Titre: Title:	Algorithmes de coupes pour la programmation mathématique linéaire à deux niveaux
Auteur: Author:	Walid Zghal
Date:	2002
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Zghal, W. (2002). Algorithmes de coupes pour la programmation mathématique linéaire à deux niveaux [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/7016/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/7016/
Directeurs de recherche: Advisors:	Gilles Savard, & Charles Audet
Programme: Program:	Non spécifié

UNIVERSITÉ DE MONTRÉAL

ALGORITHMES DE COUPES POUR LA PROGRAMMATION MATHÉMATIQUE LINÉAIRE À DEUX NIVEAUX

WALID ZGHAL
DÉPARTEMENT DE MATHÉMATIQUES
ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(MATHÉMATIQUES APPLIQUÉES)
NOVEMBRE 2002

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

ALGORITHMES DE COUPES POUR LA PROGRAMMATION MATHÉMATIQUE LINÉAIRE À DEUX NIVEAUX

présenté par : ZGHAL, Walid
en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de :

M. GAUVIN Jacques, Ph.D., président
M. SAVARD Gilles, Ph.D., membre et directeur de recherche
M. AUDET Charles, Ph.D., membre et codirecteur de recherche
M. DESAULNIERS Guy, Ph.D., membre

REMERCIEMENTS

Je tiens tout d'abord à remercier mes deux directeurs, Gilles Savard et Charles Audet pour leur soutien constant et les précieux conseils qu'ils m'ont prodigués.

Je remercie également tous les membres du GERAD pour l'aide qu'ils m'ont apportée dans la réalisation de ce travail.

Enfin, je désire remercier mes parents, ma fiancée Meriem et tous mes amis pour leurs encouragements et leur grande disponibilité.

RÉSUMÉ

Ce travail traite de la programmation mathématique à deux niveaux. Dans ce cadre, nous proposons un algorithme de résolution pour le problème linéaire à deux niveaux BLP basé sur les coupes de Gomory.

Dans un premier temps, nous présentons le modèle mathématique du BLP et ses propriétés. En particulier, nous mettons l'accent sur les liens qui existent entre la programmation linéaire à deux niveaux et la programmation mixte $0-1$.

Dans un second temps, nous proposons un ensemble de coupes valides pour le problème BLP. Ces coupes sont générées à partir de la reformulation mixte du problème BLP. Nous étudions leurs profondeurs respectives.

Dans un troisième temps, nous exposons une approche de résolution pour le problème BLP. Notre algorithme $Al(coupes)$ comprend deux étapes. La première étape correspond à une phase de coupes visant à réduire le domaine réalisable et la deuxième correspond à une phase d'énumération. Nous donnons les détails de ces deux étapes. Nous étudions l'ensemble des paramètres de l'algorithme en particulier le critère de sélection des coupes et le critère de branchement. Nous présentons également l'ensemble des tests et des procédures permettant d'améliorer la performance de l'algorithme.

Finalement, les résultats détaillés sont fournis pour déterminer les critères optimaux de l'algorithme et évaluer l'efficacité des coupes. Testé sur une série de problèmes générés aléatoirement, l'algorithme $Al(coupes)$ s'avère très performant en comparaison avec la résolution CPLEX de la reformulation mixte.

ABSTRACT

This work deals with the bilevel programming problem. Within this framework, we propose an algorithm which uses Gomory cuts to solve the linear bilevel problem BLP.

First, we present the mathematical model of BLP and its properties. We emphasize on the links between bilevel linear programming and linear mixed $0-1$ programming.

Next, we propose a set of valid cuts to the BLP. These cuts are generated from the mixed reformulation of the BLP. We examine their depths.

Then, a new algorithm $Al(coupes)$ for bilevel linear programming problem is introduced. Our algorithm includes two steps. The first one generates Gomory cuts to reduce the feasible region. The second step is a branch and bound procedure. Both steps are described in details. We study the different features of the algorithm, in particular the cut selection criterion and the branching criterion. We also propose a set of tests and procedures that streamline the algorithm.

Finally, we feature the optimal algorithmic criteria and evaluate the effectiveness of the cuts through experimental results. Our algorithm $Al(coupes)$ outperforms the CPLEX resolution of mixed reformulation when tested on a series of randomly generated test problems.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

CHAPITRE 1

Introduction

La programmation mathématique à deux niveaux traite des problèmes à structure hiérarchisée comportant deux agents de décision. Le modèle mathématique s'écrit comme suit :

$$\max_{x,y} f^1(x,y)$$

$$\text{s.c.} \begin{cases} g^1(x,y) \leq 0, \\ y \in \arg\max_{w} f^2(x,w), \\ \qquad \text{s.c.} \begin{cases} g^2(x,w) \leq 0, \end{cases} \end{cases}$$

Plusieurs travaux de recherche se sont intéressés au cas linéaire BLP, c.à.d celui où les fonctions $f^1(\cdot,\cdot)$ et $f^2(\cdot,\cdot)$ sont linéaires et les fonctions $g^1(\cdot,\cdot)$ et $g^2(\cdot,\cdot)$ sont affines. Dans ce cadre, Audet et al. [4] a démontré l'équivalence entre la programmation linéaire à deux niveaux et la programmation mixte $0-1$. Cette équivalence laisse entrevoir une structure intrinsèque et des techniques de résolution communes. Ce présent travail se veut une reconnaissance de ce fait. Ainsi, nous considérons une technique de résolution inhérente à la programmation mixte à savoir les coupes de Gomory et nous étudions leur intégration au sein des problèmes linéaires à deux niveaux. À partir de ces coupes, nous développons de nouvelles inégalités valides pour le BLP. Nous présentons finalement un algorithme de résolution pour le problème

BLP exploitant à la fois les techniques de coupes et celles d'énumération. Les résultats numériques illustrent la bonne performance de notre approche.

Le travail est organisé comme suit : le deuxième chapitre présente la programmation linéaire à deux niveaux. Nous exposons l'ensemble des propriétés du BLP ainsi qu'une revue des algorithmes de résolution. Nous mettons en valeur l'équivalence entre la programmation linéaire à deux niveaux et la programmation mixte $0-1$. Au troisième chapitre, nous générons un ensemble de coupes valides à partir de la reformulation mixte $0-1$. Nous discutons de la profondeur des différentes coupes résultantes. Au quatrième chapitre, nous développons un algorithme de coupes pour le problème BLP. Nous y présentons ses différentes étapes. Au cinquième chapitre, nous présentons les résultats de l'algorithme en analysant ses différents paramètres et en évaluant sa performance.

CHAPITRE 2

La programmation linéaire à deux niveaux

2.1 Présentation de la programmation biniveau

La programmation biniveau est une branche de l'optimisation globale traitant les problèmes à deux agents de décision. Le modèle mathématique est hiérarchisé et comprend deux niveaux d'optimisation. Sous sa formulation linéaire, il s'écrit comme suit :

$$\max_{x,y} c^{1T}x + d^{1T}y$$

$$(BLP) \qquad \text{s.c.} \begin{cases} A^1x + B^1y \leq b^1, \\ x \geq 0, \\ y \in \arg\max_{w} d^{2T}w, \\ \qquad \text{s.c.} \begin{cases} A^2x + B^2w \leq b^2, \\ w \geq 0 \end{cases} \end{cases}$$

où

$$
\begin{array}{ll}
x \in \mathbb{R}^{n_1}; & y, w \in \mathbb{R}^{n_2}; \\
c^1 \in \mathbb{R}^{n_1}; & d^1, d^2 \in \mathbb{R}^{n_2}; \\
A^i : \mathbb{R}^{m_i} \times \mathbb{R}^{n_1} & i = 1, 2; \\
B^i : \mathbb{R}^{m_i} \times \mathbb{R}^{n_2} & i = 1, 2; \\
n_1, n_2, m_1, m_2 \in \mathbb{N}. &
\end{array}
$$

Cette formulation a été introduite pour la première fois par Candler et Norton [14]. On note la présence de deux variables de décision à savoir la variable x contrôlée par le premier agent et la variable y contrôlée par le deuxième. Dans ce modèle, la prise de décision se fait de manière séquentielle. C'est l'agent du premier niveau, appelé meneur, qui commence par choisir son vecteur de décision $\hat{x}$. Ensuite, le second agent, appelé suiveur, réagit en résolvant à l'optimum le second niveau pour $x = \hat{x}$ afin d'évaluer son propre vecteur y. Ainsi, le problème consiste à déterminer une solution optimale pour le premier niveau en tenant compte de la réaction optimale du second. Nous supposons une coopération entre les deux agents en cas de présence de plusieurs vecteurs optimaux y pour le second niveau. Le suiveur choisira celui qui maximise la fonction objectif du meneur.

Cette structure hiérarchisée apparaît dans plusieurs applications notamment en économie [27] et dans le domaine du transport [21]. En économie, nous pouvons donner comme exemple le modèle de Stackelberg [27] qui représente deux firmes en concurrence. La première firme joue le rôle de meneur et annonce son niveau de production (variable de premier niveau x) avant le deuxième producteur qui joue le rôle de suiveur. Ce dernier tient compte du niveau x annoncé préalablement pour déterminer sa propre production (variable de second niveau y). Ce modèle suppose également que le meneur agit en connaissant la réaction optimale du suiveur. On y retrouve ainsi les caractéristiques fondamentales du problème biniveau notamment la

prise séquentielle de décision. Dans le domaine du transport, la conception de réseaux constitue un autre cadre d'application pour la programmation à deux niveaux. En particulier, on s'intéresse à la planification d'un système routier compte tenu de la réacation des utilisateurs.

2.1.1 Définitions et notations

Afin de définir le concept de solution optimale du problème BLP, nous présenterons dans ce qui suit certaines notations et définitions inhérentes à la programmation biniveau qui seront utilisées tout au long de ce mémoire.

Le problème relaxé : Il est obtenu à partir du problème biniveau en faisant abstraction de l'optimalité du second niveau. Il s'écrit comme suit :

$$(LRP)\begin{cases} \max\limits_{x,y} & c^{1T}x + d^{1T}y \\ \text{s.c.} & A^1x + B^1y \leq b^1, \\ & x \geq 0, \\ & A^2x + B^2y \leq b^2, \\ & y \geq 0. \end{cases}$$

Le domaine réalisable : L'ensemble des solutions réalisables pour le problème relaxé est :

$$\Omega = \left\{(x,y) \in \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \mid x \geq 0,\ y \geq 0,\ A^ix + B^iy \leq b^i,\ i = 1,2\right\}.$$

Un point $(x,y) \in \Omega$ est appelé *point réalisable.*

La trace : On appelle la trace de Ω sur l'ensemble des variables de premier niveau, notée Ω_x^2, l'ensemble :

$$\Omega_x^2 = \left\{x \in \mathbb{R}^{n_1} \mid\ x \geq 0, \exists\, y \in \mathbb{R}^{n_2}\ A^2x + B^2y \leq b^2\right\}.$$

Le domaine réalisable du second niveau : Pour $x \in \Omega_x^2$ donné, il correspond à l'ensemble des y satisfaisant les contraintes du second niveau :

$$\Omega_y(x) = \left\{ y \in \mathbb{R}^{n_2} \mid y \geq 0,\ A^2 x + B^2 y \leq b^2 \right\}.$$

L'ensemble de solutions optimales du problème de second niveau : Il correspond à la réaction du second niveau pour un x donné appartenant à Ω_x^2 :

$$M(x) = \left\{ y \in \arg\max d^{2T} w, w \in \Omega_y(x) \right\}.$$

La fonction valeur optimale : Pour $x \in \Omega_x^2$, elle correspond à la valeur de la fonction objectif du second niveau associée à tout $y \in M(x)$:

$$v(x) = d^{2T} y, y \in M(x).$$

Point rationnel : On appelle *point rationnel* un point (x, y) tel que $x \in \Omega_x^2$ et $y \in M(x)$.

Point admissible : On appelle point *admissible* un point $(x, y) \in \Omega$ tel que $y \in M(x)$.

Le domaine induit : Le domaine induit DI est l'ensemble des points admissibles.

Solution optimale : Une solution (x^*, y^*) est optimale si elle est admissible et si, pour toute solution (x, y) admissible, on a $c^{1T} x^* + d^{1T} y^* \geq c^{1T} x + d^{1T} y$.

2.1.2 Exemple

Le problème suivant illustre les définitions précédentes notamment la différence entre les concepts d'admissibilité et de rationalité en présence de contraintes au premier niveau.

Considérons l'exemple proposé par Savard [26] ainsi que sa représentation graphique donnée à la figure 2.1.

Exemple 1 $\max\limits_{x,y} -x - 4y$

$$\text{s.c} \begin{cases} 3x + 2y \leq 36, \\ y \in \arg\max\limits_{w} w \\ \qquad \text{s.c.} \begin{cases} -x - w \leq -8, \\ -3x + 2w \leq 6, \\ 3x + 4w \leq 48, \\ 2x - 5w \leq 9. \end{cases} \end{cases}$$

Dans cet exemple, le domaine induit constitue l'enveloppe supérieure du domaine réalisable. On remarque que le point A est un point admissible. En effet, il est à la fois réalisable et rationnel. Bien que le point B soit rationnel, il n'est pas admissible puisqu'il ne satisfait pas la contrainte de premier niveau. De même, pour tout $x \in]m, n]$, le second niveau retourne une solution non réalisable. Le point C est par contre réalisable et non rationnel. Par conséquent, il n'est pas admissible. Le point X^* constitue la solution optimale du problème biniveau. Il correspond à la meilleure solution admissible pour le premier niveau. Notons, finalement, la présence d'un maximum local (X).

2.2 Propriétés des problèmes BLP

2.2.1 Caractéristiques du domaine induit

Le domaine induit est généralement non convexe et il est parfois non connexe. Néanmoins, en considérant des problèmes BLP sans contraintes de premier niveau, Bard [9] a montré que cet ensemble est connexe et qu'il est l'union de faces de poly-

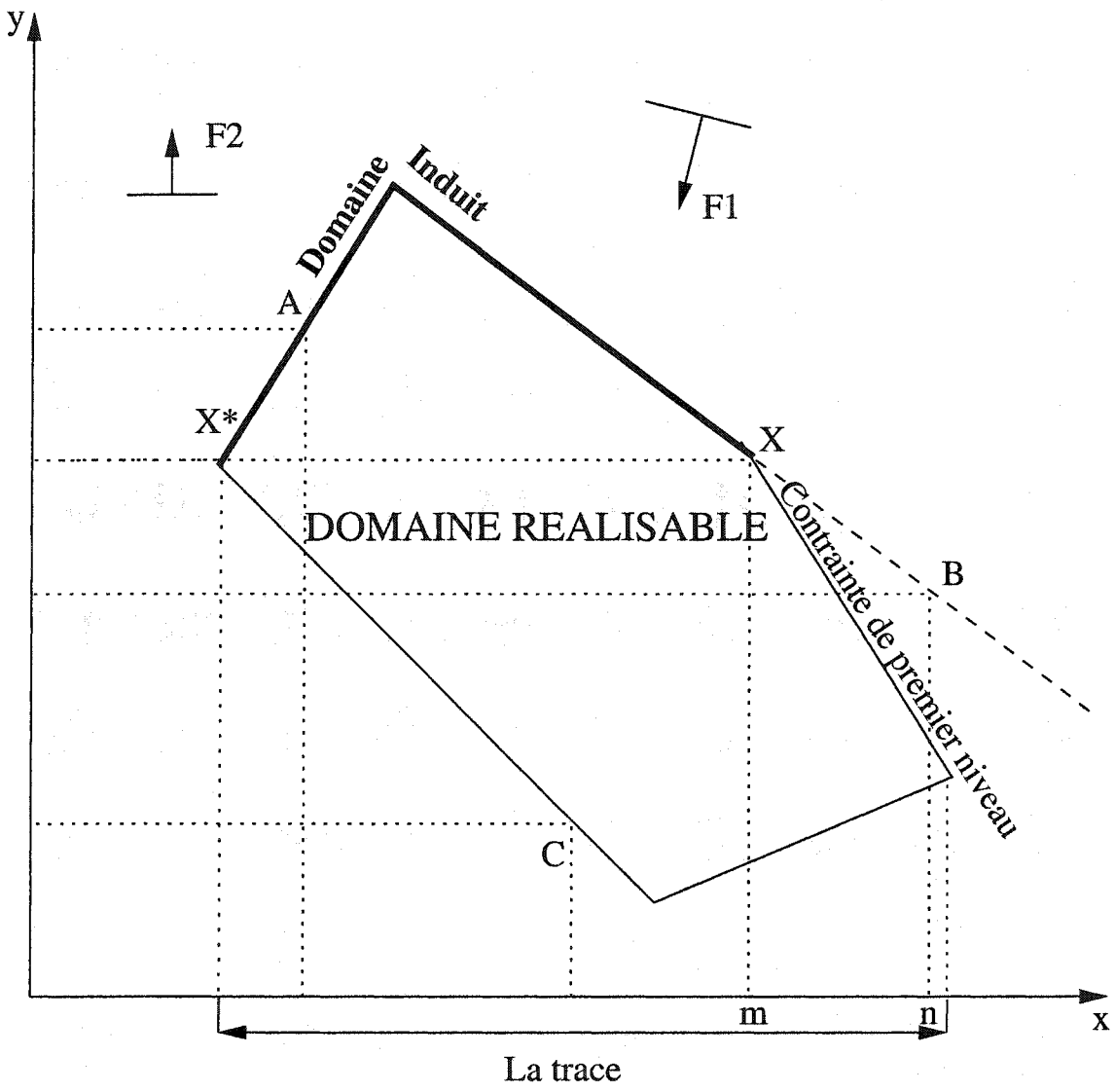

Figure 2.1 – Exemple d'un problème biniveau

topes défini par les contraintes du second niveau. Savard [26] a prouvé que ce résultat n'est plus valable avec la présence de contraintes couplantes au premier niveau. Audet [3] renforce ce dernier théorème en donnant un exemple où le domaine induit est discret :

Exemple 2

$$\max_{x,y} x$$

$$\text{s.c} \begin{cases} 0 \leq x \leq 1, \\ y = 0, \\ y \in \arg\max_z z \\ \qquad \text{s.c.} \begin{cases} z \leq x, \\ z \leq 1 - x. \end{cases} \end{cases}$$

Soit $(\hat{x}, \hat{y})$ une solution admissible. $\hat{y}$ est alors optimal pour le second niveau pour $x = \hat{x}$. Puisqu'à l'optimum une des deux contraintes doit être saturée, on aura $\hat{y} = \min(\hat{x}, 1 - \hat{x})$. Or la variable y est contrainte par le premier niveau à être nulle. D'où, $\hat{x} \in \{0, 1\}$. Ainsi $DI \subset \{(1, 0), (0, 0)\}$.
Réciproquement, on peut vérifier que $(0, 0)$ et $(1, 0)$ sont admissibles. En conclusion, le domaine induit contient uniquement les deux éléments : $\{(0, 0), (1, 0)\}$.

2.2.2 Complexité

Définition 2.2.2.1 *Une classe de problème est dite NP-difficile (resp. fortement NP-difficile), s'il ne peut exister un algorithme permettant la résolution d'un problème de cette classe en temps borné par une fonction polynomiale de la taille du problème (resp. fonction polynomiale de la taille et de la valeur maximale des données du problème).*

Jeroslow [19] a montré que le problème BLP est NP-difficile. Ce résultat a été confirmé par Ben-Ayed et Blair [11] en donnant une preuve plus simple. Hansen, Jaumard et Savard [18] ont montré que le problème biniveau est fortement NP-difficile en réduisant un problème max-min au problème de NOYAU. Le problème max-min est un cas particulier du BLP. Finalement, Judice, Savard et Vicente [20] ont montré que la vérification de l'optimalité locale est également NP-difficile.

2.3 Revue des algorithmes

Plusieurs algorithmes ont été proposés pour la résolution des problèmes biniveaux. Parmi ces algorithmes, on distingue deux classes permettant de résoudre efficacement le modèle linéaire.

2.3.1 Méthode d'énumération implicite de points extrêmes

Cette classe exploite le résultat qui indique que la solution optimale se trouve en un point extrême du domaine réalisable [26]. Partant de cette propriété, Candler et Townsley [15] et Bialas et Karwan [13] proposent des algorithmes de résolution se basant sur l'énumération des points extrêmes. Les deux approches diffèrent selon la procédure de recherche de ces points. Nous présentons brièvement ces deux algorithmes.

L'algorithme de Candler et Townsley

L'algorithme considère les problèmes BLP sans contraintes dites couplantes au premier niveau faisant intervenir à la fois la variable de premier niveau x et celle de second niveau y. L'approche consiste à énumérer les bases optimales (BOB) du problème du second niveau pour x fixé que les auteurs appellent "*behavioral LP problem*" :

$$(P_1) \begin{cases} \max\limits_{y} & d^{2T}y \\ \text{s.c.} & B^2y \leq b^2 - A^2x, \\ & y \geq 0. \end{cases}$$

Parmi ces bases, l'algorithme détecte celles qui améliorent la fonction objectif en considérant le problème suivant "*policy LP problem*" :

$$(P_2) \begin{cases} \max\limits_{x,y_k} & c^{1T}x + d^{1T}y_k \\ \text{s.c.} & A^2x + {B^2}_k y_k \leq b^2, \\ & x \geq 0 \;\; y_k \geq 0. \end{cases}$$

où ${B^2}_k$ est matrice de base optimale du problème P_1.
Candler et Townsley ont proposé une méthode d'exploration implicite des bases optimales BOB en utilisant les coûts réduits des variables hors base de second niveau (par rapport à la base optimale (BOB)) du problème P_2. Selon les tests réalisés par Bard [8] pour des problèmes de taille moyenne, l'algorithme énumère un nombre important de bases optimales et nécessite un temps assez élevé pour converger.

L'algorithme "Kth-Best" de Bialas et Karwan

Cet algorithme se limite également aux problèmes sans contraintes de premier niveau. Mais à l'encontre de l'approche proposée par Candler et Townsley [15], le

"Kth-Best" énumère les bases du problème relaxé (LRP).

L'exploration de ces bases se fait selon l'ordre décroissant de la valeur de la fonction objectif de premier niveau. Bialas et Karwan définissent T comme l'ensemble des bases traitées, W comme l'ensemble des bases à traiter. L'algorithme se divise en quatre étapes :

1. Poser $i = 1$. Soit $(x^*_{[1]}, y^*_{[1]})$ la solution optimale du problème LRP. Poser $T = \emptyset$ et $W = \{(x^*_{[i]}, y^*_{[i]})\}$.
2. Vérifier l'admissibilité de la solution en résolvant le problème de second niveau pour $x^*_{[i]}$ fixé.$(x^*_{[i]}, y^*_{[i]})$ est une solution admissible, terminer avec $(x^*_{[i]}, y^*_{[i]})$ solution optimale pour BLP ; sinon aller à l'étape 3.
3. Soit $W_{[i]}$ l'ensemble des points extrêmes (x, y) adjacents à $(x^*_{[i]}, y^*_{[i]})$ tels que $c^{1T}x + d^{1T}y \leq c^{1T}x^*_{[i]} + d^{1T}y^*_{[i]}$. Poser $T = T \cup \{(x^*_{[i]}, y^*_{[i]})\}$ et $W = (W \cup W_{[i]}) \cap T^c$.
4. Poser $i = i + 1$ et choisir $(x^*_{[i]}, y^*_{[i]}) \in \arg\max_{(x,y)\in W} \{c^{1T}x + d^{1T}y\}$. Aller à l'étape 2.

Il est à noter que la performance de l'algorithme dépend de la position relative de la solution optimale par rapport à la première solution trouvée.

2.3.2 Méthode d'énumération basée sur les conditions de complémentarité

Considérons le second niveau du problème BLP pour x fixé à $\bar{x}$:

$$\begin{aligned} \max_{y} \quad & d^{2T}y \\ \text{s.c.} \quad & B^2 y \leq b^2 - A^2\bar{x}, \\ & y \geq 0. \end{aligned}$$

D'après les conditions nécessaires et suffisantes de Karush-Kuhn-Tucker (KKT),

une solution réalisable y_{opt} est optimale pour le second niveau s'il existe λ tel que :

$$\begin{aligned}
&B^{2t}\lambda \geq d^2,\\
&\lambda(b^2 - A^2x - B^2y_{opt}) = 0,\\
&y_{opt}(B^{2T}\lambda - d^2) = 0,\\
&\lambda \geq 0.
\end{aligned}$$

Ainsi, en exprimant l'optimalité du second niveau à l'aide des conditions KKT, le problème BLP se reformule comme suit :

$$(LKKT) \begin{cases} \max\limits_{x,y,\lambda} & c^{1t}x + d^{2T}y \\ \text{s.c.} & A^1x + B^1y \leq b^1, \\ & A^2x + B^2y \leq b^2, \\ & B^{2t}\lambda \geq d^2, \\ & \lambda(b^2 - A^2x - B^2y) = 0, \\ & y(B^{2t}\lambda - d^2) = 0, \\ & x \geq 0 \;\; y \geq 0 \;\; \lambda \geq 0. \end{cases}$$

Plusieurs algorithmes d'énumération exploitent la reformulation $LKTT$ pour la résolution du problème BLP. L'approche consiste à utiliser les conditions KKT comme éléments de séparation. Parmi ces algorithmes, on distingue celui de Bard et Moore [10] et celui de Hansen et al. [18]. Ce dernier est, à notre connaissance, le plus performant de la littérature.

Bard et Moore

L'algorithme considère le problème $LKKT$ en éliminant les contraintes de complémentarité. Une procédure d'énumération est alors utilisée pour ce problème relaxé. Le

principe de séparation se fait d'une part sur les variables d'écart du second niveau s définies par $s = b^2 - A^2x - B^2y$ et d'autre part sur les variables duales correspondantes λ. Plusieurs critères de séparation ont été testés. Les auteurs ont finalement retenu celui qui consiste à séparer sur les variables λ_i et s_i tel que $i \in \arg\max_k\{\lambda_k.s_k\}$.

L'algorithme HJS

L'algorithme HJS de Hansen et al. [18] s'avère le plus performant de la littérature. Ce dernier considère la relaxation LRP et le problème du second niveau initial $FSP(\hat{x})$ avec $(\hat{x}, \hat{y})$ solution optimale du problème LRP.

$$(FSP(\hat{x})) \begin{cases} \max\limits_{y} & d^{2T}y \\ \text{s.c.} & B^2y \leq b^2 - A^2\hat{x}, \\ & y \geq 0. \end{cases}$$

L'idée de base consiste à détecter par énumération l'ensemble des contraintes saturées à l'optimum (incluant les bornes de non négativité). Pour ce faire, les auteurs associent une variable binaire α_i pour chaque contrainte du second niveau y compris les contraintes de signe. La séparation se fait sur ces variables. Si une variable α_i est fixée à 1, la contrainte correspondante (ou borne) est fixée à égalité ce qui permet d'éliminer une variable y. Si α_i est fixée à 0, la contrainte ou la variable duale associée est fixée à égalité. Pour chaque noeud, l'algorithme HJS considère le problème du second niveau courant $FRP(\hat{x})$ pour x fixé à $\hat{x}$. En tenant compte de l'élimination des variables, ce problème s'écrit comme suit :

$$(FRP(\hat{x})) \begin{cases} \max\limits_{\tilde{y}} & \tilde{d}^{2T}\tilde{y} \\ \text{s.c.} & \tilde{B}^2\tilde{y} \leq \tilde{b}^2 - \tilde{A}^2\hat{x}, \\ & \tilde{y} \geq 0. \end{cases}$$

où $\tilde{y}$ correspond à la partie du vecteur du second niveau non éliminée et les vecteurs $\tilde{d}^2$, $\tilde{b}^2$ et matrices $\tilde{A}^2$, $\tilde{B}^2$ correspondent aux vecteurs et matrices initiaux obtenus après élimination de variables.

Afin de réduire la taille de l'arbre de recherche, l'algorithme applique le principe de monotonie sur le problème du second niveau $FRP(\hat{x})$, ce qui permet d'obtenir un ensemble de relations exprimées à l'aide des variables α_i. En effet, ce principe affirme que toute solution rationnelle est telle que les contraintes du second niveau satisfont :

$$\sum_{i:B^2_{ij}>0} \alpha_i \geq 1 \quad \text{si} \quad d^2_j > 0,$$

$$\sum_{i:B^2_{ij}<0} \alpha_i + \alpha_{m_2+i} \geq 1 \quad \text{si} \quad d^2_j < 0.$$

Ainsi, dans l'exemple 1, nous pouvons déduire d'après le principe de monotonie que la deuxième ou la troisième contrainte de second niveau est serrée. On note finalement l'utilisation d'une série de tests ainsi que des pénalités semblables à celles exploitées en programmation linéaire mixte en nombres entiers. D'ailleurs, Audet [3] a prouvé le plongement (voir définition à la section 2.4.2) d'un algorithme conçu pour le problème de programmation linéaire mixte MIP_{0-1} dans l'algorithme HJS. Ce résultat conforte l'équivalence qui existe entre la programmation linéaire biniveau et la programmation linéaire mixte $0-1$.

2.4 Lien entre la programmation linéaire biniveau et la programmation linéaire mixte $0-1$

Il existe une équivalence entre la programmation linéaire mixte $0-1$ et la programmation linéaire biniveau. Cette équivalence a été démontrée par Audet [3] et Audet et al. [4] en utilisant deux approches à savoir la reformulation des classes de problèmes qui sera étudiée dans la section 2.4.1 et le plongement des algorithmes qui sera traité dans la section 2.4.2.

La reformulation est une transformation permettant de réécrire de façon équivalente un problème A comme cas particulier d'une autre classe. Ainsi, pour résoudre le problème A, il suffit de résoudre la reformulation et transférer le résultat obtenu au problème initial.

Le concept de plongement permet de comparer des algorithmes définis pour des problèmes de classes distinctes mais reliées par l'intermédiaire d'une reformulation. L'approche consiste à comparer les suites de sous-problèmes engendrés par ceux-ci.

En résumé, ces deux concepts servent à comprendre la nature des liens qui unissent deux classes de problèmes, ce qui permet de transférer les connaissances et les méthodes de résolution d'une classe à une autre.

2.4.1 Reformulation des problèmes

Définition 2.4.1.1 *Soient P_A et P_B deux classes de problèmes d'optimisation. Une reformulation $B(\cdot)$ de P_A en P_B est une transformation de P_A à P_B telle que pour tout problème A de P_A ainsi qu'une solution optimale de $B(A)$, on peut obtenir une*

solution optimale de A en temps polynomial.

Audet et al. [4] ont démontré à la fois la reformulation d'un problème linéaire mixte $0-1$ en un problème linéaire biniveau ainsi que la reformulation d'un problème linéaire biniveau en un problème linéaire mixte $0-1$.

Reformulation du problème mixte $0-1$ en un problème biniveau

Le problème de programmation linéaire mixte $0-1$ (MIP_{0-1}) est un problème linéaire où certaines variables sont restreintes à prendre des valeurs binaires. Il s'écrit comme suit :

$$(MIP_{0-1}) \begin{cases} \max\limits_{x,u} & c^T x + e^T u \\ \text{s.c.} & Ax + Eu \leq b, \\ & x \geq 0, \;\; u \in \{0,1\}. \end{cases}$$

où $x, c \in \mathbb{R}^{n_1}; e, u \in \mathbb{R}^{n_2}; b \in \mathbb{R}^m; A \in M_{m \times n}$.

Ce problème peut être réduit en un problème biniveau BLP via la transformation suivante :

$$\max_{x,u,v} c^T x + e^T u$$

$$(BLPm) \quad s.c. \begin{cases} Ax + Eu \leq b, \\ 0 \leq u \leq 1, \\ x \geq 0, \\ v = 0, \\ \quad v \in \arg\max\limits_{y} y, \\ \quad s.c. \begin{cases} y \leq u, \\ y \geq 1 - u. \end{cases} \end{cases}$$

En effet, si (x^*, u^*) est une solution optimale de MIP_{0-1}, alors $(x^*, u^*, 0)$ est une solution optimale de $BLPm$. Réciproquement, si (x^*, u^*, v^*) est une solution optimale de ce dernier, alors (x^*, u^*) est une solution optimale de MIP_{0-1}, et $v^* = 0$.

Reformulation du problème biniveau en un problème mixte $0-1$

Le problème BLP peut être également reformulé en un problème MIP_{0-1}. Cette réduction se fait via deux transformations. On a vu que le problème BLP se réduit à un problème à un seul niveau $LKKT$ en remplaçant le problème de second niveau par ses conditions nécessaires et suffisantes d'optimalité. La deuxième transformation consiste à linéariser ces conditions de KKT en introduisant deux vecteurs de variables binaires u et v ainsi qu'une grande constante L. Les conditions de complémentarité se reformulent alors comme suit :

$$\begin{aligned} & b^2 - A^2x - B^2y \leq L(1-u), && \lambda \leq Lu, \\ & y \leq L(1-v), && B^2\lambda - d^2 \leq Lv. \end{aligned}$$

On obtient alors la reformulation mixte (BLP_r) :

$$(BLP_r) \begin{cases} \max\limits_{x,y,\lambda,u,v} & c^{1T}x + d^{1T}y & \\ & A^1x + B^1y \leq b^1, & \\ & x \geq 0, & \\ & A^2x + B^2y \leq b^2, & -B^{2t}\lambda \leq -d^2, \\ & y \geq 0, & \lambda \geq 0, \\ & & \\ & -A^2x - B^2y + Lu \leq Le - b^2, & \lambda - Lu \leq 0, \\ & y + Lv \leq Le, & B^{2T}\lambda - Lv \leq d^2, \\ & u \in \{0,1\}, & v \in \{0,1\}. \end{cases}$$

où e est le vecteur unité.

Cette reformulation suppose que la valeur optimale de BLP est bornée [4] afin d'assurer l'existence de la constante L. Ainsi, pour toute solution optimale finie (x^*, y^*) de BLP, il existe une constante $L \geq 0$ et des vecteurs $\lambda^* \in \mathbb{R}^{m_2}$, $u^* \in \mathbb{R}^{m_2}$ et $v^* \in \mathbb{R}^{n_2}$ tel que $(x^*, y^*, \lambda^*, u^*, v^*)$ est solution optimale de BLP_r. Réciproquement, pour un tel L, si $(x^*, y^*, \lambda^*, u^*, v^*)$ est une solution optimale de BLP_r alors (x^*, y^*) est solution de BLP.

Notons que la constante L est fréquemment utilisée en programmation biniveau afin d'exprimer l'optimalité du second niveau via le théorème des écarts complémentaires. Anandalingam et White [2] s'en servent pour introduire une pénalité $p = -L\pi$ au niveau de la fonction objectif de premier niveau. π est l'écart dual pour un x donné, y satisfaisant les contraintes primales du second niveau et λ satisfaisant les contraintes duales du même niveau :

$$\pi = \lambda^T(b^2 - A^2x) - d^2y.$$

Anandalingam et White [2] démontrent que l'optimalité est atteinte pour une valeur finie de L qui maximise la fonction objectif du premier niveau $F(x, y)$ et annule l'écart π. Les auteurs montrent que $F(x, y)$ et π sont toutes les deux fonctions décroissantes de la constante L. Ainsi, ils proposent un algorithme permettant de résoudre le problème biniveau en augmentant à chaque itération la valeur de la constante L.

2.4.2 Plongement d'algorithmes

Définition 2.4.2.1 *Soient P_A et P_B deux classes de problèmes d'optimisation. L'algorithme $Al(P_A)$ est plongé dans $Al(P_B)$ via la transformation $B : P_A \rightarrow P_B, A \mapsto B(A)$ si pour tout problème A_0 de P_A, la suite $\{A_l\}_{l\geq 0}$ engendrée par l'application de $Al(P_A)$ à A_0, et la suite $\{B_l\}_{l\geq 0}$ engendrée par l'application de $Al(P_B)$ à $B(A_0)$ sont telles que pour tout $l \geq 0$, $B_l = B(A_l)$.*

Audet [3] et Audet et al. [4] considère à la fois l'algorithme HJS [18] conçu pour la résolution du problème biniveau et celui de Beale et Small [12] conçu pour la programmation mixte. Ce dernier est un algorithme d'énumération implicite utilisant le calcul de pénalités. Audet [3] démontre son plongement dans l'algorithme HJS via la reformulation $MIP_{0-1} \rightarrow BLP$. En effet, l'algorithme HJS appliqué à la reformulation biniveau d'un problème mixte $0-1$ reprend le même schéma de résolution de l'algorithme de Beale et Small appliqué au même problème :

L'élaboration d'une arborescence : Dans l'algorithme HJS, le critère de séparation basé sur le calcul de pénalités est similaire à celui adopté par l'algorithme Beale et Small.

L'évaluation de la borne supérieure : Afin d'évaluer une borne supérieure à chaque noeud de l'arborescence, les deux algorithmes définissent la même relaxation linéaire.

L'exploration d'un noeud : Dans l'algorithme de HJS, les tests utilisés pour l'examen d'un noeud sont soit vérifiés soit identiques à ceux utilisés par l'algorithme

de Beale et Small.

En conclusion, l'algorithme HJS appliqué à la reformulation biniveau d'un problème mixte exécute exactement les mêmes étapes que l'algorithme de Beale et Small appliqué au problème mixte. Le résultat inverse n'est pas vrai. Le plongement de l'algorithme de Beale et Small dans l'algorithme HJS présente un double intérêt. D'une part, d'un point de vue algorithmique, ce résultat montre une similitude entre la classe des problèmes biniveaux et celle des problèmes mixtes $0-1$. D'autre part, du point de vue complexité, le plongement suggère que le problème biniveau est plus difficile que le problème mixte $0-1$ d'autant plus que l'algorithme de Beale et Small n'est pas plongé dans HJS via la transformation $BLP \rightarrow MIP_{0-1}$.

CHAPITRE 3

Génération de coupes en programmation biniveau

L'équivalence qui existe entre la programmation biniveau et la programmation mixte laisse entrevoir une structure intrinsèque et des techniques de résolution communes. Parmi les techniques utilisées en programmation mixte, on distingue les coupes de Gomory [17]. Ces coupes ont été introduites au début des années soixantes. Elles ont suscité un intérêt considérable mais elles ont vite montré leur limite en pratique [7]. Dans ce chapitre, nous allons étudier l'intégration de ces coupes dans les problèmes biniveaux. Nous donnerons en premier lieu leur expression à partir de la reformulation mixte du problème BLP. En deuxième lieu, nous exposerons une deuxième approche pour retrouver la même expression à partir de la formulation initiale du problème.

3.1 Coupes de Gomory en programmation mixte

Considérons le problème de programmation linéaire mixte en nombres entiers :

$$(MIP)\begin{cases} \max\limits_{x,y} & c^T x + d^T y \\ \text{s.c.} & Ax + By \leq b, \\ & x \in \mathbb{Z}^n_+, \ y \geq 0. \end{cases}$$

Deux approches ont été adoptées pour la résolution du problème MIP : l'approche d'énumération et l'approche de coupes.

L'approche d'énumération

L'approche d'énumération consiste à diviser l'ensemble des solutions en x en sous-ensemble S^i et résoudre les sous-problèmes correspondants :

$$(MIP^i) \begin{cases} \max\limits_{x,y} & c^T x + d^T y \\ \text{s.c.} & Ax + By \leq b, \\ & x \in S^i, \ \ y \geq 0. \end{cases}$$

Deux principes caractérisent les procédures d'énumération : le principe de séparation et le principe d'évaluation.

Le principe de séparation sert à définir le schéma d'exploration des sous-ensembles S^i. Il est généralement illustré par une arborescence. Ce principe comprend notamment le critère de branchement qui permet de déterminer la variable selon laquelle s'effectue la séparation.

Le principe d'évaluation sert à évaluer une borne supérieure Z_R^i pour le sommet courant en résolvant une relaxation (RP^i) du sous-problème.

Plusieurs tests sont utilisés pour réduire la taille de l'arborescence. Un test consiste en l'application d'une condition permettant de sonder un sommet. Un sommet est dit sondable s'il n'a pas être séparé. Tel que mentionné dans Nemhauser et Wolsey [22], il existe trois tests pour sonder un noeud i :

1. RP^i est non réalisable.
2. Une solution optimale X_R^i de RP^i satisfait $x_R^i \in S^i$ et $Z_R^i = c^T x_R^i + d^T y_R^i$.

3. $Z_R^i \leq \underline{Z}_{MIP}$, avec $\underline{Z}_{MIP}$ est la valeur d'une solution réalisable de MIP.

L'énumération s'arrête lorsqu'il n'y a plus de sommets à évaluer. La solution optimale correspond alors à la meilleure solution jusqu'à date.

L'approche de coupes

L'approche de coupes consiste à considérer la relaxation continue du problème MIP et d'y ajouter des inégalités valides. Ces dernières permettent d'éliminer la solution courante sans couper les solutions réalisables de MIP. L'application successive d'une telle procédure permettrait éventuellement l'obtention d'un domaine réalisable dont le point extrême optimal satisfait la condition d'intégrité. Le premier algorithme de coupes a été développé par Gomory [17] en utilisant la coupe dite de Gomory. Cette dernière peut être générée à partir d'une équation valide :

$$\sum_{j \in N} a_j x_j + \sum_{j \in J} g_j y_j = b$$
$$x \in \mathbb{Z}_+^{n_1},\ y \in \mathbb{R}_+^{n_2}.$$

Soient $J^+ = \{j \in J, g_j > 0\}$, $J^- = \{j \in J, g_j < 0\}$, $f_j = a_j - \lfloor a_j \rfloor, \forall j \in N$ et $f_0 = b - \lfloor b \rfloor$, alors la coupe de Gomory associée s'écrit :

$$\sum_{j \in N:\, f_j \leq f_0} f_j x_j + \frac{f_0}{1-f_0} \sum_{j \in N:\, f_j > f_0} (1 - f_j) x_j + \sum_{j \in J^+} g_j y_j - \frac{f_0}{1-f_0} \sum_{j \in J^-} g_j y_j \geq f_0.$$

Dans l'algorithme de Gomory, les coupes sont associées à des lignes du tableau optimal obtenu par la méthode du simplexe. Toutefois, il est possible de générer les coupes à partir de n'importe quel tableau réalisable.

Le schéma général de l'algorithme est le suivant :

L'algorithme de Gomory

Étape 1 :

Résoudre la relaxation continue de MIP.

Étape 2 :

Si $\exists i \mid x_i \notin \mathbb{Z}$

Déterminer $k = \min\{i \mid x_i \notin \mathbb{Z}\}$.

Ajouter la coupe de Gomory associée à la ligne correspondante à x_k.

Aller à l'étape 1.

Sinon la solution est optimale.

La convergence de l'algorithme est assurée sous quatre conditions [28] :

1. La valeur optimale Z_{opt} de la fonction objectif doit être entière.
2. La valeur de la fonction objectif Z est bornée pour tout point réalisable vérifiant la propriété d'intégrité.
3. Une seule coupe est ajoutée à la fois à partir du tableau simplexe optimal.
4. La ligne source choisie pour générer une coupe est la première équation associée à une variable entière dont le membre de droite est non entier y compris l'équation correspondante à la fonction objectif Z.

Notons finalement le scepticisme de la communauté scientifique envers les algorithmes de coupes [7]. En pratique, ces derniers s'avèrent peu efficaces notamment pour la résolution de problème de grande taille [23] principalement à cause des instabilités numériques qui en découlent [25]. L'intégration de ces coupes au sein même de l'arbre d'énumération s'avère également difficile à gérer puisqu'il nécessite un grand espace de mémoire pour stocker les coupes relatives à chaque noeud d'énumération [24].

3.2 Coupes valides pour le BLP

Dans cette section, nous présentons, pour la première fois, quatre familles de coupes valides pour le problème de programmation linéaire à deux niveaux. La première famille de coupes est la coupe de Gomory. Nous étudions son expression via la reformulation mixte à la section 3.2.1. Nous générons la même coupe via la formulation initiale à la section 3.2.2. La seconde famille de coupes est la coupe simple donnée à la section 3.2.3. La troisième famille de coupes est la coupe étendue présentée à la section 3.2.4. La quatrième famille de coupes est la coupe disjonctive étudiée à la section 3.2.5.

3.2.1 Coupes de Gomory via la reformulation mixte

À partir de la reformulation mixte BLP_r, nous considérons la relaxation obtenue en enlevant les contraintes de complémentarité. Cette relaxation correspond au problème LRP enrichi des contraintes du dual du second niveau :

$$(LRP) \begin{cases} \max\limits_{x,y,\lambda} & c^{1T}x + d^{1T}y \\ & A^1x + B^1y \leq b^1, \\ & x \geq 0, \\ & A^2x + B^2y \leq b^2, \quad -B^{2T}\lambda \leq -d^2, \\ & y \geq 0, \qquad\qquad \lambda \geq 0. \end{cases}$$

Résoudre le LRP par la méthode du simplexe revient à résoudre le BLP initial relaxé et à chercher un point extrême du dual du second niveau. Si toutes les contraintes de complémentarité sont satisfaites, alors l'optimum global est atteint. Sinon, nous notons i l'indice d'une contrainte pour laquelle la condition de complémentarité est

non satisfaite. Nous discuterons du choix de i, à la section 4.1.2, dans le cas où ce choix n'est pas unique. Dans un souci de simplification de notation, nous supposons que la contrainte i est une contrainte du primal du second niveau. Notons alors s_i la variable d'écart qui lui est associée, nous avons :

$$\lambda_i s_i \neq 0.$$

À partir du tableau du simplexe, nous pouvons déduire l'expression de s_i :

$$s_i = s_{i0} - \sum_{j \in HB} g_j z_j.$$

où z_j est une variable générique qui désigne soit une des variables originelles du programme, soit une variable d'écart associée à une contrainte d'inégalité. HB est l'ensemble des variables hors base, s_{i0} et g_j sont des constantes.
De même,

$$\lambda_i = \lambda_{i0} - \sum_{j \in HB} h_j z_j.$$

où, $lambda_{i0}$ et h_j sont des constantes. Définissons ${J_1}^+ = \{j \in HB; g_j > 0\}$ et ${J_2}^+ = \{j \in HB; h_j > 0\}$. La proposition suivante donne deux inégalités valides pour le BLP associées à la contrainte de complémentarité i.

Proposition 3.2.1 *Les inégalités :*

$$\sum_{j \in {J_1}^+} g_j z_j \geq u_i s_{i0} \tag{3.1}$$

$$\sum_{j \in {J_2}^+} h_j z_j \geq (1 - u_i)\lambda_{i0} \tag{3.2}$$

sont valides pour le problème BLP.

Démonstration: La condition de complémentarité associée à la contrainte i étant

violée, nous considérons les contraintes suivantes :

$$\begin{cases} s_i \leq L(1-u_i), \\ \lambda_i \leq Lu_i, \\ u_i \text{ binaire} \end{cases}$$

où L est une constante positive de grande taille. En introduisant les variables d'écart ω et ω', on trouve

$$\begin{cases} s_i + \omega = L(1-u_i), \\ \lambda_i + \omega' = Lu_i, \\ u_i \text{ binaire}, \end{cases}$$

ce qui donne en combinant avec les expressions de s_i et de λ_i,

$$Lu_i + \omega - \sum_{j \in HB} g_j z_j = L - s_{i0}. \tag{3.3}$$

$$Lu_i + \sum_{j \in HB} h_j z_j - \omega' = \lambda_{i0}. \tag{3.4}$$

En divisant l'équation (3.3) par L, nous trouvons :

$$u_i + \tfrac{\omega}{L} - \sum_{j \in HB} \tfrac{g_j}{L} z_j = 1 - \tfrac{s_{i0}}{L}. \tag{3.5}$$

En dérivant la coupe de Gomory de cette équation, nous obtenons :

$$\tfrac{\omega}{L} - \sum_{j \in J_1^-} \tfrac{g_j}{L} z_j + \tfrac{1 - s_{i0}/L}{s_{i0}/L} \sum_{j \in J_1^+} \tfrac{g_j}{L} z_j \geq 1 - \tfrac{s_{i0}}{L}, \tag{3.6}$$

où $J_1^+ = \{j \in HB \mid g_j > 0\}$ et $J_1^- = \{j \in HB \mid g_j < 0\}$. En combinant (3.5) et (3.6), nous obtenons la première inégalité (3.1) :

$$\sum_{j \in J_1{}^+} g_j z_j \geq u_i s_{i0}.$$

De la même manière, nous obtenons la deuxième inégalité (3.2) à partir de l'équation (3.4) en faisant le changement de variable $v_i = 1 - u_i$. ∎

Les inégalités (3.1) et (3.2) forment une coupe de Gomory pour le problème BLP. Avec ces inégalités, la solution courante devient non réalisable pour le problème LRP. Notons que les coupes de Gomory ne dépendent pas de la constante L.

3.2.2 Coupes de Gomory via la formulation initiale

Les mêmes coupes de Gomory vues à la section 3.2.1 peuvent être générées à partir de la formulation initiale du BLP. Pour ce faire, nous considérons le programme linéaire suivant :

$$(LP) \begin{cases} \max\limits_{x} & c^t x \\ \text{s.c.} & Ax \leq b, \\ & x \geq 0. \end{cases}$$

Soit x_i une variable non nulle présente dans la base à l'optimalité, alors x_i s'écrit :

$$x_i = x_{i0} - \sum_{j \in HB} g_j x_j.$$

Imposer à x_i de prendre la valeur 0 revient à poser :

$$0 = x_{i0} - \sum_{j \in HB} g_j x_j,$$

ce qui implique,

$$x_{i0} - \sum_{j \in J^- \cap HB} g_j x_j = \sum_{j \in J^+ \cap HB} g_j x_j.$$

Ainsi la coupe

$$\sum_{j \in J^+ \cap HB} g_j x_j \geq x_{i0}.$$

est une condition nécessaire pour que la variable x_i prenne la valeur 0.

Les coupes établies à la section 3.2.1 se retrouvent donc facilement à partir de la relaxation LRP. Si une contrainte de complémentarité i n'est pas satisfaite, alors l'une des deux variables en jeu (s_i ou λ_i) doit être contrainte à valoir zéro. Ainsi, une des deux inégalités suivantes est valide pour le BLP :

$$\sum_{j \in J_1^+ \cap HB} g_j z_j \geq s_{i0},$$

$$\sum_{j \in J_2{}^+ \cap HB} h_j z_j \geq \lambda_{i0}.$$

En introduisant la variable binaire u_i, nous retrouvons les inégalités valides (3.1) et (3.2) :

$$\sum_{j \in J_1{}^+} g_j z_j \geq u_i s_{i0},$$

$$\sum_{j \in J_2{}^+} h_j z_j \geq (1 - u_i)\lambda_{i0}.$$

3.2.3 Coupes simples

Nous avons vu que la coupe de Gomory nécessite l'introduction d'une variable binaire u_i. Néanmoins, nous pouvons éliminer cette variable en combinant les deux inégalités des coupes 3.1 et 3.2. La proposition 3.2.2 génère une coupe moins profonde mais qui ne contient pas de variables binaires.

Proposition 3.2.2 *La coupe simple :*

$$\frac{1}{s_{i0}} \sum_{j \in J_1{}^+} g_j z_j + \frac{1}{\lambda_{i0}} \sum_{j \in J_2{}^+} h_j z_j \geq 1, \tag{3.7}$$

est valide pour le problème BLP.

Démonstration:

En divisant l'inégalité (3.1) par s_{i0} et l'inégalité (3.2) par λ_{i0}, nous obtenons :

$$\frac{1}{s_{i0}} \sum_{j \in J_1{}^+} g_j z_j \geq u_i, \tag{3.8}$$

$$\frac{1}{\lambda_{i0}} \sum_{j \in J_2{}^+} h_j z_j \geq 1 - u_i. \tag{3.9}$$

La coupe simple (3.7) est obtenue en additionnant les deux inégalités (3.8) et (3.9). ∎

3.2.4 Coupes de Gomory étendues

Les variables binaires introduites lors de la génération des coupes de Gomory peuvent apparaître dans les expressions de s_i et de λ_i tirées du tableau optimal associé au problème LRP. On pourrait alors en tenir compte dans l'expression de coupes. Nous notons z_j les variables continues génériques et w_i les variables binaires. Définissons J l'ensemble des variables continues et N l'ensemble des variables binaires. Nous pouvons supposer sans perte de généralité que les variables hors-bases sont nulles. Ainsi, les expressions de s_i et de λ_i seraient,

$$s_i = s_{i0} - \sum_{j \in HB \cap J} g_j z_j - \sum_{j \in HB \cap N} g'_j w_j \quad \text{et}$$

$$\lambda_i = \lambda_{i0} - \sum_{j \in HB \cap J} h_j z_j - \sum_{j \in HB \cap N} h'_j w_j.$$

En dérivant les coupes de Gomory de ces deux équations, on trouve :

$$\sum_{j \in J_1^+ \cap J} g_j z_j + \sum_{\substack{j \in N \\ g'_j > 0,\, g'_j < s_{i0}}} g'_j w_j + \sum_{\substack{j \in N \\ g'_j > 0,\, g'_j \geq s_{i0}}} s_{i0} w_j \geq u_i s_{i0} \quad \text{et} \qquad (3.10)$$

$$\sum_{j \in J_2^+ \cap J} h_j z_j + \sum_{\substack{j \in N \\ h'_j > 0,\, h'_j < \lambda_{i0}}} h'_j w_j + \sum_{\substack{j \in N \\ h'_j > 0,\, h'_j \geq \lambda_{i0}}} \lambda_{i0} w_j \geq (1 - u_i)\lambda_{i0}. \qquad (3.11)$$

3.2.5 Coupes disjonctives

La procédure disjonctive est une approche souvent utilisée en programmation mixte afin de générer des coupes valides [29]. Le principe de cette approche repose sur la proposition suivante :

Proposition 3.2.3 *Considérons un ensemble $X = X_1 \cup X_2$. Soient :*

$$\sum_i {\pi_i}^1 z_i \geq {\pi_0}^1$$

une coupe valide pour X_1 et

$$\sum_i {\pi_i}^2 z_i \geq {\pi_0}^2$$

une coupe valide pour X_2.

Alors :

$$\sum_i max({\pi^1}_i, {\pi^2}_i) z_i \geq min({\pi_0}^1, {\pi_0}^2)$$

est une coupe valide pour X.

Démonstration: Voir référence [29]. ∎

Appliquons maintenant la procédure disjonctive aux problèmes biniveaux. Notons X l'ensemble des solutions admissibles. Pour un entier i donné, soient X_{i1} l'ensemble des solutions de X tel que $s_i = 0$ et X_{i2} l'ensemble des solutions admissibles tel que $\lambda_i = 0$. D'après la section 3.2.2, nous avons montré que la coupe

$$\sum_{j \in J_1^+ \cap HB} g_j z_j \geq s_{i0}$$

est valide pour X_{i1} et que la coupe

$$\sum_{j \in J_2^+ \cap HB} h_j z_j \geq \lambda_{i0}$$

est valide pour X_{i2}. Or $X = X_{i1} \cup X_{i2}$. En appliquant la procédure disjonctive, nous obtenons :

$$\sum_{j \in ({J_1}^+ \cup {J_2}^+) \cap HB} \max(g_j, h_j) z_j \geq \min(\lambda_{i0}, s_{i0}). \tag{3.12}$$

3.3 Profondeur des coupes

Nous avons développé dans la section précédente quatre familles de coupes différentes pour le problème biniveau. Leurs formules relatives sont résumées au tableau 3.1.

Tableau 3.1 – Expression des coupes

Coupe	réf	Formules
Gomory	3.1	$\sum_{j \in J_1^+} g_j z_j \geq u_i s_{i0},$
	3.2	$\sum_{j \in J_2^+} h_j z_j \geq (1-u_i)\lambda_{i0}.$
Simple	3.7	$\frac{1}{s_{i0}} \sum_{j \in J_1^+} g_j z_j + \frac{1}{\lambda_{i0}} \sum_{j \in J_2^+} h_j z_j \geq 1.$
Gomory étendue	3.10	$\sum_{j \in J_1^+ \cap J} g_j z_j + \sum_{\substack{j \in N \\ g'_j > 0,\ g'_j < s_{i0}}} g'_j w_j + \sum_{\substack{j \in N \\ g'_j > 0,\ g'_j \geq s_{i0}}} s_{i0} w_j \geq u_i s_{i0},$
	3.11	$\sum_{j \in J_2^+ \cap J} h_j z_j + \sum_{\substack{j \in N \\ h'_j > 0,\ h'_j < \lambda_{i0}}} h'_j w_j + \sum_{\substack{j \in N \\ h'_j > 0,\ h'_j \geq \lambda_{i0}}} \lambda_{i0} w_j \geq (1-u_i)\lambda_{i0}.$
Disjonctive	3.12	$\sum_{j \in J_1^+ \cup J_2^+ \cap HB} \max(g_j, h_j) z_j \geq \min(\lambda_{i0}, s_{i0}).$

Dans cette section, nous allons classer ces coupes selon leurs profondeurs respectives. Une coupe est dite plus profonde si elle coupe une plus grande partie du domaine réalisable. La définition suivante permet de comparer deux coupes à partir de leurs expressions respectives [28].

Définition 3.3.0.1 *Soient*

$$\Pi_1 : \quad \sum_i {\pi_i}^1 z_i \geq {\pi_0}^1,$$

et

$$\Pi_2 : \quad \sum_i {\pi_i}^2 z_i \geq {\pi_0}^2$$

deux coupes valides. La coupe Π_1 *est dite plus profonde que* Π_2 *si* ${\pi_i}^1 \leq {\pi_i}^2$, $\forall i$ *et* ${\pi_0}^1 \geq {\pi_0}^2$.

En se basant sur cette dernière définition, les trois propositions suivantes permettent d'hiérarchiser l'ensemble des quatre familles de coupes (présentées au tableau 3.1) selon leurs profondeurs.

Proposition 3.3.1 (Comparaison entre coupe simple et coupe de Gomory) *La coupe de Gomory est plus profonde que la coupe simple.*

Démonstration: Le résultat découle directement de la définition de la coupe simple donnée à la section 3.2.3. ∎

Proposition 3.3.2 (Comparaison entre coupe de Gomory et coupe de Gomory étendue)
La coupe Gomory étendue est plus profonde que la coupe de Gomory.

Démonstration: Considérons l'expression de s_i tiré du tableau optimal associé au problème LRP :

$$s_i = s_{i0} - \sum_{j \in HB} g_j z_j.$$

En dérivant la coupe primale (3.1) de Gomory, nous obtenons :

$$\sum_{j \in J_1{}^+} g_j z_j + \sum_{g_j{}' > 0} g_j{}' w_j \geq u_i s_{i0}.$$

Or la formule de Gomory étendue diffère de celle de Gomory uniquement pour les coefficients des variables binaires. Soit π_j le coefficient associé à la variable binaire w_j dans la coupe primale (3.10) de Gomory étendue. On a $\pi_j = \min\{g_j{}', s_{io}\}$. D'où, $\pi_j \leq g_j{}'$ et, selon la définition, la coupe primale de Gomory étendue est plus profonde que celle de Gomory. Le même raisonnement est valable pour les coupes associées à la variable duale, ce qui complète la démonstration. ∎

Proposition 3.3.3 (Comparaison entre coupe simple et coupe disjonctive) *Soit* $J_1^+ = \{j \in HB; g_j \geq 0\}$ *et* $J_2^+ = \{j \in HB; h_j \geq 0\}$. *Supposons que* $J_1^+ \cap J_2^+ = \emptyset$ *alors la coupe simple est plus profonde que la coupe disjonctive.*

Démonstration: Pour tout $j \in J_1^+$, $g_j/s_{i0} \leq \max\{g_j, h_j\}/\min\{s_{i0}, \lambda_{i0}\}$. De même pour tout $j \in J_2^+$, $h_j/\lambda_{i0} \leq \max\{g_j, h_j\}/\min\{s_{i0}, \lambda_{i0}\}$.
Ainsi, d'après la définition 3.3.0.1, la coupe simple :

$$\frac{1}{s_i} \sum_{j \in {J_1}^+} g_j z_j + \frac{1}{\lambda_i} \sum_{j \in {J_2}^+} h_j z_j \geq 1$$

est plus profonde que la coupe disjonctive :

$$\sum_{j \in ({J_1}^+ \cup {J_2}^+) \cap HB} \max\{g_j, h_j\} z_j / \min(s_{i0}, \lambda_{i0}) \geq 1.$$

■

3.4 Discussion

Nous avons développé dans ce chapitre plusieurs coupes valides pour le problème linéaire biniveau. Ces coupes ont été générées à partir d'approches inhérentes à la programmation mixte. Nous les avons, ensuite, classées en utilisant le critère de profondeur de coupe. Nous avons trouvé que la coupe Gomory étendue est la plus profonde alors que la coupe disjonctive est la moins profonde. Dans le chapitre 4, nous allons utiliser ces coupes afin de mettre au point un algorithme de résolution pour le problème biniveau.

CHAPITRE 4

Présentation de l'algorithme de coupes

Nous proposons dans ce chapitre un algorithme permettant la résolution d'un problème BLP. Cet algorithme comprend deux étapes principales. La première est une procédure de coupes alors que la deuxième est une énumération implicite des solutions admissibles. Outre *la relaxation de premier niveau* LRP du problème courant (décrite dans la section 3.2.1), l'algorithme considère *la relaxation du second niveau* du problème courant $FRP(\bar{x})$ et *le sous-problème du second niveau* du problème initial $FSP(\bar{x})$. Dans les deux problèmes, x est fixé à $\bar{x}$; où $(\bar{x}, \bar{y})$ est la solution optimale de la relaxation LRP. La première étape de l'algorithme correspond à une phase de *pré-traitement* qui permet de réduire la valeur de la borne supérieure à chaque noeud lors de la phase d'énumération. C'est pourquoi, nous imposons un nombre maximal d'itérations pour l'étape de coupes, au-delà duquel si l'algorithme ne converge pas, nous passons à l'étape de branchement.

Afin d'assurer la faisabilité du problème LRP enrichi des contraintes duales du second niveau, nous émettons l'hypothèse suivante :

H1 : Le domaine réalisable du BLP est borné et non vide.

Cette hypothèse garantit l'existence d'une base optimale à partir de laquelle on peut

générer les coupes.

Ce chapitre présente en premier lieu la procédure de coupes à la section 4.1 avant d'aborder l'étape d'énumération à la section 4.2. Une description de l'algorithme est ensuite donnée à la section 4.3. Finalement, afin d'illustrer les étapes de l'algorithme, un exemple est détaillé à la fin du chapitre.

4.1 Présentation de la procédure de coupes

La procédure de coupes permet d'enrichir le système de contraintes par un ensemble de coupes valides de même type. Elle se compose essentiellement de trois parties : l'application des tests, le choix de la contrainte source et l'introduction de la coupe.

Notons que cette procédure n'assure pas nécessairement la convergence vers la solution optimale, puisque la reformulation mixte du problème BLP ne satisfait pas, en général, la première condition de Taha [28] présentée à la section 3.1 pour montrer la convergence de l'algorithme de Gomory. Rappelons que cette condition suppose l'intégrité de la valeur optimale Z. Cependant, l'intérêt de la procédure de coupes réside plutôt dans la réduction de la valeur de la borne supérieure pour les noeuds d'énumération.

4.1.1 Application des tests

La première étape de la procédure consiste à examiner l'ensemble des contraintes de complémentarité en appliquant à la fois un test d'admissibilité et un test conditionnel.

Le test d'admissibilité permet de vérifier la rationalité de la solution optimale de LRP en calculant les produits $s_i\lambda_i$ pour $0 \leq i \leq n_2 + m_2$, s_i étant la variable d'écart associée à la contrainte primale i en incluant les contraintes de signe et λ_i la variable duale complémentaire.

Le test conditionnel est appliqué sur les contraintes de complémentarité violées. Ce test repose sur la proposition suivante :

Proposition 4.1.1 *Soit i l'indice d'une contrainte de complémentarité violée. Considérons les expressions de s_i et de λ_i tirées du tableau optimal associé au problème LRP :*

$$s_i = s_{i0} - \sum_{j \in HB} g_j z_j,$$
$$\lambda_i = \lambda_{i0} - \sum_{j \in HB} h_j z_j.$$

(i) Si, $\forall j \in HB$, $g_j \leq 0$ alors à l'optimum, $\lambda_i = 0$;

(ii) Si, $\forall j \in HB$, $h_j \leq 0$ alors à l'optimum, $s_i = 0$.

Démonstration: Puisque $s_i\lambda_i \neq 0$ alors $s_{i0} > 0$ et $\lambda_{i0} > 0$. D'après la section 3.2.2, la condition $\sum_{j \in J^+ \cap HB} g_j z_j \geq s_{i0}$ est nécessaire pour avoir $s_i = 0$. Cette condition ne serait jamais satisfaite si pour tout $j \in HB$, g_j est non positif. En particulier, à l'optimum, on aura $s_i \neq 0$. Or, toute solution optimale est rationnelle. D'où $\lambda_i = 0$ à l'optimum.

De la même manière, on démontre la deuxième partie de la proposition. ∎

Ainsi, le test conditionnel examine les coefficients des expressions de s_i et λ_i. Si l'expression d'une des deux variables ne comprend pas de coefficients positifs, on peut fixer l'autre variable à 0. Le test conditionnel est illustré à l'exemple 4.1.1.

Exemple 3

Soit le problème biniveau :

$$\max_{x,y} -x - y$$
$$s.c. \begin{cases} x \leq 1, \\ x \geq 0, \\ y \in \arg\max_{v} v, \\ \qquad s.c. \begin{cases} x + v \leq 2, \\ v \geq 0. \end{cases} \end{cases}$$

Considérons sa relaxation LRP :

$$\max_{x,y,\lambda_1} -x - y$$
$$s.c. \begin{cases} x \leq 1, \\ x \geq 0, \\ x + y \leq 2, \quad \lambda_1 \geq 1, \\ y \geq 0, \qquad \lambda_1 \geq 0. \end{cases}$$

Notons s_0 l'écart associé à la contrainte primale du premier niveau, s_1 l'écart associé à celle du second niveau et enfin r celui associé à la seule contrainte duale. Partant de ces notations, l'admissibilité peut être vérifiée en calculant les produits des écarts complémentaires $s_1\lambda_1$ et yr.

En résolvant ce problème à l'aide de la méthode du simplexe, nous trouvons comme base optimale $\{s_0, s_1, \lambda_1\} = \{1, 2, 1\}$. Cependant, cette solution n'est pas rationnelle car $s_1\lambda_1 \neq 0$. Appliquons alors le test conditionnel. Considérons les lignes du tableau

simplexe optimal associées aux variables complémentaires s_1 et λ_1 :

$$\begin{cases} s_1 = 2 - x - y \\ \lambda_1 = 1 + r \end{cases}$$

On remarque que dans la ligne associée à λ_1, la seule variable hors-base r est affecté d'un coefficient négatif $(1 - (-r))$. D'après la proposition 4.1.1, on aura $s_1 = 0$. Ainsi, en fixant la variable s_1 et en réoptimisant le problème LRP, nous trouvons une base optimale $\{x, y, \lambda_1\} = \{1, 1, 1\}$. Nous vérifions que la solution qui lui est associée est admissible car toutes les contraintes de complémentarité sont satisfaites. D'où la solution $(x, y) = (1, 1)$ est optimale pour le problème BLP. Notons que l'optimalité est atteinte en utilisant uniquement le test d'admissibilité sans l'ajout de coupes.

4.1.2 Sélection de l'indice de la contrainte source

Soit V l'ensemble des contraintes de complémentarité violées. L'efficacité de la procédure dépend essentiellement du choix de la contrainte $i \in V$, appelée contrainte source, à partir de laquelle la coupe est générée. Pour cela, nous avons établi un ensemble de critères :

1. CS1 : Ce critère correspond au principe de séparation utilisé dans l'algorithme de Bard et Moore [10]. Il consiste à choisir comme contrainte source une contrainte de complémentarité $i \in \arg\max_{j \in V}(\lambda_j s_j)$.
2. CS2 : Il s'agit d'un critère hybride qui tient compte des valeurs des variables binaires u_j. En attribuant la valeur 0 aux variables u_j qui ne sont pas introduites dans le problème, ce critère choisit comme contrainte source une contrainte d'indice $i \in \arg\min_j\{|\frac{1}{2} - u_j|\}$. Si $\min_j\{|\frac{1}{2} - u_j|\} = \frac{1}{2}$, la contrainte i est choisie selon le critère CS1.

3. CS3 : La contrainte source est la première contrainte de complémentarité violée. On aura $i = \min_j\{j \in V\}$. Ce critère est équivalent à celui adopté par l'algorithme de Gomory [17] pour la résolution du problème mixte. Il figure également parmi les conditions nécessaires utilisées par Taha [28] pour montrer la convergence de l'algorithme.

4. CS4 : Le critère CS4 choisit comme contrainte source une contrainte d'indice $i \in \arg\max_{j\in V}\{\min(\lambda_j, s_j)\}$. Il est utilisé lorsque l'ordre de grandeur des variables duales est le même que celui des variables primales.

5. CS5 : Soit *it* une itération de coupe. Considérons U l'ensemble des contraintes de complémentarité utilisées comme contraintes sources dans les itérations précédant *it*. Le choix de la contrainte source i pour l'itération *it* est établi selon la formule suivante :

$$i = \begin{cases} \min\limits_{j\in V/U}\{j\}, \text{ si } V/U \neq \emptyset. \\ \min\limits_{j\in U}\{j\}, \text{ sinon.} \end{cases}$$

Ce critère permet de varier les contraintes sources afin d'enrichir le problème LRP par des coupes de formules différentes.

6. CS6 : Le critère CS6 repose sur une technique utilisée en programmation mixte à savoir le calcul de pénalité [12]. Soit k une contrainte de complémentarité violée. Considérons les équations tirées du tableau optimal et associées à la la fonction objectif Z ainsi qu'aux variables s_k et λ_k :

$$\begin{aligned} Z &= Z_j^* - \sum_{j\in HB} c_j^* z_j, \\ s_k &= s_{k0} - \sum_{j\in HB} g_j z_j, \\ \lambda_k &= \lambda_{k0} - \sum_{j\in HB} h_j z_j, \end{aligned}$$

où c_j^* est le coût réduit de la variable z_j.

Nous définissons $p_{s_k} = s_{k0} \min_{j \in HB: g_j > 0} \frac{c_j^*}{g_j}$ comme la variation de la fonction objectif Z lors de la première itération duale de la méthode du simplexe après l'ajout de la contrainte $s_k = 0$. Elle correspond à la pénalité pour fixer s_k à 0. De même la pénalité associée pour fixer λ_k à 0 est $p_{\lambda_k} = \lambda_{k0} \min_{j \in HB: h_j > 0} \frac{c_j^*}{h_j}$.

Ainsi, nous associons à la contrainte de complémentarité k la pénalité suivante :

$$P_k = \min\{p_{s_k}, p_{\lambda_k}\}.$$

Cette dernière est une mesure approximative de la décroissance de la fonction objectif suite à l'ajout d'une coupe générée à partir de la contrainte source k. En particulier, elle donne une borne inférieure pour la décroissance de Z si on introduit une coupe simple. Le critère CS6 choisit comme contrainte source une contrainte de complémentarité de pénalité maximale :

$$i \in \arg\max_j P_j.$$

En effet, ce choix pourrait correspondre à la plus forte décroissance de la fonction objectif.

4.1.3 Introduction de la coupe

L'introduction de la coupe correspond à la dernière étape d'une itération de *prétraitement*. Elle consiste à ajouter la coupe associée à la contrainte source. Contrairement aux coupes simples et disjonctives, le processus d'intégration de la coupe de Gomory et de la coupe étendue dépend néanmoins de la variable binaire u. En notant

i l'indice de la contrainte source et en considérant une constante suffisamment grande L, l'introduction de ces coupes particulières se fait selon le schéma suivant :

1. Si u_i n'existe pas à l'itération courante alors nous l'introduisons et nous ajoutons la coupe de Gomory (ou étendue) associée à la contrainte source i.
2. Si u_i est déjà présente à l'itération courante alors nous vérifions sa valeur. Trois cas se présentent :
   - Si $0 < u_i < 1$, alors nous introduisons la coupe de Gomory (ou étendue) associée à la contrainte source.
   - Si $u_i = 1$, alors nous rajoutons la coupe $s_i \leq L(1 - u_i)$
   - Si $u_i = 0$, alors nous rajoutons la coupe $\lambda_i \leq Lu_i$.

Ce schéma permet d'intégrer les contraintes assurant la complémentarité des variables s_i et λ_i.

4.1.4 Extension : nettoyage des coupes

En rajoutant les coupes, la taille du problème LRP augmente et sa résolution devient de plus en plus compliquée. Sachant que pendant la phase d'énumération, on fera appel à ce problème à chaque noeud, il serait plus approprié de définir une stratégie permettant de garder les coupes les plus efficaces. Nous avons adopté une stratégie utilisée par Alarie et al. [1] pour la résolution des problèmes bilinéaires.
Deux critères permettent de caractériser l'efficacité d'une coupe $a_i x \leq b_i$ à une itération i avec $(a_i \in \mathbb{R}^n, b_i \in \mathbb{R})$. Le premier correspond à la profondeur de la coupe évaluée en terme de décroissance de la fonction objectif. Le deuxième repose sur la valeur du cosinus de l'angle entre la coupe et chacune des coupes générées avant l'itération i. Rappelons que le cosinus de deux vecteurs normaux π et τ est égal à $\frac{\pi.\tau}{\|\pi\|.\|\tau\|}$ avec $\pi.\tau$ est le produit scalaire de π et τ. En nous basant sur ces deux critères,

nous avons adopté une stratégie de nettoyage pour les coupes simples vu la simplicité de leur formule qui consiste en une seule ligne de contrainte.
Nous avons constaté, d'après les tests, que les coupes générées pendant les premières itérations sont les plus profondes. C'est pourquoi, nous avons défini un paramètre de décision [1] : $C_i = \sum_{\tau \in \Omega} \frac{\pi.\tau}{\|\pi\|.\|\tau\|}$, Ω étant l'ensemble des coupes déjà générées. Si C_i est inférieur à la moitié de la cardinalité de Ω, nous interrompons la phase de coupes. Ainsi, tout en gardant les coupes les plus profondes, ce critère assure une large variété de celles-ci. En effet, une coupe n'est rajoutée que si la moyenne des cosinus des angles qu'elle forme avec celles qui la précèdent est inférieur à $\frac{1}{2}$. Elle n'est donc introduite que si la moyenne des angles est supérieure à $\frac{\pi}{3}$.

4.2 Présentation de la procédure d'énumération

La deuxième phase de l'algorithme consiste en une procédure d'optimisation par séparation (*Branch and Bound*) inspirée de l'algorithme HJS [18] pour la résolution des problèmes biniveaux. La procédure considère *la relaxation de premier niveau LRP* enrichi éventuellement des coupes générées pendant la première phase ainsi que *la relaxation du second niveau FRP* et *le sous-problème du second niveau FSP*. Elle consiste en une énumération implicite des solutions admissibles. La séparation se fait selon les contraintes de complémentarité violées. Pour ce faire, nous associons une variable binaire α_i à la $i^{ème}$ contrainte de second niveau en incluant les contraintes de signe. Fixer α_i à 1 revient à fixer dans LRP et FRP la contrainte primale à égalité. Si la variable est à 0, la contrainte duale correspondante est fixée à égalité. Au cas où LRP contient une variable u_i, le fait de fixer α_i à une de ses bornes revient à fixer, dans ce même problème, u_i à la même borne. Notons que les variables α ne sont pas explicitement ajoutées aux programmes. Elles sont utilisées pour la définition d'un

sommet. Ainsi, si on considère l'ensemble des solutions en α

$$S = \{\{\alpha_1, .., \alpha_{n_2+m_2}\} | \alpha_i \in \{0, 1\}\}$$

alors chaque sommet de l'arborescence correspond à un sous-ensemble de S où certaines variables booléennes α_i sont fixées soit à 0 soit à 1 alors que les autres sont libres. Le sous-problème correspondant à un sommet est identique au problème biniveau initial dont certaines contraintes seront prises à égalité et d'autres à inégalité stricte. La résolution de ce problème à deux niveaux étant difficile, l'étude du sommet sera effectuée à partir des relaxations LRP et FRP.

Ainsi, un sommet sera examiné par l'évaluation de la borne supérieure qui correspond à la valeur optimale de LRP et par application d'un ensemble de tests. Rappelons qu'un test consiste en l'application d'une condition permettant de sonder un sommet. Trois tests ont été utilisés dans notre algorithme :

1. Test de faisabilité : Ce test vérifie la faisabilité du sous-problème courant en résolvant les problèmes LRP et le dual de FRP. Si l'un des deux est non réalisable alors le sommet est sondable.

2. Test d'optimalité : Le test d'optimalité détermine si la valeur optimale de la relaxation LRP du sommet courant est inférieure à la valeur de la meilleure solution obtenue jusqu'à date. Dans le cas échéant, le sommet est sondable.

3. Test de résolution : Ce test détermine si la solution $(\bar{x}, \bar{y})$ de la relaxation LRP du sous-problème courant est une solution admissible du problème initial. Ainsi, si $(\bar{x}, \bar{y})$ est dans le domaine induit, alors le sommet est sondable.

Par ailleurs, le schéma d'exploration des différents sommets se fait selon un principe dit *de séparation* qui permet d'élaborer une arborescence de solutions. Ce principe comprend à la fois le critère de séparation et la méthode d'exploration.

Le critère de séparation :

Le critère de séparation est une stratégie permettant de choisir la variable α_i à fixer à chaque sommet de l'arborescence. Plusieurs critères peuvent être adoptés. On y distingue :

1. Le critère de Bard et Moore : Ce critère choisit comme variable de branchement la variable α_i telle que $i \in \arg\max_j\{s_j\lambda_j\}$. La valeur de s_j et celle de λ_j sont obtenues par la résolution du problème relaxé LRP. Ce critère ne puise pas d'information du problème $FRP(\bar{x})$. Par conséquent, il ne tient pas compte de la fonction objectif du second niveau.

2. Le critère HJS : Il est identique au critère de Bard et Moore à la seule différence que la valeur de λ_j est obtenue en résolvant le dual du problème $FRP(\bar{x})$. Ce critère permet, ainsi, d'exploiter à la fois l'information disponible dans la relaxation primale du sous-problème et celle donnée par le dual de la relaxation du second niveau.

3. Le critère max-min : Il choisit comme variable de branchement la variable α_i telle que $i \in \arg\max_j\{\min\{s_j, \lambda_j\}\}$. La valeur de s_j étant fournie par le problème relaxé LRP et celle de λ_j est obtenue en résolvant le dual du problème $FRP(\bar{x})$.

La méthode d'exploration :

On distingue trois méthodes d'exploration :

1. Méthode en profondeur d'abord : Dans cette méthode, on choisit d'explorer le sommet le plus profond dans l'arborescence.

2. Méthode en largeur d'abord : La méthode en largeur d'abord, contrairement à celle de profondeur d'abord, donne une plus grande priorité aux niveaux moins profonds du graphe de recherche, en explorant progressivement les sommets par couche de même profondeur.

3. Méthode meilleure d'abord : Dans cette méthode, on choisit d'explorer le sommet ayant la meilleure borne supérieure étant susceptible de contenir une solution optimale parmi ses successeurs.

Les deux dernières méthodes nécessitent, généralement, l'exploration d'un nombre plus important de sommets avant l'obtention d'une solution admissible. Nous avons implanté la première méthode pour sa simplicité d'implantation et car c'est la méthode choisie par HJS [18], Audet et al. [5] et Bard et Moore [10] pour la résolution des problèmes biniveaux ou maxmin.

4.3 Schéma général de l'algorithme

À partir des procédures vues en 4.1 et en 4.2, nous avons conçu un algorithme $Al(coupes)$ pour la résolution du problème biniveau sous l'hypothèse **H1** intégrant à la fois la technique de coupes et celle de l'énumération. Le schéma général de algorithme est le suivant :

PHASE I : Procédure de coupes

Pour $it = 1, \ldots, Max$

1. Test d'admissibilité

Résoudre le problème LRP.

Si toutes les contraintes de complémentarité sont satisfaites alors **fin** : la solution obtenue est optimale.

2. Test conditionnel

Appliquer le test conditionnel sur l'ensemble des contraintes de complémentarité :

S'il existe une variable s_i dont l'expression tirée du tableau optimal ne comporte pas des coefficients positifs, alors fixer α_i à 0. De même, s'il existe une variable λ_i dont l'expression tirée du tableau optimal ne comporte pas des coefficients positifs, alors fixer $\alpha_i = 1$.

Si, à l'issue de ce test, une ou plusieurs variables ont été fixées, alors aller à l'étape 1.

3. Sélection de la contrainte source

Soit V l'ensemble des contraintes de complémentarité violées. Appliquer un critère de sélection prédéterminé pour choisir la contrainte $i \in V$ à partir de laquelle la coupe sera générée.

4. Introduction de la coupe

Ajouter la coupe associée à la contrainte i.

Si $it = Max$ aller à Phase II.

PHASE II : Énumération

Considérer le problème LRP enrichi des coupes générées pendant la Phase I ainsi que les problèmes FRP et FSP.

1. Initialisation

Fixer z_{opt} à $-\infty$.

2. Premier test de faisabilité

Si LRP est non réalisable, aller en **7**.

3. Évaluation de la borne supérieure

Soit $(\bar{x}, \bar{y})$ une solution optimale de LRP et $\bar{z}$ la valeur optimale correspondante. Appliquer le test d'optimalité :

Si $\bar{z} < z_{opt}$, aller à l'étape **7**.

4. Second test de faisabilité

Résoudre le dual de $FRP(\bar{x})$.

S'il n'est pas réalisable, aller à l'étape **7**.

5. Test de résolution

Vérifier si $(\bar{x}, \bar{y})$ est rationnel : Résoudre $FSP(\bar{x})$ et poser y_{FS} une solution optimale.

Si $d^{2T}\bar{y} = d^{2T}y_{FS}$ alors $(\bar{x}, \bar{y})$ est rationnel. Mettre à jour z_{opt} et (x_{opt}, y_{opt}) et aller en **7**.

6. Branchement

Choisir une contrainte de branchement α_i où $i \in \{1, 2..., m_2 + n_2\}$ en utilisant un critère de séparation prédéterminé. Fixer $\alpha_i = 1$, ce qui crée un nouveau sous-problème et aller à l'étape **2** (pour résoudre complètement ce sous-problème).

Lorsque le sous-arbre enraciné à $\alpha_i = 1$ est complètement exploré, libérer toutes les variables fixées à l'intérieur de ce sous-arbre. Fixer $\alpha_i = 0$, ce qui crée un nouveau sous-problème et aller à l'étape **2** (pour résoudre complètement ce sous-problème).

Lorsque le sous-arbre enraciné à $\alpha_i = 0$ est complètement exploré, libérer toutes les variables fixées à l'intérieur de ce sous-arbre.

7. Retour arrière

Le noeud courant est sondable.

S'il s'agit de la racine, arrêter ; (x_{opt}, y_{opt}) est une solution optimale de valeur z_{opt}.

Sinon, continuer le branchement à l'étape **6** au noeud père de l'arbre d'exploration.

Notons que la génération des coupes pendant la phase I est effectuée à partir de bases optimales. L'hypothèse **H1** garantit l'existence d'une telle base à chaque itération. Sous cette hypothèse, nous montrons à la proposition suivante la convergence de notre algorithme.

Proposition 4.3.1 *Sous l'hypothèse* **H1**, *l'algorithme de coupes résout le problème BLP en un temps fini.*

Démonstration:

D'après Hansen et al. [18], le BLP admet soit une solution optimale en un point extrême du domaine induit soit il n'admet pas de solution. L'algorithme exécute en premier lieu la phase de coupes. L'hypothèse **H1** assure l'existence d'une base optimale à chaque itération permettant de générer une coupe valide qui n'élimine pas la solution optimale. La génération d'une coupe se fait en un temps fini et le nombre total de coupes générées est inférieur à une constante Max. Il s'ensuit que la procédure de coupes se termine en un temps fini.

A l'issue de cette première phase, deux possibilités se présentent selon que la phase d'énumération est exécutée ou pas :

1. Si la phase d'énumération n'est pas appelée, la solution obtenue par application de la phase de coupes est admissible et donc optimale puisqu'elle constitue une solution optimale de la relaxation LRP. Ainsi, l'algorithme résout le problème BLP en un temps fini.

2. Si la phase d'énumération est appelée, alors il doit résoudre le même problème BLP enrichi d'un nombre fini de coupes valides au premier niveau. Brancher selon les contraintes de complémentarité α_i implique une énumération implicite de toutes les bases du second niveau. Ainsi, une solution optimale est éventuellement trouvée.

   Chaque noeud de l'arbre traite un nombre fini de programmes linéaires dont la taille est finie. Or, le nombre de noeuds est inférieur au nombre des combinaisons possibles des variables α_i. Le nombre des variables α_i étant fini, l'algorithme se termine en un temps fini.

■

Afin d'illustrer l'algorithme de coupes, nous proposons de résoudre l'exemple donné par Candler et Townsley [15] :

Exemple 4

$$\max_{x,y} \; 8x_1 + 4x_2 - 4y_1 + 40y_2 + 4y_3$$

$$\text{s.c.} \begin{cases} x_1, x_2 \geq 0, \\ y \in \arg\max_{w} -x_1 - 2x_2 - w_1 - w_2 - 2w_3, \\ \quad \text{s.c.} \begin{cases} w_1 - w_2 - w_3 \geq -1, \\ -2x_1 + w_1 - 2w_2 + 0.5w_3 \geq -1, \\ -2x_2 - 2w_1 + w_2 + 0.5w_3 \geq -1, \\ w_1, w_2, w_3 \geq 0. \end{cases} \end{cases}$$

Remarquons que la fonction objectif du second niveau contient le vecteur du premier niveau x. Ce dernier, considéré comme une constante dans la résolution du sous-problème du second niveau, peut être enlevé. Appliquons l'algorithme $Al(coupes)$ en utilisant les coupes de Gomory. Pour ce faire, nous imposons un nombre maximal d'itérations de coupes $Max = 5$, nous choisissons le critère $CS1$ comme critère de sélection et nous retenons le critère HJS comme critère de séparation.

PHASE I :

Considérons la relaxation LRP :

$$\max_{x,y} 8x_1 + 4x_2 - 4y_1 + 40y_2 + 4y_3$$

$$\text{s.c.} \begin{cases} y_1 - y_2 - y_3 \geq -1, & (\alpha_1) \\ -2x_1 + y_1 - 2y_2 + 0.5y_3 \geq -1, & (\alpha_2) \\ -2x_2 - 2y_1 + y_2 + 0.5y_3 \geq -1, & (\alpha_3) \\ y_1 \geq 0, & (\alpha_4) \\ y_2 \geq 0, & (\alpha_5) \\ y_3 \geq 0, & (\alpha_6) \\ x_1, x_2 \geq 0. \end{cases}$$

Pour chaque contrainte de second niveau (i), nous associons la variable d'écart s_i, la contrainte de complémentarité associée à α_i et la variable duale λ_i. Notons que pour $i > 3$, on a $s_i = y_{i-3}$. A la première itération de coupes, l'algorithme trouve une solution optimale pour LRP : $x = (0, 0)$, $y = (1.5, 1.5, 1)$ et $\lambda = (0, 0, 0, 1, 1, 2)$ avec $z = 58$. D'après le test d'admissibilité, cette solution n'est pas rationnelle car les contraintes de complémentarité associées à α_4, α_5 et α_6 sont violées. L'étape **3** choisit comme contrainte source la contrainte α_6 selon le critère **CS1**. En effet, on a $s_6\lambda_6 = y_3\lambda_6 = 2 = \max\{\lambda_i s_i \mid 1 \leq i \leq 6\}$. En dérivant la coupe de Gomory à partir des lignes du tableau du simplexe associées aux variables y_3 et λ_6, l'étape **4** introduit la variable binaire u_6 et les deux contraintes suivantes :

$$\begin{cases} \frac{2}{3}x_2 + s_1 + \frac{1}{3}s_3 - u_6 \geq 0 \\ \frac{1}{2}\lambda_2 + \frac{1}{2}\lambda_3 + 2u_6 \geq 2. \end{cases}$$

Ainsi, s'achève la première itération de la phase de coupes. Les autres itérations se déroulent de la même manière. Les résultats sont donnés dans le tableau 4.1. La première colonne du tableau correspond au numéro de l'itération *it* de la phase de coupes. Pour chaque itération i, nous donnons la valeur optimale de la fonction objectif $\bar{z}$, l'indice de la contrainte source i, l'écart primal s_i et la valeur de la variable duale λ_i. Enfin, un commentaire sera donné à la dernière colonne concernant la variable binaire u_i.

Tableau 4.1 – Illustration de la procédure de coupe de l'exemple 4

it	$\bar{z}$	i	s_i	λ_i	commentaire
1	58	6	1	2	ajout de u_6
2	58	5	1	6	ajout de u_5
3	58	4	1	6	ajout de u_4
4	43	4	0.75	3	$u_4 = 0.5$
5	38.19	5	1.05	1.8	$u_5 = 0.3$

Notons que la solution duale obtenue au terme de la phase de coupes n'est pas

admissible. Cependant, la valeur de la fonction objectif a diminué. On entame ainsi la phase d'énumération avec une meilleure borne supérieure qui sera évaluée à la première itération de la procédure de séparation.

PHASE II :

z_{opt} est initialisée à $-\infty$. La relaxation du premier niveau LRP étant réalisable, permet de fournir une borne supérieure $\bar{z} = 38.2$. Cette valeur est obtenue pour la solution $x = (0, 0)$ et $y = (1.05, 1.05, 0.1)$. Le second test vérifie la faisabilité du dual du problème $FRP(\bar{x})$ où $\bar{x} = (0, 0)$:

$$\max_{x,y} \quad -y_1 - y_2 - 2y_3$$

$$\text{s.c.} \begin{cases} y_1 - y_2 - y_3 \geq -1, & (\alpha_1) \\ y_1 - 2y_2 + 0.5y_3 \geq -1, & (\alpha_2) \\ -2y_1 + y_2 + 0.5y_3 \geq -1, & (\alpha_3) \\ y_1 \geq 0, & (\alpha_4) \\ y_2 \geq 0, & (\alpha_5) \\ y_3 \geq 0, & (\alpha_6). \end{cases}$$

La résolution du dual de $FRP(\bar{x})$ permet de donner une solution duale optimale pour le sous-problème : $\lambda = (1, 1, 1, -1, -1, -2)$. Le test de résolution du problème $FSP(\bar{x})$ montre que la solution $(\bar{x}, \bar{y})$ n'est pas admissible. L'application du critère de séparation HJS permet de brancher selon la contrainte associée α_4 car $s_4\lambda_4 = y_1\lambda_4 = \max\{s_i\lambda_i \mid 0 \leq i \leq 6\}$, λ étant la solution du problème $FRP(\bar{x})$.

Nous obtenons un premier noeud fils en fixant α_4 à 1. L'exploration de ce noeud montre que la solution de LRP du sous problème est rationnelle et fournit une

première solution admissible $x_{opt} = (0, 0.9)$, $y_{opt} = (0, 0.6, 0.4)$ ayant pour valeur $\bar{z} = 29.2$. Ce sommet étant sondé, nous explorons le deuxième fils obtenu en fixant α_4 à 0. Le test d'optimalité montre que la borne supérieure associée à ce sommet $\bar{z} = 28.88$ est inférieur à la valeur de la meilleure solution à date. Ainsi le sommet est sondable.

En conclusion, l'algorithme fournit comme solution optimale $x_{opt} = (0, 0.9)$, $y_{opt} = (0, 0.6, 0.4)$ et $z_{opt} = 29.2$. L'optimalité est atteinte en explorant seulement 2 noeuds alors que l'algorithme de Bard et Moore résout le même problème en 10 noeuds. Notons que l'application de la seule procédure d'énumération nécessite également l'utilisation de 10 sommets de branchement. La figure 4.1 illustre l'arbre de branchement associé à l'algorithme $Al(coupes)$.

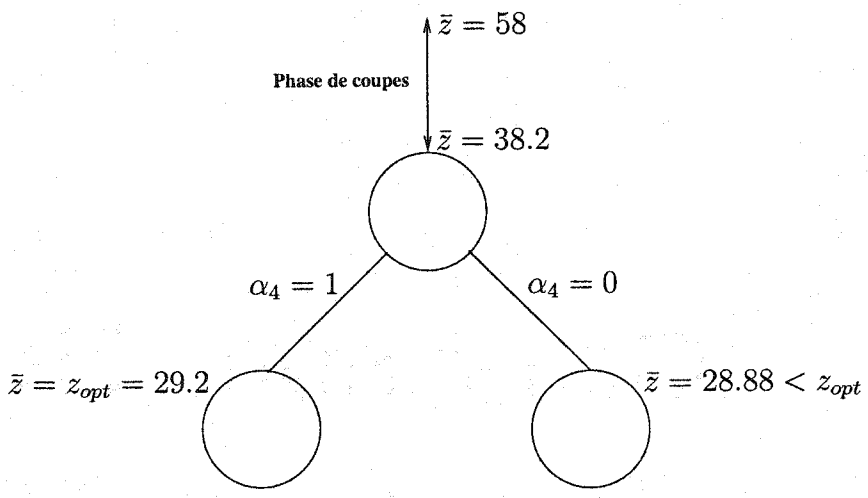

Figure 4.1 – L'arbre de branchement de l'exemple 4

CHAPITRE 5

Résultats de l'algorithme

5.1 Introduction

Ce chapitre est organisé comme suit. Dans la section 5.2, nous présentons la méthode utilisée pour la générer les problèmes tests. Dans la section 5.3, nous présentons un ensemble de tests comparatifs sur les différents paramètres de coupes à savoir le critère de sélection, le type de coupe et le nombre de coupes introduites. Dans ces séries de tests, nous adoptons le critère HJS puisqu'il constitue le meilleur critère de séparation selon Hansen et al. [18]. Dans la section 5.4, nous confirmons ce résultat à travers une série de tests portant sur les critères d'énumération. Finalement, nous évaluons la performance de notre algorithme $Al(coupes)$ en comparaison avec la résolution CPLEX de la reformulation mixte.

5.2 Génération des problèmes tests

L'algorithme codé en langage C utilise les librairies CPLEX 6.0 [16] pour la résolution des problèmes linéaires. Les résultats présentés dans ce chapitre ont été obtenus sur une station ULTRA 60, sous le système d'exploitation Solaris 2.7-05. Les tableaux contiennent les moyennes (μ) et les écarts types (σ) de dix problèmes

générés au hasard de densité D. La génération des problèmes biniveaux s'inspire de la méthode adoptée par Audet et al. [5] pour générer des problèmes bilinéaires. Les composantes des vecteurs c^1, d^2, b^1 et b^2 sont choisies aléatoirement et uniformément dans l'intervalle $[-10, 10]$. Celles de d^1 sont choisies dans l'intervalle $[10, 20]$. Pour chaque élément des matrices A^1, A^2, B^1 et B^2, un nombre aléatoire entre 0 et 1 est généré. Si ce nombre est supérieur à D alors l'élément est fixé à 0. Les éléments des matrices A^1 et B^2 sont choisis dans l'intervalle $[-20, 20]$. À la différence du générateur des problèmes bilinéaires [5], les éléments des matrices A^2 et B^1 sont choisis dans l'intervalle $[-12, 8]$. Cette plage de valeurs paraît la mieux adaptée pour la génération des problèmes BLP puisqu'elle assure un bon niveau de difficulté. Afin d'obtenir un domaine relaxé borné, la contrainte $\sum_j x_j + \sum_j y_j \leq n_1 + n_2$ est introduite. De plus, des coefficients non nuls sont ajoutés aux lignes ou colonnes vides. Il s'ensuit que la densité observée est supérieure au paramètre D. La densité obtenue sera indiquée entre parenthèse suite au choix du paramètre D. Finalement, le paramètre L de l'algorithme $Al(coupes)$ est fixé à 100.

5.3 Étude des paramètres de coupes

Afin d'identifier le meilleur critère de sélection, des séries de tests ont été réalisées pour les différents critères étudiés à la section 4.1.2 (CS1 à CS6) en utilisant $\lfloor \frac{m_2}{2} \rfloor$ itérations de coupes. Deux données sont retenues pour étudier la performance des critères de sélection. Le premier critère correspond à la décroissance de la fonction objectif $\delta = (F_n - F_{opt})/(F_{ini} - F_{opt})$ avec F_n la valeur de la fonction objectif à l'issue de la phase de coupes, F_{ini} la valeur initial de l'objectif et F_{opt} la valeur de la fonction objectif à l'optimum. Le second critère correspond au nombre de noeuds

d'énumération. Notons que pour le même type de coupes, le temps de branchement est proportionnel au nombre de noeuds. Les tableaux 5.1, 5.2, 5.3 et 5.4 présentent les résultats des tests pour les différents types de coupes à savoir les coupes de Gomory, les coupes étendues, les coupes simples et les coupes disjonctives. Pour chacun des critères, nous y présentons la réduction relative de la fonction objectif (δ) et le nombre de noeuds requis par la phase d'énumération (noeuds). Notons que dans les séries de tests donnant les résultats des coupes simples (tableau 5.3) et disjonctives (tableau 5.4), le critère hybride CS2 est identique à CS1 puisque les deux types de coupes ne font pas appel aux variables binaires u_i.

Tableau 5.1 – Tests sur le critère de sélection avec $\lfloor \frac{m_2}{2} \rfloor$ coupes de Gomory

Moyenne de 10 problèmes pour une densité de 10%(14%)												
$n_1 = n_2 =$		CS1		CS2		CS3		CS4		CS5		CS6	
$m_1 = m_2$		δ	noeuds	δ	noeuds	δ	noeuds	δ	noeuds	δ	noeuds	δ	noeuds
15	μ	0.24	342	0.26	346	0.12	304	0.21	314	0.12	312	0.28	350
	σ	0.25	398	0.27	397	0.17	308	0.26	340	0.17	324	0.26	417
18	μ	0.16	1121	0.19	1216	0.07	1203	0.13	1158	0.09	1227	0.17	1300
	σ	0.17	1134	0.20	1255	0.13	1394	0.19	1372	0.13	1374	0.19	1457
20	μ	0.19	1016	0.21	819	0.11	1095	0.13	1090	0.10	1098	0.14	1026
	σ	0.22	1257	0.23	903	0.23	1466	0.23	1477	0.23	1476	0.23	1265
25	μ	0.09	6788	0.11	6644	0.00	6873	0.03	6149	0.00	6930	0.05	6858
	σ	0.07	9143	0.08	7862	0.00	9400	0.06	7447	0.00	9422	0.05	9334
28	μ	0.09	23590	0.10	24747	0.04	24832	0.08	24774	0.04	24932	0.09	24123
	σ	0.12	25000	0.13	25162	0.06	26970	0.12	26944	0.06	26811	0.12	24485

Les résultats obtenus en utilisant les coupes étendues sont identiques à ceux de Gomory. Ayant des expressions assez semblables, les deux coupes sont souvent équivalentes : la formule des deux coupes ne diffère que pour les coefficients des variables binaires et le nombre de ces derniers est limité ne dépassant pas le nombre d'itérations de coupes. Néanmoins, en augmentant la densité et le nombre de coupes introduites, les résultats peuvent différer. À titre d'exemple, les résultats sont différents pour le problème $P1$ ($P1 : n_1 = n_2 = m_1 = m_2 = 10$, densité= 50% et germe aléatoire = 1) en utilisant dix itérations de coupes. En introduisant les coupes étendues à la

Tableau 5.2 – Tests sur le critère de sélection avec $\lfloor \frac{m_2}{2} \rfloor$ coupes étendues

Moyenne de 10 problèmes pour une densité de 10%(14%)												
$n_1 = n_2 =$		CS1		CS2		CS3		CS4		CS5		CS6	
$m_1 = m_2$		δ	noeuds	δ	noeuds	δ	noeuds	δ	noeuds	δ	noeuds	δ	noeuds
15	μ	0.24	342	0.26	346	0.12	304	0.21	314	0.12	312	0.28	350
	σ	0.25	398	0.27	397	0.17	308	0.26	340	0.17	324	0.26	417
18	μ	0.16	1121	0.19	1216	0.07	1203	0.13	1158	0.09	1227	0.17	1294
	σ	0.17	1134	0.20	1255	0.13	1394	0.19	1372	0.13	1374	0.19	1452
20	μ	0.19	1016	0.21	819	0.11	1095	0.13	1090	0.10	1098	0.14	1026
	σ	0.22	1257	0.23	903	0.23	1466	0.23	1477	0.23	1476	0.23	1265
25	μ	0.09	6788	0.11	6644	0.00	6873	0.03	6149	0.00	6930	0.05	6858
	σ	0.07	9143	0.08	7862	0.00	9400	0.06	7447	0.00	9422	0.06	9334
28	μ	0.09	23590	0.10	24747	0.04	24832	0.08	24774	0.04	24932	0.09	24123
	σ	0.12	25000	0.13	25162	0.06	26970	0.12	26944	0.06	26811	0.12	24485

place des coupes de Gomory, le nombre de noeuds explorés par l'algorithme passe de 74 à 52. Les tests concernant les coupes de Gomory et étendues montrent que

Tableau 5.3 – Tests sur le critère de sélection avec $\lfloor \frac{m_2}{2} \rfloor$ coupes simples

Moyenne de 10 problèmes pour une densité de 10%(14%)											
$n_1 = n_2 =$		CS1 = CS2		CS3		CS4		CS5		CS6	
$m_1 = m_2$		δ	noeuds	δ	noeuds	δ	noeuds	δ	noeuds	δ	noeuds
15	μ	0.22	314	0.12	306	0.19	319	0.13	308	0.18	350
	σ	0.25	335	0.17	308	0.27	338	0.17	313	0.25	401
18	μ	0.13	1336	0.07	1205	0.12	1241	0.09	1225	0.15	1143
	σ	0.17	1433	0.13	1403	0.17	1446	0.13	1380	0.18	1347
20	μ	0.12	1015	0.10	1099	0.12	1100	0.09	1099	0.12	1076
	σ	0.23	1256	0.23	1475	0.23	1487	0.23	1476	0.24	1325
25	μ	0.01	6904	0.00	6902	0.02	6916	0.00	6896	0.01	6925
	σ	0.03	9404	0.00	9404	0.04	9412	0.00	9405	0.02	9409
28	μ	0.04	25133	0.04	25025	0.04	25029	0.04	24986	0.05	25140
	σ	0.06	27066	0.06	26872	0.06	26882	0.06	26793	0.06	27079

les critères CS2 et (à degré moindre) CS1 sont les plus performants en terme de décroissance de la fonction objectif. Cependant le critère CS1 s'avère le plus efficace en ce qui concerne la taille de l'arbre d'énumération. Ce même critère donne les meilleurs résultats pour les tests effectués avec les coupes simples et disjonctifs. C'est pourquoi, il sera adopté pour la suite des séries de tests.

Tableau 5.4 – Tests sur le critère de sélection avec $\lfloor \frac{m_2}{2} \rfloor$ coupes disjonctives

Moyenne de 10 problèmes pour une densité de 10%(14%)											
$n_1 = n_2 =$		CS1=CS2		CS3		CS4		CS5		CS6	
$m_1 = m_2$		δ	noeuds	δ	noeuds	δ	noeuds	δ	noeuds	δ	noeuds
15	μ	0.13	312	0.12	312	0.13	307	0.12	300	0.13	292
	σ	0.18	312	0.17	312	0.17	311	0.17	295	0.17	306
18	μ	0.10	1180	0.07	1207	0.10	1180	0.07	1195	0.07	1207
	σ	0.19	1380	0.13	1403	0.19	1381	0.13	1392	0.13	1403
20	μ	0.11	1092	0.09	1090	0.09	1106	0.09	1094	0.09	1096
	σ	0.23	1478	0.23	1473	0.22	1471	0.23	1472	0.23	1478
25	μ	0.00	6901	0.00	6902	0.00	6901	0.00	6891	0.00	6901
	σ	0.00	9404	0.00	9404	0.00	9404	0.00	9399	0.00	9404
28	μ	0.04	24898	0.04	25133	0.04	24906	0.04	25133	0.04	24943
	σ	0.06	26657	0.06	27066	0.06	26656	0.06	27066	0.06	26752

Le tableau 5.5 présente les résultats des quatre types de coupes étudiés au chapitre 3. Les résultats sont donnés pour cinq classes de problèmes de tailles différentes. Notons Δ l'écart initial relatif : $\Delta = (F_{ini} - F_{opt})/F_{opt}$. Pour chaque classe de problèmes, nous introduisons un nombre de coupes $n = \lfloor \frac{m_2}{2} \rfloor$. Pour chaque type de coupes, nous évaluons le nombre de contraintes ajoutées (Nb ctes ajoutées), la décroissance de la fonction objectif (décroissance δ) et le nombre de tests conditionnels utilisés (Nb tests). Nous donnons également à titre indicatif le nombre de noeuds explorés (Nb noeuds) et le temps de branchement en secondes (temps branch(s)). L'analyse de ces deux données sera entreprise ultérieurement pour évaluer la performance de l'algorithme. Étant de l'ordre de $0.05s$, le temps de la phase de coupes sera omis tout au long des séries de tests. La décroissance de la fonction objectif confirme les résultats théoriques donnés à la section 3.3 concernant la profondeur des coupes. Elle montre que les coupes de Gomory et étendues sont les plus profondes alors que les coupes disjonctives sont les moins profondes. Il s'ensuit que le nombre de noeuds explorés en utilisant les coupes de Gomory et étendues est inférieur à celui en utilisant les coupes simples et disjonctives. Cependant, les coupes simples introduisent moins de contraintes grâce notamment à l'utilisation de la procédure de nettoyage.

Tableau 5.5 – Tests comparatifs sur le type de coupes utilisant CS1

Moyenne de 10 problèmes pour une densité de 10%(14%)							
Taille Δ	type de coupes		Nb ctes ajoutées	décroissance δ	Nb tests	Nb noeuds	temps branch(s)
$n_1 = 15$ $n_2 = 15$ $m_1 = 15$ $m_2 = 15$ Δ : $\mu = 0.4$, $\sigma = 0.2$	Gomory	μ	11	0.24	1	342	0.4
		σ	1.7	0.25	1.4	398.1	0.5
	Étendue	μ	11	0.24	1	342	0.4
		σ	1.7	0.25	1.4	398.1	0.5
	Simple	μ	4	0.22	1	314	0.3
		σ	2.4	0.25	1.4	335.9	0.3
	Disjonctive	μ	6	0.13	1	312	0.36
		σ	0.6	0.18	1.2	312.9	0.3
$n_1 = 18$ $n_2 = 18$ $m_1 = 18$ $m_2 = 18$ Δ : $\mu = 0.5$, $\sigma = 0.3$	Gomory	μ	14	0.16	1	1121	2.0
		σ	2.1	0.17	1.4	1134.6	2.1
	Étendue	μ	14	0.16	1	1121	2.0
		σ	2.1	0.17	1.4	1134.6	2.1
	Simple	μ	5	0.13	1	1336	2.3
		σ	2.9	0.17	1.4	1433.0	2.5
	Disjonctive	μ	7	0.10	1	1180	2.1
		σ	1.1	0.18	1.6	1380.2	2.6
$n_1 = 20$ $n_2 = 20$ $m_1 = 20$ $m_2 = 20$ Δ : $\mu = 0.3$, $\sigma = 0.2$	Gomory	μ	16	0.19	2	1016	2.2
		σ	2.5	0.22	2.1	1257.1	2.9
	Étendue	μ	16	0.19	2	1016	2.2
		σ	2.5	0.22	2.1	1257.1	2.9
	Simple	μ	5	0.12	2	1015	2.0
		σ	3.5	0.23	2.0	1256.5	2.7
	Disjonctive	μ	8	0.11	2	1092	2.2
		σ	0.8	0.23	1.8	1478.9	1.8
$n_1 = 25$ $n_2 = 25$ $m_1 = 25$ $m_2 = 25$ Δ : $\mu = 0.3$, $\sigma = 0.2$	Gomory	μ	21	0.09	1	6788	22.1
		σ	2.1	0.07	1.7	9143.4	29.7
	Étendue	μ	21	0.09	1	6788	21.9
		σ	2.1	0.07	1.7	9143.4	29.5
	Simple	μ	5	0.01	0	6904	17.8
		σ	4.9	0.03	1.1	9404.2	22.7
	Disjonctive	μ	11	0.00	0	6901	20.1
		σ	0.6	0.00	1.1	9404.9	26.7
$n_1 = 28$ $n_2 = 28$ $m_1 = 28$ $m_2 = 28$ Δ : $\mu = 0.4$, $\sigma = 0.2$	Gomory	μ	23	0.09	1	23590	88.4
		σ	2.3	0.12	1.2	25000.7	92.7
	Étendue	μ	23	0.09	1	23590	87.9
		σ	2.3	0.12	1.2	25000.7	92.0
	Simple	μ	4	0.04	0	25133	79.2
		σ	5.4	0.06	0.8	27066.0	88.5
	Disjonctive	μ	13	0.04	1	24898	95.0
		σ	0.6	0.06	0.9	26657.8	103.7

L'apport des coupes est mis en évidence à travers une série de tests comparatifs sur le nombre d'itérations de la Phase I. Deux familles de coupes ont été utilisées : les coupes de Gomory et les coupes simples. Le tableau 5.6 donne la décroissance de la fonction objectif (δ), le nombre de noeuds d'énumération (nds) et le temps de branchement en secondes (t(s)) en fonction du nombre d'itérations de la Phase I.

Tableau 5.6 – Tests comparatifs sur le nombre de coupes utilisant CS1

Moyenne de 10 problèmes pour une densité de 8%(12%)															
Nb coupes		0		$\lfloor \frac{m_2}{8} \rfloor$			$\lfloor \frac{m_2}{4} \rfloor$			$\lfloor \frac{m_2}{3} \rfloor$			$\lfloor \frac{m_2}{2} \rfloor$		
Taille	type	nds	t(s)	δ	nds	t(s)	δ	nds	t(s)	δ	nds	t(s)	δ	nds	t(s)
$n_1 = 25$	Gomory μ	3177	6.0	0.07	3266	6.5	0.13	2207	4.8	0.19	2203	5.0	0.22	1959	5.0
$n_2 = 25$	σ	4825	8.7	0.17	5295	10.0	0.16	2607	5.7	0.17	2609	6.0	0.17	2239	6.3
$m_1 = 25$	Simple μ	3177	6.0	0.07	3266	6.5	0.11	2270	4.6	0.11	2119	4.4	0.14	2119	4.5
$m_2 = 25$	σ	4825	8.7	0.17	5295	9.9	0.17	2724	5.6	0.17	2455	5.2	0.19	2455	5.3
$n_1 = 25$	Gomory μ	9948	21.6	0.07	7917	18.1	0.11	7809	18.9	0.12	7810	20.12	0.13	7794	22.1
$n_2 = 25$	σ	11002	23.7	0.13	8085	18.5	0.12	8082	19.6	0.11	8084	21.0	0.12	8075	23.6
$m_1 = 25$	Simple μ	9948	21.6	0.07	7917	18.2	0.08	7917	18.4	0.08	7917	18.8	0.09	7916	19.4
$m_2 = 30$	σ	11002	23.7	0.13	8085	18.7	0.12	8085	18.8	0.12	8085	19.0	0.12	8084	19.4
$n_1 = 30$	Gomory μ	44943	117.2	0.06	41422	114.3	0.11	36100	108.7	0.13	36054	115.9	0.15	33306	118.7
$n_2 = 30$	σ	66527	179.0	0.14	67713	191.3	0.19	52753	166.1	0.21	52784	177.3	0.23	44966	165.7
$m_1 = 30$	Simple μ	44943	117.1	0.05	41549	115.4	0.10	41484	115.1	0.11	41437	116.2	0.11	41437	119.0
$m_2 = 30$	σ	66527	178.6	0.14	67743	192.2	0.19	67785	189.7	0.20	67814	189.5	0.20	67813	190.4
$n_1 = 60$	Gomory μ	46864	105.6	0.06	36658	89.2	0.09	36754	98.2	0.10	36741	101.5	0.11	30520	89.8
$n_2 = 30$	σ	110737	251.1	0.05	73621	181.1	0.08	73587	199.5	0.08	73592	207.4	0.08	51192	154.1
$m_1 = 9$	Simple μ	46864	110.8	0.05	36663	89.2	0.08	36746	92.5	0.08	36745	95.1	0.09	37534	103.6
$m_2 = 27$	σ	110737	264.2	0.05	73610	179.7	0.06	73584	186.2	0.06	73584	192.6	0.06	73464	207.6
$n_1 = 70$	Gomory μ	4182	8.7	0.08	3890	8.3	0.12	3528	8.1	0.13	3315	7.7	0.18	3371	8.6
$n_2 = 30$	σ	4774	9.6	0.16	4468	9.1	0.17	4000	8.6	0.18	3501	7.7	0.24	3453	8.2
$m_1 = 20$	Simple μ	4182	8.7	0.08	3890	8.4	0.11	3947	8.7	0.12	3947	8.8	0.13	3938	9.1
$m_2 = 20$	σ	4774	9.7	0.16	4468	9.2	0.17	4487	9.4	0.18	4487	9.6	0.18	4495	10.0

Les tests comparatifs sur le nombre de coupes montrent une bonne décroissance de la fonction objectif. Ceci est illustré dans la figure 5.1 qui représente la décroissance moyenne de l'objectif pour l'ensemble des cinq classes de problèmes en fonction du nombre d'itérations de coupes. Cette décroissance atteint une valeur de saturation pour un nombre d'itérations $n \cong \lfloor \frac{m_2}{2} \rfloor$. La diminution de la valeur de la fonction objectif se traduit par un gain plus notable en nombre de noeuds d'énumération.

Notons que les coupes de Gomory permettent un meilleur gain de noeuds que les coupes simples. Ce gain, illustré dans la figure 5.2, engendre un gain moins important en temps de branchement. En effet, l'introduction des coupes augmente la taille du problème LRP. Il s'ensuit une augmentation du temps de calcul à chaque noeud de l'arbre d'énumération. En optant pour les coupes simples, la Phase I introduit moins de contraintes qu'en utilisant les coupes de Gomory, ce qui permet de réduire la taille des problèmes à chaque noeud d'énumération et par la suite le temps de branchement.

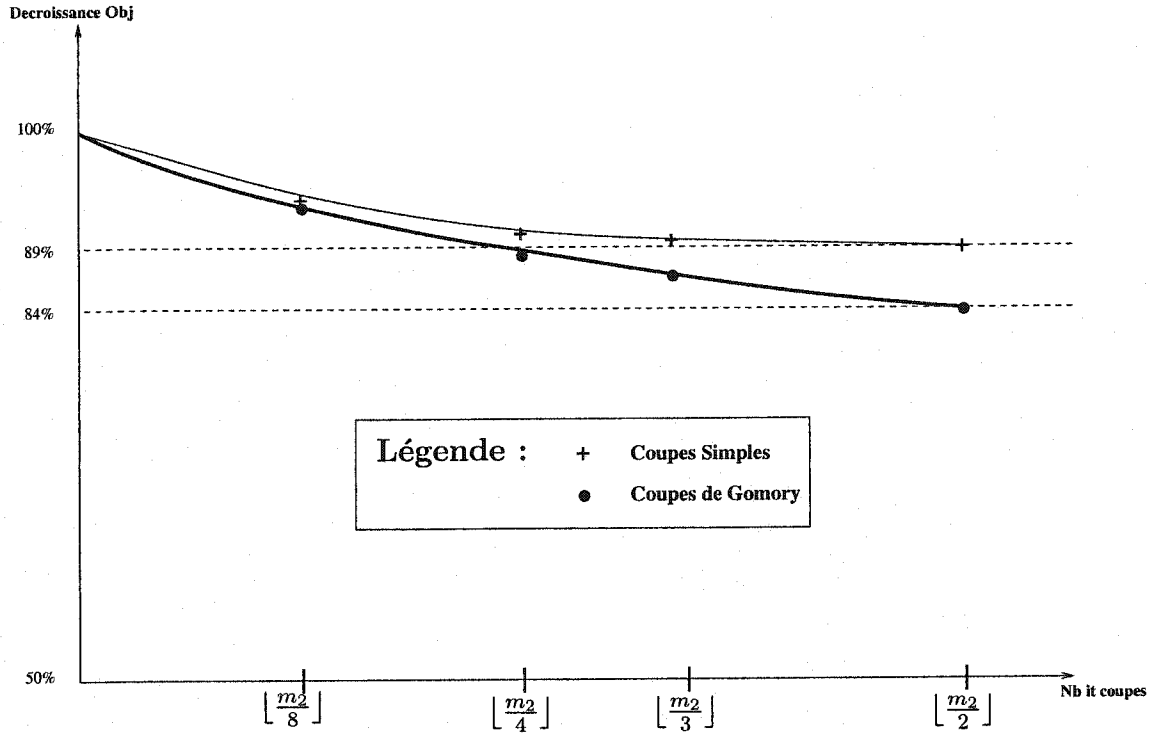

Figure 5.1 – Décroissance moyenne de l'objectif en fonction du nombre d'itérations de coupes. Critère de sélection : CS1.

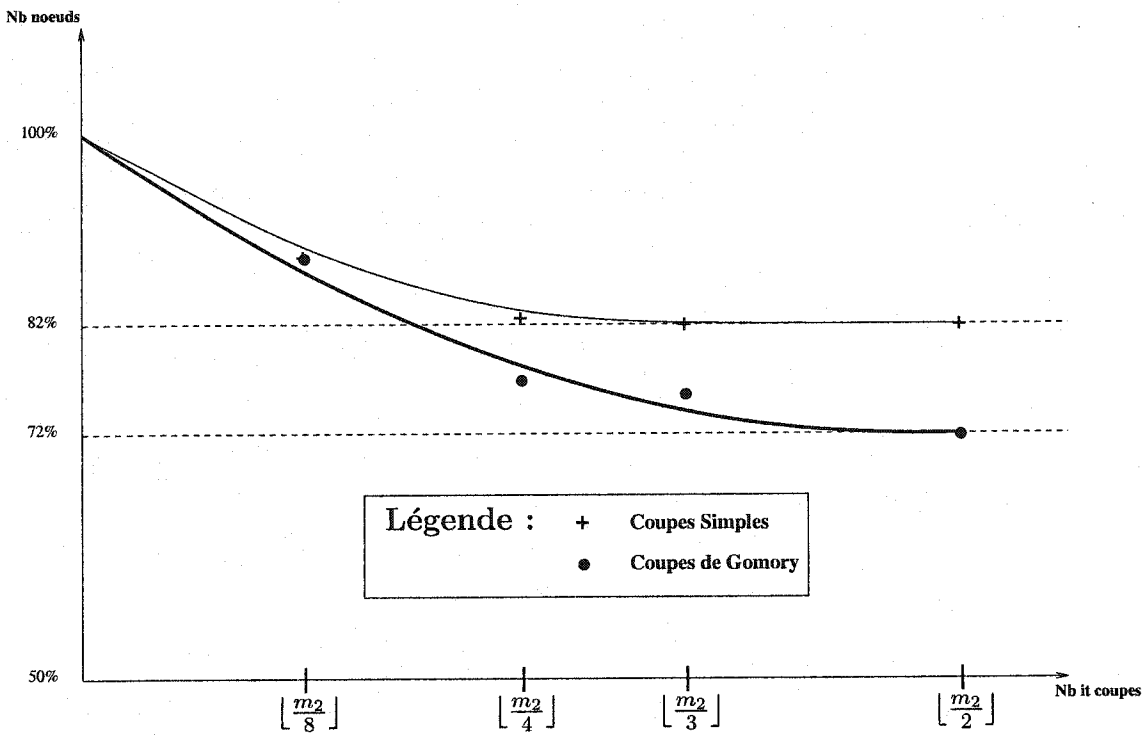

Figure 5.2 – Décroissance moyenne du nombre de noeuds en fonction du nombre d'itérations de coupes. Critère de sélection : CS1.

5.4 Résultats de l'énumération implicite

Une quatrième série de tests est effectuée pour choisir le meilleur critère de branchement. Le tableau 5.7 donne le nombre de noeuds d'énumération pour différents critères de séparation en utilisant $\lfloor \frac{m_2}{3} \rfloor$ itérations de coupes de Gomory.

D'après les résultats, le critère HJS permet un gain assez important en nombre de noeuds allant jusqu'à 67% par rapport au critère de Bard et Moore et jusqu'à 88% en comparaison avec le critère maxmin. Le critère maxmin s'avère aussi performant que le critère HJS pour les petits problèmes. Mais, il est peu efficace pour les problèmes

Tableau 5.7 – Tests comparatifs sur le critère d'énumération avec $\lfloor \frac{m_2}{3} \rfloor$ coupes de Gomory

Moyenne de 10 problèmes pour une densité de 10%(14%)				
$n_1 = n_2 =$ $m_1 = m_2$		nombre de noeuds		
		Critère HJS	Critère maxmin	Critère Bard et Moore
15	μ	339	323	469
	σ	386	339	488
18	μ	1121	1771	1729
	σ	1133	1569	1563
20	μ	1030	2448	2013
	σ	1262	3503	1548
25	μ	6828	31858	7351
	σ	9126	47530	5059
28	μ	23848	200811	72771
	σ	24971	237694	38635

de grande taille. En conclusion, le critère HJS est le plus performant. Il sera retenu pour la suite des séries de tests.

Une dernière série de tests est réalisée afin d'évaluer la performance de l'algorithme *Al(coupes)*. La série de tests porte sur des problèmes de densité 8%(12%). C'est la densité utilisée par Hansen et Al. [18] pour générer la majorité des problèmes tests contenant des contraintes de premier niveau. Le tableau 5.8 contient les résultats de la résolution de CPLEX 6.6 de la reformulation mixte $0-1$ et ceux de notre algorithme en utilisant $\lfloor \frac{m_2}{2} \rfloor$ itérations de coupes de Gomory et le critère de séparation CS1. Les deux approches utilisent le même ordinateur et langage de programmation. Pour la reformulation mixte, la plus petite valeur expérimentale de L que nous avons pu utiliser sans couper la solution optimale est 10000 (notons que les meilleurs résultats sont obtenus avec la petite valeur possible de la constante L). Le nombre de coupes

utilisées dans la résolution mixte dépend d'un paramètre P. Lorsque ce paramètre est fixé à -1, CPLEX 6.6 ne génère pas de coupes. S'il est à 1, CPLEX 6.6 génère un nombre limité de coupes. En le fixant à 2, CPLEX 6.6 introduit un nombre élevé de coupes. La valeur par défaut de P est 0. Dans ce cas, CPLEX 6.6 détermine automatiquement le nombre de coupes à intégrer. Notons que les coupes sont introduites aux niveaux supérieures de l'arbre d'énumération.

Les résultats montrent que l'algorithme de coupes génère beaucoup moins de noeuds dans l'arbre d'énumération et met moins de temps de calcul que la résolution de CPLEX de la reformulation mixte. Contrairement à notre algorithme, l'introduction des coupes dans la résolution CPLEX s'avère peu efficace. Notre algorithme gère mieux ces coupes. Ceci est dû à l'exploitation de la structure biniveau du problème, en particulier au choix adéquat des critères de branchement et de sélection.

Tableau 5.8 – Comparaison entre *Al(coupes)* pour $\lfloor \frac{m_2}{2} \rfloor$ itérations de coupes et résolution CPLEX de la reformulation mixte

Moyenne de 10 problèmes pour une densité de 8%(12%)															
algorithme		$n_1=25, n_2=25$ $m_1=25, m_2=25$			$n_1=25, n_2=25$ $m_1=25, m_2=30$			$n_1=30, n_2=30$ $m_1=30, m_2=30$			$n_1=60, n_2=30$ $m_1=9, m_2=27$			$n_1=70, n_2=30$ $m_1=20, m_2=20$		
		cts	nds	t(s)	cts	nds	t(s)	cts	nds	t(s)	cts	nds	t(s)	cts	nds	t(s)
Al(coupes)	μ	20	1959	5	25	7794	22	25	33306	118	21	30520	89	16	3371	8
	σ	1	2239	6	1	8075	23	1	44966	165	2	51192	154	2	3453	8
CPLEX	μ	0	4575	7	0	25176	45	0	107908	230	0	108328	211	0	15764	26
$P=-1$	σ	0	6626	12	0	28107	52	0	129391	297	0	260688	519	0	40301	65
CPLEX	μ	5	6732	11	6	30555	57	6	107687	259	8	199980	405	8	10714	18
$P=0$	σ	2	10023	16	2	41063	74	3	129484	339	2	431675	897	2	25013	42
CPLEX	μ	7	6518	11	8	46169	94	7	161616	386	10	284912	595	9	17425	33
$P=1$	σ	1	9264	15	3	76308	172	2	164246	427	2	760768	1597	3	47623	89
CPLEX	μ	9	6679	11	11	21976	41	9	179125	463	11	427191	947	12	17816	34
$P=2$	σ	5	9208	15	5	18832	38	6	198618	572	3	1202919	2712	4	47506	89

CHAPITRE 6

Conclusion

Nous avons étudié dans ce mémoire le problème de programmation linéaire à deux niveaux du point de vue algorithmique et théorique.

Dans un premier temps, nous avons présenté les principales caractéristiques du problème BLP. En particulier, nous avons mis l'accent sur l'équivalence qui existe entre les problèmes BLP et les problèmes de programmation linéaire mixte en nombres entiers.

Dans un deuxième temps, nous avons exploité les liens entre ces deux classes de problèmes pour développer des inégalités valides au sein du problème BLP. Plusieurs types de coupes ont été définies à savoir : les coupes de Gomory, les coupes étendues, les coupes simples et les coupes disjonctives.

Enfin, nous avons mis au point un algorithme exact pour la résolution du problème BLP utilisant à la fois les techniques de coupes et les méthodes d'énumération. Plusieurs tests comparatifs ont été effectués afin d'étudier l'effet des différents paramètres de l'algorithme. Les résultats démontrent l'efficacité de notre méthode. D'une part, l'exploitation de la structure biniveau réduit les temps de calcul. D'autre part, l'ajout des coupes entraîne une diminution du nombre de noeuds de l'arbre d'énumération. Cependant, tout comme en programmation linéaire mixte $0-1$, l'ajout de coupes n'améliore pas significativement les temps de calcul totaux.

Ainsi, notre contribution se situe à deux niveaux : théorique et algorithmique. Cependant, il y a des possibilités d'extension. Notamment, on peut se demander s'il est possible d'intégrer les coupes au sein même de l'arbre d'énumération. Cette démarche s'est avérée valable pour la résolution des problèmes mixtes $0-1$ [7]. Quelle serait alors l'effet de l'application d'une telle approche en programmation linéaire à deux niveaux ? On pourrait également étudier l'intégration des coupes de Gomory au sein du sous-problème dual afin de réduire la taille du problème relaxé. Une autre possibilité d'extension consiste à déterminer, à partir de la reformulation mixte, les sous classes de problèmes BLP sur lesquels les coupes de Gomory donnent les meilleurs résultats. Le champ d'étude peut être élargi en examinant l'expression d'autres coupes mixtes au sein du problème BLP. Dans ce cadre, il serait intéressant d'exploiter la technique de *lift and project* [6] en programmation linéaire à deux niveaux.

RÉFÉRENCES

[1] ALARIE, S., AUDET, C., JAUMARD,B. et SAVARD, G. (2001). "Concavity Cuts for Disjoint Bilinear Programming", *Mathematical Programming*, 90, 373-398.

[2] ANANDLINGAM, G. et WHITE, D. (1990). "A Solution Method for Linear Static Stackelberg Problem using Penalty Functions", *IEEE Transactions on Automatic Control*, 35, 1170-1173.

[3] AUDET, C. (1997). " Optimisation Globale Structurée : Propriétés, Équivalences et Résolution", *Thèse de Doctorat, École Polytechnique de Montréal.*

[4] AUDET, C., HANSEN, P., JAUMARD, B. et SAVARD, G. (1997). "Links between Linear Bilevel and Mixed $0-1$ Programming Problems", *Journal of Optimization Theory and Applications*, 93, 2, 273-300.

[5] AUDET, C., HANSEN, P., JAUMARD, B. et SAVARD, G. (1999). "A Symmetrical Linear Maxmin Approach to Disjoint Bilinear Programming", *Mathematical Programming*, 85, 573-592.

[6] BALAS, E., CERIA, S., CORNUÉJOLS, G. (1993). "A Lift-and-Project Cutting Plane Algorithm for Mixed $0-1$ Programs", *Mathematical Programming*, 58, 295-324.

[7] BALAS, E., CERIA, S., CORNUÉJOLS, G., NATRAJ, N. (1996). "Gomory Cuts Revisited", *Operations Research Letters*, 19,1-9.

[8] BARD, J.F. (1983). "An Efficient Point Algorithm for a Linear Two-Stage Optimization Problem", *Operations Research*, 31, 670-684.

[9] BARD, J.F. (1984). "Optimality Conditions for the Bilevel Programming Problem", *Naval Research Logistic Quarterly*, 31, 13-26.

[10] BARD, J.F. et MOORE, J. (1990). "A Branch and Bound Algorithm for the Bilevel Programming Problem", *SIAM Journal on Scientific and Statistical Computing* 11, 281-292.

[11] BEN-AYED, O. et BLAIR, C. (1990). "Computational Difficulties of Bilevel Linear Programming", *Operations Research*, 38, 556-560.

[12] BEALE, E.M.L. et SMALL, R.E. (1965). "Mixed Integer Programmation by a Branch and Bound Technique", *Proceeding of the 3rd IFIP Congress 1965*, 2, 450-451.

[13] BIALAS, W. et KARWAN, M. (1984). "Two-Level Linear Programming", *Management Science*, 30, 1004-1020.

[14] CANDLER, W. et NORTON, R. (1977). "Multi-Level Programming", *World Bank Development Research Center Discussion Paper*, 6, Washington, DC.

[15] CANDLER, W. et TOWNSLEY, R. (1982). "A Linear Two-Level Programming Problem", *Computers and Operations Research*, 9, 59-76.

[16] CPLEX Division (1998). "Using the CPLEX Callable Library", *ILOG*, 6, USA.

[17] GOMORY, R. (1960). "An Algorithm for the Mixed Integer Problem", *Technical Report RM-2537, The Rand Corporation.*

[18] HANSAN, P., JAUMARD, B. et SAVARD, G. (1992). "New Branch-and-Bound rules for Linear Bilevel Programming", *SIAM Journal on Scientific and Statistical Computing*, 13, 1194-1217.

[19] JEROSLOW, R. (1985). "The Polynomial Hierarchy and a Simple Model for Competitive Analysis", *Mathematical Programming*, 32, 146-164.

[20] JUDICE, J., SAVARD, G. et VICENTE, L. (1994). "Descent Approaches for Quadratic Bilevel Programming", *Journal of Optimization Theory and Application*, 81, 379-399.

[21] MAGNANTI, T.L. et WONG, R.T. (1984). "Network Design and Transportation Planning- Models and Algorithms", *Transportation Science*, 18.

[22] NEMHAUSER, G.L. et WOLSEY, L.A. (1988). "Integer and Combinatorial Optimization", *Wiley-Interscience Publication*, New-York.

[23] NEMHAUSER, G.L. et WOLSEY, L.A. (1989). "Integer Programming", *Handbooks in Operations Research and Management Science 1 : Optimization* , North-Holland, Amsterdam, 447-527.

[24] PADBERG, M. et RINALDI, G. (1991). "A Branch-and-Cut Algorithm for Resolution of Large-Scale Symmetric Traveling Salesman Problems", *SIAM*, 33, 60-100.

[25] PARKER, R.G. et RARDIN, R.L. (1988). "Discrete Optimization", *Academic Press*, New-York.

[26] SAVARD, G. (1988). "Contributions à la Programmation Mathématiques à Deux Niveaux", *Thèse de Doctorat, École Polytechnique de Montréal.*

[27] STACKELBERG, V.H. (1952). "The Theory of the Market Theory", *Oxford University Press*, Oxford, England.

[28] TAHA, H.A. (1975). "Integer Programming Theory, Applications and Computations", *Academic Press*, New York.

[29] WOLSEY, L.A. (1988). "Integer Programming", *Collection Wiley-Interscience Series in Discrete Mathematics and Optimization.*