vie courante. Les programmes en (0,1) constituent un cas très important de la programmation en nombres entiers. La structure particulière de ces programmes a amené les scientifiques à développer des techniques spécifiques pour leurs résolutions.

Considérons le problème suivant :

Min Z = f (x_1…..x_n).

$g_i(x_1 \ldots x_n) \leq b$ avec $i \in I=\{1, \ldots, m\}$

$x_j = (1, 0)$ pour $j \in J_1 \in \{1, \ldots, p\}$

$x_j \geq 0$ pour $j \in J- J_1$

Les fonctions f et g_i sont linéaires pour toutes les solutions réalisables de ce programme. Lorsque $J= J_1$, ce programme est 0-1 pur, sinon il sera mixte.

Tout programme linéaire à variables entières peut être ramené à un programme linéaire bivalent. En effet, soit x_k une variable entière tel que $x_k \leq u_k$ où u_k est une borne supérieure alors nous pouvons toujours réécrire $x_k= \sum_{j=1}^{k} y_j$ avec $y_j = (0, 1)$ pour j=1, …, k.

Une autre transformation consiste à réécrire $x_k = \sum_{j=1}^{k} 2^j y_j$ avec $y_j = (0, 1)$ pour j=1, …, k et k est le plus petit entier tel que $2^{(k+1)} \geq u_k +1$. A titre d'exemple si $x_k \leq 5$, nous pouvons réécrire $x_k= y_0+ 2\, y_1 + 2^2\, y_2$ et k=2 est le plus petit entier vérifiant $2^{2+1}= 8 \geq 6 = 5+1$.

L'inconvénient de ces transformations réside dans le faite que le nombre de variable augmente.

2. Exemples de problème en 0.1

2.1. Le problème de recouvrement et de partionnement

Supposons que l'on doit réaliser un nombre de « tâches » s_1…, s_m et pour les effectuer nous disposons d'un nombre de moyens E_1…, E_n. Un moyen E_j réalise une ou plusieurs tâches, à un cout $c_j = c_j(Ej)$. Le problème de recouvrement consiste à réaliser toutes les tâches par les moyens au moindre coût.

Dans le cas où les moyens sont exclusifs, c'est-à-dire qu'une tâche est réalisée par un et un seul moyen, le problème devient alors une partition de l'ensemble des tâches par les moyens et est appelé problème de partitionnement.

Posons: S={s_1…, s_m} l'ensemble des tâches et ζ ={E_1, …, E_m} l'ensemble des moyens où Ej $\subset$ S (Ej $\neq$ 0) et $\bigcup_{j=1}^{n} E_j$ = S. H=(S, ζ) est appelé **hypergraphe**.

Le problème de recouvrement consiste à déterminer une sous famille $F^* \subset \zeta$ qui recouvre S et qui minimise le coût total c(F^*) = $\sum_{E_j \in F^*} c_j(E_j)$. En d'autres termes,

$$\begin{cases} Chercher\ F* \subset \zeta \\ \bigcup_{j=1}^{n} E_j = S \\ c(F*) = \text{Min } c(F)\ \forall F \subset \zeta \end{cases}$$

Dans le cas où F* est une partition de S alors ce problème devient un problème de partionnement.

Soit A= (a_{ij}) la matière d'incidence tâches-moyens de l'hypergraphe H définie comme suit :

$$a_{ij} = \begin{cases} 1 \ si\ s_i \in E_j \\ 0 \quad sinon \end{cases}$$

Dans la matrice A, les lignes désignent les différentes tâches et les colonnes les différents moyens. Soit X= (x_1, x_2, …, x_n)^t le vecteur caractéristique représentant les variables de décision de la sous famille $F \subset \zeta$ où

$$x_j = \begin{cases} 1 \ si\ E_j \in F \\ 0 \quad sinon \end{cases}$$

Le problème de recouvrement à coût minimum se modélise comme suit :

$$\begin{cases} Min\ Z = \sum_{j=1}^{n} c_j x_j \\ \sum_{j=1}^{n} a_{ij} x_j \geq 1,\ \forall i \in \{1, \dots, m\} \\ x_j = (0,1),\ \forall j \in \{1, \dots, n\} \end{cases}$$

La modélisation du problème de partionnement est presque identique à celle du problème de recouvrement à l'exception du système de contraintes qui devient :

$$\sum_{j=1}^{n} a_{ij} x_j = 1, \qquad \forall i \in \{1, \dots, m\}$$

Plusieurs situations pratiques se ramènent à cette classe de problèmes :

- *Les problèmes de tournées :*

Etant donné un dépôt où nous nous proposons de déterminer un ensemble de tournées permettant la livraison d'un produit à un certain nombre de clients. Dans le cas où nous imposons que chaque client est approvisionné par une et une seule tournée, ce problème devient un problème de partionnement des clients au moindre coût. En effet, soit S l'ensemble des clients (tâches) et E l'ensemble des tournées possibles (moyens). On associe à chaque tournée E_j un coût c_j. Il s'agit de déterminer un sous ensemble de tournées de manière qui minimise le coût total et qui recouvre (ou à partitionne) l'ensemble des clients en respectant différentes contraintes de tonnage, de volume, de temps etc.

Parmi les exemples des problèmes de tournées nous citons :

- Problème de ramassage scolaire.
- Problème d'organisation de circuit de transport.
- Problème de distribution à partir d'un ou plusieurs dépôts.
- Etc.

- *Problème de localisation*

Etant donné un service qui doit être effectué en un nombre de points (par exemple, localiser des stations d'émetteurs d'un réseau de téléphonie mobile, les stations de secours d'urgence, les centres de traitement de l'information, etc.). Il s'agit de sélectionner un nombre d'emplacements parmi un ensemble d'emplacements possibles (moyens) de façon à desservir la totalité des points.

Soient S l'ensemble des tâches à effectuer sur un nombre de points et E l'ensemble des emplacements possibles. Un emplacement E_j correspond à un sous ensemble de tâches qu'il peut couvrir. Si nous affectons à chaque emplacement E_j un cout c_j, ce problème devient un problème de recouvrement des points à moindre coût.

2.2 Le problème d'ordonnancement d'ateliers

Considérons le problème de la réalisation de n différentes activités sur une seule machine en un temps minimum possible. Chaque produit passe par une séquence de différentes opérations dont l'ordre d'exécution doit être respecter. De plus, ces produits doivent satisfaire une date de livraison. Nous rencontrons souvent trois types de contraintes dans ce type de situation :

- Contrainte d'ordre
- Contrainte de non interférence (Deux opérations ne peuvent être effectuées simultanément)
- Contrainte sur le délai de livraison

Si nous dénotons par x_j la date de début de l'opération j et a_j le temps nécessaire pour la réalisation de l'opération j, alors la première contrainte peut s'exprimer : $x_i + a_i \leq x_j$. La seconde contrainte qui traduit le fait que les opérations i et j n'occupent pas la machine simultanément peut s'exprimer comme suit : $x_i - x_j \geq a_i$ ou $x_j - x_i \geq a_i$. La présence de ce type de contrainte pose un problème car le domaine des solutions n'est pas convexe. Pour surmonter cette difficulté, nous introduisons une variable binaire y_{ij} définie comme suit :

$$y_{ij} = \begin{cases} 0 \; si \; l'opération \; j \; précède \; i \\ 1 \; si \; l'opération \; i \; précède \; j \end{cases}$$

Nous pouvons ré exprimer les deux contraintes précédentes comme suit :

$M\, y_{ij} + (x_i - x_j) \geq a_j$ et $M(1 - y_{ij}) + (x_j - x_i) \geq a_i$ avec M un nombre suffisamment grand.

Si t est le temps global pour l'achèvement de toutes les opérations, ce problème simplifié d'ordonnancement d'ateliers peut être modélisé comme suit :

$$\begin{cases} Min\, Z = t \\ M\, y_{ij} + (x_i - x_j) \geq a_j \\ M(1 - y_{ij}) + (x_j - x_i) \geq a_i \\ x_i + a_i \geq x_j \\ x_j + a_j \leq d_j \\ y_{ij} = (1,\ 0) \\ x_j \geq 0 \end{cases} \qquad \forall\, i, j \in \{1, \ldots, n\}$$

Nous remarquons que ce programme est mixte où une partie des variables est astreinte à être booléenne et une autre positive ou nulle (souvent fractionnelle).

3 Méthodes de résolution

En général, l'espace des solutions d'un programme linéaire à variables booléennes est supposé fini. Une méthode naïve consiste à énumérer selon un ordre lexicographique toutes les solutions possibles puis celles qui sont réalisables et de choisir enfin parmi elles celle qui est meilleure. Cette approche est obsolète car le nombre de variables et de contraintes peuvent être tellement grand qu'il faudrait plus d'un siècle pour résoudre un tel programme. D'autre

part, nous pouvons substituer les contraintes de bivalence des variables par des contraintes continues ($0 \leq xj \leq 1$) alors nous avons les résultats suivants :

i) L'espace des solutions du domaine continue ne contient aucune solution réalisable à l'intérieur de ce domaine (évident)

ii) La solution réalisable optimale est atteinte en un point extrême de l'espace convexe continue (résultat de la théorie sur la programmation linéaire).

En effet, i) provient du fait qu'une solution réalisable doit obéir à la restriction $xj = 0$ ou $xj=1$. Comme ces contraintes constituent des hyperplans délimitant les frontières du domaine continu, il est donc impossible qu'une telle solution soit à l'intérieur.

Ces résultats suggèrent que le problème à variables booléennes peut être traité comme un programme continu car le fait que la solution optimale est atteinte en un point extrême, il est donc possible d'utiliser une méthode similaire à la méthode du simplexe pour sa résolution. Cependant, la difficulté réside dans le fait que la solution optimale du problème continue n'est pas forcément entière.

Nous allons adapter l'approche de Branch and Bound pour la résolution de ce type de problème, à savoir, l'algorithme de Ballas puis nous présenterons une amélioration de cette technique pour traiter le problème de l'explosion combinatoire des nœuds de l'arborescence des solutions.

3.1) Algorithme de Ballas

L'algorithme de Ballas explore l'arborescence des solutions où chaque sommet donne lieu à deux branches ; sur une branche la variable de séparation est nulle et sur l'autre la variable est égal à 1. La racine correspond à toutes les variables nulles. La solution initiale n'est pas réalisable car si non elle serait optimale. Nous améliorons l'infaisabilité de la solution à une itération donnée en astreignant un sous ensemble de variables adéquates à prendre la valeur 1. Une branche est prospectée jusqu'à obtenir une feuille dont le sommet correspond à une solution réalisable, ou l'infaisabilité de la solution ne peut être améliorée ou enfin la valeur de la fonction objectif est plus grande que la borne supérieur déjà trouvée (pour un problème de minimisation). Cette technique introduit un ensemble de règle pour une exploration intelligente de l'arborescence des solutions.

Considérons le programme linéaire à variables booléennes (P) suivant :

$$Min\ Z = \sum_{j=1}^{n} c_j x_j \quad avec\ c_j \geq 0$$

$$\sum_{j=1}^{n} a_{ij} x_j + S_i = b_i$$

$$x_j = (0,1), \qquad \forall j \in \{1, \dots, n\}$$

$$S_i \geq 0 \ \forall i \in \{1, \dots, m\}$$

Si un programme n'est pas sous cette forme, nous devons effectuer des transformations adéquates pour obtenir la forme donnée dans le programme (P) précédent.

Si un paramètre $c_j < 0$ alors il faut transformer sa variable correspondante $x'_j = (1 - x_j)$.

Dans toute la suite, nous adopterons les notations suivantes ;

+i : La variable x_i est fixée à 1 et –i si elle prend la valeur 0

J_t : La solution partielle obtenue au nœud t, si $J_t = \varnothing$ alors toutes les variables sont libres

N_t : L'ensemble des variables libres au nœud t

Z^t : La valeur de z à l'itération t.

Z^* : La meilleure solution trouvée (au départ $Z^* = \infty$)

Si^t : La valeur de la variable d'écart à l'étape t.

Nous pouvons exprimer la ligne i au nœud t comme suit :

$$\sum_{j=1}^{n} a_{ij}^t \ x_{ij}^t + S_i^t = b_i^t$$

L'infaisabilité de la variable x_j au nœud t est définie par :

$$I_j^t = \sum_{i=1}^{m} Min\ (0, S_i^t - a_{ij}^t)$$

L'algorithme procède par élimination des branches inutiles par une série de tests. Le principe de cette procédure d'élimination est de ramener une variable libre x_j à 1 que si elle améliore son infaisabilité.

A chaque nœud de l'algorithme, nous effectuons les tests suivants dans l'ordre présenté:

Test 1 : Toute variable libre x_r vérifiant $a_{ir} \geq 0 \ \forall$ i ayant $S_i < 0$ doit être ignorée car la présence d'une telle variable dont les coefficients de toutes les variables libres sont positifs ou nuls ne permet pas d'améliorer son infaisabilité et pire encore elle l'aggraverait. Mettre à jour Nt

Test 2 : Toute variable libre x_r vérifiant $c_r + Z^t \geq Z^*$ ne peut améliorer la valeur de la fonction objectif. La variable est ignorée. Mettre à jour Nt.

Test 3: S'il existe une variable d'écart $S_i^t < 0$ à vérifiant $\sum_{j\epsilon Nt} Min\left(0,\ a_{ij}^t\right) > S_i^t$ alors aucune variable libre ne peut améliorer l'infaisabilité et aucun branchement à partir de ce nœud ne peut mener vers une solution réalisable. On supprime J_t.

Test 4: Si Nt ≠∅, nous sélectionnons la variable x_k vérifiant $I_k^t = Max\ (I_j^t,\ j \in N_t)$

Deux cas se présentent :

Cas 1 : Si $I_k^t = 0$ et $x_k=1$ alors $J_{t+1} = J_t \cup \{+ k\}$ correspond à une solution réalisable. Nous remplaçons Z* par $Z_{(t+1)}$ si $Z_{(t+1)} < Z^*$. On supprime $J_{(t+1)}$.

Cas 2 : Si $I_k^t < 0$, on applique les précédents tests sur $J_{(t+1)}$.

Lorsqu'un nœud est supprimé, nous faisons un retour arrière à un autre nœud non encore exploité. L'algorithme s'achève lorsque tous les nœuds sont coupés. Si Z* reste infinie alors le problème ne possède pas de solution réalisable.

Remarque :

Glover et Zionts ont montré qu'un ensemble N_t peut passer le test 3 même si un coût élevé de la solution réalisable en résulte, ils suggèrent que le test soit modifié comme suit :

$$D_t = \{j\epsilon\ N_t / \frac{c_j}{|a_{ij}|} \geq \frac{Z_{min} - Z_t}{|S_i^t|}, S_i^t < 0\ et\ a_{ij} < 0\}$$

Si $D_t = N_t$, nous supprimons J_t car chaque variable de N_t lorsqu'elle est ramenée à 1 va contribuer à l'augmentation de la fonction objectif par un ratio.

3.2 Exemple

Soit le programme linéaire à variables booléennes

$$\text{Max } Z = 3x_1+2x_2 - 5x_3 -2x_4 +3x_5$$
$$x_1+x_2 + x_3 +x_4+x_5 \leq 4$$
$$7x_1+ 3\ x_3 - 4x_4+ 3\ x_5 \leq 8$$
$$11\ x_1 – 6x_2 +3x_4 – 3x_5 \geq 3$$
$$x_j = (0,1)$$

Après transformation de la fonction objectif en un problème de minimisation à savoir : Max Z= - Min (-Z) qui est équivalent à Min (-Z). Pour avoir des coûts positifs ou nuls, nous faisons un changement de variable pour x_1, x_2 et x_5 en posant $x_j= 1- x_j$ pour j=1, 2 et 5, nous obtenons le programme suivant :

Min Z = $3x_1+2x_2 + 5x_3 +2x_4 +3x_5$

$- x_1 \quad - x_2 + x_3 \quad +2 x_4 - x_5 + S_1 \quad = 1$

$- 7x_1 \quad + 3 x_3 - 4x_4 - 3x_5 \quad + S_2 \quad = -2$

$11 x_1 – 6x_2 \quad -3x_4 – 3x_5 \quad + S_3 = -1$

$x_j = (0,1),\ j=1, \ldots, 5$

$S_i \geq 0,\ i= 1, 2, 3$

Les différentes itérations de l'algorithme sont résumées comme suit et l'arborescence deolutions est présentée dans la figure 4.1 suivante :

Itération 0: $J_0 = \emptyset$, $N_0 =\{1, 2, 3, 4, 5\}$, $Z^0= 0,\ \ Z^*=\infty$
$S_1^0 = 1, S_2^0 = -2,\ S_3^0 = -1$
Test 1: Nous supprimons x_3 et $N_0 =\{1, 2, 4, 5\}$
Test 2 : -
Test 3 : On garde $N_0 =\{1, 2, 4, 5\}$
Test 4 : Nous trouvons V_5= Max (V_j, j∈ N_0)= 0
La séparation se fera sur x_5

Itération 1: x_5= 1, $J_1 = \{+5\}$, $N_1 =\{1, 2, 3, 4\}$, Z^1= 3< Z*=∞
$S_1^1 = 2, S_2^1 = 1,\ S_3^1 = 2$
On pose Z*=3. On supprime J_1

Itération 2: x_5= 0, $J_2 = \{-5\}$, $N_2 =\{1, 2, 3, 4\}$, Z^2= 0, Z*=3
$S_1^2 = 1, S_2^2 = -2,\ S_3^2 = -1$
Test 1: Nous supprimons x_3 et $N_2 =\{1, 2, 4\}$
Test 2 : Nous supprimons x_1 et $N_2 =\{2, 4\}$
Test 3 : On garde $N_2 =\{ 2, 4\}$
Test 4 : Nous trouvons V_4= Max {V_j, j∈ N_0}= -1
La séparation se fera sur x_4

Itération 3: x_4= 1, $J_3 = \{-5, +4\}$, $N_3 =\{1, 2, 3\}$, Z^3= 2, Z*=3
$S_1^3 = -1, S_2^3 = 2,\ S_3^3 = 2$,
Test 1: Supprimons x_3 et $N_3 =\{1, 2\}$
Test 2 : Supprimons x_1 et x_2 , $N_3=\emptyset$, J_3 à supprimer
Nous faisons un retour arrière au nœud 4.

Itération 4: x_4= 0, $J_4 = \{-5, -4\}$, $N_2 =\{1, 2, 3\}$, Z^4= O, Z*=3
$S_1^4 = 1, S_2^4 = -2,\ S_3^4 = -1,$
Test 1: Nous supprimons x_3 et $N_2 =\{1, 2\}$
Test 2 : Nous supprimons x_1 et $N_2 =\{2\}$
Test 3 : Nous supprimons x_2. On supprime J_4.
Terminer, la solution optimale est atteinte au nœud 1.

La solution optimale du programme transformé est : $x_1=x_2=x_3=x_4=0$ et $x_5=1$. Il ne faut pas oublier de revenir aux variables d'origines à savoir: $x_1=1$ $x_2=1$ et $x_3=x_4=0$ et $x_5=0$ avec Z=5.

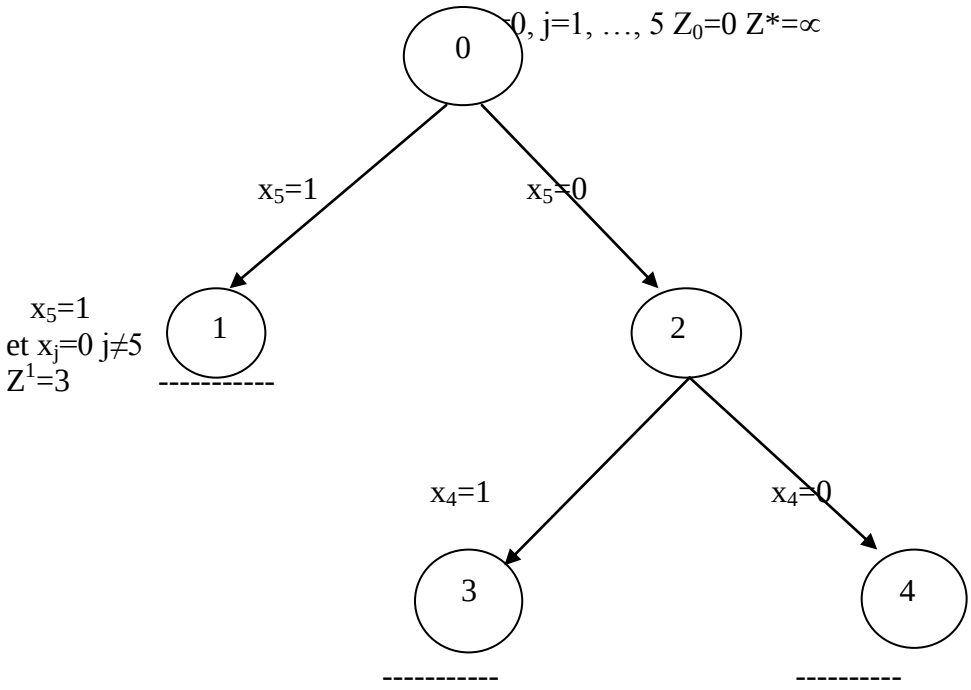

Figure 4.1 : Arborescence des solutions

4 Amélioration de Ballas

En programmation booléennes, il est utile de construire d'une contrainte cumulative (surrogate constraint, en Anglais) ou une combinaison de contrainte à partir de l'ensemble de contraintes du problème pour réduire l'explosion combinatoire des sommets de l'arborescence des solutions. En effet, la construction d'une telle contrainte permet de mieux nous renseigner sur les valeurs de certaines variables et de les astreindre. La difficulté est la construction d'une telle contrainte. Nous présenterons dans cette section la construction de la meilleure contrainte selon Glover et selon Geoffrion.

4.1 Construction d'une contrainte cumulative

Considérons le programme linéaire à variables Booléennes suivant :

Min Z = c x Min Z = c x

Ax ≤ b ou encore $\sum_{j=1}^{n} a_{ij} \leq b_i \;\; i = 1, \ldots, m$

x = (0, 1) x_j = (0, 1), j= 1, …, n

Soient J_t l'ensemble des indices des variables de la solution partielle à une itération donnée (au départ $J_t=\varnothing$) et N_t l'ensemble des indices des variables libres (au départ, $N_t=\{1, \ldots, n\}$).

Supposons qu'à une itération donnée, nous souhaitons remplacer l'ensemble de contraintes suivantes: $\sum_{j \in Nk} a_{ij}^k \; x_j \leq b_i^k$, i = 1, …, m (1) par une contrainte unique sous la forme :

$\sum_{i=1}^{m} \sum_{j=1}^{n} \mu_i \; a_{ij}^k \; x_j \leq \sum_{i=1}^{m} \mu_i \; b_i^k$ (2) où μ est un vecteur positif non négatif

Nous remarquons que (2) est plus faible que (1) car $D_k \subseteq D_k(x)$ où D_k et $D_k(x)$ représente les solutions réalisables respectivement de (1) et (2). Cependant, la contrainte (2) a l'avantage de contenir des informations différentes que les contraintes individuelles de (1).

Nous pouvons appliquer les différents tests de l'algorithme de Ballas sur la contrainte (2) et si J_k correspondant est vide alors nous coupons le nœud k car si $D_k(x) = \varnothing$ il s'ensuivra de même pour D_k. A titre d'exemple, considérons les deux contraintes suivantes :

$x_1 - x_2 \leq -1$

$- x_1 + 2\, x_2 \leq -1$

Chacune des deux contraintes prise séparément possède une solution réalisable mais en les combinant en prenant $\mu_1 = \mu_2 = 1$ qui donne la contrainte $x_2 \leq -2$ et ne possède aucune solution réalisable.

Glover a montré deux propriétés importantes que doit posséder la contrainte combinée à savoir :

i) Si x' est solution réalisable du problème d'origine (P) alors x' est aussi réalisable pour la contrainte (2)

ii) Si la contrainte (2) ne possède pas de solution réalisable, il en sera de même pour le problème d'origine (P).

En effet, si on a $A\, x' \leq b \Rightarrow \mu\, A\, x' \leq \mu\, b \;\; \forall\; \mu \geq 0$. Ces deux propriétés sont utiles du fait qu'elles montrent que la contrainte (2) peut être utilisée pour anticiper l'existence de solution réalisable.

4.2 Théorème de Geoffrion

i) La contrainte $\beta_i - \sum_{j \in Nt} a_j \; x_j \geq 0 \; (>0)$ où $x_j = (0, 1)$ pour j=1, …, n est dite bi variablement irréalisable si et seulement si :

$$Max_{\; i} \left\{ \beta_i - \sum_{j \in Nt} a_j \; x_j \;\; où \;\; x_j = (0,1)\; j = 1,...,n \right\} = \beta_i - \sum_{j} Min\; (0,\, a_{ij}) < 0 \; (\leq)$$

ii) Pour une solution réalisable de $\beta_i - \sum_{j \in Nt} a_j \; x_j \geq 0$ (>0), si $\beta_i - \sum_j Min\,(0, a_{ij}) - \left|a_{ij0}\right| < 0$ (≤)

alors : x_{j0}= pour a_{ij0}> et x_{j0}=1 si a_{ij0} <0

La démonstration est triviale. On remarquera que i) est identique au test 3. Pour illustrer ce théorème, considérons les contraintes a) $-5 - (-x_1 - 2\,x_2 + 3\,x_3) \geq 0$ et b) $-1 - (-2\,x_1 + x_2 + x_3 + x_4) \geq 0$.

La contrainte a) ne possède pas de solution binaire réalisable car -5 – (-1 -2) = - 2 < 0. Pour la contrainte b) on conclut que x_1= 1 car -1 – (-2) - |-2| = -1 < 0.

Ce théorème suggère que si i) est satisfaite pour la contrainte combinée (2) alors la solution partielle correspondante doit être coupée. De plus, ii) offre un moyen simple de fixer les valeurs de certaines variables libres à 1 ou 0.

Le problème crucial est de déterminer la meilleure contrainte combinée dans le sens où elle offre le maximum d'information à propos de l'optimalité et de la faisabilité de solution du problème relativement à la solution partielle à une itération donnée.

Selon Glover, la contrainte combinée « μ^1 (A x – b) ≤ 0 » est meilleure que « μ^2 (A x – b) ≤ 0 » relativement à la solution partielle J_t si :

$$\underset{x}{Min}\left\{ c\;x \; avec \; \mu^1\;(A\;x - b) \leq 0 \right\} \geq \underset{x}{Min}\left\{ c\;x \; avec \; \mu^2\;(A\;x - b) \leq 0 \right\}$$

Où la minimisation est faite sur les variables libres et les autres variables sont fixées par la solution partielle.

Geoffrion suggère en plus de la construction de la combinaison de contrainte de rajouter la contrainte $c\,x \leq Z_{Min}$.

Selon Geoffrion, la contrainte combinée « μ^1 (A x – b) + (Z_{Min} – cx) ≥ 0 » est meilleure que « μ^2 (A x – b) + (Z_{Min} – cx) ≥ 0 » relativement à la solution partielle J_t si :

$$\underset{x}{Max}\left\{ c\;x \; avec \; \mu^1\;(A\;x - b) + (Z_{\min} - c\;x) \right\} \leq \underset{x}{Max}\left\{ c\;x \; avec \; \mu^2\;(A\;x - b) + (Z_{\min} - c\;x) \right\}$$

Selon Geoffrion, la meilleure contrainte combinée est celle qui satisfait :

$$\underset{\mu}{Min}\;\underset{x}{Max}\left\{ \mu\;(A\;x - b) + (Z_{\min} - c\;x) \right\}$$

Nous pouvons transformer ce problème de Min-Max à un problème de la programmation linéaire. En effet,

$$\underset{x}{Max}\left\{ \sum \mu_i \left(b_i - \sum_{j\in Jt} a_{ij}\ x_j^* - \sum_{j\in Jt} a_{ij}\ x_j + \left(Z_{Min} - \sum_{j\in Jtt} cj\ x_j^* - \sum_{j\in Jt} c_j\ x_j \right) tel\ que\ \mu_i \geq 0 \right\}$$

$$= \sum_{i=\&}^{m} \mu_i\ S_i^t + Z_{Min} - Z^t + Max\left\{ \sum_{j\in Nt} (-1)\ (\sum_{i=1}^{m} a_{ij}\ \mu_i \quad + c_j)\ x_j\ avec\ j \in N_t \right\}$$

En utilisant les résultats de la dualité et en particulier le théorème des écarts complémentaires on obtient :

$$= \sum_{i=\&}^{m} \mu_i\ S_i^t + Z_{Min} - Z^t + Min\left\{ \sum_{j\in Nt} y_j\ tel\ que\ y_j \geq 0\ et\ y_j\ \geq -(\ \sum_{i=1}^{m} a_{ij}\ \mu_i + c_j\ ,\ j \in N_t \right\}$$

Nous remarquons que l'expression non linéaire des μ_i et x_j est transformée en une expression linéaire (Q) des μ_i et y_j et devient alors :

$$\text{Min } w^t = \sum_{i1i=1}^{m} S_i^t\ \mu_i + \sum_{j\in Nt} y_j + (Z_{Min} - Z^t)$$

$$-\sum_{i=1}^{m} a_{ij}\ \mu_i - y_j \leq c_j$$

$$\mu_i,\ y_j \geq 0\ i \in M\ et\ j \in N_t$$

Supposons que w_{Min}^t est l'optimum de ce programme linéaire (Q). Deux cas peuvent se présenter :

i) Si $w_{Min}^t < 0$ alors d'après i) du théorème de Geoffrion, J_t doit être coupée car nous ne pouvons pas espérer une meilleure solution réalisable à partir de ce sommet.

ii) Si $w_{Min}^t \geq 0$, nous utilisons les valeurs optimales de μ pour construire la contrainte combinée qui est ensuite ajoutée au problème et on utilise le ii) du théorème précédent pour fixer les valeurs de certaines variables libres ensuite nous utilisons l'algorithme de Ballas pour le programme augmenté.

4.3 Exemple

Soit le programme linéaire Booléen suivant :

Min Z = (3, 2, 5, 2, 2) $(x_1, x_2, x_3, x_4, x_5)^t$

$$\begin{pmatrix} -1 & -1 & 1 & 2 & -1 \\ -7 & 0 & 3 & -4 & -3 \\ 11 & -6 & 0 & -3 & -3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} \leq \begin{pmatrix} 1 \\ -2 \\ -1 \end{pmatrix}$$

x_j= (0, 1) pour j=1, 2, 3, 4, 5

Avant de construire la contrainte combinée, il est nécessaire d'avoir une solution réalisable pour fixer la valeur de Z_{Min}. En utilisant Ballas, nous obtenons une solution réalisable suivante : J_1={5} avec S^1={2, 1, 2) avec Z_{Min} = 3 nous coupons J_1 et 2_1={-5} avec S^1={1,-2, -1) avec Z^2 = 0.

Nous résolvons le programme linéaire suivant :

Min $w^2 = \mu_1 - 2\ \mu_2 - \mu_3 + y_1 + y_2 + y_3 + y_4 + (Z_{Min} - Z^t) = \mu_1 - 2\ \mu_2 - \mu_3 + y_1 + y_2 + y_3 + y_4 + 3$

$\mu_1 + 7\ \mu_2 - 11\ \mu_3 - y_1 \leq 3$

$\mu_1 + 6\ \mu_3 - y_2 \leq 2$

$-\mu_1 - 3\ \mu_2 - y_3 \leq 5$

$-2\ \mu_1 + 4\ \mu_2 + 3\ \mu_3 - y_4 \leq 2$

$\mu_i, y_j \geq 0$ i, j =1, 2, 3

La solution optimale de ce problème est :

μ_1 = - 0. 99 μ_2 = 0. 48 et μ_3 = 0. 03

w^2_{Min}= - 0.99 + (Z_{Min} – 1 – Z2)= 0. 1 > 0.

Comme $w^2_{Min} > 0$, nous construisons la coupe en fixant x_5=0, la contrainte combinée est :

0.48 (-2 – (-7 x_1 + 3 x_3 – 4 x_4)) + 0.03 (-1 – (11 x_1 – 6 x_2 -3 x_4)) + ((3-1) – (3x_1 + 2 x_2 + 5 x_3 + 2 x_4)) ≥ 0. Ou encore : -0.03 x_1 + 1.82 x_2 + 6.44 x_3 -0.01 x_4 ≤ 1.01.

En utilisant le théorème de Geoffrion, pour cette contrainte combinée il s'ensuivra que x_2 et x_3 doivent être fixés à 0 pour espérer avoir une solution réalisable.

J_3={-5, -2, -3}, le problème résultant ne possède que deux variables x_1 et x_4. Le test 4 de Ballas donne x_4=1 et J_4= {-5, -2, -3, +1} avec Z_4=2 et S_4= (-1, 2, 2)^t.

Une nouvelle contrainte combinée est construite après résolution du programme linéaire suivant :

Min $w^4 = - \mu_1 + 2 \ \mu_2 + 2 \mu_3 + y_1 + \ 3\text{-}2)$

$\mu_1 + 7 \ \mu_2 - 11 \mu_3 - y_1 \qquad \leq 3$

$\mu_{i,} \ y_1 \geq 0$ i, j =1, 2, 3

Ce problème ne possède pas de solution optimale finie, $w^4 = -\infty$. On coupe le sommet associé à J_4.

De la même manière l'algorithme montre qu'il n'existe pas de solution réalisable meilleure que J_1 avec Z_1=3.

Remarque : Geoffrion a testé cette méthode sur des Benchmarks dont le nombre de variables ne dépasse pas 90, l'a comparé avec les autres techniques et a montré qu'elle offre des résultats largement meilleurs surtout en termes de gain en CPU time lorsque le nombre de variables augmente. Elle est donc efficace pour la résolution de problèmes de dimension importante.

5 La programmation 0-1 mixte

Soit le programme linéaire mixte (P) suivant :

$$(P)\begin{cases} Max\ Z = c\,x + d\,y \\ A\,x + B\,y \le b \\ x \ge 0 \\ y = (0,\ 1) \end{cases}$$

Si le vecteur y est fixé alors ce programme devient un programme linéaire que l'on notera (P_x) :

$$(P_x)\begin{cases} Max\ \ Z = c\,x + d\,y \\ A\,x \le b - B\,y \\ x \ge 0 \end{cases}$$

Le programme dual de (P_x) que l'on notera (D) est :

$$(D)\begin{cases} Min\ W = u\,(b - By) \\ u\,A \ge c \\ u \ge 0 \end{cases}$$

Nous remarquons que l'espace des solutions de (D) est indépendant de y. D'après les résultats de la dualité en programmation linéaire, si (P_x) possède une solution réalisable alors son programme dual (D) possède également une solution réalisable et à l'optimum leurs valeurs sont égales. On introduit, par nécessité, une contrainte régulatrice sous la forme:

$\sum_{i=1}^{m} u_i \le M$, où M est un nombre réel grand.

Nous savons également de la théorie de la programmation linéaire que la solution optimale de (D) lorsqu'elle existe est atteinte en un point extrême u^k de son polyèdre convexe.

Par conséquent, le programme (P) peut être réécrit comme suit :

$Max\ Z = Dy + Min_{\,u^k}\ \{u^k\,(b - By) | u^k \ge 0\ k = 1, \ldots,\ K\}$ sous la contrainte y = (0, 1). On désignera cette forme (P*).

Considérons le programme (P^r) suivant:

$$(P^r)\begin{cases} Max\ \ Z \\ Z \le D\,y + u^k\,(b - B\,y) \quad \text{avec} \quad k = 1, \ldots, r \\ u^k \ge 0\,,\ y = (0, 1) \end{cases}$$

où $1 \le r \le K$. Les contraintes de (P^r) sont appelées « coupes de BENDER ».

Lorsque r=K alors (P^r) est identique à (P*) et permet de résoudre (P).

Notons que si u_k, k= 1, …, r sont connus alors (P^r) est un programme booléen pur à l'exception de la variable Z continue mais dont l'effet est absorbé par une modification de

l'algorithme d'énumération. L'idée est que lors de la résolution de (P^K) comme un programme binaire, on détermine la valeur optimale de y et on utilise les variable binaires optimales dans le programme (P_x) pour résoudre ensuite le programme linéaire pour obtenir la solution optimale x dans (P). Il n'est pas nécessaire de déterminer tous les points extrêmes u^k pour résoudre (P). Ces points extrêmes sont générés seulement en cas de besoin en utilisant un l'algorithme itératif.

Algorithme :

Soient $\underline{Z}$ et $\bar{Z}$ respectivement une borne inférieure et une borne supérieure sur la valeur de Z* du programme (P). Posons Z^r la solution optimale de (P^r)

0) Soit u0 une solution réalisable de D, pas nécessairement un point extrême. Poser r=1 et aller en 1).

1) Résoudre (Pr) en Z et y. Soit (Z^r, y^r) la solution optimale alors $\bar{Z}= Z^r$ puis aller en 2)

2) Résoudre (D) en posant $y = y^r$ et soit u^r le point extrême correspondant solution optimale alors $\underline{Z} = D\, y^r + u^r (b - b\, y^r)$. Aller en 3).

3) Tester si $\underline{Z} = \bar{Z}$

Si oui, y^r est optimale et aller en 4)

Si non, poser r= r+ 1 et aller en 1)

4) En posant $y = y^r$, résoudre (P_x) et soit x^r la solution optimale. La solution optimale de (P) est : (x^r, y^r) et $Z^* = \underline{Z} = \bar{Z}$

Exemple : Soit le (P) le programme linéaire mixte suivant :

Max $Z = - 3\, x_1 - 3\, x_2 - y_2 + 10$

$- x_1 + x_2 + y_1 - 0{,}4\, y_2 = 0{,}7$

$x_1,\ x_2 \geq 0$

$y_1,\ y_2 = (0, 1)$

Le programme (P_x) correspondant est :

Max $Z = (- 3\, x_1 - 3\, x_2) + (10 - y_2)$

$- x_1 + x_2 = 0{,}7 - y_1 + 0{,}4\, y_2$

$x_1,\ x_2 \geq 0$

Le programme dual (D) correspondant à (P_x) est :

Min $W= u_1 (0{,}7 - y_1 + 0{,}4\, y_2)$

$-3 \leq u_1 \leq 3$

u_1 quelconque

Soit $u_1^0 = 1$ alors P^1 devient :

Max Z

$Z \leq (10 - y_2) + (0{,}7 - y_1 + 0{,}4\ y_2) = 10{,}7 - y_1 - 0{,}6\ y_2$

$y_1,\ y_2 = (0, 1)$

Ce qui entraine forcément $y_1^1 = y_2^1 = 0$ et $Z^1 = 10{,}7 = \bar{Z}$. En fixant les valeurs de y_1^1 et y_2^1 dans D on obtient :

Min $W = 0{,}7\ u_1$

$-3 \leq u_1 \leq 3$

u_1 quelconque

On obtient $u_1^1 = -3$ et $\underline{Z} = 10 + (-2{,}1) = 7{,}9$

Comme $\underline{Z} < \bar{Z}$, nous construisons P^2

Max Z

$Z \leq 10{,}7 - y_1 - 0{,}6\ y_2$

$Z \leq (10 - y_2) + (-3)\ (0{,}7 - y_1 + 0{,}4\ y_2) = 7{,}9 + 3\ y_1 - 2{,}2\ y_2$

$y_1,\ y_2 = (0, 1)$

Ce qui entraine forcément $y_1^2 = 1\ \ y_2^2 = 0$ et $Z^2 = 9{,}7 = \bar{Z}$. En remplaçant les valeurs de y_1^2 et y_2^2 dans D, on obtient :

Min $W = -0{,}3\ u_1$

$-3 \leq u_1 \leq 3$

u_1 quelconque

D'où $u_1^2 = 3$ et $\underline{Z} = (10 - 1) + 3(0{,}7 - 1 + 0{,}4 * 0) = 9{,}1$

Comme $\underline{Z} < \bar{Z}$, nous construisons P^3

Max Z

$Z \leq 10{,}7 - y_1 - 0{,}6\ y_2$

$Z \leq 7{,}9 + 3\ y_1 - 2{,}2\ y_2$

$Z \leq (10 - y_2) + 3\ (0{,}7 - y_1 + 0{,}4\ y_2) = 12{,}1 - 3\ y_1 + 0{,}2\ y_2$

$y_1,\ y_2 = (0, 1)$

Ce qui entraine forcément $y_1^3 = 1\ \ y_2^3 = 0$ et $Z^3 = 9{,}1 = \bar{Z} = \underline{Z}$.

La solution optimale en y est $y_1=1$ et $y_2=0$. En remplaçant ces valeurs dans P_x et en résolvant le programme linéaire on obtient : $x_1 = 0{,}3$ et $x_2=0$ et $Z=9{,}1$.