THÈSE

présentée à l'

UNIVERSITÉ DES SCIENCES ET TECHNIQUES

DE LILLE I

pour obtenir le titre de

DOCTEUR DE SPÉCIALITÉ
(Mathématiques appliquées)

par

Mana KORTAS

PROGRAMMATION LINÉAIRE EN NOMBRES ENTIERS : COMPARAISON DE MÉTHODES

Thèse soutenue le 5 Mars 1971, devant la Commission d'Examen :

Monsieur P. BACCHUS, Président
Monsieur P. HUARD, Rapporteur
Monsieur C. CARREZ, Examinateur

UNIVERSITE DE LILLE

FACULTE DES SCIENCES

DOYENS HONORAIRES

MM. H. LEFEBVRE, M. PARREAU.

PROFESSEURS HONORAIRES

MM. ARNOULT, BROCHARD, CAU, CHAPPELON, CHAUDRON, CORDONNIER, DEHEUVELS, DEHORNE, DOLLE, FLEURY, P. GERMAIN, KAMPE DE FERIET, KOURGANOFF, LAMOTTE, LELONG, Mme LELONG, MM. MAZET, MICHEL, NORMANT, PARISELLE, PAUTHENIER, ROIG, ROSEAU, ROUBINE, ROUELLE, WIEMAN, ZAMANSKY.

PROFESSEURS TITULAIRES

M. BACCHUS Pierre	Astronomie et Calcul
M. BEAUFILS Jean-Pierre	Chimie Générale
M. BLOCH Vincent	Psychophysiologie
M. BONNEMAN Pierre	Chimie Industrielle
M. BONTE Antoine	Géologie Appliquée
M. BOUGHON Pierre	Mathématiques
M. BOURIQUET Robert	Biologie Végétale
M. CELET Paul	Géologie Générale
M. CONSTANT Eugène	Electronique
M. CORSIN Pierre	Paléobotanique
M. DECUYPER Marcel	Mathématiques
M. DEDECKER Paul	Mathématiques
M. le Doyen DEFRETIN René	Directeur du Laboratoire de Biologie Maritime de Wimereux
M. DELATTRE Charles	Géologie Générale
M. DURCHON Maurice	Biologie Animale
M. FOURET René	Physique
M. GABILLARD Robert	Electronique
M. GLACET Charles	Chimie Organique
M. GONTIER Gérard	Mécanique des Fluides
M. GUILLAUME Jean	Biologie Végétale
M. HEUBEL Joseph	Chimie Minérale
Mme LENOBLE Jacqueline	Physique
M. MONTREUIL Jean	Chimie Biologique
Mme SCHWARTZ Marie Hélène	Mathématiques
M. TILLIEU Jacques	Physique
M. TRIDOT Gabriel	Chimie Minérale Appliquée E.N.S.C.L.
M. VIDAL Pierre	Automatique
M. VIVIER Emile	Biologie Animale
M. WATERLOT Gérard	Géologie et Minéralogie
M. WERTHEIMER Raymond	Physique

PROFESSEURS A TITRE PERSONNEL

M. BOUISSET Simon	Physiologie Animale
M. DELHAYE Michel	Chimie Physique et Minérale 1er Cycle
M. LINDER Robert	Biologie Végétale
M. LUCQUIN Michel	Chimie Physique
M. PARREAU Michel	Mathématiques
M. SAVARD Jean	Chimie Générale
M. SCHALLER François	Biologie Animale
M. SCHILTZ René	Physique

PROFESSEURS SANS CHAIRE

M. BELLET Jean	Physique
M. BODART Marcel	Biologie Végétale
M. BOILLET Pierre	Physique
M. DERCOURT Jean-Michel	Géologie et Minéralogie
M. DEVRAINNE Pierre	Chimie Minérale
Mle MARQUET Simone	Mathématiques
M. MONTARIOL Frédéric	Chimie Minérale Appliquée
M. PROUVOST Jean	Géologie et Minéralogie
M. VAILLANT Jean	Mathématiques

MAITRES DE CONFERENCE (et chargés des fonctions)

M. AUBIN Thierry	Mathématiques Pures
M. BEGUIN Paul	Mécanique des Fluides
M. BILLARD Jean	Physique
M. BKOUCHE Rudolphe	Mathématiques
M. BOILLY Bénoni	Biologie Animale
M. BONNOT Ernest	Biologie Végétale
M. CAPURON Alfred	Biologie Animale
M. CARREZ Christian	Calcul
M. CORDONNIER Vincent	Calcul
M. CORTOIS Jean	Physique
M. COULON Jean-Paul	Electrotechnique
M. DEBRABAN Pierre	Sciences Appliquées
M. ESCAIG Bertrand	Physique
M. FROLICH Daniel	Sciences Appliquées
M. GOBLOT Rémi	Mathématiques
M. GOUDMAND Pierre	Chimie Physique
M. GRUSON Laurent	Mathématiques
M. GUILBAULT Pierre	Physiologie Animale
M. HERMAN Maurice	Physique
M. HUARD de la MARRE Pierre	Calcul
M. JOURNEL Gérard	Sciences Appliquées
Mle KOSMANN Yvette	Mathématiques
M. LABLACHE COMBIER Alain	Chimie Générale
M. LACOSTE Louis	Biologie Végétale
M. LANDAIS Jean	Chimie Organique
M. LAURENT François	Automatique
M. LEHMANN Daniel	Mathématiques
Mme LEHMANN Josiane	Mathématiques
M. LOCQUENEUX Robert	Physique
M. LOUAGE Francis	Sciences Appliquées

M. LOUCHEUX Claude	Chimie Physique
M. MAES Serge	Physique
M. MAIZIERES Christian	Automatique
M. MESSELYN Jean	Physique
M. MIGEON Michel	Sciences Appliquées
M. MONTEL Marc	Physique
M. OUZIAUX Roger	Sciences Appliquées
M. PANET Marius	Electrotechnique
M. PAQUET Jacques	Sciences Appliquées
M. PARSY Fernand	Mécanique des Fluides
M. POVY Jean-Claude	Sciences Appliquées
M. RACZY	Radioélectrique
M. ROUSSEAU Jean-Paul	Biologie Animale
M. ROYNETTE Bernard	Mathématiques
M. SALMER Georges	Electronique
M. SMET Pierre	Physique
M. VANDORPE Bernard	Sciences Appliquées
M. WATERLOT Michel	Géologie Générale
Mme ZINN **JUSTIN** Nicole	Mathématiques

Je tiens à exprimer toute ma gratitude,

à Monsieur le Professeur BACCHUS, directeur de l'UER Informatique, Electronique, Electrotechnique, Automatique de l'Université des Sciences et Techniques de Lille, qui m'a fait l'honneur d'accepter la présidence du jury,

à Monsieur CARREZ et Monsieur HUARD, tous deux maître de conférences à l'Université des Sciences et Techniques de Lille, qui ont bien voulu faire partie du jury.

Je souhaite que Monsieur HUARD trouve ici l'expression de ma profonde reconnaissance pour m'avoir donné l'idée de ce travail et m'avoir permis par ses conseils de le mener à bien, en ne ménageant ni son temps, ni sa peine.

Je tiens également à remercier la Direction des Etudes et Recherches E.D.F. de Clamart qui grâce au concours de Monsieur HUARD m'a aidé matériellement et techniquement dans la réalisation de ce travail.

Je remercie Mademoiselle DRIESSENS, qui, par sa rapidité et sa gentillesse, a permis la réalisation matérielle de cette thèse.

T A B L E des M A T I E R E S

***

***

E R R A T A

Chapitre	Paragraphe	Page	Ligne	Lire
II	II.3.2	II.6	4	$X_i = A_i^0 + A_i^{\bar{I}}(-X_{\bar{I}})$.
II	II.4.4	II.14	3	tel que le rang soit .
III	III.2.4	III.2	dernière	colonne A^{k_0} des constantes .
IV	IV.3.2	IV.6	19	ce qui conduit à $r_{i0} < 0$.
IV	IV.3.2	IV.7	4	$r_{i0} \leq A_i^{\bar{I}} x'_{\bar{I}\, k_0}$.
IV	IV.7	IV.12	14	$x_0^{k_0+1} = x_0^{k_0} - \frac{A_0^{k_0}}{r_{0s}} \cdot r_{00}$.
IV	IV.7	IV.12	16	$x_0^{k_0+1} \leq x_0^{k_0} - r_{00} = A_0^{k_0} - r_{00} = N_0^0$.
IV	IV.8.1	IV.14	7	$r_{i0} \leq A_i^{\bar{I}_i} x^{k}_{\bar{I}_i^+} + A_i^{\bar{I}_i} x^{k}_{\bar{I}_i^-} \leq A_i^{\bar{I}_i} x^{k}_{\bar{I}_i^-}$.
IV	IV.8.1	IV.14	10	$r_{i0}-1 \geq A_i^{\bar{I}_i} x^{k}_{\bar{I}_i^+} + A_i^{\bar{I}_i} x^{k}_{\bar{I}_i^-} \geq A_i^{\bar{I}_i} x^{k}_{\bar{I}_i^-}$.
V	V.3.1	V.5	5	suivant : Tableau V.3
VI	VI.1.2	VI.3	4	tels que leurs composantes soient
VII	VII.2.1	VII.5	5	En effet selon cette règle
VII	VII.2.3	VII.13	20	de la même façon que dans les
VII	VII.4	VII.16	27	la règle du (§ IV.4.5)
VIII	VIII.2.1	VIII.2	12	de la référence [11]
VIII	VIII.2.2	VIII.4	9	du graphe

LISTE des SYMBOLES et NOTATIONS EMPLOYEES

N	:	ensemble des entiers $\{0,1,2,...\}$
$\mathbb{R}$	:	ensemble des nombres réels
$[a]$	:	partie entière inférieure ou égale à a
A	:	matrice des coefficients des variables du programme linéaire
a	:	matrice colonne, second
A^j	:	colonne n° j de la matrice A
A_i	:	ligne n° i de la matrice A
A_i^j	:	élément (i,j) de la matrice A
L	:	ensemble des indices des lignes de la matrice A
J	:	ensemble des indices des colonnes de la matrice A
a	:	vecteur colonne second membre de composantes a_i
f	:	vecteur ligne de la fonction économique de composantes f^j
X	:	vecteur colonne de variable primale
X_i	:	composante n° i du vecteur X
u	:	vecteur ligne de variables duales u_i
d^j	:	critère de candidature
X_I	:	ensemble des composantes de X dont les indices appartiennent à I
A_iX	:	produit scalaire de la ligne i de la matrice A par le vecteur colonne X équivalent à $\sum_{j=1}^{n} A_i^j X_j$
$\|I\|$	:	cardinal de I
A^S	:	matrice constituée par les colonnes d'indice appartenant à l'ensemble S
A^I	:	matrice de base ou associée à la base I
V	:	variété linéaire engendrée par les équations du système $AX=a$
Γ	:	cône polyèdrique engendré par les inégalités $X \geq 0$
E	:	ensemble des solutions réalisables et réelles
$G(X,U)$	:	graphe dont l'ensemble des sommets X_i est X et l'ensemble des arcs u_i est u.

Chapitre 0

PROGRAMMATION LINEAIRE EN VARIABLES ENTIERES

- Généralités -

0.1 Introduction

En économie, un nombre élevé de productions et d'activités se présentent en unités indivisibles. En effet une entité telle qu'une demi-locomotive ou un quart de navire est dépourvue de tout sens.

Les résultats en nombres rationnels de la programmation linéaire habituelle ne conviennent pas lorsqu'il s'agit par essence, de réponses entières. De nombreux exemples montrent que la solution optimale entière n'est pas du tout voisine de la solution optimale rationnelle et même parfois ces solutions n'ont aucun rapport ; c'est le cas où toutes les variables de base de la solution continue ont des valeurs nulles dans la solution entière. Par conséquent, arrondir ne conduit pas toujours à une solution acceptable, d'où la nécessité d'étudier la résolution numérique exacte des programmes linéaires en nombres entiers.

0.2 Champ d'application de la programmation linéaire en variables entières

Le domaine de la programmation linéaire en nombres entiers est très vaste. L'énumération suivante des problèmes qui doivent par essence être résolus en variables entières n'est nullement exhaustive. Nous la donnons à titre indicatif [34], [46].

0.2.1 Problèmes dits de transport

La principale caractéristique de ces problèmes est que la matrice de formulation des contraintes est totalement unimodulaire [12] c'est-à-dire que tout sous-déterminant de cette matrice est nul ou égal à ±1.

Cette section comprend entre autres les problèmes

- d'affectation et de constitution d'équipes compatibles
- du plus court chemin [8]

0.2.2 Problèmes liés à la théorie des graphes tels que :

- Recherche des nombres chromatiques d'un graphe ([8], page 31)
- Détermination de chemins ou circuits hamiltoniens : ceci revient à trouver un ordre optimum pour effectuer un certain nombre d'opérations (page 103 de [8]).

Dans cette rubrique rentrent :

* Le problème du voyageur de commerce qui cherche un ordre minimisant le coût de transport pour la visite d'un groupe de villes

* Les problèmes d'ordonnancement dans les ateliers de production de pièces nécessitant des opérations sur différentes machines.

- Couverture minimale d'un graphe (page 171 de [8]) :

Etant donné un graphe G(X,U) de n sommets X_i et m arêtes U_j, on appelle couverture tout sous-ensemble $V \subset U$, tel que tout sommet $X_i \in X$ soit l'extrémité d'au moins une arête de V.

Cette section comprend les problèmes de :

* distribution de marchandises avec minimisation des coûts de transport entrepôts-clients

* établissement de systèmes de relais dans un central téléphonique ou de postes d'aiguillage sur un réseau ferrovière etc.

0.2.3 Problèmes de planification et d'investissement :

Détermination du nombre optimal d'unités de production de différents types dans une région déterminée et selon un projet d'implantation établi d'avance.

Définition de l'ensemble des investissements indépendants et compatibles qui rapportent le maximum de profit ou qui miniminisent les coûts ou charges

fixes ; le capital d'investissement étant fixé à l'avance.

0.2.4 D'autres types de problèmes à variables discrètes tels que :

- Répartition des containers dans les cales d'un port-container de manière à maximiser le fret.

- Approvisionnement avec remise variable par paliers et où les coûts varient linéairement de seuil en seuil.

0.3 Synthèse mathématique et types généraux de problèmes [46] :

Vus sous un angle mathématique, les programmes linéaires en variables bivalentes ou entières appartiennent à un nombre limité d'ensembles généraux non disjoints. Ce sont d'ailleurs les ensembles des :

0.3.1 Problèmes à contraintes mutuellement exclusives ou à groupes de contraintes mutuellement exclusives :

Du point de vue pratique, le domaine des solutions réalisables est une union et non une intersection de domaines convexes. Cette union est en général non convexe.

Citons quelques exemples [46] dans R^2.

0.3.1.1 Les contraintes d'un programme donné étant :

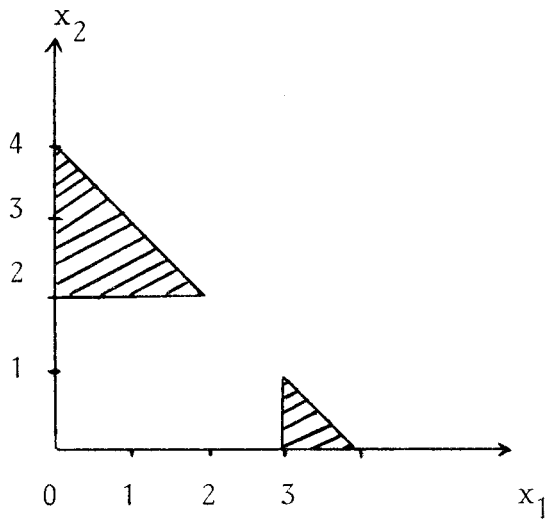
Domaine non connexe :

$$\begin{cases} x_1 + x_2 \leq 4 \\ x_1 \geq 3 \quad \underline{\text{ou}} \quad x_2 \geq 2 \\ x_1, x_2 \geq 0 \end{cases}$$

Le programme équivalent est le suivant :

$$\begin{cases} x_1 + x_2 \leq 4 \\ x_1 - 3 + 3(1-y_1) \geq 0 \\ x_2 - 2 + 2(1-y_2) \geq 0 \\ y_1 + y_2 = 1 \\ y_1, y_2 \geq 0 \text{ et entières} \end{cases}$$

Le domaine des solutions réalisables de ce dernier programme est le suivant, il est non connexe : (domaine hachuré).

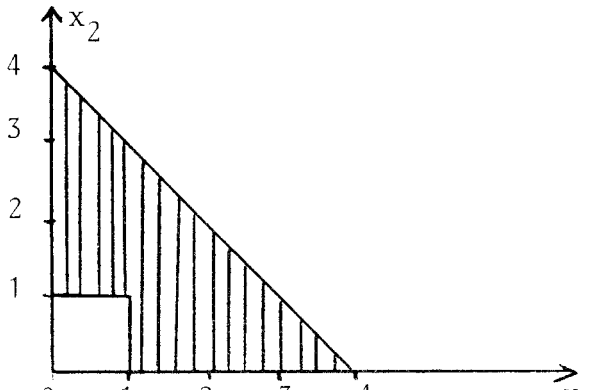


0.3.1.2 Domaine non convexe :

$$\begin{cases} x_1 + x_2 \leq 4 \\ x_1 \geq 1 \quad \underline{ou} \quad x_2 \geq 1 \\ x_1, x_2 \geq 0 \end{cases}$$

Le programme équivalent à ce dernier étant :

$$\begin{cases} x_1 + x_2 \leq 4 \\ x_1 - 1 + (1-y_1) \geq 0 \\ x_2 - 1 + (1-y_2) \geq 0 \\ y_1 + y_2 = 1 \\ y_1, y_2 \in N \end{cases}$$



Domaine (hachuré) non convexe

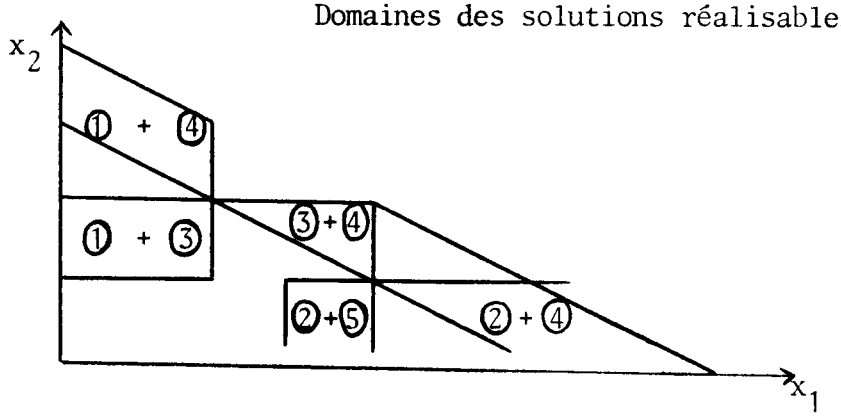
0.3.2 Problèmes à ensembles combinatoires de contraintes [46] :

Les variables sont astreintes à vérifier ℓ des m contraintes du programme linéaire :

exemple : x_1 et x_2 doivent vérifier deux des quatre ensembles de contraintes suivants :

$$① \begin{cases} x_1 \leq 2 \\ x_2 \geq 1 \end{cases} \qquad ② \begin{cases} x_1 \geq 3 \\ x_2 \leq 1 \end{cases} \qquad ③ \begin{cases} x_1 \leq 4 \\ x_2 \leq 2 \end{cases} \qquad ④ \begin{cases} x_1 + 2x_2 \geq 6 \\ x_1 + 2x_2 \leq 8 \end{cases}$$

Domaines des solutions réalisables

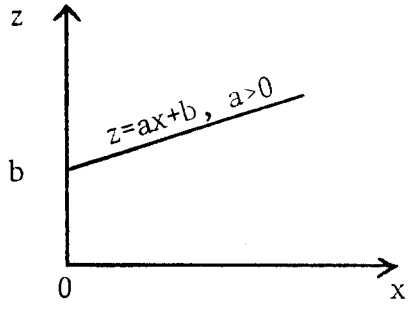


0.3.3 Problèmes à contraintes conditionnelles :

Ce sont les problèmes du type charges fixes dont la formulation est la suivante :

$$\min z$$

$$\begin{cases} z = ax + b & \text{si } x > 0 \\ z = 0 & \text{si } x = 0 \end{cases}$$



0.4 Méthodes de programmation linéaire en nombres entiers :

A la différence de la méthode universelle de programmation linéaire classique (variables continues), il existe autant de méthodes de programmation linéaire en variables entières que de cas de problèmes à traiter. Cette diversité et ce nombre élevé de méthodes témoignent d'une part du caractère insuffisant de chacune d'elles et d'autre part rendent difficile toute comparaison entre elles.

Notons que ces méthodes utilisent des critères d'investigation heuristiques, qui sont souvent dépendants du cas de programme traité.

Les méthodes sont de deux types :

- les méthodes utilisant la technique de la programmation linéaire habituelle [22], [23], [24], [7]

- les méthodes utilisant des techniques exploratoires (booléennes, arborescentes, combinatoires,...) [2], [3], [18] ,[20] ,[21], [32].

0.4.1 Les méthodes à base simpliciale :

Ces méthodes utilisent l'algorithme de Dantzig ou ses résultats en variables continues. Afin d'atteindre la solution entière, elles procèdent :

- soit par troncature du polyèdre des solutions réalisables en continu (§ I.2.2), [22], [23], [24], [7] ...

- soit en prenant la fonction économique comme paramètre ou contrainte de troncature [18].

0.4.2 Les méthodes exploratoires :

Les techniques employées sont diverses :

- la procédure booléenne [18], [19], [20]

- les procédures combinatoires-arborescentes [2], [3], [21], [32]

- les procédures basées sur la méthode des centres [36].

Dans les chapitres suivants nous étudierons les méthodes à base simpliciale et utilisant la technique de troncature des polyèdres des solutions réalisables en continu et nous nous intéresserons particulièrement aux méthodes de Gomory [22], [23], [24], et à leur convergence.

Le chapitre I rappelle les méthodes simpliciale et duale-simpliciale, fixe les différentes notations et formulations employées par la suite et définit les relations d'ordre lexicographique.

Les chapitres II, III, IV et V seront consacrés à l'exposé des méthodes de Gomory et à une interprétation géométrique de la troncature du polyèdre des solutions réalisables continues.

Dans le chapitre VI nous décrirons sommairement quelques codes linéaires basés sur les principales méthodes énumérées précédemment.

Quant aux chapitres VII et VIII ils seront consacrés aux comparaisons des résultats numériques des principaux codes linéaires basés sur les algorithmes de Gomory et sur les algorithmes exploratoires. Les codes basés sur les méthodes de Gomory et les méthodes exploratoires dont les résultats numériques sont donnés au chapitre VIII, ont été programmés dans le même esprit par une équipe homogène et testés sur une même série d'exemples numériques sur le CDC 6600 de la Direction des Etudes et Recherches de l'E.D.F. à Clamart.

Chapitre I

DEFINITIONS ET RAPPELS

Dans ce chapitre nous nous proposons de rappeler certaines définitions et de préciser les formulations que nous utiliserons par la suite.

I.1 Formulation du problème et notations :

Le problème de programmation linéaire consiste en la recherche de l'optimum d'une fonction linéaire de n variables x_j, liées par des relations linéaires appelées contraintes.

I.1.1 Forme canonique

Symboliquement cette forme s'écrit de la manière suivante :

$$\begin{cases} \min fx & (I.1) \\ Ax = a & (I.2) \\ x > 0 \end{cases}$$

où

- f est un vecteur à n composantes f^j, $j \in J$ et $|J| = n$
- A matrice $(L \times J)$ à coefficients A_i^j, $i \in L$, $j \in J$ et $|L| = m$
- a vecteur colonne à m composantes a_i, $i \in L$
- x vecteur colonne à n composantes x_j, $j \in J$

I.1.2 Forme standard :

La forme standard se présente symboliquement comme suit :

$$\begin{cases} \min f^s x_s & (I.4) \\ A^s x_s \leq a & (I.4) \\ x_s \geq 0 & (I.5) \end{cases}$$

où cette fois-ci

- f^s est un vecteur ligne à n composantes f^j, $j \in J$
- A^s est une matrice (L×J) à coefficients A_i^j, $i \in L$, $j \in J$
- x_s est un vecteur colonne à n composantes x_j, $j \in J$
- a est un vecteur colonne à m composantes a_i, $i \in L$.

Nous verrons plus loin que cette forme se prête particulièrement bien à la théorie de la dualité.

I.1.3 Forme mixte :

La forme mixte contient simultanément des contraintes des formes canonique et standard :

$$\begin{cases} \min fx & (I.7) \\ A_{L_1} x \leq a_{L_1} & (I.8) \\ A_{L_2} x = a_{L_2} & (I.9) \\ x \geq 0 \end{cases}$$

avec $L_1 \cup L_2 = L$ et $L_1 \cap L_2 = \emptyset$.

I.1.4 Remarques pratiques :

Le passage de l'une des formes précédentes à l'autre est assuré par des opérations élémentaires :

- *Opération n°1*

Toute équation de la forme $A_i x = a_i$ (A_i représente la ligne n°i de A) peut être remplacée par deux inéquations :

$$\begin{cases} A_i x \leq a_i \\ -A_i x \geq -a_i \end{cases}$$

- *Opération n°2*

Toute inéquation $A_i x \leq a_i$ peut donner une équation par l'addition au premier membre d'une variable x_{i+n}, appelée variable d'écart positive ou nulle

$$A_i x + x_{i+n} = a_i .$$

Cette opération assure le passage de la forme standard à la forme canonique.

I.2 Solution réalisable - Solution optimale-base et Solution de base :

Considérons la forme canonique déduite par l'opération 2 de la forme standard :

$$\begin{cases} \min f^S x_S & (I.10) \\ A^S x_S \leq a & (I.11) \\ x_S \geq 0 & (I.12) \end{cases}$$

par addition des variables d'écart x_{i+n}, $i \in L$ on a la forme canonique suivante (P_0) :

$$P_0 \quad \begin{cases} \min fx & (I.13) \\ Ax = a & (I.14) \\ x \geq 0 & (I.15) \end{cases}$$

telle que

- $f = (f^S, 0)$
- $x = (x_1, x_2 \ldots x_n, x_{1+n}, \ldots, x_{m+n})$
- $A = (A^S, U)$

où U est une matrice unité (L×L) de dimension .

I.2.1 Base :

Les définitions et propriétés qui suivent sont relatives au programme linéaire P_0. Nous supposons en outre que le système (I.14) est régulier c'est-à-dire rang (A) = m et que $m \leq n$.

Nous appelons base du programme linéaire P_0 tout sous-ensemble I de J tel que $|I| = m$. Les m colonnes A^j, $j \in I$ constituent une sous-matrice régulière de A. Nous notons cette sous-matrice par A^I, m étant son rang ; A^I est appelée matrice de base du programme P_0.

I.2.2 Solution basique - Solution réalisable

On appelle solution de base ou solution basique du programme P_0 tout vecteur x vérifiant les relations (I.14). Toute solution basique vérifiant (I.15) est dite réalisable.

Soit V la variété linéaire engendrée par l'ensemble des équations du système (I.14).

$$V = \{x \mid Ax = a\}$$

Soit Γ le cône polyèdrique engendré par les contraintes de non négativité (I.15).

$$\Gamma = \{x \mid x \geq 0\}.$$

L'ensemble des solutions réalisables est constitué par le polyèdre E, intersection de V et de Γ soit :

$$E = V \cap \Gamma$$

I.2.3 Solution optimale :

Une solution réalisable est dite optimale si elle minimise fx sur E.

I.2.4 Variables de base - Variables hors base :

Nous appellerons variable de base toute variable x_j associée à un vecteur colonne A^j de A^I, $j \in I$. Les variables x_ℓ, $\ell \in \bar{J} = J-I$ sont appelées variables hors base ou variables secondaires.

Les composantes x_j, $j \in J$ de X peuvent donc être partitionnées en deux sous-ensembles disjoints notés x_J et $x_{\bar{J}}$ représentant respectivement les m variables de base et les n variables secondaires.

Posons $x(I) = (x_I, x_{\bar{I}})$. La solution de base, associée à la base I de P_0, sera représentée par le vecteur x(I) défini par

$$x(I) = \begin{cases} A^{I}_{x_I}(I) = a & (I.16) \\ x_{\bar{I}}(I) = 0 \end{cases}$$

Une solution de base est dite dégénérée si une ou plusieurs composantes de x_I sont nulles.

La solution x(I) représente graphiquement un sommet du polyèdre E.

.3 Rappel de la méthode simplicial [35]

I.3.1 Généralités :

Il est toujours possible d'exprimer les variables de base x_I en fonction des variables hors base $x_{\bar{I}}$.

Le système (I.14) est équivalent au suivant

$$(A^I, A^{\bar{I}}) \begin{pmatrix} x_I \\ x_{\bar{I}} \end{pmatrix} = a \qquad (I.18)$$

soit

$$A^I x_I + A^{\bar{I}} x_{\bar{I}} = a \qquad (I.19)$$

Soit en multipliant les deux membres de (I.19) par $(A^I)^{-1}$ (A^I régulière) on a

$$(A^I)^{-1} A^I x_I + (A^I)^{-1} A^{\bar{I}} x_{\bar{I}} = (A^I)^{-1} a \qquad (I.20)$$

Posons

$$t(I) = (A^I)^{-1} a \qquad (I.21)$$

$$T(I) = (A^I)^{-1} A \qquad (I.22)$$

d'où l'expression de x_I en fonction de $x_{\bar{I}}$

$$x_I = t(I) - T^{\bar{I}}(I)\, x_{\bar{I}} \qquad (I.23)$$

L'expression de la fonction économique est alors

$$fx = (f^I, f^{\bar{I}}) \begin{pmatrix} x_I \\ x_{\bar{I}} \end{pmatrix} = f^I x_I + f^{\bar{I}} x_{\bar{I}}$$

soit en tenant compte de (I.23)

$$fx = f^I\, t(I) + (f^{\bar{I}} - f^I\, T^{\bar{I}}(I))\, x_{\bar{I}} \qquad (I.24)$$

posons $$d(I) = f - f^I\, T^{\bar{I}}(I) \qquad (I.25)$$

par définition d(I) est le vecteur critère de candidature de l'algorithme simplicial (§ I.3.3).

Compte tenu de (I.25) l'équation (I.24) devient

$$fx = f^I\, t(I) + d^{\bar{I}}(I)\, x_{\bar{I}} \qquad (I.26)$$

Notons au passage que $d^I(I) = 0$ pour toute base I.

Si x(I) est une solution basique réalisable ($x(I) \geq 0$) on a, compte tenu de (I.16) ($x_{\bar{I}}(I)=0$) :

$$t(I) \geq 0 \qquad (I.27)$$

et

$$fx(I) = f^I t(I) \qquad (I.28)$$

I.3.2 Théorème

Si pour une solution de base x(I) on a $d^{\bar{I}} \geq 0$ alors la solution de base réalisable x(I) est optimale pour le programme P_0.

En effet d'après (I.25) et (I.28) on a

$$\cdot \quad fx(I) = f^I t(I)$$

$$fx = f^I t(I) + d^{\bar{I}} x_{\bar{I}}(I)$$

soit par différence :

$$fx(I) - fx = - d^{\bar{I}} x_{\bar{I}}(I) \qquad (I.29)$$

Comme $x_{\bar{I}} \geq 0$ et $d^{\bar{I}} \geq 0$ le second membre de (I.29) est ~~non~~ négatif c'est-à-dire :

$$fx(I) \leq fx \qquad (I.30)$$

La solution x(I) pour laquelle $d^{\bar{I}} \geq 0$ est dite alors primale réalisable, elle est optimale.

I.3.3 Algorithme simplicial :

Si la solution x(I) n'est pas optimale, c'est-à-dire, qu'il existe au moins un indice $s \in \bar{I}$ tel que $d^s < 0$, alors il est possible de trouver une solution de base x(I') telle que $fx(I') \leq fx(I)$. La nouvelle base étant I'.

Le procédé d'amélioration de la solution de base consiste en l'échange entre une variable de base et une variable hors-base : c'est un changement de base.

Considérons une nouvelle solution de base $x(\theta, I)$, fonction du paramètre θ et telle que :

$$x(\theta,I) = \begin{cases} x_s = \theta & s \in \bar{I} \\ x_{\bar{I}-\{s\}} = 0 & \\ x_I = t(I) - T^s(I)\,\theta & \end{cases} \qquad (I.31)$$

Pour tout $\theta > 0$ il est clair que pour que $x(\theta, I)$ soit réalisable il faut et il suffit que

$$x_i \geq 0 \qquad \forall\, i \in I \qquad (I.32)$$

ce qui est équivalent à $T_i^s \theta \leq t_i \qquad \forall\, i \in \bar{I}$; deux cas sont possibles :

- $T_i^s \leq 0$: (I.32) est vérifié pour tout i tel que $T_i^s \leq 0$, $i \in I$

- $T_i^s > 0$: (I.32) reste réalisable quand la valeur de θ ne dépasse pas une certaine valeur θ_M donnée par :

$$\theta_M = \min_{i \in I \cap \{i \mid T_i^s > 0\}} \left(\frac{t_i}{T_i^s}\right) \qquad (I.33)$$

La variation de la fonction économique

$$\Delta fx(\theta, I) = fx(\theta, I) - fx(I) = d^s \theta \qquad (I.34)$$

d^s étant négatif $\Delta fx(\theta,I)$ est d'autant plus grande que θ est grand.

Soit $r \in I$ tel que : $\theta_M = \dfrac{t_r}{T^s_r}$ (valeur donnée par (I.33)).

La variable x_r, devenue nulle, quitte la base et la variable $x_s = \theta_M$ entre dans la base. La base est alors constituée par l'ensemble $I' = I - \{r\} + \{s\}$.

Les nouvelles variables de base sont alors :

$$x_I(I') = \begin{cases} x'_i = x_i - \dfrac{T^s_i}{T^s_r} t_r & \forall\, i \in I - \{r\} \\ x'_s = \dfrac{t_r}{T^s_r} \end{cases}$$

et

$$x'_{\bar{I}'}(I') = \begin{cases} x'_r = 0 \\ x'_{\bar{I}-\{s\}} = 0 \end{cases}$$

D'après (I.34) on vérifie bien que $x(I')$ est meilleure que $x(I)$ $(fx(I') \leq fx(I))$.

Si l'ensemble $\{i \mid T^s_i > 0\} \cap I$ est vide, la solution $x(\theta,I)$ devient infinie quand θ augmente indéfiniment. De même $fx(\theta,I)$ devient infini. Le programme P_0 a alors un minimum infini. Dans ces conditions nous dirons que P_0 n'a pas de solution optimale finie.

Si de nouveau $x(I')$ n'est pas solution optimale c'est-à-dire qu'il existe au moins un $s' \in \bar{I}'$ tel que $d^{s'} < 0$, on applique une fois de plus la procédure précédente. Cette procédure constitue l'algorithme simplicial dont le vecteur critère de candidature est le vecteur $d(I)$.

I.3.4 Convergence de l'algorithme simplicial :

La variation de la base I ne se poursuit pas indéfiniment. En effet les valeurs successives de la fonction économique forment une suite non croissante et à chaque base I la valeur de la fonction économique est unique.

Si à l'itération k, Δfx est non nulle on est certain de ne pas rencontrer

une base déjà explorée (unicité et monotonie de la fonction économique).

Le nombre de bases possible étant fini, (la suite monotone des valeurs de la fonction économique est bornée inférieurement), l'algorithme ne peut, éventuellement, tourner indéfiniment que dans le cas de dégénérescence, cas où une base peut être rencontrée plusieurs fois.

I.4 Méthode de Lemke ou méthode duale-simpliciale [40]

I.4.1 Généralités et propriétés de la dualité :

Avant de décrire l'algorithme dual-simplicial, rappelons brièvement les principales propriétés de la dualité. Nous nous sommes inspirés de la référence [46] quant au développement des paragraphes suivants.

Considérons un programme primal sous la forme suivante et essayons de définir le programme dual correspondant :

Programme primal		Programme dual
$A_i x \geq a_i$	$i \in L_1$	$u_i \geq 0$
$A_i x = a_i$	$i \in L - L_1 = \bar{L}_1$	(u_i) quelconque
$x_j \geq 0$	$j \in J_1$	$A^j u \leq f_j$
(x_j) quelconque	$j \in \bar{J}_1 = J - J_1$	$A^j u = f_j$
min fx		max ua

où u est un vecteur ligne à m composantes $u = (u_{L_1}, u_{\bar{L}_1})$.

Cette définition possède un caractère involutif :

- à chaque contrainte-inéquation correspond une variable duale
- à chaque contrainte-équation correspond une variable duale quelconque
- à la forme minimiser correspond la forme duale maximiser de la fonction économique.

Les deux programmes précédents constituent un couple dual ; les variables x_j sont des variables primales, quant aux u_i elles sont dites variables duales.

Considérons maintenant le couple suivant de programmes duals écrits sous la forme matricielle :

$$\text{Primal} \begin{cases} Ax \geq a \\ x \geq 0 \\ \min fx \end{cases} \qquad \text{Dual} \begin{cases} u \geq 0 \\ uA \leq f \\ \max ua \end{cases} \qquad \text{(I.35)}$$

Les résultats suivants peuvent être aisément démontrés :

- Si $\bar{x}$ et $\bar{u}$ constituent un couple de solutions réalisables des deux programmes duals, on a $f\bar{x} \geq \bar{u}a$.

- Si en plus $f\bar{x} = u\bar{a}$, ces solutions réalisables sont optimales.

- Une affirmation et une seule des propriétés suivantes est vraie :

  * Les programmes duals ont des solutions optimales $\bar{x}, \bar{u}$ telles que $f\bar{x} = u\bar{a}$

  * Aucun des programmes ne possède une solution optimale
    - si l'un au moins des programmes n'a pas de solution réalisable
    - si l'un des programmes a au moins une solution réalisable, l'ensemble de ces solutions réalisables n'est pas borné et l'optimum de ce programme est infini.

I.4.2 Théorème de la Dualité [46]

Etant donné un couple de programmes duals, une condition nécessaire et suffisante pour qu'une solution $\bar{x}$ (ou $\bar{u}$) de l'un des programmes soit optimale est qu'il existe une solution $\bar{u}$ (ou $\bar{x}$) de l'autre programme tel que $f\bar{x} = \bar{u}a$. La solution $\bar{u}$ (ou $\bar{x}$) est alors elle-même optimale.

I.4.3 Algorithme dual-simplicial [40]

Soit un programme primal écrit sous la forme canonique (§ I.1.1) et son programme dual :

$$\text{Primal} \begin{cases} Ax = a & \text{(I.36 a)} \\ x \geqslant 0 & \text{(I.37 a)} \\ \min fx & \text{(I.38 a)} \end{cases} \qquad \text{Dual} \begin{cases} (u) \text{ quelconque} & \text{(I.36 b)} \\ uA \leqslant f & \text{(I.37 b)} \\ \max ua & \text{(I.38 b)} \end{cases}$$

Supposons connue une base I du primal, et soit A^I la matrice de base associée à I :

- une solution est dite duale réalisable si $\forall\, j \in \bar{I}$ le critère de candidature d^j est non négatif

- une solution est dite primale réalisable si $x(I) \geqslant 0$

- une solution x(I) à la fois primale et duale réalisable est dite optimale.

Une solution duale réalisable implique des $d^j \geqslant 0$, $\forall\, j \in \bar{I}$ c'est-à-dire d'après (I.25)

$$f^I (T^{\bar{I}}(I))^j \leqslant f^j$$

ou

$$f^I (A^I)^{-1} A^j \leqslant f^j \qquad \text{(I.39)}$$

en posant $$u = f^I (A^I)^{-1} \qquad \text{(I.40)}$$

(I.39) devient $$uA^j \leqslant f^j \qquad \text{(I.41)}$$

La comparaison de (I.41) et (I.37 b) montre que u est une solution réalisable du programme dual.

D'après le théorème de Dualité (§ I.4.2) si u n'est pas solution optimale du programme dual, la solution x(I) du primal n'est pas réalisable, c'est-à-dire qu'il existe au moins un $i \in I$ tel que $x_i < 0$. Réciproquement si pour $r \in I$, $x_r < 0$, il existe une solution réalisable u' du dual meilleure que u.

En effet, considérons un vecteur $u' = u - (A^I)^{-1}_r \ominus$ (I.42) , multiplions chaque membre de (I.42) par a.

$$u'a = ua - \theta(A^I)_r^{-1}a = ua - \theta t_r \qquad (I.43)$$

ce qui montre que $u'a > ua$ pour tout $\theta > 0$.

u' est une solution réalisable si $u'A \leqslant f$.

Détaillons (I.42) après avoir multiplié à gauche chaque membre par A^j

$$\cdot\ u'A^j = uA^j - \theta T_r^j = f^j - \theta T_r^j \qquad \forall\ j \in \bar{I}$$

$$\cdot\ u'A^j = uA^j - 0 \quad = f^j \leqslant f^j \qquad \forall\ j \in I - \{r\}$$

$$\cdot\ u'A^r = uA^r - \theta \quad = f^r - \theta \leqslant f_r$$

La première des trois relations précédentes montre qu'il pourrait exister une valeur θ pour laquelle la solution n'est plus duale réalisable ; deux cas sont possibles :

- $T_r^j \geqslant 0 \quad \forall\ j \in \bar{I}$ la solution u' est duale réalisable mais quand θ croît indéfiniment la fonction économique n'a pas de maximum fini et par suite le primal et le dual n'ont pas de solution.

- $T_r^j < 0$, pour un $j \in \bar{I}$, u' est une solution duale réalisable si et seulement si $u'A^j - \theta T_r^j \leqslant f^j$, $\forall\ j \in \bar{I}$, c'est-à-dire θ borné supérieurement par

$$\theta_M = \min_{j \in \bar{I} \cap \{j \mid T_r^j < 0\}} \frac{uA^j - f^j}{T_r^j} \qquad (I.44)$$

soit s tel que $$\theta_M = \frac{uA^s - f^s}{T_r^s} \qquad (I.45)$$

La variable x_s devient une variable de base tandis que x_r quitte la base.

Le critère de sortie de la base peut être pour l'instant :

$$x_r = \min_{i \in I} x_i$$

La nouvelle base devient alors $I' = I - \{r\} + \{s\}$ et la solution duale réalisable est $u' = f^{I'}(A^{I'})^{-1}$.

On a ainsi trouvé une base duale réalisable I' meilleure que I et on vérifie bien que $u'A \lesssim f$, ($u'A^I = f^I$ et $u'A^{\bar{I}'} \lesssim f^{\bar{I}'}$).

La solution du primal correspondant à u' est x(I') :

- si $x(I') \gtrsim 0$ la solution primale est optimale

- s'il existe au moins un $i \in I'$ tel que $x_i < 0$ il est possible d'itérer le processus précédent et de trouver une solution u'' meilleure que u' ou montrer la non "réalisabilité" du programme.

I.4.4 Convergence de l'algorithme dual-simplicial :

La variation de la fonction économique est Δf telle que

$$\Delta f = - t_r \Theta.$$

La fonction économique du dual étant strictement croissante ($t_r < 0$) pour $\Theta > 0$, aucune base rencontrée ne peut donc être revue du fait de la monotonie de la fonction économique duale. Comme le nombre de base est fini on converge nécessairement vers la solution optimale.

Le cas de $\Delta f = 0$, désigné sous le nom de dégénérescence duale, fait en sorte qu'une base peut être rencontrée plusieurs fois et favorise un cyclage.

5 Relation d'ordre lexicographique :

Nous avons signalé au (§ I.4.4) le danger de cyclage de l'algorithme dual-simplicial à la suite de dégénérescence de l'une des solutions de base, cas très fréquent dans la pratique. Afin d'éviter tout cyclage indéfini il est nécessaire de classer les candidats relatifs à une règle de choix donnée. Ce classement se fait par une relation d'ordre dite lexicographique, relation d'ordre généralisée.

I.5.1 Définitions et notations

I.5.1.1 Définition :

Un vecteur x est dit lexicographiquement positif si sa première composante non nulle est positive. Cette propriété sera notée par $x \overset{\ell}{\gtrsim} 0$. Le vecteur x est dit strictement lexicographiquement positif si sa première composante est strictement positive soit $x \overset{\ell}{>} 0$.

I.5.1.2 Une matrice A est dite lexicographiquement positive si tous les vecteurs colonnes de A sont lexicographiquement positifs.

I.5.1.3 Un vecteur x est dit lexicographiquement supérieur à un vecteur y si $x-y \overset{\ell}{\gtrdot} 0$.

Notons cette propriété par $x \overset{\ell}{\gtrdot} y$ (ou x ℓ-sup y).

I.5.1.4 Propriétés :

La relation d'ordre lexicographique est une relation d'ordre réflexive, antisymétrique et transitive.

1.5.1.5 Définition :

Deux ou plusieurs vecteurs lexicographiquement positifs sont dits ex aequo si leurs composantes de rang 1 sont identiques.

Nous dirons qu'ils sont ex aequo d'ordre k si leurs k premières composantes sont identiques. Ces premières composantes identiques peuvent être nulles en partie ou en totalité.

Remarquons que nous n'avons parlé que de vecteurs lexicographiquement positifs, il est possible de définir parallèlement des vecteurs lexicographiquement négatifs.

I.5.2 Théorème [22]

Etant donné un ensemble de n vecteurs $({}_1x, {}_2x, {}_3x, \ldots, {}_nx)$ non identiques et du même espace, il est toujours possible de les ordonner lexicographiquement s'ils sont tous lexicographiquement positifs (ou négatifs).

Dans la démonstration suivante nous supposons qu'il s'agit d'un ensemble de vecteurs lexicographiquement positifs.

1- Le cas d'ensemble de vecteurs tous strictement lexicographiquement positifs et non ex aequo revient à ordonner une suite de nombres positifs.

2- Cas de vecteurs ex aequo et non strictement lexicographiquement positifs :

Cherchons à les classer par ordre lexicographiquement croissant. Nous dirons que ${}_rx$ est l'inf-lexicographique des ${}_ix$, $i \in I - \{r\}$ ($I=\{1,2,\ldots r-1,r,r+1,\ldots n\}$) si ${}_ix - {}_rx \overset{\ell}{\geq} 0 \ \forall\ i \in I-\{r\}$.

Soit $J \subset I$ l'ensemble des indices des vecteurs ex aequo d'ordre k, si ${}_\ell x_j$ est la j^e composante de ${}_\ell x$, l'inf-lexicographique du sous-ensemble J est ${}_hx$ tel que

$$ {}_ix_{k+1} - {}_hx_{k+1} > 0 \qquad \forall\ i \in J - \{h\} \qquad (I.46) $$

En tenant compte de (I.46) on peut classer les vecteurs ex aequo et non strictement lexicographiquement positifs. L'ensemble I étant auparavant partitionné en sous-ensembles J_k de vecteur ex aequo d'ordre k. L'application répétée de (I.46) à la suite finie J_k permet d'ordonner tous les vecteurs ex aequo de l'ensemble I.

I.5.3 Exemples :

Soit à classer par ordre lexicographique croissant les vecteurs $\overset{\ell}{\geq} 0$ suivants :

$$ {}_1x = \begin{pmatrix} 0 \\ 1 \\ 2 \\ 3 \end{pmatrix}, \quad {}_2x = \begin{pmatrix} 1 \\ -3 \\ 4 \\ 0 \end{pmatrix}, \quad {}_3x = \begin{pmatrix} 2 \\ 3 \\ 2 \\ 1 \end{pmatrix}, \quad {}_4x = \begin{pmatrix} 2 \\ -2 \\ 3 \\ 0 \end{pmatrix}, \quad {}_5x = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 4 \end{pmatrix} $$

on a

$$ I = \{1,2,3,4,5\} \qquad I = J_1 \cup J_2 \cup J_0 $$

tels que

$$ J_1 = \{1,3,4\} \qquad J_0 = \{2\} \qquad J_2 = \{5\} $$

Les résultats suivants sont évidents :

- ${}_1x$ = inf. lex de $\{{}_1x, {}_3x, {}_4x\}$
- ${}_2x$ = inf. lex $\{{}_3x, {}_4x\}$
- ${}_5x$ = inf. lex $\{{}_1x, {}_2x, {}_3x, {}_4x, {}_5x\}$.

Le classement de ces vecteurs est le suivant

$$ {}_3x \overset{\ell}{\geq} {}_4x \overset{\ell}{\geq} {}_2x \overset{\ell}{\geq} {}_1x \overset{\ell}{\geq} {}_5x $$

I.5.4 Définition :

Un vecteur x lexicographiquement positif (ou négatif) est dit lexicographiquement nul de degré p si ses p premières composantes sont nulles :

$$\begin{cases} x_i = 0 & i=1,2,\ldots p \\ x_{p+1} > 0 \end{cases}$$

Les vecteurs ${}_5x$, ${}_1x$, ${}_2x$ des exemples du paragraphe précédent sont respectivement lexicographiquement nuls d'ordre 2, 1, 0.

Chapitre II

ALGORITHME N°2 DE GOMORY

I.1 Généralités :

En dehors de son application limitée au programme dont les coefficients de la fonction économique sont non négatifs, la deuxième méthode de Gomory [23] est une méthode itérative caractérisée par les deux points essentiels suivants :

a) Les coefficients du tableau simplicial au démarrage de l'algorithme, et après chaque itération sont tous entiers.

b) La méthode itérative de cet algorithme est basée sur l'algorithme dual simplicial (§ I.4.3) avec la particularité suivante : l'acheminement vers la solution optimale se fait par des solutions basiques entières.

Pour réaliser ces conditions supplémentaires, le pivot doit être à chaque itération égal à -1, (pivot négatif exigé par la méthode duale simpliciale (§ I.4.3) et égal à -1 pour que le tableau simplicial reste entier). Ceci exige le choix d'une ligne de pivot ayant cette propriété c'est-à-dire un pivot égal à -1 ou à défaut, la création d'une contrainte artificielle redondante, dont la génération se fera à partir de l'une des contraintes du programme linéaire.

Géométriquement cette contrainte artificielle est une contrainte de troncature du domaine E (§ I.2.2) des solutions réalisables du programme linéaire, contrainte qui réduit ce domaine mais qui n'exclut pas pour autant aucune des solutions réalisables entières.

Nous verrons au chapitre IV que le premier algorithme de Gomory est constitué de deux phases :

- une phase d'optimisation en variables continues
- une phase d'optimisation par l'algorithme dual simplicial en variables entières analogue à la procédure du second algorithme de Gomory ;

c'est la raison pour laquelle nous avons d'abord traité le 2ème algorithme.

II.2 Formulation du problème et notations :

II.2.1 Soit le programme linéaire P_0 formulé au § I.2 auquel on ajoute les contraintes supplémentaires d'intégrité de la solution x. Dans la formulation suivante nous supposons avoir tenu compte des variables d'écart dans x et dans Ax=a et de la non négativité des coefficients de la fonction économique :

$$\min fx \qquad \text{(II.1)}$$

$$Ax = a \qquad \text{(II.2)}$$

$$x \geq 0 \qquad \text{(II.3)}$$

$$x \text{ entier} \qquad \text{(II.4)}$$

$$f \geq 0 \qquad \text{(II.5)}$$

Introduisons dans (II.2) la contrainte redondante dont les coefficients sont ceux de la fonction économique :

$$\sum_{j=1}^{n} f_j^j x_j - x_0 = 0$$

Le programme obtenu est dit complété. La première composante de a, d'indice 0, sera $a_0=0$; x_0 est une variable supplémentaire de x, d'indice 0.

Pour avoir des notations homogènes posons :

$$A_0^j = f^j \qquad j=1,2,\ldots,n$$

$$A_i^0 = a_i \qquad i=1,2,\ldots,m.$$

II.2.2 Remarques :

a) Les coefficients A_i^j de A et a_i de a sont supposés entiers.

b) La matrice A est indicée en ligne de 0 à m (m+1 lignes) et en colonne de 0 à m+n (m+n+1 colonnes) $|J| = m+n+1$.

c) Désormais le vecteur x aura m+n+1 composantes et sera partitionné en 3 ensembles x_0, x_I et $x_{\bar{I}}$.

II.2.3 Solution de démarrage :

Comme les A_0^j sont non négatifs (II.5), $\forall\, j \in J$ une base duale admissible de démarrage est constituée par l'ensemble des indices $I = \{n+1,\ldots, n+m\}$, indices des variables d'écart $x_I(x_{n+1},\ldots,x_{n+m})$.

Les variables hors base étant $x_{\bar{I}} = (x_1,\ldots,x_n)$. La valeur de la fonction économique relative à cette base est $x_0 = A_0^0 = a_0 = 0$.

D'après (§ I.4.3) si $A^0 \geq 0$ la solution $(x_0, x_I, x_{\bar{I}})$ est optimale, sinon la méthode duale simpliciale déterminera une autre base duale admissible.

Explicitons les variables x_0, x_I en fonction des variables $x_{\bar{I}}$ et formulons comme suit le problème :

$$P_I \begin{cases} \max x_0 = A_0^0 + A_0^{\bar{I}}\,(-x_{\bar{I}}) & \text{(II.6)} \\ \text{tels que } x_i = A_i^0 + A_i^{\bar{I}}(-x_{\bar{I}}) \;\; i \in I & \text{(II.7)} \\ x_{\bar{I}} \text{ entiers positifs} & \text{(II.8)} \end{cases}$$

Compte tenu de (II.8) et de la remarque a du (§ II.2.2) les variables de base x_I et x_0 sont entières.

D'après le (§ 1.2.2) l'ensemble des solutions réalisables est constitué par le polyèdre E. Ce domaine E peut ne contenir aucune solution entière réalisable, dans ce cas le programme P_I n'a pas de solution entière.

II.3 Itération et troncature :

Rappelons que la base est dite :

- duale réalisable si $A_0^j \geq 0 \quad \forall\, j \in \bar{I}$
- primale réalisable si $A_i^0 \geq 0 \;\forall\, i \in I$
- optimale si la base est à la fois duale et primale réalisable.

Si la solution de départ du (§ II.2.3) n'est pas optimale, nous itérons l'algorithme dual-simplicial (§ I.4) pour trouver une autre solution, éventuel-

lement optimale.

Comme le pivot doit être égal à -1 nous serons probablement amenés à construire une contrainte supplémentaire ayant cette propriété (pivot = -1).

A l'itération n°k, notons, comme suit, les éléments du tableau simplicial A et de la solution x :

- $\overset{k}{x}$ le vecteur x de composante $\overset{k}{x}_j$
- $\overset{k}{A}$ le tableau simplicial A d'éléments $\overset{kj}{A}_i$
- $\overset{k}{E}$ le domaine des solutions réalisables.

Le programme ayant comme base de départ une base duale réalisable ($A_0^j \geq 0 \quad \forall\, j \in \bar{I}$), la solution trouvée x_I n'est pas optimale s'il existe au moins un indice $i \in I$ tel que $A_i^0 < 0$. (De part sa définition x_I peut avoir des composantes négatives). Dans ces conditions, la ligne i peut servir comme ligne de pivot de l'algorithme dual-simplicial ; cela ne suffit pas pour démarrer une itération de la méthode de Gomory si le pivot n'est pas égal à -1. Il est nécessaire, dans ce cas, de générer une ligne de pivot convenable à partir de la ligne i. Cette génération se traduit géométriquement par la troncature du domaine des solutions réalisables.

II.3.1 Méthode de troncature :

Convenons de noter le domaine initial des solutions réalisables E par $\overset{0}{E}$ (itération n°0).

En partant de $\overset{0}{E}$ la méthode des troncatures consiste à ajouter, à chaque itération, une contrainte supplémentaire à l'ensemble des contraintes initiales. Cette contrainte additionnelle réduit le domaine des solutions réalisables en variables continues sans exclure une seule solution réalisable entière.

A la k^e itération on aboutit à la suite suivante :

$$\overset{0}{E} \supset \overset{1}{E} \supset \overset{2}{E} \supset \ldots \supset \overset{k}{E} \qquad \text{(II.9)}$$

D'une manière générale si la base est optimale à l'itération k la solution de base correspondante $\overset{k}{x}_{(I)}$ constitue la solution entière cherchée. Si ce n'est

pas le cas on tronque le domaine E^k par un plan donné par une contrainte supplémentaire ; de nouveau on constitue un domaine $E^{k+1} \subset E^k$ et on teste l'intégrité de la solution x^{k+1} (I).

Dans ce qui suit, nous étudierons de près ces containtes de troncature et les justifierons.

II.3.2 Contraintes additionnelles de Gomory

y étant un nombre quelconque, notons par [y] la partie entière de y, y peut s'exprimer par

$$y = [y] + r \quad \text{où} \quad 0 \leq r < 1.$$

Etant donnés deux entiers quelconques a et b il existe toujours deux autres entiers c et r tels que :

$$a = c \times b + r \quad \text{où} \quad 0 \leq r < b.$$

Il s'en suit qu'on peut écrire a sous la forme suivante :

$$a = [a/b] \times b + r \qquad \text{(II.10)}$$

Comme $r \geq 0$ il en résulte que pour tout $b>0$ on a

$$[a/b] \leq -1 \quad \text{pour tout } a<0.$$

Exemples :

$$[-5/5] = -1 \quad \text{et} \quad [-1/5] = -1$$

$$[-2/5] = -1 \quad \text{et} \quad [7/5] = 1$$

par contre

$$[1/5] = [2/5] = [3/5] = [4/5] = 0.$$

Par extension $[A_i/\lambda]$ représentera un vecteur ligne de composantes $[A_i^j/\lambda]$, λ étant un nombre quelconque non nul.

Considérons maintenant la ligne d'indice i du tableau simplicial A, susceptible de générer la contrainte additionnelle de Gomory.

Cette ligne est choisie parmi celle du système (II.7)

$$x_i = A_i^0 + A_i^{\bar{I}}(-x_{\bar{I}}) \quad i \in I \qquad (II.11)$$

Soit λ un réel positif tel que :

$$A_i^j = [A_i^j/\lambda] \times \lambda + r_j \quad \text{où} \quad 0 \leq r_j < \lambda, \ \forall\, j \in \bar{I} \qquad (II.12)$$

$$\text{et} \quad 1 = [1/\lambda] + r \quad \text{où} \quad 0 \leq r < \lambda \qquad (II.13)$$

En tenant compte de (II.12) et (II.13) l'équation (II.11) écrite sous la forme $A_i^0 + A_i^{\bar{I}}(-x_{\bar{I}}) + 1\ (-x_i) = 0$ devient

$$r(-x_i) + \sum_{j \in \bar{I}} r_j\ (-x_j) + r_0 + \lambda \left\{ \left[\frac{A_i^0}{\lambda} \right] + [A_i^{\bar{I}}/\lambda]\ (-x_{\bar{I}}) + [1/\lambda]\ (-x_i) \right\} = 0 \qquad (II.14)$$

soit aussi

$$\sum_{j \in \bar{I}} r_j\, x_j + r x_i = r_0 + \lambda \left\{ \left[\frac{A_i^0}{\lambda} \right] + [A_i/\lambda]\ (-x_{\bar{I}}) + \left[\frac{1}{\lambda} \right] (-x_i) \right\} \qquad (II.15)$$

Désignons par $\bar{x}_i$ la quantité entre accolades de l'équation (II.15) :

$$\bar{x}_i = \left[\frac{A_i^0}{\lambda} \right] + [A_i/\lambda]\ (-x_{\bar{I}}) + [1/\lambda]\ (-x_i) \qquad (II.16)$$

Examinons le premier membre de (II.15)

- r_j et r sont tous non négatifs
- x est entier non négatif.

Il en résulte que le premier membre de (II.15) est non négatif

$$\sum_{j \in \bar{I}} r_j x_j + r x_i \geq 0 \qquad (II.17)$$

II.3.4 Propriétés de l'équation (II.16) [44] :

1- $\bar{x}_i$ a une valeur entière :

En effet les coefficients $b_j = [A_i^j/\lambda]$, $j \in \bar{I}$ et $[1/\lambda]$ sont entiers ; il en est de même pour les variables x_j, $j\in\bar{I}$ et x_i, $i\in I$. $\bar{x}_i$ étant une somme de produits de nombres entiers est entier.

2- $\bar{x}_i$ est un entier non négatif :

En effet comme x_i est entier, toute valeur $x_i<0$ c'est-à-dire $x_i=-1$ ou -2, etc... est contradictoire avec la propriété de (II.17) : $r_0<\lambda$, le second membre de (II.15) est strictement négatif si $\bar{x}_i$ est inférieur à 0.

3- Compte tenu des deux propriétés précédentes il en résulte que :

$$r_0 \leqslant \sum_{j\in\bar{I}} r_j x_j + r x_i \qquad \text{(II.18)}$$

4- L'équation (II.16) est équivalente à (II.11) pour tout $\lambda>1$.

En effet si $\lambda>1$ la valeur de r est 1 et d'après l'équation (II.15) la relation entre x_i et $\bar{x}_i$ est la suivante :

$$x_i = \lambda\bar{x}_i + r_0 + \sum_{j\in\bar{I}} r_j(-x_j) \qquad \text{(II.19)}$$

ou en d'autres termes, en divisant les 2 membres de (II.19) par λ :

$$\frac{x_i}{\lambda} = \bar{x}_i + \frac{r_0}{\lambda} + \sum_{j\in\bar{I}} r_j/\lambda\ (-x_j)$$

soit aussi

$$\frac{x_i}{\lambda} \leqslant \bar{x}_i + \frac{r_0}{\lambda} \qquad \text{(II.20)}$$

c'est-à-dire

$$\bar{x}_i \geqslant \frac{x_i}{\lambda} - \frac{r_0}{\lambda}$$

ou

$$\bar{x}_i > \frac{x_i}{\lambda} - 1$$

soit finalement

$$\bar{x}_i \geqslant \frac{x_i}{\lambda} \qquad \text{(II.21)}$$

ce qui montre l'équivalence de (II.16) et (II.11).

5- L'équation (II.17) peut être considérée comme une ligne de pivot :

En effet (II.16) est une contrainte surabondante et comme A_i^0 est négatif il en est de même de $[A_i^0/\lambda]$; cette propriété montre aussi que la génération de (II.16) est toujours possible.

La contrainte additionnelle de Gomory sera définitivement déterminée une fois qu'on a fixé la valeur de λ.

Le cas de $\lambda=1$ est trivial, on retrouve la contrainte génératrice initiale (II.11).

Seul le cas $\lambda>1$ est intéressant.

Il s'en suit que $[1/\lambda]=0$ et que l'équation (II.16) devient

$$\bar{x}_i = [A_i^0/\lambda] + [A_i^{\bar{I}}/\lambda](-x_{\bar{I}}) \qquad \text{(II.22)}$$

II.4.1 Choix et détermination de la valeur de λ :

Comme pour l'algorithme dual-simplicial (§ 1.4.3) le problème de la règle de choix des lignes pivot se pose, sous une forme d'ailleurs plus complexe. On examinera plus loin les critères des règles de choix. (chapitre III)

Supposons que par une règle de choix bien déterminée, on a choisi une ligne de pivot, soit r l'indice de cette ligne ($r \in I$).

Cette ligne vérifie le critère de candidature :

$$A_r^0 < 0$$

Supposons qu'il existe aussi au moins un indice $j \in \bar{I}$ tel que $A_r^j < 0$, sinon le problème serait impossible et le programme linéaire P_r n'aurait pas de solution.

Soit J_r l'ensemble défini par des $j \in \bar{I}$ tels que

$$J_r = \{j \in \bar{I} \mid A_r^j < 0\}$$

II.4.2 Justification de l'équation générée comme ligne de pivot :

L'équation (II.16) a les mêmes propriétés que la ligne de pivot (II.11) d'indice r.

En effet pour toute valeur de λ positive les propriétés suivantes sont vérifiées :

- $[A_r^0/\lambda] \leq -1$ du fait que $A_r^0 < 0$
- $[A_r^j/\lambda] < -1$ pour tout $j \in J_r$.
- J_r' étant l'ensemble des indices j tels que $[A_r^j/\lambda] < -1$ nous avons $J_r \equiv J_r'$.

D'autre part d'après la propriété 5 du (§ II.3.4) l'addition de la contrainte (II.16) à l'ensemble des contraintes initiales limitant le domaine des solutions réalisables n'exclut aucun point à coordonnées entières de E.

Il en résulte que (II.16) est une ligne de pivot et que le choix de $\lambda=A_r^j$, $j \in J_r$ donne un pivot $[A_r^j/A_r^j] = -1$. Ceci montre que les conditions supplémentaires du (§ II.1) sont réalisées.

Montrons aussi qu'il y a un choix de λ à faire :

Soit λ_1 et λ_2 deux valeurs différentes de λ. Les variations correspondantes de la fonction économique sont respectivement $[A_r^0/\lambda_1]\ A_0^s$ et $[A_r^0/\lambda_2]\ A_0^s$, $s \in J_r$, indice de la colonne pivot. Dans le cas de non dégénérescence ($A_0^s \neq 0$) la décroissance de la fonction économique est plus importante pour la plus petite valeur de (λ_1,λ_2), ce qui est notre but dans un programme de minimisation.

En conclusion le choix des λ doit être tel que :

- il réalise dans l'équation (II.16) un pivot égal à -1
- il assure la plus grande variation possible de la fonction économique.

II.4.3 Choix de λ :

La variable sortant de la base étant x_r, la ligne qui lui est associée étant celle du pivot :

$$x_r = A_r^0 + A_r^{\bar{I}}(-x_{\bar{I}}),\ r \in I \qquad (II.24)$$

La ligne de pivot générée est

$$\bar{x}_r = \left[\frac{A_r^0}{\lambda}\right] + [A_r^{\bar{I}}/\lambda]\ (-x_{\bar{I}}) \qquad (II.25)$$

Le choix de la variable x_j, $j \in \bar{I}$, entrant dans la base est provisoirement le suivant :

Soit $L(J_r)$ l'ensemble des vecteurs colonnes A^j tels que $j \in J_r$. Soit A^s l'inf-lexicographique des éléments de $L(J_r)$ (§ I.5.2)

$$A^s = \inf_{j \in J_r \subset \bar{I}}^{\ell} A^j \qquad (II.26)$$

A^s sera choisi comme colonne pivot; la variable qui entre dans la base sera alors x_s, $s \in \bar{I} \cap J_r$.

Remarquons que ce choix contient le cas particulier du choix habituel de la méthode duale-simpliciale.

En effet ce dernier choix définit s de la manière suivante :

$$-\frac{A_0^s}{[A_r^s/\lambda]} = -\min_{j \in J_r} A_0^j/[A_r^j/\lambda] \qquad (II.27)$$

c'est-à-dire

$$-A_0^s/[A_r^s/\lambda] \leq -A_0^j/[A_r^j/\lambda] \quad \forall\ j \in J_r$$

Comme nous devons avoir $[A_r^s/\lambda] = -1$ nous pouvons déduire le résultat suivant :

$$A_0^s \leq -\frac{A_0^j}{[A_r^j/\lambda]} \leq +A_0^j \qquad \forall\ j \in J_r$$

donc d'une façon générale :

$$A^s \overset{\ell}{\leq} -A^j/[A_r^j/\lambda] \qquad (II.28)$$
$$s, j \in J_r$$

Soit maintenant $\mu_j > 0$, le plus grand entier tel que :

$$(1/\mu_j)\ A^j \overset{\ell}{\gtrless} A^s \quad \forall\ j \in Jr \qquad (II.29)$$

D'après les relations (II.28) et (II.29) et compte tenu du choix de μ_j, la valeur de λ est telle qu'on ait

$$- [A_r^j/\lambda] \leqslant \mu_j$$

ou aussi

$$\mu_j + [A_r^j/\lambda] \geqslant 0 \qquad (II.30)$$

La plus petite valeur de λ déduite de (II.30) est alors :

$$\lambda_j = - A_r^j/\mu_j \qquad (II.31)$$

Un raisonnement identique appliqué à tous les indices $j \in J_r$ nous donne un ensemble $J(J_r) = \{\lambda_j \mid j \in J_r\}$.

La valeur définitive de λ sera

$$\lambda = \max_{\lambda_j \in J(J_r)} \lambda_j \qquad (II.32)$$

Nous pouvons vérifier que le pivot $[A_r^s/\lambda] = -1$.

En effet $\mu_s = 1$ et d'après (II.30) et (II.32)

$$\lambda \geqslant - \frac{A_r^s}{\mu^s} = - A_r^s \quad \Longrightarrow \quad [-A_r^s/\lambda] = -1.$$

II.4.4 Propriétés principales du choix lexicographique :

Le tableau simplicial A au démarrage de l'algorithme est dual réalisable ($A^j \overset{\ell}{\gtrless} 0$, $j \in \bar{I}$) ou lexicographiquement positif.

Une condition supplémentaire de l'algorithme de Gomory étant la conservation de cette propriété après chaque itération.

Compte tenu des conditions du choix de λ nous pouvons énoncer le théorème suivant.

Théorème [1] :

La propriété d'un tableau initialement lexicographiqmement positif est conservéepar le choix lexicographique de la colonne pivot à chaque itération :

Pour la démonstration de ce théorème nous procédons par un raisonnement par récurrence.

Le tableau simplicial A de départ est par hypothèse lexicographiquement positif. A l'itération k-1 l'équation :

$$\overset{k-1}{x}_{I} = \overset{k-1}{A^{0}} + \overset{k-1}{A^{\bar{I}}}\ \overset{k-1}{x}_{\bar{I}} \qquad \text{(II.33)}$$

Le passage de $\overset{k-1}{A^{\bar{I}}}$ à $\overset{k}{A^{\bar{I}}}$ se fait de la manière suivante :

Le système de contraintes est éventuellement augmenté d'une contrainte additionnelle (cas où $A_r^s \neq -1$). Soit r(k) son indice de ligne (indice de la dernière ligne du tableau simplicial). La nouvelle base est

$$I' = \overset{k}{I} = \overset{k-1}{I} + \{s\} - \{r(k)\} \qquad \text{(II.34)}$$

et

$$\bar{I}' = \overset{k}{\bar{I}} = \overset{k-1}{\bar{I}} - \{s\} + \{r(k)\} \qquad \text{(II.35)}$$

Les opérations de substitution étant :

$$\overset{k}{A}{}_i^j = \overset{k-1}{A}{}_i^j + \overset{k-1}{A}{}_i^s \cdot \left[\overset{k-1}{A}{}_r^j/\lambda\right] \quad \forall\, i \in \overset{k-1}{\bar{I}} + \{0\},\ \forall\, j \in \overset{k-1}{\bar{I}} - \{s\} \qquad \text{(II.36)}$$

$$\overset{k}{A}{}_i^s = \overset{k-1}{A}{}_i^s \qquad \forall\, i \in \overset{k-1}{\bar{I}} + \{0\} \qquad \text{(II.37)}$$

Nous passerons en revue tous les cas possibles :

1- $\overset{k-1}{A}{}_r^j \geq 0,\ \forall\, j \notin J_r \cap \overset{k-1}{\bar{I}} \quad \Longrightarrow \quad \left[\overset{k-1}{A}{}_r^j/\lambda\right] \geq 0 \quad \forall\, j \notin J_r \cap \overset{k-1}{\bar{I}}$

d'après (II.36) $\overset{k}{A}{}^j \overset{\ell}{\geq} 0 \qquad \forall\, j \notin J_r \cap \overset{k-1}{\bar{I}}$

---

1 L'établissement de ce théorème a pour origine l'énoncé d'une propriété équivalente dans [22] et [44].

2- $\forall\, j \in J_r \quad \Longrightarrow \quad \left[{}^{k-1}_{\ }A^{j}_{i} /\lambda \right] \leqslant -1$

a) Il n'existe pas de colonnes ${}^{k-1}_{\ }A^{j}$ ex aequo et aussi comme tous les ${}^{k-1}_{\ }A^{j}_{0} > 0$ et différents et comme ${}^{k-1}_{\ }A^{s} \overset{\ell}{\leqslant} {}^{k-1}_{\ }A^{j}$, $j \in J_r$ c'est-à-dire aussi ${}^{k-1}_{\ }A^{s}_{0} \leqslant {}^{k-1}_{\ }A^{j}_{0}$, $\forall\, j \in J_r$

on déduit que :

$$ {}^{k-1}_{\ }A^{j}_{0} = {}^{k-1}_{\ }A^{j}_{0} + {}^{k-1}_{\ }A^{s}_{0} \left[{}^{k-1}_{\ }A^{j}_{r} /\lambda \right] $$

soit

$$ {}^{k-1}_{\ }A^{j}_{0} = {}^{k-1}_{\ }A^{s}_{0} \left[\frac{{}^{k-1}_{\ }A^{j}_{0}}{{}^{k-1}_{\ }A^{s}_{0}} + \left[\frac{{}^{k-1}_{\ }A^{j}_{r}}{\lambda} \right] \right] \qquad j \in J_r . $$

D'après (II.28) la quantité entre crochets de l'expression précédente est non négative. Compte tenu de (II.29) on déduit que

$$ {}^{k-1}_{\ }A^{j}_{0} \geqslant {}^{k-1}_{\ }A^{s}_{0} \left(\mu_j + \left[\frac{{}^{k-1}_{\ }A^{j}_{r}}{\lambda} \right] \right) j \in J_r \qquad \text{(II.38)} $$

Comme ${}^{k-1}_{\ }A^{s}_{0}$ est non négative on a

$$ {}^{k}_{\ }A^{j}_{0} \geqslant 0 $$

et par suite

$$ {}^{k}_{\ }A^{j} \overset{\ell}{\geqslant} 0 \qquad \forall\, j \in J_r . $$

b) Il existe deux ou plusieurs colonnes ${}^{k-1}_{\ }A^{j}$, $j \in J_r$ ex aequo.

Soit H_ℓ l'ensemble des indices des colonnes ex aequo dont le plus grand ordre est ℓ. (§ I.5.2).

- Aucune des colonnes d'indice appartenant à H_ℓ n'est lexicographiquement nulle (§ I.5.4).

La colonne ${}^{k-1}_{\ }A^{s}$ est nécessairement telle que $s \in H_\ell \cap J_r$. On a pour les ℓ premières composantes de chaque colonne h de ${}^{k-1}_{\ }A$

$$ {}^{k}_{g_i}A^{h} = 0, \; g_i = 0,1,\ldots,\ell-1, \; \forall\, h \in H_\ell \cap J_r \qquad \text{(II.39)} $$

en effet :

$$A^{k\,h}_{g_i} = A^{k-1\,h}_{g_i} + A^{k-1\,s}_{g_i} \cdot \left[A^{k-1\,h}_{r} / \lambda \right] \qquad \text{(II.40)}$$

où g_i est un indice repérant le rang de la ligne $i \in {}^{k}I$ tel que le rang soit inférieur ou égal à ℓ-1.

(II.40) peut s'écrire aussi sous la forme suivante :

$$A^{k\,h}_{g_i} = A^{k-1\,s}_{g_i} \left[\frac{A^{k-1\,h}_{g_i}}{A^{k-1\,s}_{g_i}} + \left[A^{k-1\,h}_{r} / \lambda \right] \right]$$

or $\dfrac{A^{k-1\,h}_{g_i}}{A^{k-1\,s}_{g_i}} = 1$ et $\left[A^{k-1\,h}_{r} / \lambda \right] \leqslant -1.$

D'autre part d'après (II.30) $\left[A^{k-1\,h}_{r} / \lambda \right] \geqslant - \mu_j$ et comme $\mu_j = 1$

on a $\left[A^{k-1\,h}_{r} / \lambda \right] = -1$

d'où $$A^{k-1\,h}_{g_i} = 0 \qquad g_i = 0,1,2,\ldots,\ell-1.$$

A partir de $g_i = \ell$ nous nous trouvons dans l'un des cas envisagés précédemment et par suite $A^{k\,h}_{g_i} > 0 \quad \forall\, g_i \geqslant \ell$

ce qui permet d'écrire :

$$A^{k\,h} \overset{\ell}{\geqslant} 0 \qquad \forall\, h \in H_\ell \cap J_r$$

et par conséquent

$$A^{k\,j} \overset{\ell}{\geqslant} 0 \qquad \forall\, j \in J_r$$

- Il existe une ou plusieurs colonnes d'indices appartenant à $H_\ell \cap J_r$ lexicographiquement nulle et dont le plus grand degré est ℓ.

Il est évident que A^{k-1}_{s} est un vecteur lexicographiquement nul de degré ℓ.

La ℓ^e ligne du tableau simplicial $^{k-1}\bar{A}$ joue le même rôle que la ligne de la fonction économique pour la détermination des μ_j. Les cas éventuels de $A^{k-1}_{\ell j} < A^{k-1}_{\ell s}$, $j \in H_\ell \cap J_r$ donneraient des μ_j négatifs, par définition nous prendrons les λ_j correspondants nuls.

Une fois λ déterminé nous nous trouvons dans le cas précédent.

Dans le dernier cas (A^{k-1}_s lexicographiquement nul d'ordre ℓ) la variation de la fonction économique est nulle. C'est le cas de la dégénérescence duale. Il y a risque éventuel de cyclage.

II.4.5 Convergence du second algorithme de Gomory :

La solution $\overset{k}{x}(I) = (\overset{k}{x}_0, \overset{k}{x}_I, \overset{k}{x}_{\bar{I}})$ étant la solution de base à l'itération k, le vecteur $\overset{k}{x}(I)$ est un vecteur lexicographiquement décroissant.

En effet la première composante de $\overset{k}{x}$ représente la valeur de la fonction économique dont l'expression est

$$x^k_0 = A^{k_0}_0 + A^{k_{\bar{I}}}_0 \, (-\overset{k}{x}_{\bar{I}}) \qquad \text{(II.41)}$$

Le tableau simplicial étant lexicographiquement positif après chaque itération (§ II.4.4), $A^{k_j}_0 \geq 0$, $\forall\, j \in \bar{I}$, et $\overset{k}{x}_{\bar{I}} \geq 0$, il en résulte que $A^{k_{\bar{I}}}_0 \, (-\overset{k}{x}_{\bar{I}}) \leq 0$, ce résultat est valable quel que soit k d'où la suite vérifiée par les solutions de base :

$$\overset{0}{x}(I) \overset{\ell}{\geq} \overset{1}{x} \overset{\ell}{\geq} \cdots \geq \overset{k}{x}(I) \ldots \overset{\ell}{\geq} \overset{M}{x}(I) \overset{\ell}{\geq} Z \qquad \text{(II.42)}$$

$\overset{M}{x}(I)$ étant la solution optimale qui est obligatoirement bornée inférieurement par une certaine colonne Z de même dimension.

Supposons que nous ayons la suite précédente (II.52), le passage d'une solution de base $\overset{k}{x}(I)$ à $^{k+1}x(I)$ ne peut s'accompagner que d'une variation finie et entière des composantes de $\overset{k}{x}(I)$ dont $\overset{k}{x}_0$ est la première composante.

En effet l'examen des différentes composantes de $\overset{k}{x}(I)$ montre que :

- Si $\overset{k}{x}_0$ décroissait indéfiniment on aurait $\overset{k}{x}_0 < z_0$ ce qui est contradictoire avec les relations (II.52). Il existe alors une valeur k_0 de k telle que $\overset{k}{x}_0 = \overset{M}{x}_0$, $\forall\ k \geq k_0+1$ et $\overset{k}{A}{}^s_0=0$, $s \in \bar{I}$.

- Les autres composantes ou elles atteignent une valeur finale et restent constantes à partir d'une certaine valeur de k ou elles décroissent indéfiniment.

- Supposons que la deuxième éventualité ait lieu :

Ce cas ne serait possible que si une ligne candidate n'était jamais choisie comme une ligne génératrice de la contrainte additionnelle. Lorsque cette ligne est choisie, r étant son indice, on aurait $\overset{k}{A}{}^s_r < 0$ et comme $[\overset{k}{A}{}^0_r /\lambda] < -1$ on aura d'après (II.36) :

$$\overset{k+1}{A}{}^0_r = \overset{k}{A}{}^0_r + \overset{k}{A}{}^s_r \,.\, [\overset{k}{A}{}^0_r /\lambda\ ]$$

Le produit du second membre de la relation précédente est positif. Il en résulte que $\overset{k+1}{A}{}^0_r > \overset{k}{A}{}^0_r$ et après un nombre fini d'itérations $\overset{k}{A}{}^0_r$ devient non négatif et à partir d'une valeur k_1 de k, $\overset{k}{A}{}^0_r$ atteint $\overset{M}{x}_r$ et reste constante. La deuxième éventualité n'a lieu donc que si la règle de choix est telle que des lignes candidates ne soient jamais prises pour générer des contraintes additionnelles.

Il en résulte qu'avec des règles de choix bien déterminées l'algorithme converge, théoriquement, après un nombre fini d'itérations . Dans la pratique le nombre d'itérations est parfois prohibitif, ce cas correspond en général à une situation de dégénérescence de la base duale ; cette dégénerescence peut conduire parfois à un cyclage indéfini.

La règle de choix des lignes génératrices des contraintes additionnelles est souvent déterminante quant à la convergence de l'algorithme.

Chapitre III

REGLES DE CHOIX DES LIGNES GENERATRICES. HYPOTHESES ET CONCLUSIONS PRATIQUES DU SECOND ALGORITHME DE GOMORY

II.1 Généralités

Il existe un très grand nombre de règles de choix souvent heuristiques et plus ou moins fondées. Aucune règle ne possède cependant une efficacité stable dans la résolution des divers problèmes rencontrés, efficacité qui serait le seul but souhaité et recherché, dans ce domaine de programmation linéaire en nombres entiers. Nous allons passer en revue les règles les plus courantes et donner une idée de leur domaine d'application. Les critères de jugement étant le nombre d'itérations et le temps d'exécution qui sont d'ailleurs plus ou moins liés.

Rappelons que toute ligne i du tableau simplicial ayant un terme A_i^0 constant négatif et au moins un terme $A_i^j < 0$, $j \in \bar{I}$, est dite candidate, elle est susceptible de générer la contrainte additionnelle de l'itération suivante.

III.2 Règles de choix

L'inventaire suivant des règles de choix n'est nullement exhaustif. Nous nous contentons de citer les principales règles. Ces règles par ailleurs ont été programmées et testées sur un nombre assez réduit d'exemples numériques. Les conclusions tirées ne sont données qu'à titre indicatif.

III.2.1 Choisir comme ligne génératrice de contrainte additionnelle la première ou la dernière ligne candidate du tableau simplicial. Cette règle dans le cas de dégénérescence conduit souvent à un grand nombre d'itérations avant de pouvoir sortir de la situation de dégénérescence.

III.2.2 Processus cyclique consistant à choisir à l'itération n°k la ℓ^e ligne telle que

$$\ell \equiv k \text{ (module (m)) ou } m = |I|$$

Si la ℓ^e ligne n'est pas candidate il est conseillé de choisir celle qui la suit immédiatement.

Ce critère occasionne les mêmes inconvénients que pour le choix précédent (§ 2.1), cependant les risques de cyclage sont moins grands.

III.2.3 Etant donné que les contraintes définissant le problème ont été écrites dans un ordre quelconque (dû à l'effet du hasard), cet ordre se répercute sur les lignes du tableau simplicial et par suite sur l'ordre lexicographique des colonnes (§ I.5).

Le choix classique de la méthode duale n'est pas sensible à cet ordre. En effet, cette règle de choix consiste à prendre comme ligne génératrice toute ligne candidate ayant le plus petit terme constant négatif.

Soit

$$H = \bigcup_{i \in \{i \in I \mid A_i^0 < 0\}} H_i \quad \text{et telle que } H_i = \{j \in \bar{I} \mid A_i^j < 0\}$$

Cette règle est basée sur le fait que la ligne générée a une forte probabilité d'avoir un terme constant très grand également en valeur absolue. Ce qui, dans le cas de non dégénérescence seulement assure une importante variation de la fonction économique.

Cette règle est particulièrement efficace pour les problèmes de petite taille, bien que, en cas de dégénérescence, le nombre d'itérations soit parfois prohibitif.

III.2.4 Jusqu'ici nous avons toujours porté notre attention sur les lignes du tableau simplicial plutôt que sur les colonnes. Les résultats surprenants obtenus par les choix préalables de la colonne pivot, notamment dans les cas de dégénérescence, donnent plus de sens à ce critère de choix.

Deux raisons justifient le choix préalable de la colonne pivot [22].

La première est qu'à chaque itération un multiple entier de la colonne pivot est ajouté à la colonne ~~A^k des contraintes~~ A^{k_0} des constantes du tableau simplicial et plus exactement

$$ {}^{k+1}A^0 = {}^{k}A^0 + {}^{k}A^s \left[{}^{k}A^s_r / \lambda \right] \qquad \text{(III.2)} $$

soit au moins on a $\left[{}^{k}A^s_r / \lambda \right] = -1$

et $${}^{k+1}A^0 = {}^{k}A^0 - {}^{k}A^s.$$

La variation de la fonction économique est au moins $\left({}^{k}_{-}A^s_0 \right)$. Le choix d'une colonne pivot ayant le plus grand terme ${}^{k}A^j_0$, $j \in \bar{I}$ peut assurer une plus grande variation de la fonction économique.

La deuxième raison est plus importante car elle concerne le cas de dégénérescence duale. Quand il y a choix entre plusieurs vecteurs lexicographiquement nuls de degrés différents (§ I.5.4), il est très intéressant de prendre comme colonne pivot la colonne ayant le plus petit degré. Ce choix permet d'éviter tout cyclage indéfini.

Cette règle est connue sous le nom du "maximum lexicographique". Elle consiste à ranger les colonnes par ordre lexicographique croissant selon la suite $N = \{1,2,...n\}$, notons cette application par γ telle que

$$ \bar{I} \overset{\gamma}{\rightarrow} N \qquad \text{(III.3)} $$

Soit $\gamma(\gamma)$, $j \in \bar{I}$ le rang de la colonne j, $(\gamma(j) \in N)$. Chaque ligne i candidate, $i \in I$ contient un ou plusieurs éléments $A^j_i < 0$, soit en reprenant la notation du (§ II 4.1) $H_i = \{j \in \bar{I} \mid A^j_i < 0\}$, $\forall\, i \in I$, chaque ligne candidate sera caractérisée par la variable $R(i)$ dont la valeur est :

$$ R(i) = \min_{j \in H_i} \gamma(j) \qquad \text{(III.4)} $$

Complétons cette définition par : $R(i) = 0$ pour tout $H_i = \emptyset$.

La variable $R(i)$ est le rang de la colonne pivot qui pourrait être choisie si la ligne i était choisie comme ligne génératrice de la contrainte de Gomory.

En choisissant par conséquent

$$ R(r) = \max_{i \in I} \; \min_{i \in H_i} R(i) \qquad \text{(III.5)} $$

on a à la fois la ligne génératrice r et la colonne pivot de rang p = R(r).

III.2.5 Les deux dernières règles de choix semblent l'emporter sur les autres du point de vue efficacité et acheminement rapide vers la solution optimale.

Une règle dédoublée consiste à prendre en cas de non dégénérescence la règle du (§ III.2.3) et dans le cas contraire la règle du (§ III.2.5). Cette règle constitue la règle du code TOKOSI 3 (chapitre VI).

III.3 Hypothèses et conclusions du deuxième algorithme de Gomory

III.3.1 Hypothèses et conditions d'application du deuxième algorithme de Gomory.

Le deuxième algorithme de Gomory exige pour des raisons de convergence certaines conditions particulières.

a) Le tableau simplicial doit être entier au démarrage de l'algorithme d'ailleurs, toujours réalisable, car il suffit pour cette fin de multiplier tous les éléments du tableau simplicial par le P.P.C.M. des coefficients rationnels.

b) Le tableau simplicial doit être au démarrage de l'algorithme dual réalisable, tous les coefficients de la fonction économique doivent être par conséquent, pour un programme de minimisation non négatifs . Dans le cas contraire le problème est résolu par l'introduction d'une équation d'amorçage [6].

c) Le tableau simplicial initial doit être lexicographiquement positif; cette condition complète la condition précédente (b), elle est réalisable grâce à un choix judicieux de l'ordre dans lequel on écrit les différentes contraintes constituant le tableau simplicial.

III.3.2 Conclusions du deuxième algorithme de Gomory en matière de règles de choix

La condition (c) du (§ III.3.1) montre qu'il y a autant de tableaux lexicographiquement positifs que de permutations différentes des contraintes du programme linéaire (au maximum).

Le deuxième algorithme de Gomory paraît moins sensible à l'ordre des contraint initiales lorsqu'on utilise la règle du maximum lexicographique.

D'autre part, contrairement à la procédure du premier algorithme de Gomory [22] la solution basique intermédiaire de l'algorithme n°2 n'est pas réalisable (il existe au moins un x_i, $i \in I$, tel que $x_i < 0$). Il est donc impossible d'obtenir une solution réalisable approchée dans le cas où la convergence n'est pas assez rapide.

Rappelons que tous les calculs sont effectués en nombres déclarés entiers (tableau simplicial de départ entier et pivot égal à -1).

Chapitre IV

PREMIER ALGORITHME DE GOMORY ET ALGORITHME MIXTE

IV.1 Généralités

Le premier algorithme de Gomory [22] est une méthode itérative basée sur l'algorithme simplicial. Elle consiste en une préoptimisation classique en variables continues ; cette approche permet la détermination du sommet du polyèdre convexe correspondant à la valeur optimale de la fonction économique. Ce polyèdre étant déterminé par les hyperplans représentant les contraintes.

Une fois l'optimisation rationnelle terminée, ledit polyèdre, qui contient tous les points entiers,(éventuelles solutions réalisables entières), est tronqué, autant de fois que nécessaire, jusqu'à l'obtention de la solution entière ou l'affirmation de sa non existence. Cette troncature réduit le polyèdre convexe primitif en polyèdre convexe contenant toujours tous les points à coordonnées entières et facilite l'acheminement vers la solution entière optimale. La réduction des polyèdres successifs se fait par des contraintes additionnelles, contraintes de Gomory, dont les coefficients sont issus de ceux de la contrainte génératrice par une congruence modulo 1.

Dans ce qui suit nous donnons :

- une description générale du premier algorithme de Gomory
- le principe de troncature et de choix des contraintes de Gomory
- les propriétés de ces contraintes et leurs justifications
- une description de l'algorithme mixte de Gomory et le principe de construction des contraintes additionnelles.

IV.2.1 Formulation du problème :

Nous adoptons la formulation représentée par les équations (II.6) et (II.7) auxquelles nous ajoutons les contraintes $x_I, x_{\bar{I}} > 0$ seulement

$$P_1 \begin{cases} \max x_0 = A_0^0 + A_0^{\bar{I}}(-x_{\bar{I}}) & \text{(IV.1)} \\ x_i = A_i^0 + A_i^{\bar{I}}(-x_{\bar{I}}) \quad i \in I & \text{(IV.2)} \\ \begin{pmatrix} x_I \\ x_{\bar{I}} \end{pmatrix} \geq 0 & \text{(IV.3)} \end{cases}$$

et sous une forme matricielle, P_1 devient

$$P_1 \begin{cases} x_I = A^0 + A^{\bar{I}}(-x_{\bar{I}}) \quad i \in \{0\} \cup I & \text{(IV.4)} \\ \begin{pmatrix} x_{\bar{I}} \\ x_I \end{pmatrix} \geq 0 & \text{(IV.5)} \\ \max x_0 & \text{(IV.6)} \end{cases}$$

L'application de l'algorithme simplicial (§ I.3) permet l'obtention d'une solution optimale $\mathring{x}(I) = (\mathring{x}_0, \mathring{x}_I, \mathring{x}_{\bar{I}})$. C'est la phase de préoptimisation du programme P_1 en variables continues. Cette phase est suffisante si $x_i \in N$, $\forall\, i \in I$, la solution optimale $\mathring{x}(I)$ est donc entière. Si pour au moins un $i \in I$, $x_i \notin N$ nous serons ramenés à la résolution du programme suivant :

$$P_2 \begin{cases} \overset{0}{x}_I = \overset{0}{A}^0 + \overset{0}{A}_I^{\bar{I}}(-\overset{0}{x}_{\bar{I}}) \quad i \in \{0\} \cup I & \text{(IV.7)} \\ \begin{pmatrix} \overset{0}{x}_I \\ 0 \\ \overset{0}{x}_{\bar{I}} \end{pmatrix} \geq 0 & \text{(IV.8)} \\ \overset{0}{x} \in N & \text{(IV.9)} \\ \max x_0 & \text{(IV.10)} \end{cases}$$

La deuxième phase est amorcée par l'adjonction aux contraintes initiales d'une contrainte supplémentaire, contrainte générée par l'une des lignes du tableau simplicial $\overset{0}{A}$ tel que $\overset{0}{A}_i^0 \notin N$ et par l'application de la méthode duale simpliciale (§ I.4).

IV.2.2 Justification de l'utilisation de la 2ème phase :

A la fin de la 1ère phase, la base de P_2 est à la fois duale et primale réalisable, il est donc toujours possible d'introduire des contraintes supplé-

mentaires et de continuer la résolution de P_2 augmentée, sans reprendre le problème d'optimisation dès le début.

Rappelons que l'addition de nouvelles contraintes ne peut jamais se traduire par une "amélioration" de la fonction économique. Il s'ensuit que, dans le cas de maximisation, après chaque troncature la valeur initiale de $\overset{\circ}{x}_0$ est inférieure ou égale à la nouvelle valeur. Notons que toute troncature correspond au moins à une itération de l'algorithme. En effet comme l'acheminement se fait de sommet en sommet, suivant les arêtes du polyèdre, il se peut qu'une troncature crée des sommets intermédiaires n'appartenant pas au domaine des solutions réalisables ; cette situation rend la base non primale réalisable et nécessite une ou plusieurs itérations supplémentaires (§ V.3.2).

IV.3.1 Construction des contraintes de Gomory [22], [4] :

La solution $\overset{\circ}{x}(I)$ n'est pas totalement entière : soit $\overset{0}{D}$ le P.P.C.M. des coefficients $\overset{0}{A}{}^{j}_{i}$ de $\overset{0}{A}$: on a $\overset{0}{D}.\overset{0}{A}{}^{j}_{i} \equiv 0 \pmod 1$ $i \in I$, $j \in \bar{I}$ (IV.11). La relation liant $\overset{0}{x}_i$ aux $\overset{0}{x}_j$, $j \in \bar{I}$ étant :

$$\overset{0}{x}_i = \overset{0}{A}{}^{0}_{i} + \overset{0}{A}{}^{\bar{I}}_{i}(-\overset{\circ}{x}_{\bar{I}}) \qquad i \in \{0\} \cup I \qquad \text{(IV.12)}$$

Si la solution $\overset{0}{x}(I)$ n'est pas entière, tout en étant duale et primale réalisable, c'est-à-dire pour au moins un indice $i \in I$ la constance $\overset{0}{A}{}^{0}_{i}$ est non entière.

D'après le (§ II.3.2) la relation (IV.12) s'écrit aussi :

$$\overset{0}{x}_i = [\overset{0}{A}{}^{0}_{i}] + r_{io} + \sum_{j \in \bar{I}} ([\overset{0}{A}{}^{j}_{i}] + r_{ij}).(-\overset{0}{x}_j) \qquad \text{(IV.13)}$$

Sachant que les r_{ij} sont positifs ou nuls et inférieurs à 1. ($0 \leq r_{ij} \leq 1$).
La relation (IV.13) s'écrit aussi de la manière suivante :

$$\overset{0}{x}_i + \sum_{j \in \bar{I}} r_{ij}\, \overset{0}{x}_j = [\overset{0}{A}{}^{0}_{i}] + r_{io} + [\overset{0}{A}{}^{\bar{I}}_{i}]\ (-\overset{0}{x}_{\bar{I}}) \qquad \text{(IV.14)}$$

Le 1^{er} membre de la relation (IV.14) est positif ou nul :

En effet les $r_{ij} \geq 0$ et $\overset{0}{x}(I) \geq 0$

$$\overset{0}{x}_i + \sum_{j\in\bar{I}} r_{ij} \overset{0}{x}_j \geq 0 \qquad \text{(IV.15)}$$

Comme $r_{io}<1$ il en résulte que :

$$[\overset{0}{A}_i] + \begin{bmatrix} 0\bar{I} \\ A_i \end{bmatrix} \left(-\overset{0}{x}_{\bar{I}}\right) \in N \qquad \text{(IV.16)}$$

soit aussi

$$\begin{bmatrix} \bar{I} \\ A_i \end{bmatrix} \overset{0}{x}_{\bar{I}} \leq [\overset{0}{A}_i] \qquad \text{(IV.17)}.$$

Nous avons déjà étudié cette relation (chapitre II) ; c'est la contrainte génératrice des contraintes additionnelles du second algorithme de Gomory.

L'écriture de la relation (IV.13) sous la forme donnée par (IV.18) permet d'avoir la contrainte de Gomory relative au premier algorithme.

$$\sum_{j\in\bar{I}} r_{ij} \overset{0}{x}_j = \{[\overset{0}{A}_i] + \begin{bmatrix} 0\bar{I} \\ A_i \end{bmatrix} \left(-\overset{0}{x}_{\bar{I}}\right) - \overset{0}{x}_i\} + r_{io} \qquad \text{(IV.18)}$$

La quantité entre accolades est entière, somme de valeurs entières. Elle est aussi non négative:

En effet, le 1^{er} membre de (IV.18) est non négatif, d'autre part r_{io} est non négatif mais inférieur à 1. Par suite, si la quantité entre accolades était négative, c'est-à-dire égale à -1, -2, ..., le 1^{er} membre serait négatif, ce qui est contradictoire avec le fait que $\sum_{j\in\bar{I}} r_{ij} \overset{0}{x}_j \geq 0$.

Il en résulte que :

$$\sum_{j\in\bar{I}} r_{ij} \overset{0}{x}_j \geq r_{io} \qquad \text{(IV.19)}.$$

La relation (IV.19) constitue le 2^e type de contraintes de Gomory.

Soit R le tableau formé par les parties fractionnaires r_{ij} des éléments $\overset{0j}{A_i}$ du tableau simplicial A. Ce tableau après la préoptimisation en variables continues étant à la fois dual et primal réalisable, on démontre alors [22], [4] et [28] que l'ensemble de toutes les équations additionnelles associées au système (IV.12), sont des combinaisons linéaires et entières des lignes de la matrice R.

On démontre aussi que si la matrice A, au démarrage, est constituée par des coefficients rationnels ou entiers, l'ensemble des contraintes de Gomory (IV.19) est fini et forme un groupe additif d'inégalités.

D'autre part, étant donné que la matrice $\overset{k}{A}$ à l'itération k est le produit de la matrice $\overset{0}{A}$ par une certaine matrice P^{-1}, elle-même produit de k matrices du type P_ℓ [22] tel que :

$$P_\ell = \frac{1}{A_r^{s_\ell}} \begin{pmatrix} & & 1 & & \\ & & & 1 & \\ -A_r^0 & \cdots\cdots & & \ddots & -1 \\ & & & & -A_r^n \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{pmatrix} \qquad \text{(IV.20)}$$

$A_r^{s_\ell}$ étant le pivot à l'itération n° ℓ et :

$$P^{-1} = \prod_{\ell=1}^{k} P_\ell \qquad \text{(IV.21)}$$

et

$$\overset{k}{A} = AP^{-1} \qquad \text{(IV.22)}$$

D'après le théorème de A.W. Tucker [22], tout sous-déterminant de $\overset{k}{A}$ est le produit d'un sous-déterminant défini de A par ± le déterminant de P^{-1}.

Le déterminant de P^{-1} n'est pas autre chose que le produit des déterminants des matrices P_ℓ :

$$|P^{-1}| = \prod_{\ell=1}^{k} \frac{1}{A_r^{s_\ell}}$$

et

$$|P| = \prod_{\ell=1}^{k} A_r^{s_\ell} = D \qquad \text{(IV.23)}$$

Les coefficients de $\overset{0}{A}$ étant entiers (s'ils sont rationnels il est toujours possible de les réduire au même dénominateur), le nombre D est un entier et par conséquent tout sous-déterminant de $\overset{k}{A}$ est de la forme $\frac{h}{D}$, $h \in N$.

Cette conclusion reste vraie pour la matrice R, en effet D.R. est une matrice entière et à toute combinaison (modulo 1) des lignes de R correspond une combinaison modulo D des lignes de D.R. Les éléments de R sont de la forme $\frac{d}{D}$ où d est premier avec D [22] et on démontre que si D est un produit de nombres premiers à la puissance 1, l'ensemble des inégalités donnant les contraintes de Gomory est généré par une seule d'entre elles par simple multiplication (modulo 1) des coefficients de cette dernière par des entiers positifs. Le groupe additif fini ainsi obtenu est alors dit cyclique et au maximum il contient D inégalités distinctes [22] du type (IV.19).

IV.3.2 Justification et propriétés des contraintes de Gomory :

Proposition 1.

Chaque contrainte de Gomory ajoutée au tableau simplicial introduit une variable d'écart s_i^k telle que :

$$s_i^k = - r_{io} + \sum_{j \in \bar{I}} (-r_{ij}) \; (-\overset{k}{x}_j) \qquad \text{(IV.24)}$$

Son adjonction aux contraintes initiales provoque effectivement un changement dans la solution de base, c'est-à-dire que toute contrainte de Gomory réduit le domaine des solutions réalisables continues :

En effet la solution de base $\overset{k}{x}(I)$ à l'itération k ne peut pas satisfaire la contrainte (IV.24) : on a $\overset{k}{x}_{\bar{I}} = 0$ et $s_i^k \geq 0$, ce qui conduit à $r_{io} \leq 0$ au lieu de $r_{io} > 0$.

Proposition 2.

Toute solution réalisable entière vérifie la contrainte de Gomory (IV.24) :

En effet toute solution réalisable entière x'(I) est telle que

$$x'_i \equiv 0 \; (\text{mod.}1) \quad i \in I + \{0\} \qquad \text{(IV.25)}$$

D'après (IV.7) et (IV.25) on a :

$$\overset{k}{A}_i^0 + \overset{k}{A}_i^{\bar{I}} \; (-x'_{\bar{I}}) \equiv 0 \quad (\text{mod.}1) \qquad \text{(IV.26)}$$

ou aussi

$$\left[\overset{k0}{A_i} \right] + r_{io} \equiv \overset{k\bar{}}{A_i} \; (+x'_{\bar{I}})$$

soit

$$r_{io} \leq \overset{k\bar{}}{A_i} \; x'_{\bar{I}} \qquad \text{(IV.27)}$$

ou d'après (IV.18)

$$r_{io} \leq \sum_{j \in \bar{I}} r_{ij} \; x'_j \qquad \text{(IV.28)}$$

Il existe en somme, une fonction permettant le passage de (III.26) à (IV.28) et donnant à cette dernière les mêmes propriétés. ((IV.28) est vérifiée par toute solution entière $x'_{(I)}$).

(IV.28) désormais sera associée à la variable d'écart s_i^k

$$s_i^k = - r_{io} - \sum_{j \in \bar{I}} r_{ij} \; (-\overset{k}{x_j}) \qquad \text{(IV.29)}$$

V.4 Règles de choix des lignes génératrices des contraintes de Gomory :

Comme dans le deuxième algorithme de Gomory [23], il existe aussi un grand nombre de règles de choix quant à la génération des contraintes de troncature du premier algorithme de Gomory [22].

Tout comme les précédentes (§ III.2) ces règles relèvent davantage du domaine pratique.

Notons d'autre part qu'à la différence du deuxième algoritime, il est possible d'ajouter, à chaque itération, une ou plusieurs contraintes de troncature et que la ligne représentant la fonction économique dans le tableau simplicial peut, elle aussi, générer des contraintes de Gomory. Dans les paragraphes suivants nous énumérons les principales règles de choix :

IV.4.1 R étant la matrice dont les coefficients r_{ij} sont les parties fractionnaires des éléments du tableau simplicial.

Règle n°1 :

La ligne génératrice de la contrainte de Gomory est indicée par g tel que

$$r_g = \max_{i \in I+\{0\}} r_{io} \qquad \text{(IV.30)}$$

Ce critère de choix est celui du code TOKOSI 1 (chapitre VI). Cette règle a le mérite d'être facile à mettre en oeuvre. Cependant la convergence n'est pas souvent assez rapide (chapitres VII et VIII).

IV.4.2 Une autre règle de choix consiste à prendre comme ligne génératrice celle qui donne le plus grand rapport r_{io}/r_{ij} , $i \in I + \{0\}$ et $j \in \bar{I}$.

Règle n°2 :

La ligne génératrice est indicée par g tel que :

$$r_{go}/r_{gj} = \max_{\substack{i \in I \cup \{0\} \\ j \in \bar{I}}} r_{io}/r_{ij} \qquad \text{(IV.31)}$$

IV.4.3 D'autres règles dérivées de la précédente sont :

Règle n°3 :

- g est l'indice de ligne génératrice telle que :

$$r_{go} \Big/ \sum_{j \in \bar{I}} r_{gj} = \max_{i \in I \cup \{0\}} r_{io} \Big/ \sum_{j \in \bar{I}} r_{ij} \qquad \text{(IV.32)}$$

Règle n°4 :

- s est un indice donné de $\bar{I}$:

$$r_{go} / r_{gl} = \max_{i \in I \cup \{0\}} r_{io} / r_{is} \qquad \text{(IV.33)}$$

Les trois dernières règles nécessitent trop de calculs mais paraissent plus efficaces que la règle n°1.

IV.4.4 Soit p l'indice de la colonne inf-lexicographique du tableau simplicial, une règle voisine des précédentes consiste à choisir comme ligne génératrice la ligne g telle que :

Règle n°5 :

$$r_{gs} = \min_{i \in I \cup \{0\} \cap \{i \in I \cup \{0\} \mid r_{io} > 0\}} r_{is} \qquad \text{(IV.34)}$$

IV.4.5 Cette règle consiste à prendre, chaque fois que cela est possible, c'est-à-dire $r_{oo}>0$, la ligne de la fonction économique pour la génération de la contrainte de Gomory. Si $r_{oo}=0$, la contrainte de troncature est générée par la règle n°1.

Cette règle (Règle n°6) est utilisée dans le code TOKOSI. L'expérience numérique [47] et chapitres VII et VIII laissent paraître qu'elle est beaucoup plus efficace que les précédentes règles.

IV.4.6 Remarque : Etant donné que les algorithmes de Gomory sont basés sur l'algorithme euclidien [43] quant au processus de troncature, il existe pour toute règle de choix une procédure de transformation de la contrainte de Gomory en une contrainte plus "forte" permettant l'accélération de la convergence [35].

IV.5 Principe du premier algorithme de Gomory :

Deux possibilités s'offrent après la phase d'optimisation en variables continues ou chaque fois que la base est optimale (duale et primale réalisable simultanément) :

- La solution basique est entière : l'algorithme a atteint sa dernière phase de convergence.

- La solution basique optimale n'est pas entière, il existe au moins un $i \in I \cup \{0\}$ tel que $A_i^\circ \notin N$. L'une des règles de choix (§ IV.4) permet la génération d'une contrainte de Gomory du type (IV.29).

Cette contrainte ajoutée au système initial des contraintes augmente la dimension en ligne du tableau simplicial de 1 et introduit une variable d'écart $s^k_{i(k)}$. $-r_{io}$ étant négatif, la base complétée par l'indice $i_{(k)}$ de la ligne correspondant à la contrainte (IV.29), n'est plus optimale, mais duale réalisable seulement. On est alors en mesure d'appliquer l'algorithme dual-simplicial de manière à rendre la base de nouveau optimale :

1- Si dans (IV.29) on a : $r_{ij}=0$, $\forall\, j \in \bar{I}$ le programme ne possède pas de solution entière.

2- Il existe au moins un $j \in \bar{I}$ tel que $r_{ij}>0$. Dans ce cas, la variable d'écart $s^k_{i(k)}$ quitte la base et elle est remplacée par une variable x_j, $j \in \bar{I}$.

A la fin du cycle d'optimisation qui, d'ailleurs, comporte une ou plusieurs itérations, la base est optimale. De nouveau on examine l'intégrité de la solutio optimale obtenue et si nécessaire on poursuit la troncature du domaine des solutions réalisables jusqu'à l'obtention d'une solution entière, on prouvera son inexistence.

Une remarque faite par Gomory, très utile sur le plan pratique, consiste à éliminer du tableau simplicial la ligne correspondant à une variable d'écart du type $s^k_{i(k)}$ chaque fois que cette dernière rentre dans la base ; pour pouvoir faire cette opération il faut que la base soit à la fois duale et primale réalisable. On démontre [22] que cette opération n'altère pas l'ordre lexicographique des colonnes du tableau simplicial (La contrainte de Gomory constitue toujours la dernière ligne du tableau simplicial) et ne ralentit en aucun cas l'acheminement vers la solution entière.

Compte tenu du nombre des variables hors bases (n) le nombre de lignes du tableau simplicial est au plus égal à m+n+1.

IV.6 Formules de changement de base :

Le passage d'une base $\overset{k}{I}$ à une base $\overset{k+1}{I}$ obéit aux règles suivantes lorsque les calculs sont effectués en fixe :

$$\begin{cases} A^{j}_{i}{}^{k+1} = \left(A^{j}_{i}{}^{k} A^{s}_{r}{}^{k} - A^{j}_{r}{}^{k} A^{s}_{i}{}^{k} \right) / \pm |\overset{k}{D}| & i \in I \cup \{0\} \text{ et } i \neq r \\ & j \in \bar{I} \quad \text{et } j \neq s \\ A^{j}_{r}{}^{k+1} = \pm A^{j}_{r}{}^{k} & j \in \bar{I},\ j \neq s. \\ A^{s}_{i}{}^{k+1} = \pm \left(- A^{s}_{i}{}^{k} \right) & i \in I \cup \{0\} \text{ et } i \neq r \\ A^{s}_{r}{}^{k+1} = \pm |\overset{k}{D}| & \\ |\overset{k+1}{D}| = |A^{s}_{r}{}^{k}| & \end{cases}$$

$\overset{k}{D}$ étant le pivot de l'itération n° k, $\overset{\circ}{D} = 1$.

Dans ces expressions on prend le signe + ou - suivant qu'il s'agit d'un pivot A^{s}_{r} positif ou négatif. D'autre part, si les éléments $\overset{\circ}{A}{}^{j}_{i}$ sont entiers, les éléments $A^{j}_{i}{}^{k+1}$, $\overset{k+1}{D}$ le sont aussi.

Il arrive quelquefois, selon le mode de calculs, des dépassements de capacité d'une ou de plusieurs mémoires du calculateur contenant les $A_i^{j\,k+1}$. Cette situation correspond au cas où les valeurs des $A_i^{j\,k+1}$ dépassent la quantité 2^n-1, n étant le nombre de bits du mot mémoire du calculateur utilisé.

La décomposition des termes $A_i^{j\,k+1}$ peut se faire de la manière suivante [38] :

$$A_i^{j\,k} = D^k q_{ij} + r_{ij}$$

on a comme expression des éléments $A_i^{j\,k+1}$:

$$A_i^{j\,k+1} = T_1 + T_2 + T_3$$

avec

$$T_1 = \pm |D^k| \, (q_{ij} \cdot q_{rs} - q_{rj} \, q_{is})$$

$$T_2 = \pm \, (q_{ij} \, r_{rs} - q_{rj} \, r_{is} + r_{ij} \, q_{rs} - r_{rj} \, q_{is})$$

$$T_3 = (r_{ij} \, r_{rs} - r_{rj} . r_{is}) \; / \; \pm |D^k|$$

T_3 est nécessairement un entier. T_1, T_2, T_3 sont relativement inférieurs à $A_i^{j\,k} A_r^{s\,k}$ ou $A_r^{j\,k} A_i^{s\,k}$. On s'affranchit de cette manière des dépassements de capacité de certaines mémoires. Cependant ce risque n'est pas totalement écarté. Pour ce faire il est nécessaire d'utiliser la double précision ce qui modifie le mode de calculs.

Les codes TOKOSI 1 et TOKOSI 2 utilisent cette décomposition.

IV.7 Convergence du premier algorithme de Gomory [22] :

Lorsque le programme possède une solution entière la valeur de la fonction économique x_0 est bornée inférieurement. Soit M_0 cette borne inférieure.

L'introduction d'une contrainte de Gomory du type (IV.29) conduit à un cycle de réoptimisation par la méthode duale simpliciale qui nécessite au moins une itération. La troncature du polyèdre des solutions réalisables continues se poursuit tant que $A_i^\circ \notin N$, $i \in I \cup \{0\}$. Deux possibilités exclusives se présentent alors :

1- La convergence est terminée : il n'existe pas de solution entière.

2- La réoptimisation est toujours possible : on aboutit dans ce cas à une solution entière en un nombre fini d'itérations.

En effet, considérons toutes les composantes x_i du vecteur solution x. A l'itération k la solution de base est représentée par $\overset{k}{x}$. La suite $\overset{k}{x}$ est décroissante au sens lexicographique :

$$\overset{0}{x} > \overset{1}{x} > \overset{2}{x} > \ldots > \overset{k}{x} \qquad \text{(IV.35)}$$

Comme la première composante $\overset{k}{x}_0$ est bornée inférieurement par M_0, N_0 étant égal à $[A_0^0]$, il existe une certaine valeur k_0 de k telle que $\forall\ k > k_0$, $A_0^{k_0} = N_0^0$:

En effet si $A_0^{k_0}$ n'est pas entier, la ligne d'indice 0 du tableau simplicial peut être choisie pour générer la contrainte de Gomory (IV.29).

D'après les formules de changement de base (§ IV.6) on a l'itération k_0+1

$$\overset{k+1}{x}_0 = \overset{k_0}{x}_0 - \frac{A_0^{k_0 s}}{r_{0s}} \cdot r_{00}$$

or par définition $A_0^{k_0 s} \geq r_{0s}$, il en résulte que :

$$\overset{k_0+1}{x}_0 \leq \overset{k_0}{x}_0 - r_{00} = A_0^{k_0} - r_{00} = N_0^0$$

ce qui démontre la propriété énoncée.

En outre chaque fois que la base est optimale, les autres composantes de $\overset{k}{x}$ sont non négatives (donc bornes inférieures =0).

Pour tout $k>k_0$ les composantes d'indice 1 des vecteurs solutions $\overset{\ell}{x}$, $\ell=k+1,\ldots,k+n$, forment une suite monotone décroissante minorée par zéro. Le raisonnement fait pour la composante $\overset{k}{x}_0$ peut s'appliquer à $\overset{k}{x}_1$ où $N_1^0 = \left[A_1^k \right]$, à partir d'une valeur k_1 de k on a $A_1^{k_0}=N_1^0$, $\forall\ k \geq \sup(k_1,k_0)$:
Si $A_1^{k_0}$ n'est pas entier, la contrainte de Gomory générée à partir de la ligne n°1 donne :

$$A_1^{k+1} = A_1^{k_1} - \frac{A_1^{k_1 s}}{r_{1s}} \cdot r_{10}$$

Comme le tableau simplicial est toujours lexicographiquement positif, $A_1^{k_1 s} \geq 0$ car $A_0^{k_1 s} = 0$ (sinon $A_0^{k_1 s}$ devient < à $A_0^{k_0}$ pour $k_1 > k_0$) on a alors $A_1^{k_1 s} \geq r_{1s}$ et par suite

$$A_1^{k_1+1} \leq A_1^{k_1} - r_{10} = N_1^0 \qquad \forall\, k > k_1$$

L'application de ce raisonnement à toutes les autres composantes de x^k démontre que pour une valeur finie de $k \geq \sup (k_0, k_1, \ldots, k_{n+m})$ les composantes de x^k sont entières et finies.

IV.8 Algorithme mixte [24], [6], [46]

L'algorithme mixte de Gomory [24] à la différence de son premier algorithme [22] procède autrement quant à la façon de construire la contrainte de troncature. En effet étant donné qu'une partie seulement des variables sont astreintes à être entières il parait plus avantageux de procéder autrement :

Soit K l'ensemble des indices des variables astreintes à être entières, c'est-à-dire $x_i \in N$, $\forall\, i \in I \cap K$.

IV.8.1 Construction de la contrainte de troncature :

Reprenons l'équation (IV.26) à l'itération k :

$$A_i^{k_0} + A_i^{k_{\bar{I}}} (-x_{\bar{I}}^k) \geq 0 \quad (\text{mod.}1)$$

On peut écrire sous la forme suivante $A_i^{k_0} \geq r_{i0}$ (modulo 1)

$$A_i^{k_{\bar{I}}} x_{\bar{I}}^k \geq r_{i0} \quad (\text{modulo } 1) \qquad (IV.36)$$

Partitionnons $\bar{I}$ en deux ensembles disjoints $\bar{I}^+$ et $\bar{I}^-$ tels que

$$\bar{I}_i^+ = \{j \in \bar{I} \mid A_i^{k_j} \geq 0\} \qquad i \in I \qquad (IV.37)$$

et

$$\bar{I}_i^- = \{j \in \bar{I} \mid A_i^{k_j} < 0\} \qquad (IV.38)$$

Remarquons que $H_i = \bar{I}_i$.

En tenant compte de ces notations, (IV.35) devient :

$$A_i^{k_{\bar{I}_i^+}} \; x^{k}_{\bar{I}_i^+} + A_i^{k_{\bar{I}_i^-}} \; x^{k}_{\bar{I}_i^-} = r_{i0} \qquad \text{(IV.39)}$$

Deux cas sont à considérer :

1- Le premier membre de (IV.39) est non négatif, il est de la forme $r_{i0}+\ell$, $\ell \in N$, alors on a :

$$r_{i0} \leq A_i^{k_{\bar{I}_i^+}} \; x^{k}_{\bar{I}_i^+} + A_i^{k_{\bar{I}_i^-}} \; x^{k}_{\bar{I}_i^-} \leq A_i^{k_{\bar{I}_i}} \; x^{k}_{\bar{I}_i} \qquad \text{(IV.40)}$$

2- Le premier membre de (IV.39) est négatif, c'est-à-dire de la forme $r_{i0}-\ell$, $\ell \in N$, alors on a :

$$r_{i0}-1 \geq A_i^{k_{\bar{I}_i^+}} \; x^{k}_{\bar{I}_i^+} \quad A_i^{k_{\bar{I}_i^-}} \; x^{k}_{\bar{I}_i^-} \geq A_i^{k_{\bar{I}_i^-}} \; x^{k}_{\bar{I}_i^-} \qquad \text{(IV.41)}$$

La multiplication des membres extrêmes de (IV.41), par $-r_{i0}/1-r_{i0}$ permet d'écrire :

$$r_{i0} \leq \frac{r_{i0}}{1-r_{i0}} \left(- A_i^{k_{\bar{I}_i^-}} \right) x^{k}_{\bar{I}_i^-} \qquad \text{(IV.42)}$$

Il en résulte qu'on peut grouper (IV.40) et (IV.41) dans la relation suivante :

$$r_{i0} \leq A_i^{k_{\bar{I}_i^+}} \; x^{k}_{\bar{I}_i^+} + \frac{r_{i0}}{1-r_{i0}} \left(- A_i^{k_{\bar{I}_i^-}} \right) x^{k}_{\bar{I}_i^-} \qquad \text{(IV.43)}$$

L'équation de troncature de Gomory est alors :

$$s = - r_{i0} - A_i^{k_{\bar{I}_i^+}} \left(- x^{k}_{\bar{I}_i^+} \right) - \frac{r_{i0}}{1-r_{i0}} \left(- A_i^{k_{\bar{I}_i^-}} \right) \left(- x^{k}_{\bar{I}_i} \right) \qquad \text{(IV.44)}$$

La génération de la contrainte (IV.44) est basée sur le fait que $x_i \notin N$, $i \in I \cap K$.

Supposons maintenant qu'une variable x_ℓ, $\ell \in \bar{I}$ soit astreinte à être entière

dans (IV.39). Toute variation entière de $\overset{k\ell}{A}_i$ augmenterait le 1er membre de (IV.39) d'une quantité entière, et par conséquent la relation d'équivalence est toujours vérifiée si on remplace $\overset{k\ell}{A}_i$ par une valeur équivalente (modulo 1).

- $\overset{k\ell}{A}_i \geq 0$. Dans (IV.44) Gomory prend comme valeur équivalente (modulo 1) la partie fractionnaire $r_{i\ell}$ de $\overset{k\ell}{A}_i$.

- $\overset{k\ell}{A}_i < 0$. La valeur équivalente est $1-r_{i\ell}$, telle que $\left|\overset{k\ell}{A}_i\right| \equiv r_{i\ell} \pmod 1$.

La seule différence avec la contrainte de Gomory relative au premier algorithme est alors qu'il faut remplacer dans (IV.29) $\overset{k\ell}{A}_i$ par $r_{i\ell}$ si $\ell \in \bar{I}_i^+$ et par $\frac{r_{i0}}{1-r_{i0}}(1-r_{i\ell})$ si $i \in \bar{I}_i^-$ pour obtenir (IV.44).

La contrainte additionnelle de Gomory relative à l'algorithme mixte est alors de la forme :

$$s = - r_{i0} - \sum_{j \in \bar{I}} r^*_{ij}(-x_j) \qquad \text{(IV.45)}$$

telle que :

$$r^*_{ij} = \begin{cases} \textcircled{1} & \overset{k_j}{A}_i \quad \text{si} \quad j \in \bar{I}_i^+ \text{ et } x_j \text{ non astreinte à être entière} \\ \textcircled{2} & \frac{r_{i0}}{1-r_{i0}}\left(-\overset{k_j}{A}_i\right) \text{ si } j \in \bar{I}_i^- \text{ et } x_j \text{ non astreinte à être entière} \\ \textcircled{3} & r_{ij} \qquad\qquad \text{si } r_{ij} \leq r_{i0} \text{ et } x_j \text{ astreinte à être entière} \\ \textcircled{4} & \frac{r_{i0}}{1-r_{i0}}(r_{ij}-1) \quad \text{si } r_{ij} > r_{i0} \text{ et } x_j \quad " \quad " \quad " \end{cases}$$

La considération de ③ et ④ est due au fait que la fonction $\frac{r_{i0}}{1-r_{i0}}$ est monotone croissante lorsque r_{i0} varie de 0 à 1 et au souci d'accélérer la convergence de l'algorithme. Cette accélération est obtenue par la construction d'une contrainte de troncature assurant la plus profonde troncature du polyèdre des solutions réalisables.

IV.9 Autres algorithmes à base de troncature

D'autres algorithmes à base de troncature du polyèdre des solutions réalisables ont été développées en même temps que ceux de Gomory par Dantzig, Charnes, Cooper et Ben Israel,... Nous ne ferons dans ce paragraphe qu'une comparaison sommaire avec les algorithmes de Gomory.

La troncature introduite par Dantzig [14], [15] n'assure pas nécessairement la convergence de la méthode. La troncature construite par Charnes et Cooper [13] est plus fine que celle de Dantzig et on démontre que le polyèdre tronqué par Dantzig contient celui tronqué par Charnes et Cooper. Ben Israel et Charnes [7], quant à eux, ont essayé d'abondonner la règle lexicographique dans la détermination de la colonne pivot. En supposant l'unicité de la solution, ils sont arrivés à trouver une règle très proche de la règle simpliciale (§ I.3.3) quant à la détermination de la colonne du pivot. Cette nouvelle règle nécessite entre autre la résolution d'un programme linéaire auxiliaire.

Chapitre V

INTERPRETATION GEOMETRIQUE DE LA TRONCATURE

.1 Généralités

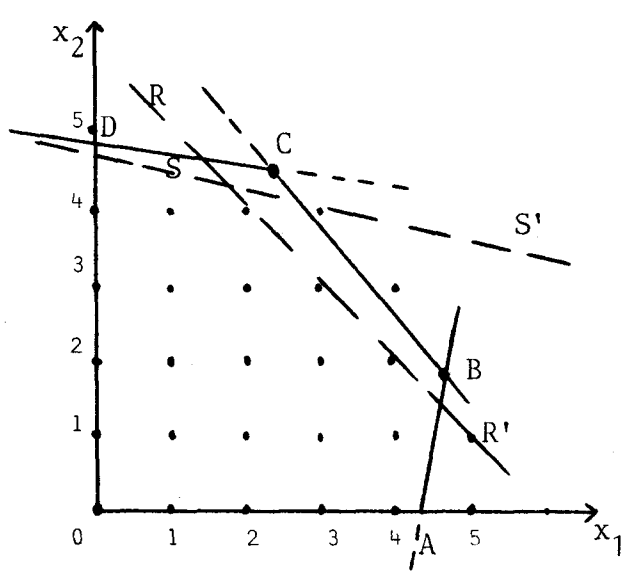
Dans ce qui suit nous allons dégager les principales propriétés des contraintes additionnelles qui ont permis aux algorithmes de Gomory [22] et [23] d'atteindre la solution entière, lorsque celle-ci existe, en un nombre fini de pas. Nous montrerons également les caractéristiques fondamentales des règles des choix les plus courantes (§ III.2) et (§ IV.4). Pour atteindre ces objectifs nous procèderons par une représentation géométrique du problème de programmation linéaire en variables entières, dans l'espace R^2.

L'interprétation de cette représentation géométrique permettra une compréhension intuitive de la méthode de troncature du domaine initial des solutions réalisables.

2 Interprétation géométrique

Soit OABCD le domaine convexe E (§ I.2.2) des solutions réalisables en variables continues d'un programme linéaire donné. Représentons par un point tous les points à coordonnées entières compris dans OABCD. Ces points forment une grille de nombres entiers (figure V.1).

La solution optimale en variables continues est constituée par les 2 coordonnées d'un des sommets A,B,C,D du domaine E. Aucun des sommets du domaine E_0, hormis 0, n'appartient à la grille des nombres entiers, la solution optimale précédente est donc rationnelle.



(figure V.1)

Supposons que la grille des nombres entiers puisse être isolée dans un domaine convexe $\bar{E}^k \subset E^0$. $\bar{E}^k$ est alors la plus petite région contenant tous les points entiers du domaine OABCD.

Le domaine $\bar{E}^k$ se déduit en fait du domaine E^0 par troncature par des droites du type RR' ou SS' qui passent par au moins un point entier de la frontière ou du plan.

Le domaine $\bar{E}^k$ est associé à un programme linéaire tel que :

- toute solution entière de ce programme en est une pour le programme associé à E^0

- toute solution de base est un sommet de $\bar{E}^k$, donc entière.

Nous pouvons alors conclure que toute solution optimale du programme linéaire associé à $\bar{E}^k$ est une solution optimale entière du programme linéaire original associé à E^0.

Cette idée fut à la base de la méthode de troncature. Le passage de E^0 à $\bar{E}^k$ est difficilement réalisable dans la pratique. Une méthode d'approche a consisté en une suite d'étapes réduisant progressivement le domaine E^0 par l'addition de contraintes supplémentaires au programme initial.

Ces contraintes vérifient les conditions suivantes [26]

- chacune d'elles réduit effectivement le domaine des solutions continues

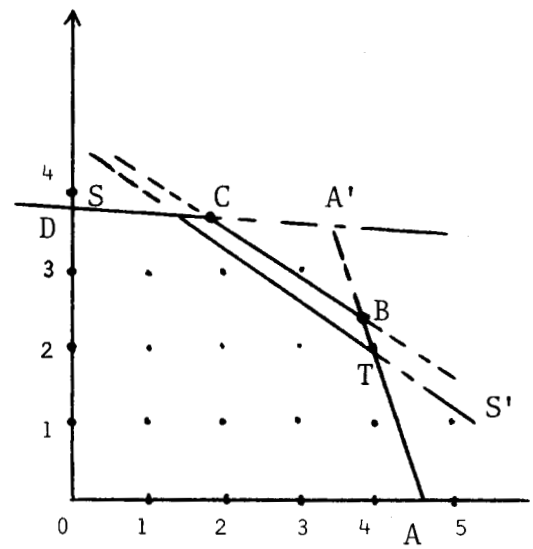
- chacune d'elles passe par au moins un point à coordonnées entières; ce point peut être un point n'appartenant pas à E^0

- elles engendrent en un nombre fini d'étapes un nouveau domaine convexe contenant la grille des points entiers et dont le programme linéaire associé a la même solution entière que le programme initial.

Les règles de choix du type (IV.30), (IV.31), (IV.34) se traduisent par le souci de réduire le plus possible le domaine des solutions rationnelles, et d'accélérer la convergence de l'algorithme [43].

Si le sommet C représente par ses coordonnées l'optimum du programme linéaire, les contraintes représentées par CD et BC sont appelées contraintes actives [35], les autres telles que AB sont dites non actives (figure V.2).

D'après un théorème relatif aux conditions d'optimalité de KUHN et TUCKER [5] : A toute contrainte active est associée une variable duale nulle. Ainsi si T représente par ses coordonnées la solution entière optimale, obtenue par troncature du domaine initial par la contrainte représentée par SS' (fig. V.2), la contrainte représentée par AB devient active et la variable duale associée n'est plus nulle, elle est positive.



(figure V.2)

Il en résulte qu'une des conséquences de la troncature est telle que certaines variables duales non nulles à l'optimum rationnel deviennent nulles à l'optimum entier et réciproquement.

V.3.1 Exemples numériques

Pour illustrer la représentation géométrique et le premier algorithme de Gomory [22] nous allons traiter manuellement le programme suivant [22] :

$$\max \quad x_0 = 3x_1 - x_2 \qquad (V.2)$$

$$\text{tel que} \begin{cases} 3x_1 - 2x_2 \leq 3 & (V.3) \\ -5x_1 - 4x_2 \leq -10 & (V.4) \\ 2x_1 + x_2 \leq 5 & (V.5) \\ x_1 , x_2 \geq 0 \text{ et entiers} & (V.6) \end{cases}$$

Soit x_3, x_4, x_5, les variables d'écart associées aux contraintes (V.3), (V.4) et (V.5).

En adoptant les notations (II.14), (II.15) nous pouvons construire le

tableau simplicial complété

	1	$-x_1$	$-x_2$
x_0	0	-3	1
x_3	3	3	-2
x_4	-10	-5	-4
x_5	5	2	1

Tableau V.1

Nous ne détaillerons pas les calculs qui nous ont conduits à la solution optimale en variables non entières, le tableau simplicial à cette étape est le suivant :

	1	$-x_3$	$-x_5$
x_0	$\frac{30}{7}$	$\frac{5}{7}$	$\frac{3}{7}$
x_1	$\frac{13}{7}$	$\frac{1}{7}$	$\frac{2}{7}$
x_4	$\frac{31}{7}$	$-\frac{3}{7}$	$\frac{20}{7}$
x_2	$\frac{9}{7}$	$-\frac{2}{7}$	$\frac{3}{7}$

$$R = \begin{pmatrix} \frac{2}{7} & \frac{5}{7} & \frac{3}{7} \\ \frac{6}{7} & \frac{1}{7} & \frac{2}{7} \\ \frac{3}{7} & \frac{4}{7} & \frac{1}{7} \\ \frac{2}{7} & \frac{5}{7} & \frac{3}{7} \end{pmatrix}$$

Tableau (V.2)

La solution optimale rationnelle est $x_0 = \frac{30}{7}$, $x_1 = \frac{13}{7}$ et $x_2 = \frac{9}{7}$

$$I = \{1,4,2\} \quad \bar{I} = \{3,5\}.$$

En choisissant comme règle de choix de ligne génératrice de contrainte de Gomory la règle (IV.3.2), nous obtiendrons en conséquence la ligne indicée par $1 \in I$; en effet R étant le tableau des parties fractionnaires positives la ligne 1 est telle que $r_{10}/r_{ij} = \max_{i \in I} r_{i0}/r_{rj}$. La contrainte de Gomory est

alors :

$$s_1 = -\frac{6}{7} + (-\frac{1}{7})\ (-x_3) + (-\frac{2}{7})\ (-x_5).$$

L'addition de cette contrainte au tableau simplicial précédent et l'application de l'algorithme dual (§ I.4) donne le tableau simplicial suivant : Tableau(V.3)

De nouveau, l'application de l'algorithme dual au tableau (V.3) donne le tableau (V.4)

	1	$-x_3$	$-s_1$
x_0	3	$\frac{1}{2}$	$\frac{3}{2}$
x_1	1	0	1
x_4	-5	-2	11
x_2	0	$-\frac{1}{2}$	$\frac{3}{2}$
x_5	3	$\frac{1}{2}$	$-\frac{7}{2}$

Tableau (V.3)

	1	$-x_4$	$-s_1$
x_0	$\frac{7}{4}$	$\frac{1}{4}$	$\frac{17}{4}$
x_1	1	0	1
x_3	$\frac{10}{4}$	$-\frac{2}{4}$	$-\frac{22}{4}$
x_2	$\frac{5}{4}$	$-\frac{1}{4}$	$-\frac{5}{4}$
x_5	$\frac{7}{4}$	$\frac{1}{4}$	$-\frac{3}{4}$

Tableau (V.4)

La contrainte de Gomory est cette fois générée par la dernière ligne du tableau précédent (V.4)

$$s_2 = -\frac{3}{4} + (-\frac{1}{4})\ (-x_4) + (-\frac{1}{4})\ (-s_1)$$

De nouveau l'application de l'algorithme dual simplicial au tableau (V.4) donne le tableau suivant :

	1	$-s_2$	-11
x_0	1	1	4
x_1	1	0	1
x_3	4	2	-5
x_2	2	1	-1
x_5	1	-1	-1
x_4	3	-4	1

Tableau (V.5)

La solution optimale entière est alors

$$x_0 = 1 \ , \ x_1 = 1 \quad \text{et} \quad x_2 = 2$$

V.3.2 Représentation géométrique :

Le domaine des solutions réalisables en continu est représenté par la région E_0 = ABCD. La solution optimale continue étant donnée par les coordonnées de B(x_1 = 13/7 , x_2 = 9/7), valeur de la fonction économique 30/7, fig. V.3.

AB et BC représentent deux contraintes actives.

La grille des points entiers est formée par les points E, F, G, H, C.

La première contrainte de Gomory de variables d'écart s_1 réduit le domaine E_0 et passe par les points entiers E et F. JF représente à son tour une contrainte active avec AB.

La solution basique correspondante est donnée par les coordonnées de I (x_1 = 1, x_2 = 0) (fig. V.4). Le point I n'appartient pas au domaine de solution réalisables car la base n'est pas primale réalisable (x_4 = -5) (tableau V.3). La phase de réoptimisation nécessite un pivotage supplémentaire (acheminement de I vers J).

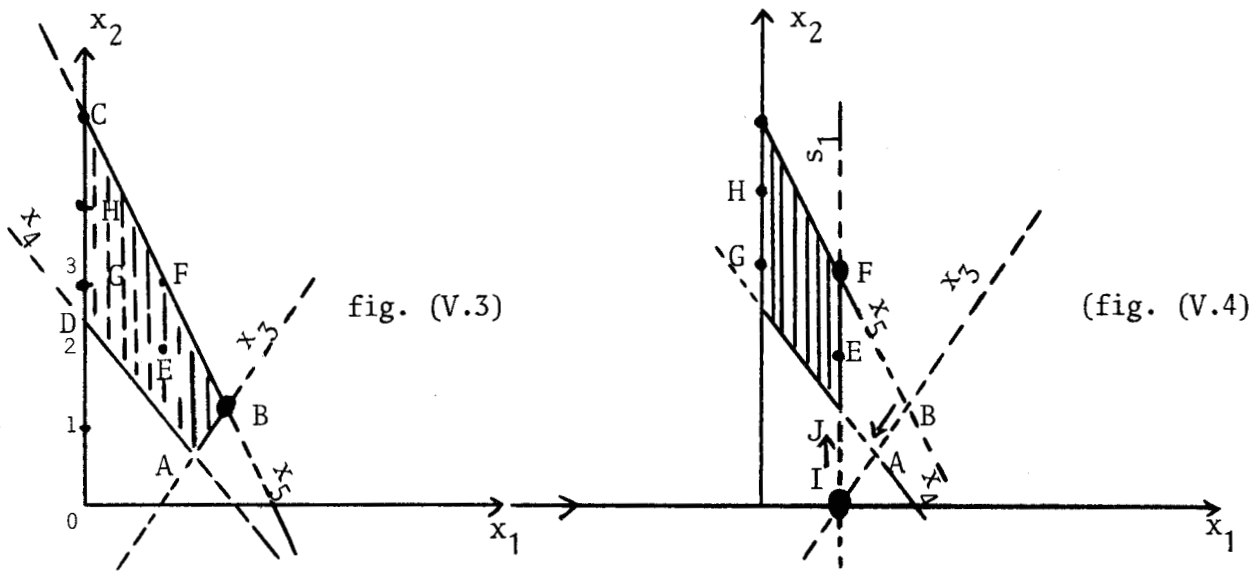


fig. (V.3)

(fig. (V.4)

La seconde contrainte de Gomory associée à la variable d'écart s_2 passe par E et G (points entiers de la grille).

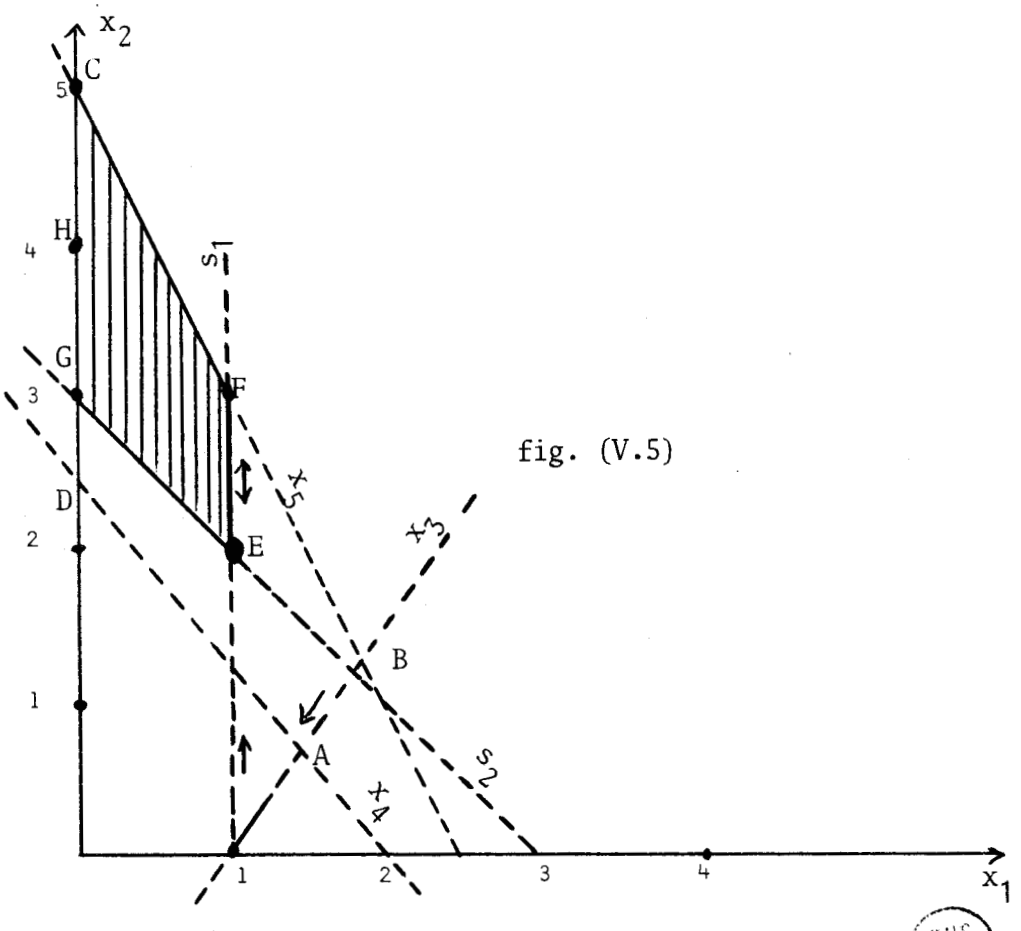


fig. (V.5)

La solution basique donnée par E (x_1=1, x_2=2) appartient au domaine des solutions réalisables. La base est simultanément primale et duale réalisable : un seul pivotage est suffisant (acheminement de J vers E), (fig. V.5). E représente la solution optimale entière du programme initial.

Chapitre VI

PRINCIPAUX CODES DE RESOLUTION DES PROGRAMMES LINEAIRES EN VARIABLES ENTIERES.

DESCRIPTION DES CODES

I.1 Généralités

VI.1.1 Les méthodes de Gomory :

La parution des différents algorithmes de Gomory [22], [23] a été accueillie avec espoir dans les milieux intéressés par la programmation linéaire en nombres entiers. Cet espoir, malheureusement, ne fut pas confirmé car ces algorithmes se révèlèrent non efficaces pour un bon nombre de problèmes, parfois même, de petite taille.

Bien que Gomory ait montré l'existence de règles de choix (§ III.4) permettant l'obtention de la solution entière en un nombre fini d'itérations (IV.6), la plus grande difficulté réside encore dans la façon de trouver la règle de choix qui garantit la convergence rapide de l'algorithme utilisé pour un type de problèmes donnés.

L'application du 1er algorithme de Gomory [22] s'est révélé inefficace à cause des difficultés rencontrées sur le plan pratique. Ces difficultés sont telles que lorsque les calculs sont menés en virgule flottante on peut, d'une part accepter comme entier un rationnel et rejeter comme rationnel un entier, et d'autre part se trouver devant un dépassement de capacité de certaines mémoires quand le pivot est très petit. Le danger est aussi grave lorsque les calculs sont menés en fixe, il arrive quelquefois de perdre la solution de base courante par dépassement de capacité de certaines mémoires du calculateur, même lorsqu'il s'agit de problèmes de petite taille. La décomposition du (§ IV.c) n'a même pas suffi et il a été parfois nécessaire d'effectuer une partie des calculs en virgule flottante et en double précision pour aboutir à la solution entière optimale.

Pour palier à ces difficultés Gomory a été amené à développer son deuxième algorithme [23] où les problèmes d'arrondi et de dépassement de

capacité de quelques mémoires n'ont plus raison d'être. Les calculs sont toujours menés en fixe: d'une part le pivot à chaque itération est égal à -1 (§ II.4.1), d'autre part le tableau simplicial de départ est entier (condition supplémentaire de l'application de l'algorithme n°2 de Gomory (chapitre II)).

De nouveau, l'espoir revenait mais les difficultés majeures subsistaient malgré les facilités du deuxième algorithme quant aux procédés de calcul. Le temps mis par les codes basés sur cet algorithme dépasse toute prévision même pour certains problèmes de petite taille. Le nombre d'itérations est parfois prohibitif et la convergence est très lente. Ceci a été à la base du développement des critères heuristiques dont le but principal est l'accélération de la convergence des codes utilisés.

Pour faire face à cette diversité dans les règles de choix basés sur des critères heuristiques plusieurs codes linéaires utilisant les algorithmes de Gomory ont été programmés et testés par des centres universitaires et des constructeurs de calculateurs.

Afin de mieux situer les résultats numériques (chapitre VIII) des codes utilisant les algorithmes de Gomory, nous mentionnerons d'autres codes basés sur des esprits différents. Ces codes utilisant des méthodes du type Branch and Bound [39] sont à caractères exploratoires.

VI.1.2 Les méthodes du type Branch and Bound :

Ces méthodes n'utilisant pas, en principe, la technique simpliciale dans la recherche des solutions entières, sont très nombreuses et variées. Bien que ces méthodes traitent théoriquement les variables bornées, dans la pratique elles sont destinées aux variables bivalentes (méthode S.E.P. - Séparation et évaluation progressives [9]).

Rappelons aussi que tout problème à variables entières bornées (bornes > 1) peut être ramené par la formule suivante

$$x_j = \sum_{j=1}^{q} 2^{p-1} x_{jp}$$

à un problème à variables bivalentes. q est tel que

$$2^{q-1} \leq k_j < 2^q$$

k_j étant la borne supérieure de la variable x_j. Les variables x_{jp} sont bivalentes (valeurs 0 ou 1).

En outre quel que soit l'ensemble R^n il existe un nombre fini de vecteurs $X \subset R^n$ tels que leurs composantes soient bivalentes. De tels vecteurs booléens sont au nombre de 2^n. La représentation graphique, dans le plan, de l'ensemble de ces vecteurs booléens X constitue une arborescence de solutions. Chaque sommet du graphe des solutions est représenté par un vecteur booléen X_h tel que h de ses composantes soient identiques à 1 et les n-h autres nulles. A chaque valeur de h (0 à n) correspond un niveau de l'arborescence ; (en tout il y a n+1 niveaux distincts).

Les procédures connues sous le nom de Branch and Bound (bifurcation et borné) ou S.E.P. consiste en la construction d'une sous-arborescence du graphe des solutions H (S,U,Π), S est l'ensemble de sommets du graphe,U l'ensemble des arcs joignant ces sommets et Π une application multivoque de S dans S [37]. Chaque étape de la construction a pour but d'introduire l'ensemble S(s) des suivants d'un sommet s, sélectionné parmi un ensemble V de sommets et de calculer l'évaluation z de la fonction économique dans chacun des sommets introduits de cette façon. On appelle ensemble des suivants d'un sommet s l'ensemble S(s) Π(s). Ces méthodes ont été à la base de certains algorithmes tels que ceux de BALAS, HERVE, GEOFFRIDN, etc... décrits sommairement dans les paragraphes suivants.

I.2 Codes basés sur les algorithmes de Gomory :

VI.2.1 Code I.P.M. 3 [47] :

Code écrit par R.E. Levitan et R.E. Gomory [47] tous d'eux d'I.B.M. en 1961. Il est basé sur le 1er algorithme de Gomory et utilise aussi le 2° algorithme de Gomory.

- langage FORTRAN II et FAP du 7030-94-IBM
- calculs effectués en virgule flottante
- algorithme de préoptimisation rationnelle : algorithme dual-simplicial
- critère de choix de lignes génératrices de contraintes de Gomory (§ IV.4.1)
- possibilité d'ajouter plusieurs contraintes de troncature (§ IV.4) en même temps
- taille maximum des problèmes traités (200-n)×n, n≤100

VI.2.2 Code LIP 1 [30]

Ce code a été écrit par J. Haldi de Stanford University et L. Isaacson de Standard Oil Company de Californie et est basé sur le premier algorithme de Gomory :

- langage FORTRAN II et FAP 7090-34 I.B.M.
- calculs effectués en virgule flottante
- algorithme dual-simplicial pour la préoptimisation en continu
- critère de choix de lignes génératrices de contraintes de Gomory : (§ IV.4.2)
- taille maximum : 60 x 240.

VI.2.3 Code ILP 2 [47]

Code programmé par D. Summers de C.D.C et basé sur le deuxième algorithme de Gomory :

- langage Compass C.D.C. 3600
- calculs effectués en fixe
- deux critères de choix :
  . choix lexicographique de la colonne pivot
  . choix de la ligne génératrice de la contrainte additionnelle (§ III.2.3)
- taille variable, le plus grand problème résolu fait 80 x 2700.

VI.2.4 Code IPSC [47]

Ce code a été programmé par R.E. Woolsey de Sandria Laboratories et est basé sur le deuxième algorithme de Gomory.

- langage FORTRAN II
- mêmes caractéristiques qu'ILP 2 mais adapté aux problèmes de taille moyenne.

VI.2.5 Les codes TOKOSI

Les codes précédents ont été programmés sur des calculateurs à configurations différentes. Dans un premier temps nous avons commencé par programmer les algo-

rithmes de Gomory selon des codes utilisant les critères de choix des codes énumérés précédemment. Les codes TOKOSI sont tous écrits en FORTRAN IV sur le C.D.C. 6600 (Scopes2 et 3) de la Direction des Etudes et Recherches d'E.D.F. à Clamart. Les détails techniques de ces codes sont en annexes.

VI.2.5.1 Code_TOKOSI_1 :

Code basé sur le premier algorithme de Gomory avec comme règle de choix de ligne génératrice de contrainte de troncature celle qui est donnée par le (§ IV.4.1) critère (IV.30)

- optimisation rationnelle : algorithme simplicial.

VI.2.5.2 Code_TOKOSI_2 :

Code basé sur le premier algorithme de Gomory

- règle de choix : critère donné par le (§ IV.4.5)
- optimisation rationnelle : algorithme dual simplicial

Les calculs dans les 2 codes précédents sont effectués en fixe. Si le problème d'arrondi ne s'est pas posé lors des traitements numériques, les problèmes de dépassement de capacité de certaines mémoires se sont manifestés à plusieurs reprises, parfois dans des exemples de petite taille. Nous avons détourné ces difficultés par l'utilisation de la double précision ou par la décomposition de chaque terme en un quotient et un reste (IV.6).

VI.2.5.3 Code_TOKOSI_3 :

Ce code est basé sur le 2° algorithme de Gomory

- critères de choix des lignes génératrices des contraintes additionnelles :
  - . critère du "maximum lexicographique" (§ III.2.3)
  - . critère classique de la méthode duale-simpliciale (§ III.2.4), en cas de non dégénérescence
- calculs effectués en fixe.

VI.2.5.4 Code_TOKOSI_4 :

Ce code est basé sur l'algorithme n°3 de Gomory

- critère de choix donné par le (§ IV.8)
- préoptimisation rationnelle : méthode duale simpliciale
- calculs effectués en virgule flottante et en fixe.

Signalons que ce dernier code n'a pas été suffisamment testé par manque d'exemples numériques mixtes (variables continues et variables entières). Les résultats numériques se rapportant à ce code sont très insignifiants.

VI.3 Codes basés sur les méthodes exploratoires :

Les codes basés sur ces méthodes sont du type combinatoire ou arborescent ou booléen. Les codes relatifs aux méthodes bivalentes d'HERVE, BALAS et GEOFFRION feront prochainement l'objet d'une publication [37]. Ils ont été programmés dans le même esprit que les codes TOKOSI sur le C.D.C. 6600 de l'E.D.F. Nous décrivons sommairement dans les paragraphes suivants les méthodes qui sont à la base de ces codes.

VI.3.1 Méthode de HERVE [32]

Cette méthode consiste en l'application de la méthode S.E.P. après une optimisation en variables continues par la méthode simpliciale.

VI.3.2 Méthode de BALAS [1], [2], [3]

Une première méthode a consisté en l'exploration systématique de l'ensemble des solutions représentées par les sommets du graphe des solutions et à trouver la solution optimale. Une expérience numérique de l'algorithme correspondant à cette méthode est donnée dans la référence [11].

L'autre version de la méthode de BALAS procède par une génération préalable et systématique des solutions réalisables bivalentes par une technique d'énumération. Afin de réduire le nombre de solutions réalisables à explorer dans les phases restantes de l'algorithme, BALAS utilise un "filtre" consistant à définir un nouveau programme équivalent. Ensuite, contrairement à l'algorithme d'HERVE [32], l'ensemble des solutions réalisables associées à un sommet de S est séparé en multiples sous-ensembles au lieu de deux, ce qui implique par conséquent la non-équivalence des ensembles S(s) et $\pi(s)$.

VI.3.3 Méthode de GEOFFRION [21]

Il s'agit d'une procédure d'énumération implicite qui teste les 2^n solutions du programme. Ce test n'est effectué qu'une seule fois pour chaque solution. Le développement de l'algorithme est le même que celui de la procédure S.E.P.

La méthode préconisée pour chercher toutes les solutions possibles, en faisant le moins de tests possibles, consiste en l'addition de contraintes surabondantes au système des contraintes initiales. Ces contraintes sont des combinaisons linéaires à coefficients non négatifs des contraintes initiales. Elles sont appelées contraintes "déléguées". Leur utilisation permet la réduction du nombre de solutions à étudier.

VI.3.4 Méthode de R. FAURE et Y. MALGRANGE [18]

Cette méthode n'est donnée qu'à titre comparatif pour ce qui concerne l'expérience numérique. L'algorithme, programmé et testé dans la référence [34] opère également par troncature du domaine des solutions réalisables après l'optimisation en continu par la méthode simpliciale. L'équation de l'hyperplan de troncature est définie par l'équation de la fonction économique et est vérifiée par les coordonnées d'un point entier fixé à l'avance. Ceci revient à ajouter au système des contraintes initiales une contrainte du type $fx \geq f\overset{o}{x}+1$ où $\overset{o}{x}$ est une solution réalisable entière connue. Une nouvelle solution entière réalisable est déterminée dans le polyèdre qui est de nouveau tronqué s'il contient au moins un point à coordonnées entières et si la solution courante n'est pas optimale. La détermination des solutions réalisables entières se fait par une procédure de recherche arborescente.

Chapitre VII

EXPERIENCES NUMERIQUES COMPARATIVES RELATIVES A DES CODES ET DES CALCULATEURS

II.1 Généralités

Toutes les méthodes de résolution de programmes linéaires en nombres entiers convergent théoriquement vers une solution entière et en un nombre fini d'itérations. Il n'en est pas ainsi en pratique. En effet, comme nous l'avons signalé auparavant, aucune méthode ne paraît, actuellement, universellement efficace. Il est même difficile de classer ces méthodes par type de problèmes. Toutes les méthodes développées ont un facteur commun aléatoire qui est le temps mis par chaque méthode pour aboutir à une solution entière. Ce temps de convergence peut dépasser toute prévision malgré les gros ordinateurs dont nous disposons . D'autres difficultés d'ordre technique apparaissent comme de sérieux obstacles quant aux méthodes basées sur les algorithmes de Gomory [22], [23] et [24]. Ces facteurs sont surtout, pour quelques codes, les erreurs cumulées d'arrondi et de dépassement de capacité de certaines mémoires des calculateurs. Ces complications font en sorte que de nombreux problèmes restent pratiquement insolubles. Cette solution est parfois aggravée par le fait que certains programmes sont insolubles à quelques codes et le sont facilement par d'autres.

Afin d'analyser de près les contributions des codes d'une part, et des caractéristiques des calculateurs d'autre part, dans la résolution d'un programme linéaire en nombres entiers, nous nous sommes imposés un ensemble d'exemples numériques les plus variés tant au point de vue structure que de la provenance afin que les résultats obtenus puissent être statistiquement significatifs.

Dans ce chapitre (et le chapitre VIII) nous étudierons le comportement de chaque code sur la même série d'exemples. Nous tenterons ensuite dans un premier temps de faire un parallèle entre les différentes règles de choix de pivots (§ III.2 et IV.4.3) quant aux codes à base simpliciale. Dans un deuxième temps nous donnerons une comparaison entre codes en tenant compte cette fois-ci des configurations des systèmes électroniques employés.

Les critères de comparaison qui viennent à l'esprit ne sont pas très significatifs. Nous pouvons en citer quelques uns :

- le rapport temps de résolution du programme en nombres entiers sur temps de résolution en nombres rationnels en est un ; ce dernier présente malheureusement trop de dispersion;

- les nombres d'itérations et de troncatures du domaine des solutions réalisables en continu apparaissent comme des critères plus significatifs que le premier;

- le temps de calcul en processor central en est un autre, il est plus ou moins lié au nombre d'itérations;

- les caractéristiques de chaque programme sont aussi des éléments de comparaison :

  . nombre de variables (variables d'écart non comprises) : n
  . nombre de contraintes (contraintes de troncature non comprises) : m
  . densité du tableau simplicial : d
  . densité du vecteur second membre ou fonction économique : δ.

VII.2 Comparaison des différents codes à base simpliciale

Ces codes sont basés sur le principe de troncature de Gomory et sont différenciés par les critères de choix des lignes génératrices des contraintes de troncature. Les codes développés aux paragraphes (VI.2.1,3,4,5), Tokosi 3 et Tokosi 4 utilisent respectivement les règles données par (IV.30), (§ IV.4.5), (§ III.2.1) et (§ III 2.4). En général, ces règles semblent relativement plus efficaces que les autres. Tout comme dans la référence [47] cette comparaison a été faite sur quatre types d'exemples fournis d'ailleurs par les références [47] et [30].

Ces exemples sont des types suivants :

- choix d'investissement
- charges fixées
- théorie des graphes
- quelconque.

Les tableaux récapitulatifs de chaque type d'exemple donnent

- le n° de l'exemple
- le nombre de variables N
- le nombre de contraintes M
- la densité du tableau simplicial d
- la densité du vecteur fonction économique δ
- la densité du vecteur critère de candidature final de la fonction économique δf
- la valeur rationnelle de la fonction économique Z_{rat}
- la valeur entière de la fonction économique Z_E
- le nombre d'itérations de la solution rationnelle α
- le nombre de troncatures du domaine des solutions réalisables en continu β
- le nombre total d'itérations de la solution entière γ
- la solution entière x_i, i=1,N
- le temps de résolution mis par chaque code utilisé en secondes.

VII.2.1 Problèmes de choix d'investissement :

Les exemples sont de petite taille (10x11). Le but poursuivi est d'étudier la sensibilité de chaque code par suite d'un changement progressif de la valeur d'un second membre.

Les 9 exemples sont déduits de l'exemple suivant par le changement des valeurs de a :

$$\max x_0 = 20x_1 + 18x_2 + 17x_3 + 15x_4 + 15x_5 + 10x_6 + 5x_7 + 3x_8 + x_9 + x_{10}$$

$$\text{t.q.} \quad 30x_1 + 25x_2 + 20x_3 + 18x_4 + 17x_5 + 11x_6 + 5x_7 + 2x_8 + x_9 + x_{10} \leq a$$

$$x_i \leq 1 \qquad \forall\, i = 1,10$$

$$x_i \geq 0$$

N°	N	M	d	δ	δ_f	a	Z_{rat}	Z_E	α	β	γ	x 1	2	3	4	5	6	7	8	9	10
1	10	11	.181	1	1	35	33.3	33	10	1	14	0	0	0	0	1	1	1	1	0	0
2	10	11	.181	1	1	60	54.5	52	12	12	51	0	0	1	1	1	0	1	0	0	0
3	10	11	.181	1	1	65	58.6	57	13	4	30	0	0	1	1	1	0	1	1	1	0
4	10	11	.181	1	1	70	62.9	62	14	2	23	0	0	1	1	1	1	0	1	1	1
5	10	11	.181	1	1	75	67.	67	12	0	12	0	0	1	1	1	1	1	1	1	1
6	10	11	.181	1	1	80	70.6	68	12	13	52	0	1	1	0	1	1	1	1	0	0
7	10	11	.181	1	1	85	74.25	70	12	24	110	1	0	0	1	1	1	1	1	1	1
8	10	11	.181	1	1	90	77.65	75	12	7	47	0	1	1	1	1	0	1	1	1	1
9	10	11	.181	1	1	100	102	85	11	1	12	0	1	1	1	1	1	1	1	1	1

Tableau VII.1 Choix d'investissement (TOKOSI 2)

Quant au tableau VII.2 [47] il récapitule les différents résultats des autres codes (§ VI.2) programmés et testés sur des systèmes électroniques différents.

Code	CDC 6600 TOKOSI 2		IBM 7090 LIP 1		CDC 6600 TOKOSI 1		IBM 7090 IPM 3		CDC 3600 ILP 2-1		CDC 3600 ILP 2-2		CDC 3600 IPSC	
N°	γ	temps	γ	temps	γ	temps	γ	temps	γ	temps	γ	temps	γ	temps
1	14	.175	19	2.417	14	0.025	14	3.767	54	1.937	51	1.879	46	2.217
2	51	.645	55	5.083	39	0.408	31	4.083	163	3.670	77	2.278	64	2.840
3	30	.355	41	3.900	60	0.284	30	4.033	168	3.767	59	2.011	71	3.054
4	23	.262	19	2.600	50	0.283	18	3.683	192	4.191	48	1.920	62	2.812
5	12	.150	12	1.867	12	0.249	11	3.500	139	3.188	32	1.616	50	2.463
6	52	.682	40	3.867	25	0.363	18	3.733	157	3.451	54	1.974	81	3.375
7	110	1.441	81	7.883	97	0.808	61	4.733	504	8.675	119	2.910	131	4.943
8	47	.575	51	4.517	30	0.700	21	3.833	370	6.528	57	1.952	102	4.035
9	12	.329	12	1.867	12	0.329	12	3.600	201	4.130	34	1.628	44	2.256

Tableau VII.2

Indépendamment des configurations des systèmes utilisés, tous les codes testés sur ce type de problème semblent performants

Il n'est pas question de généraliser mais il est cependant intéressant de suivre l'évolution de chaque code quand le second membre a varie.

* La règle de choix de LIP 1 ou TOKOSI 2 (§VI.2.5.2) est la plus sensible de toutes (Tableau VII.2).

En effet cette règle

* Le nombre d'itérations paraît proportionnel au temps d'exécution.

* Les temps d'exécution varient sensiblement avec la valeur a du second membre.

La règle (IV.30) des codes IPM3 et TOKOSI 1 paraît comme très peu sensible au changement de a, quoique le nombre d'itérations soit inférieur à celui des codes LIP1 et TOKOSI 2, pour chaque exemple. Les temps d'exécution évoluent dans les rapports de 1 à 2 pour certains codes, alors qu'ils varient dans les rapports 1 à 4 ou plus pour les deux premiers codes du tableau VII.2.

Ces remarques sont valables quant aux règles des codes ILP2-1, ILP2-2 et I.P.S.C.

VII.2.2 Problèmes de charges fixes : [47] [30]

Le nombre d'exemples est assez restreint, une dizaine. L'intérêt réside dans les difficultés rencontrées par tous les codes bien que la taille de ces exemples soit assez modeste.

Les différents exemples sont assez voisins, ils ne diffèrent que par le changement de quelques coefficients ou de la taille.

Exemples 1, 2, 3, 4 :

$$\text{maximiser } x_0 = x_1 + x_2 + x_3$$

$$\text{t.q.} \quad \begin{aligned} x_1 + 2x_2 + 2x_3 + 2x_4 + 3x_5 &\leq a_1 \\ 2x_1 + x_2 + 2x_3 + 3x_4 + 2x_5 &\leq a_2 \\ x_1 - p_1 x_4 &\leq 0 \\ x_2 - p_2 x_5 &\leq 0 \end{aligned}$$

$$x_1, x_2, x_3, x_4, x_5 \in N$$

$$0 \leq x_4 \leq 1$$

$$0 \leq x_5 \leq 1$$

Exemples 5, 6

$$\text{Max} \quad x_0 = x_1 + x_2 + x_3$$

$$\begin{aligned}
\text{t.q.} \quad & x_1 + 2x_2 + 2x_3 + 20x_4 + 30x_5 \leq a_1 \\
& 2x_1 + x_2 + 2x_3 + 30x_4 + 20x_5 \leq a_2 \\
& x_1 - p_1 x_4 \leq 0 \\
& x_2 - p_2 x_5 \leq 0 \\
& x_4 \leq 1 \\
& x_5 \leq 1
\end{aligned}$$

$$x_1, x_2, x_3, x_4, x_5 \in N$$

Exemples 7, 8

Ces exemples sont identiques aux exemples 5 et 6 et n'en diffèrent que par la suppression des deux dernières contraintes.

Exemples 9

$$\text{Max} \quad x_0 = x_1 + x_2 + x_3$$

$$\begin{aligned}
\text{t.q.} \quad & x_1 + x_2 + 2x_4 + 2x_5 \leq 10 \\
& x_1 + x_3 + 2x_4 + 2x_6 \leq 10 \\
& x_2 + x_3 + 2x_5 + 2x_6 \leq 10 \\
& x_1 - 8x_4 \leq 0 \\
& x_2 - 8x_5 \leq 0 \\
& x_3 - 8x_6 \leq 0
\end{aligned}$$

$$X \in N$$

Exemples 10

$$\text{Max} \quad x_0 = x_1 + x_2 + x_3 + x_4 + x_5 + x_6$$

$$\begin{aligned}
\text{t.q.} \quad & 3x_1+7x_2+8x_3+4x_4+6x_5+5x_6+9x_7+7x_8+16x_9+8x_{10}+24x_{11}+5x_{12} \leq 110 \\
& 4x_1+6x_2+3x_3+1x_4+5x_5+8x_6+12x_7+6x_8+6x_9+2x_{10}+20x_{11}+8x_{12} \leq 95 \\
& 5x_1+5x_2+6x_3+2x_4+1x_5+5x_6+15x_7+5x_8+12x_9+4x_{10}+4x_{11}+5x_{12} \leq 80 \\
& 6x_1+4x_2+2x_3+9x_4+7x_5+1x_6+18x_7+4x_8+4x_9+18x_{10}+28x_{11}+1x_{12} \leq 100 \\
& x_1 - 12x_7 \leq 0 \\
& x_2 - 15x_8 \leq 0 \\
& x_3 - 12x_9 \leq 0 \\
& x_4 - 10x_{10} \leq 0 \\
& x_5 - 11x_{11} \leq 0 \\
& x_6 - 11x_{12} \leq 0
\end{aligned}$$

$$x_i \leq 1 \qquad i=7,8,9,10,11,12$$

$$X \in N.$$

Les tableaux récapitulatifs (VII.3) et (VII.4) sont établis de la même façon que les tableaux VII.1 et VII.2.

En outre pour des raisons économiques le nombre maximum d'itérations imposé à tout programme linéaire au cours de l'exécution est de 3000 pour les codes TOKOSI et de 7000 pour les codes de la référence [40]. Cette limitation a été jugée nécessaire chaque fois que des indices de cyclage indéfini apparaissent, ou toutes les fois que la convergence vers la solution entière devient très lente.

N°	N	M	d	δ	δ_f	a_1	a_2	p_1	p_2	Z_{rat}	Z_E	α	β	γ	x 1	2	3	4	5	6	7	8	9	10	11	12
1	5	4	.7	.6	.2	18	15	6	7	8.79	7	4	17	37	0	7	0	0	1							
2	5	4	.7	.6	.2	18	17	9	7	9.625	8	4	12	29	3	5	0	1	1							
3	5	4	.7	.6	.2	21	21	9	9	11.8	10	4	22	49	4	6	0	1	1							
4	5	4	.700	.6	.2	19	25	6	8	11.88	11	4	11	26	6	0	5	1	0							
5	5	6	.533	.6	.2	180	150	60	75	88.5	76	4	244	428	22	54	0	1	1							
6	5	6	.533	.6	.2	210	210	90	90	118	106	4	543	997	52	54	0	1	1							
7	5	4	.466	.6	.2	180	150	60	75	88.5	76	4	1160	1755	22	54	0	1	1							
8	5	4	.466	.6	.2	210	210	90	90	118	106	4	1008	1586	52	54	0	1	1							
9	6	6	.5	.5	.166	-	-	-	-	12.	9	6	29	266	3	3	3	1	1	1						
10	12	10	.5	.5	.166	-	-	-	-	18.7	17	10	76	168	0	0	0	8	0	9	0	0	0	1	0	1

Tableau VII.3 Charges fixes (TOKOSI 2)

Les exemples 7,8 montrent l'effet de la réduction du nombre de contraintes. Les domaines des solutions réalisables rationnelles pour ces derniers sont plus vastes que pour les exemples 5,6 ; par suite un grand nombre de troncature s'est avéré nécessaire pour certains codes (Tableau VII.4) afin d'arriver à la solutio entière.

CODES	CDC 6600 TOKOSI 1		IBM 7090 IPM 3		CDC 6600 TOKOSI 2		IBM 7090 LIP 1		CDC 3600 ILP 2-1		CDC 3600 ILP 2-2		CDC 3600 IPSC	
N°	γ	temps	γ	temps	γ	temps	γ	temps	γ	temps	γ	temps	γ	temps
1	47	1.906	54	3.117	37	0.004	24	1.833	135	1.594	36	.852	32	0.743
2	40	1.534	81	3.767	29	0.250	15	1.350	94	1.260	47	.935	45	.886
3	28	.89	37	3.033	49	0.409	26	1.883	154	1.712	104	1.384	56	1.007
4	30	1.036	91	4.100	26	0.232	18	1.483	93	1.158	18	0.674	22	0.632
5	3000^+		7000^+	-	428	3.820	158	9.012	7000^+	-	7000^+	-	6104	79.9
6	846	6.642	7000^+	-	937	7.801	123	7.567	7000^+	-	311	3.273	3320	43.483
7	1773	12.794	7000^+	-	1755	12.640	159	7.833	7000^+	-	7000^+	-	7000^+	-
8	1182	8.528	7000^+	-	1586	11.345	126	6.417	7000^+	-	306	3.033	7000^+	-
9	99	.931	118	5.183	266	5.360	42	3.233	7000^+	-	298	3.598	339	5.483
10	3000^+	-	7396	71.100	168	3.694	102	9.150	7000^+	-	7000^+	-	7000^+	-

Tableau VII.4 Charges fixes

Il ressort des résultats numériques du tableau (VII.4) que la règle du (§ IV.4.5), utilisant la ligne de la fonction économique, comme génératrice de contraintes de troncature, est la plus efficace. Il semblerait que cette règle soit la mieux adaptée à ce type de problèmes.

VII.2.3 Problèmes de type combinatoire [40]

Etant donné un graphe G de n sommets [8] et deux couleurs différentes. Chaque arc des n(n-1)/2 arcs du graphe G est coloré par une des deux couleurs précédentes. Le problème consiste à trouver min $(n) = n(k_1,k_2)$, tel que si $n \geq n$ il existe un circuit de k_1 sommets dont toutes les arêtes qui les relient sont de la première couleur ou un circuit de k_2 sommets dont toutes les arêtes les reliant sont de la 2ème couleur.

Les principales raisons de ce type de problèmes tiennent au fait que les variables sont bivalentes d'une part, et que d'autre part on se trouve dans une situation de dégénérescence du problème après un nombre réduit d'itérations.

Exercice n°1

$$\begin{cases} k_1 = k_2 = 3 \\ n = 4 \end{cases}$$

Les circuits dans ce cas sont des triangles (3 arêtes, 3 sommets). Soit x_{ij} l'arête joignant le sommet i au sommet j ou inversement. Convenons que $x_{ij}=0$ si cette arête est de la 1ère couleur et $x_{ij}=1$ si elle est de la seconde couleur.

Les contraintes sont alors du type $1 \leq x_{ij} + x_{jk} + x_{ik} \leq 2$. Soit alors le programme suivant :

$$\min x_0 = x_{12} + x_{13} + x_{14} + x_{23} + x_{24} + x_{34}$$

$$\text{t.q.} \begin{cases} x_{12} + x_{13} + x_{23} \leq 2 \\ x_{12} + x_{14} + x_{24} \leq 2 \\ x_{13} + x_{14} + x_{34} \leq 2 \\ x_{23} + x_{24} + x_{34} \leq 2 \\ x_{12} + x_{13} + x_{23} \geq 1 \\ x_{12} + x_{14} + x_{24} \geq 1 \\ x_{13} + x_{14} + x_{34} \geq 1 \\ x_{23} = x_{24} + x_{34} \geq 1 \end{cases}$$

une solution entière

$$x_0 = 2 \qquad x_{13} = 1,\ x_{24} = 1$$
$$x_{12} = x_{14} = x_{23} = x_{34} = 0.$$

Exercice n°2

C'est la même formulation que l'exercice précédent mais n=5 points, le nombre de variables est de 10 et celui des contraintes est de 20.

Une solution entière de cet exemple est :

$$x_0 = 5 \qquad x_{12} = x_{13} = x_{24} = x_{35} = x_{45} = 1$$
$$x_{14} = x_{15} = x_{23} = x_{25} = x_{34} = 0.$$

Exercice n°3

$$\begin{cases} n = 6 \\ k_1 = 3 \\ k_2 = 4 \end{cases}$$

Les circuits sont des triangles et des quadrilatères.

Les contraintes sont du type $x_{ij}+x_{ik}+x_{j\ell}+x_{k\ell} \leq 3$ pour les quadrilatères.

Les contraintes relatives aux circuits triangulaires sont du type

$$x_{ij} + x_{ik} + x_{jk} \geq 1.$$

Nombre de variables 15.
Nombre de contraintes 65.

Le programme formulé est donné par la référence [43]. Une solution entière de cet exemple est :

$$x_0 = 6 \qquad x_{14} = x_{15} = x_{23} = x_{26} = x_{36} = x_{45} = 1$$
$$x_{12} = x_{13} = x_{16} = x_{24} = x_{25} = x_{34} = x_{35} = x_{46} = x_{56} = 0$$

Exercice n°4

$$n = 7$$
$$k_1 = 3$$
$$k_2 = 4$$

Le nombre de variables est de 21 quant au nombre de contraintes il est de 140.

Ces deux derniers exemples n'ont pas été testés par les codes TOKOSI.

CODES					CDC 6600 TOKOSI 1		IBM 7090 IPM 3		CDC 6600 TOKOSI 2		IBM 7090 LIP 1		CDC 3600 ILP 2-1		CDC 3600 ILP 2 - 2		CDC 3600 I P S C	
	M	d	δ	δ_f	γ	temps	γ	temps	γ	temps	γ	temps	γ	temps	γ	temps	γ	temps
5	8	.5	1		6	.004	4	2.200	6	.004	11	1.767	4	.891	4	.907	5	.765
0	20	.3	1		85	2.256	140	8.760	85	2.256	83	19.800	132	4.278	74	3.383	86	5.373
5	65	.185	1		-	-	42	13.917	-	-	27	22.817	114	11.336	29	7.086	70	15.819
1	140	.167	1		-	-	7000+	-	-	-	7000+	-	7000+		7000+	-	7000+	-

Tableau VII.5 Problèmes combinatoires

Il apparaît d'après les résultats du tableau VII.5 que, d'une part le nombre d'itérations n'est pas proportionnel au temps d'exécution , d'autre part le nombre d'itérations est fonction de la densité de remplissage de la matrice simpliciale pour une même structure de problèmes (1,2), (3,4) : Signalons qu'aucune des règles de génération de contraintes de troncatures n'a paru efficace pour le problème n°4.

VII.2.4 Problèmes de type quelconque [47], [30] :

Il s'agit de 6 exemples de tailles différentes et d'origines variées extraits de la référence [40]. Les raisons militant en faveur de ce choix d'exemples sont d'une part la dispersion constatée dans les temps d'exécution et dans les nombres d'itérations pour des exemples ne différant que par les seconds membres et d'autre part par la multiplicité des solutions entières optimales (Tableau VII.6).

Exemples n°s 1, 2

Les exemples sont identiques, ils ne diffèrent que par les valeurs des seconds membres ${}_1a$ et ${}_2a$:

$$ {}_1a_i = {}_2a_i + 1 \qquad i = 1,2, \ldots,7 $$

$$\min x_0 = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 \qquad a_1 \quad a_2$$

$$\text{t.q.} \begin{cases} x_1 \quad + x_4 + x_5 \quad + x_7 \geq 5 > 4 \\ + x_2 \quad + x_4 \quad + x_6 + x_7 \geq 5 \quad 4 \\ x_3 \quad + x_5 + x_6 + x_7 \geq 5 \quad 4 \\ x_1 + x_2 \quad + x_5 + x_6 \quad \geq 4 \quad 3 \\ x_1 \quad + x_3 + x_4 \quad + x_6 \quad \geq 4 \quad 3 \\ + x_2 + x_3 + x_4 + x_5 \quad \geq 4 \quad 3 \\ x_1 + x_2 + x_3 \quad + x_7 \geq 3 \quad 2 \end{cases}$$

Exemple n°3

$$\min x_0 = 13x_1 + 15x_2 + 14x_3 + 11x_4$$

$$\text{t.q.} \begin{cases} 4x_1 + 5x_2 + 3x_3 + 6x_4 - x_5 \geq 96 \\ 20x_1 + 21x_2 + 17x_3 + 12x_4 - x_6 \geq 200 \\ 11x_1 + 12x_2 + 12x_3 + 7x_4 - x_7 \geq 101 \end{cases}$$

Exemples n°s 4, 5

Exemples identiques de dimensions 15×15 différents par leur second membre ${}_5a_i = {}_4a_i + 2 \qquad i=1,\ldots,15.$

Les coefficients du tableau simplicial sont 0 ou 1 à raison de huit par colonne.

Le vecteur fonction économique est une ligne dont toutes les composantes sont égales à 1.

Chaque n° de ligne donné dans l'une des 15 colonnes du tableau simplicial indique que le coefficient situé à l'intersection de cette colonne et de cette ligne est égal à 1. Les autres coefficients étant nuls :

colonnes		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
coef. de la fonction économique		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	(Min)
	lignes i	1	2	3	4	1	1	1	2	2	3	1	1	1	2	1	
		5	5	6	7	2	3	4	3	4	4	2	2	3	3	2	
		6	8	8	9	6	5	5	5	5	6	3	4	4	4	3	
		7	9	10	10	7	7	6	6	7	7	7	6	5	5	4	
		11	11	11	12	8	8	9	9	8	8	9	8	8	6	11	
		12	12	13	13	9	10	10	10	10	9	10	10	9	7	12	
		13	14	14	14	13	12	11	12	11	11	11	12	13	14	13	
		15	15	15	15	14	14	14	13	13	12	15	15	15	15	14	

Les seconds membres sont, les contraintes étant interprêtées $\geq$,

		$4^{a}i$	$5^{a}i$
lignes	1 à 4	6	8
	5 à 10	5	7
	11 à 14	4	6
	15	3	5

Exemple N°9

Le 6° exemple porte le n°9 dans la référence [30] ; il a la même structure que les exemples 4,5 mais un nombre plus élevé de contraintes (50). Nous représentons son tableau simplicial de la même façon que les deux exemples précédents.

colonnes		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
coef. de la fonction économique		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	(Min)
	lignes i	3	4	1	1	2	1	2	3	4	5	11	12	13	14	15	
		4	5	5	2	3	6	7	8	9	10	16	17	18	19	20	
		7	6	7	8	6	13	14	11	11	12	23	24	21	21	22	
		10	8	9	10	9	14	15	15	12	13	24	25	25	22	23	
		21	22	23	24	25	17	16	17	18	16	27	26	27	28	26	
		26	27	28	29	30	20	18	19	20	19	30	28	29	30	29	
		31	32	33	34	35	31	32	33	34	35	31	32	33	34	35	
		36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	

N°	N	M	d	δ	δ_f	Z_{rat}	Z_E	α	β	γ	x 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	7	7	.572	1.	.143	7.5	8	7	1	8	1	0	0	2	2	1	2								
2	7	7	.572	1.	.143	5.75	7	7	5	18	1	0	0	2	2	1	1								
-	-	-	-	-	-	-	-	-	-	-	1	1	0	1	2	2	0								
-	-	-	-	-	-	-	-	-	-	-	1	0	0	0	2	2	2								
3	7	3	.715	.572	1	180.2	187	2	13	28	0	0	0	17	6	4	18								
4	15	15	.533	1.	0.137	9.23	10	17	29	91	0	0	0	0	1	1	1	1	1	1	1	1	1	0	1
-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	1	1	1	1	1	0	1	1	1	1	1
5	15	15	.533	1.	.067	12.820	15	19	78	332	1	0	0	0	2	2	2	0	0	0	2	2	2	0	2
-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	2	1	2	0	1	1	1	2	1	2	2
-	-	-	-	-	-	-	-	-	-	-	1	1	1	1	1	1	1	1	0	1	0	2	1	2	1
9	15	50	.16	1.	.067	5.67	9	28	114	737	1	1	1	1	1	1	1	0	1	1	0	0	0	0	0
-	-	-	-	-	-	-	-	-	-	-	0	0	1	1	1	1	1	1	1	0	0	0	1	1	0
-	-	-	-	-	-	-	-	-	-	-	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0
-	-	-	-	-	-	-	-	-	-	-	1	1	1	1	1	0	1	1	1	1	0	0	0	0	0

Tableau VII.6 Problèmes de type quelconque (TOKOSI 2)

Les 50 seconds membres sont égaux à 1. Les contraintes sont interprêtées ≥ 1 pour les lignes de 1 à 35 et ≤ 1 pour les lignes 36 à 50.

Les tableaux VII.6 et VII.8 donnent d'une part les caractéristiques de chaque exemple et les différentes solutions entières et d'autre part les résultats des différents codes programmés sur des calculateurs à configurations variées.

N°	CDC 6600 TOKOSI 1		IBM 7090 IPM 3		CDC 6600 TOKOSI 2		IBM 7090 LIP 1		CDC 3600 ILP 2-1		CDC 3600 ILP 2-2		CDC 3600 I P S G	
	γ	temps	γ	temps	γ	temps	γ	temps	γ	temps	γ	temps	γ	temps
1	9	.017	8	2.300	8	.104	11	1.866	9	1.010	11	1.087	9	.917
2	19	.017	17	2.833	18	.018	32	3.016	13	1.056	15	1.149	16	1.045
3	47	.280	22	2.633	28	.468	53	2.866	23	.705	14	.621	14	.477
4	178	6.585	24	5.933	91	2.768	73	11.666	41	3.492	18	3.079	17	3.638
5	287	12.108	1144	51.6	332	10.996	351	66.483	7000+	-	842	26.184	1020	62.817
6	724	51.933	6758	633.313	737	60.037	953	473.1	7000+	-	1105	75.121	752	95.392

Tableau VII.7 Problèmes de type quelconque

L'examen du tableau VII.7 montre que, pour ce type de problèmes, le temps du Processor central n'est pas relié à la densité de remplissage du tableau simplicial et aux dimensions du programme. Ce manque de corrélation temps d'exécution-taille de programme est manifeste pour tous les codes, mêmes pour les codes que nous avons jugés comme peu sensibles (§ VII.2.1). En effet, le passage des seconds membres de l'exemple 1 à l'exemple 2 ne modifie pas d'une façon sensible le temps d'exécution des codes IPM 3, ILP 2-1, ILP 2-2 et IPSC ; par contre pour ces mêmes codes, le passage des seconds membres de 4 aux seconds membres de 5 multiplie le temps d'exécution de l'exemple 4 par un facteur supérieur à 10 pour attendre la solution entière de l'exemple 5.

Notons aussi que ce type de programmes confirme une fois de plus la sensibilité du critère de choix de ligne génératrice de contraintes (§ IV.4.5) des codes LIP -1 et TOKOSI 2 (Tableau VII.7).

I.3 Comparaisons entre codes compte tenu de la configuration des systèmes électroniques utilisés

La comparaison des codes que nous venons de développer, abstraction faite des systèmes électroniques, de la façon de programmer et du langage de programmation est un peu artificielle. Le parallèle entre ces codes, compte tenu des systèmes électroniques utilisés, est certainement encore plus artificiel que dans le cas précédent.

En effet, il s'agit de 3 configurations différentes du point de vue Hardware et Software. Du point de vue temps de cycle de base les temps sont respectivement pour les 3 calculateurs CDC 3600, IBM 7090 et CDC 6600, 1. , 2.2 et 1 microsecondes. Quant aux systèmes, ils sont variés SCOPE et autres à des niveaux plus ou moins élevés. En ce qui concerne les langages de programmation FORTRAN ou COMPASS, les versions sont automatiquement différentes et à performances inégales.

Ces éléments faussent dans une certaine mesure les conclusions tirées quant à l'évaluation de l'efficacité de chaque critère utilisé.

VII.4 Conclusion générale

Tout d'abord nous remarquons que l'utilisation des mêmes critères de choix nous a conduit, pour le même exemple, à des nombres d'itérations différents ((tableaux VII.1, VII.2 ... VII.7) codes IPM 3, TOKOSI 1, codes LIP 1, TOKOSI 2). Cette différence est due en partie à la manière de mener les calculs. Pour les codes TOKOSI les calculs sont faits en nombres déclarés entiers (en fixe) par contre pour les autres codes, les calculs sont effectués en virgule flottante et les nombres d'itérations sont fonction des seuils d'arrondi adoptés. Notons au passage que les écarts sur les temps de Processor central sont eux dûs en partie aux cycles de base différents et aux systèmes utilisés.

Bien que certains codes soient mieux adaptés que d'autres, à certains problèmes il est difficile de classer les règles de choix (§ IV.4.3 et III.2) par domaine d'application, mais une conclusion semble évidente : il n'existe pas de choix unique et par conséquent il n'existe pas non plus de code parfaitement efficace. Il est cependant intéressant de constater que la règle de choix des codes LIP 1 ou TOKOSI 2 a permis la résolution de 28 problèmes sur 29 sans que parfois le nombre d'itérations n'excède 2000. Bien que la règle de choix de ces codes soit heuristique, elle semble toutefois bien fondée du point de vue mathématique. En effet, le fait que les coefficients de départ soient entiers et que la valeur de la fonction économique soit entière, impliquerait dans une certaine mesure que les variables sont entières (à cette étape la base est à la fois duale et primale réalisable). On pourrait penser à la faible probabilité de rencontrer, à cette étape, une combinaison des variables non entières qui donnerait une valeur entière à la fonction économique. Cela justifie probablement le choix de ce critère et sa performance. Si on pouvait classer ces codes (ou ces règles de choix) la règle du (§ IV.4.3) serait de loin la plus efficace. Le code basé sur cette règle parait expérimentalement le plus efficace des codes fondés sur les méthodes de Gomory.

Chapitre VIII

EXPERIENCES NUMERIQUES (suite)

COMPARAISON ENTRE LES METHODES DE GOMORY ET LES METHODES EXPLORATOIRES

II.1 Généralités

Lorsqu'il s'agit d'un programme à variables entières dont les bornes supérieures dépassent 1, le nombre de variables du programme bivalent correspondant se trouve accru par les variables bivalentes (exprimant les différentes variables totalement entières et bornées (§ VI.3.2)). Ceci a pour conséquence l'augmentation considérable de la taille du problème quant aux méthodes exploratoires à variables bivalentes. Le nombre de variables se trouve donc multiplié par q (2^q étant la borne supérieure des variables entières) lorsqu'on passe d'un programme totalement entier à un programme à variables bivalentes.

En outre, il faut ajouter au problème de taille le nombre fabuleux des combinaisons possibles des solutions bivalentes (booléennes). Ainsi lorsqu'il s'agit d'un problème de 30 variables bivalentes, le nombre de combinaisons possibles dépasse le milliard. Supposons qu'il faille une milliseconde pour explorer une combinaison et conclure, il faudrait passer 27 heures environ de Processor central pour tester toutes les combinaisons possibles par un puissant calculateur de 3ème génération. C'est pour cette raison que plusieurs méthodes exploratoires ont cherché à éviter cette revue systématique de toutes les solutions et cela grâce à des règles de choix et à des "filtres" [2], [3].

La comparaison entre les algorithmes de Gomory et les algorithmes exploratoires n'a pas tout à fait la même base de départ : les tailles des problèmes ne sont pas identiques. L'application des algorithmes de Gomory aux problèmes écrits sous la forme bivalente nécessite des contraintes supplémentaires du type $x_j \leq 1$, $j=1,\ldots,n$. Le nombre de ces contraintes est égal au nombre de variables bivalentes.

Dans les paragraphes suivants nous reprenons les mêmes critères de comparaison qu'au chapitre VII : la comparaison porte sur des problèmes

variés résolus à la fois par des codes basés sur les algorithmes de Gomory et des codes basés sur les méthodes exploratoires (chapitre VI).

VIII.2 Exemples numériques

VIII.2.1 Généralités :

La comparaison porte aussi dans ce chapitre sur une série d'exemples numériques d'origines diverses. Ces exemples traitent de problèmes réels et des programmes à intérêts expérimentaux.

Les exemples sont extraits en majorité des références [34] et [11]. D'autres exemples traitent des problèmes types de la théorie des Graphes [8].

Les exemples de la référence [34] sont en partie de types quelconques. D'autres traitent des problèmes agricoles, de choix d'investissement, de transport de personnel, ... Quant aux exemples de la référence [9] ils sont presque tous bivalents, soit par construction à partir de problèmes formulés en variables entières,soit par construction directe d'une manière quelconque et au hasard.

VIII.2.2 Exemples de la référence [34] :

Nous allons procéder comme au chapitre précédent. Le tableau VIII.1 récapitule les caractéristiques des exemples traités et donne les résultats numériques fournis par les codes algébriques de Gomory, TOKOSI 1, TOKOSI 2 et TOKOSI 3. Nous avons, en outre, actualisé dans ce tableau les solutions entières optimales trouvées.

Les exemples sont en majorité de petite taille et formulent des cas réels. Les nombres d'itération et de troncature sont assez modestes dans l'ensemble ; nous n'avons rencontré aucun cas de cyclage dans cette série d'exemples.

Les calculs ont été menés en virgule fixe. Ce mode de calcul a été à l'origine des dépassements de capacité de certaines mémoires notamment lors de la résolution des exemples n° 16, 17, 20, 21 par les codes TOKOSI 1 et TOKOSI 2 (chapitre VI). Ces difficultés nous ont conduit à envisager une partie des calculs en double précision et en virgule flottante afin de mener les calculs jusqu'au bout.

N°	N	M	Densité	δ	Z_{rat}	Z_{ent}	δ_f	Nit rat	N.tr	N;it	Alg. codes	temps sec.	x																										
													1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	21	22	23	24	25	26	27	28
1	3	3	1	1	34.2	38	0.3	03	19	33	TOKOSI 1	.098	4	5	2																								
2	3	4	1	1	13.97	11	1.0	2	4	9	TOKOSI 1	.241	2	0	3																								
3	4	3	.75	1	30.5	29	.5	4	15	23	"	.931	5	5	0	2																							
4	3	3	1.	1	216.	207	.1	5	1	13	"	.339	1	1	1	0																							
5	3	4	.915	1	27.35	26	.5	2	32	49	"	3.010	4	2	6																								
6	4	4	.875	1	31.95	31	.5	3	9	25	"	1.063	3	0	6	1																							
7	4	5	.85	1	63.5	61	.50	10	3	17	TOKOSI 1	.549	2	4	14	1																							
8	5	4	.9	1	61.3	63	.6	5	6	17	TOKOSI 2	.157	4	4	0	0	5																						
9	4	4	.79	.5	172.5	186	.5	-	-	76	TOKOSI 3	2.420	22	41	1	1																							
10	5	6	.6	.4	4.5	7	.4	-	-	20	"	.913	3	2	1	1	1																						
11	5	6	.868	1	23.2	19	.4	3	15	31	TOKOSI 2	1.714	2	1	0	3	1																						
12	7	4	.5	1	14.5	15	1	-	-	2	TOKOSI 3	.167	0	0	0	0	4	7	0																				
13	5	10	.54	1	29.4	23	.2	9	32	92	TOKOSI 1	7.998	2	0	2	2	1																						
14	11	4	.568	1	129.9	123	1		18	36	"	1.2	0	0	15	0	10	0	0	0	0	1	0																
15	10	6	1.	1	1505.5	1510	0.8		5	15	TOKOSI 2	.243	0	0	99	246	40	0	0	0	1	0																	
* 16	6	7	.43	1	48400	46000	.	8	-	-	"	-	0	40	0	0	0	0	1	1																			
* 17	11	12	.41	1	16700	16525	.	-	-	-	"	-	50	0	19	0	38	12	1	0	2	1	0																
18	10	12	.376	1	1162.3	1161	0.8	4	4	9	TOKOSI 2	.038	19	58	0	0	0	40	3	0	0	0																	
19	7	8	.785	1	160.06	120	.28	4	253	480	"	5.232	0	1	0	13	3	2	0	0																			
* 20	22	27	.101	1	2449.02	3013	.28	-	-	-	"	-	0	0	0	0	0	0	1	0	1	0	0	2	0	0	1	0	0	0	0	0	0	0				
* 21	20	24	.124	.95	1212.74	1224	.	-	-	-	"	-	261	51	0	0	0	51	0	0	0	0	0	22	0	0	0	0	0	11	84	0						
* 22	32	60		1							"	-																											
23	10	11	.18	1	219.92	215	1		5	25	TOKOSI 2	.320	1	1	1	0	1	1	1	1	1	0																	
24	6	8	.291	1	895.2	1070	1		13	26	"	.329	1	0	0	0	1	0																					

Tableau VIII.1 Exemples de la référence [34]

* Les exemples marqués d'un astérisque n'ont pas pu être résolu en fixe, la solution de base courante a été perdue par dépassement de capacité de certaines mémoires dès les premières itérations.

Le tableau VIII.2 donne les résultats des différents codes algébriques et exploratoires testés sur des exemples identiques (en nombre d'itération et en temps Processor central en secondes). Quoique le code de la référence [34] programmé sur le CDC 3600 n'ait pas un grand rapport avec les autres codes (chapitre VI), nous l'avons mentionné à titre indicatif et comparatif quant aux temps de résolution sur un calculateur de la famille CDC 6600.

Notons au passage que la colonne intitulée γ^* (Tableau VIII.2) ne donne pas le nombre d'itérations comme à l'habitude mais le nombre de choix de sommets de graphe des solutions selon la procédure employée dans [34].

Le peu d'exemples traités par les méthodes exploratoires est dû en grande partie à la grande taille de ces exemples obtenue par le passage des variables entières aux variables bivalentes.

Ainsi l'exemple n° 16 de la référence [34] est passé de la taille initiale (7×8) à la taille (7×27) quant à l'exemple n° 14 de la même référence il est passé de la taille (4×11) à la taille (4×30). Le 1er obstacle des algorithmes exploratoires est donc du type encombrement mémoire, le temps de résolution paraîtrait accru par la taille en général.

Les temps de Processor central mis par les autres codes pour les mêmes exemples de la référence [34] et les codes TOKOSI sont très différents. Les codes TOKOSI sembleraient plus performants et plus rapides.

Les difficultés rencontrées par les codes TOKOSI 1 et TOKOSI 2 lors de la résolution des exemples 16, 17, 20, 21 et 22 sont du type dépassement de capacité de certaines mémoires de l'ordinateur utilisé en virgule fixe et elles sont rencontrées en virgule flottante même pour l'algorithme de la référence [34] dans l'inversion de la matrice simpliciale, difficultés plus grandes quand l'inversion se fait dans le corps Q des rationnels.

CODES	CDC 3600		C D C 6600							
	FAURE & MEIGRANGE		TOKOSI		BALAS Filtre		Geoffrian		Hervé	
N°	γ	temps	γ	temps	γ	temps	γ	temps	γ	temps
1	1	1	33	.098	-	-	-	-	-	-
2	1	10.	9	.251	16	7	13	1.27	11	4.36
3	1	10.	23	.931	-	-	-	-	-	-
4	1	10.	13	.339	-	-	-	-	-	-
5	2	10.	49	3.010	-	-	-	-	-	-
6	3	10.	22	1.063	-	-	-	-	-	-
7	3	15.	36	.005	-	-	-	-	-	-
8	5	15.	17	.157	-	-	-	-	-	-
9	2	10.	76	2.420	-	-	-	-	-	-
10	2	20.	20	.913	-	-	-	-	-	-
11	5	20.	31	1.714	-	-	-	-	-	-
12	6	20.	2	.167	-	-	-	-	-	-
13	0	15.	92	7.998	-	-	-	-	-	-
14	0	20.	36	1.2	-	-	15007	140.	43	208.8
15	6	90.	15	.243	-	-				-
16	9	30.	DC		10815+	-	1829	17.	121	520
17	6	30.	DC		-	-	-	-	-	-
18	5	20.	9	.038	-	-	-	-	-	-
19	8	20.	480	5.232	-	-	-	-	-	-
20	-	600.+	DC		-	-	-	-	-	-
21	15	60.	DC		-	-	-	-	-	-
22	-	-	DC		-	-	-	-	-	-
23	8	10.	15	.320	-	-	-	-	-	-
24	4	10.	26	.329	-	-	-	-	-	-

Tableau VIII.2 Comparaison entre codes

VIII.2.3 Exemples de la référence [11]

Les tableaux VIII.3 et 4 ont les mêmes caractéristiques que celles des tableaux VIII.1 et 2.

L'algorithme programmé de la référence [11] traitant la méthode de Balas [2] a été testé sur un IBM 7090. Nous ne tenons pas compte de la configuration de cet ordinateur dont le temps de cycle de base est supérieur à celui du CDC 6600. Le nombre d'itérations des algorithmes exploratoires n'a pas tout à fait la même signification que dans les algorithmes algébriques. C'est pour cette raison que la référence [11] ne les mentionne pas. En conséquence, pour toute comparaison, nous ne prenons en considération que le temps de Processor central qui caractérise plus ou moins la convergence de l'algorithme considéré.

Le tableau VIII.4 montre que le code basé sur l'algorithme de Geoffrian paraît plus performant et plus efficace que les autres codes exploratoires et notamment du code basé sur le filtre de Balas. La méthode d'Hervé, comme celle de Geoffrian a résolu tous les exemples de la référence [11].

Le Balas filtre apparaît de plus en plus inefficace quand la taille du problème augmente, ce qui, d'ailleurs, n'est pas le cas pour le code Balas de la référence [11] où le temps de processor central diminue quand le nombre de contraintes augmente pour le même nombre de variables. Par ailleurs, notons que le temps de résolution croît avec le nombre de variables pour le même nombre de contraintes [11].

Une fois de plus il n'est pas possible de tirer une conclusion générale quant à la relation temps Processor central-taille de problèmes. Les algorithmes de Gomory ne sembleraient pas dépendre de la taille du problème. Ainsi l'exemple n°4 de la référence [11] de taille (6*12) a été résolu sans peine par tous les algorithmes exploratoires. Par contre la convergence a été trés lente pour les algorithmes de Gomory. Même le code Tokosi 2 qui s'est révélé le plus performant (chapitre VII) des autres, n'a pas convergé après 9000 itérations et 9 minutes d'unité centrale. Quant au code Tokosi 3 le nombre d'itérations a été prohibitif, plus de 20 000 itérations. Il semblerait que ces difficultés sont dues uniquement à la forme du polyèdre des solutions réalisables. En effet l'acheminement vers la solution entière est assez lent et les troncatures du polyèdre ne sont pas

assez "profondes" ; il a fallu imposer 12 contraintes supplémentaires du type $x_j \leqslant 1$; j=1,12 pour que le code TOKOSI 2 converge en 5.3 secondes et après 209 itérations dont 51 troncatures du polyèdre des solutions réalisables.

Un contre exemple est fourni par le n°11 de la référence [11], exemple tiré de "Programmes linéaires mixtes" de F. Lambert. Il a été résolu sans peine par la méthode de Gomory (15 itérations dont 5 troncatures et en 0.25 seconde d'unité centrale). Il a fallu 90 secondes pour le résoudre par la méthode de FAURE et MALGRANGE programmée et testée dans la référence [34]. L'algorithme de Balas selon la référence [11] s'est révélé inefficace quant à la résolution de cet exemple, le calcul a été abandonné après 40 minutes d'unité centrale. La taille de l'exemple en variables bivalentes a été ramenée de (6×10) à (6×60) ; la borne des variables étant 2^6.

Aussi, pour les algorithmes de Gomory, lorsque l'exemple est totalement bivalent, il a été souvent nécessaire d'introduire des contraintes du type $x_j \leqslant 1$, j=1,N. Cette opération se justifie pour deux raisons :

- soit que la solution entière contient des variables bivalentes
- soit que la convergence est assez lente.

Les derniers exemples de la référence [11] n'ont pu être résolu sans limitation du domaine des solutions réalisables par les contraintes $x_j \leqslant 1$, j=1,n par le code TOKOSI 3 basé sur le deuxième algorithme de Gomory [23]. Ainsi pour l'exemple 20 de la référence [11] après 2000 itérations et 56 secondes d'unité centrale, la fonction économique dont l'optimum vaut 47, a atteint la valeur 6.

Quant aux codes basés sur l'algorithme n°1 de Gomory [22], ces exemples n'ont pu être résolus par suite de dépassement de capacité de certaines mémoires dès les premières itérations.

N°	N	M	d	δ	Z_E	δ_f	Nit	Nb Tr	temps	Codes	x																										
											1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
1	5	3	.933	1	17	1	4	4	.239	TOKOSI 3	0	1	1	0	0																						
2	10	7	.54	1	3	.8		2	.345	"	0	0	1	0	1	0	0	0	0	0																	
3	9	4	.722	1	∤ Sol																																		
4	12	18	.268	1	13		209	51	5.283	TOKOSI 2	0	0	1	1	0	0	0	1	0	1	0	1															
5	10	1	1.	1	85	.6	8	8	.732	TOKOSI 3	0	0	0	1	0	0	0	0	0	1																	
6	2	3	1.	1	8			4	.071	TOKOSI 3	2	2																									
7	3	3	.777	1	11		2	2	.060	"	1	1	2																								
8	3	3	1.	1	52	1	4	4	.192	TOKOSI 3	1	0	2																								
9	4	4	.75	.5	186	.50	76	76	2.420	"	22	41	1	1																							
10	5	6	.6	.4	7	.2	20	20	.913	"	3	2	1	1	1																						
11	10	6	1.	1.	1510	.8	15	5	.243	TOKOSI 2	0	0	99	246	40	0	0	0	1	0																	
12	10	10	.82	1	5	.9	3	3	.547	TOKOSI 3	0	0	1	0	0	0	1	0	0	0																	
13	15	30	.468	1	26						0	1	1	0	0	0	1	1	1	0	0	1	0	0	1												
14	20	10	.895	1	2	.95	2	2	.886	TOKOSI 3	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	1	0	0							
15	20	20	.85	.85							1	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1											
16	20	23	1.	1.	47						0	0	1	1	1	0	0	1	1	0	0	0	0	0	0	1	1	1	0	1							
17	20	23	.876	1.	53						1	0	1	1	0	0	1	1	0	0	0	1	1	0	0	0	1	1	0	1							
18	20	23	.87	1	∤ Sol																																
19	20	25	.88	1.	47						0	1	0	1	1	1	1	1	0	0	0	1	1	0	0	0	1	0	0	1							
20	20	27	.88	1.	47						0	1	0	1	1	1	1	1	0	0	0	1	1	0	0	0	1	0	0	1							
21	23	20	.88	1.							0	0	1	1	0	0	0	0	0	0	0	1	0	0	1	0	1	0	1	0	0						
22	25	20	.88	1	33						0	0	1	0	1	0	0	0	1	0	0	0	0	0	1	0	1	0	0	1	0	1	0	0			
23	27	20	.88	1.	34						0	0	1	0	1	0	0	0	1	0	0	0	0	0	1	0	1	0	0	1	0	1	0	0	0	1	
24	28	20	.88	1.	38						0	0	1	0	1	0	0	0	1	0	0	0	0	0	1	0	1	0	0	1	0	1	0	0	0	0	1
25	30	20																																			

Tableau VIII.3 Exemples extraits de la référence [11]

Codes	IBM 7090		C D C 6 6 0 0							
	BALAS		TOKOSI		BALAS Filtre		GEOFFRIAN		HERVE	
N°	γ	temps	γ	temps	γ	temps	γ	temps	γ	temps
1		1.	4	.239	3	3.	5	.6	.7	5.
2		1.	2	.375		4.	9	.7	5	3.
4		1.	209	5.283		13.	91	.9	15	5.
5		1.	8	.732		3.	37	.6	35	5.
6		1.	4	.71	-	-	-	-	-	-
7		1.	2	.060	-	-	-	-	-	-
8		1.	4	.192	-	-	-	-	-	-
9		255.	76	2.420	-	-	-	-	-	-
10		1.	20	.913	-	-	-	-	-	-
11		2400*	15	.243	-	-	-	-	-	-
12		5.	3	.547	25	3.	9	.7	11	4.
13		1.				50.	765	6.	23	13.
14		1.	2	.886		5.	3	.7	5	4
15		28.				35.	1909	29.	13	24.
16		124.			3.327	26.	9047	128.	143	523.
17		51.			4215	370.	3285	48.	63	320.
19		23.			52	43.	1195	22.	23	90.
20		16.			502	42.	-	22.	21	90.
21		216.			10000+	-	15297	247.	397	48
22		196.			10985	-	13107	230	341	40.
23		426.			10867	-	21583	432	429	56.
24		912.			10846	-	18923	420	101	480.
25		960*			10935	-			-	-

Tableau VIII.4 Comparaison entre codes - Exemples de la référence [11]

VIII.2.4 Exemples de Graphes :

Il s'agit de quatre exemples numériques de tailles différentes appartenant au domaine des graphes. Les problèmes traités dans ce paragraphe sont relatifs à la recherche du nombre chromatique d'un graphe donné [8], [17].

G = (X,U) est un graphe donné :

- X ensemble des sommets X_i
- U ensemble des arcs (X_i,X_j) joignant les sommets X_i et X_j.

Le problème revient à trouver des entiers $e_i \in N$ tels que

$$X_i = e_i \quad \text{si} \quad X_i \in X \qquad \text{(VIII.1)}$$

et $$s \leq e_i \leq \vartheta \qquad \text{(VIII.2)}$$

$$(X_j,X_k) \in u \Longrightarrow e_j \neq e_k \qquad \text{(VIII.3)}$$

et rendant minimum ϑ.

La contrainte (VIII.3) peut s'écrire sous la forme $|X_j - X_k| \geq 1$.

Désormais nous confondons X_i et e_i.

Une autre formulation plus pratique à manier en calculs automatiques est donnée dans la référence [17] :

Soit n le nombre de sommets X_i du graphe G et m l'ensemble des arcs (X_i,X_j) de G.

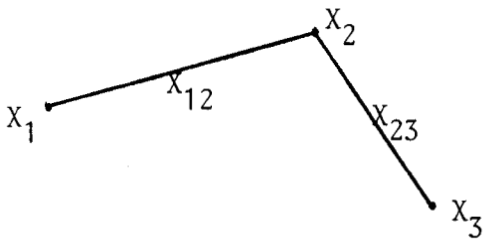
Posons $(X_i,X_j) = X_{ij}$. Les variables X_{ij} sont des variables bivalentes dans la formulation suivante :

Pour tout $(X_j,X_k) \in U$ on a :

$$\left.\begin{array}{l} \left\{\begin{array}{l} X_j - X_k \geq 1 - n\,X_{jk} \\ X_k - X_j \geq 1 - n(1-X_{jk}) \end{array}\right. \\ X_j \leq \vartheta \qquad j=1,n \\ X_{jk} \leq 1 \\ \min \vartheta \end{array}\right\} \text{ soit P } \left\{\begin{array}{l} \min \vartheta \\ X_j \leq \vartheta \qquad j=1,n \\ X_{jk} \leq 1 \\ X_j - X_k + n\,X_{jk} \geq 1 \quad \forall (X_j,X_k) \in U \\ -X_j + X_k - n X_{ik} \geq 1-n \end{array}\right.$$

Exemple n°1 :

$G_1 \begin{cases} n = 3 \\ m = 2 \end{cases}$



Le programme correspondant à G_1 est le suivant

$$\min \partial$$

$$\begin{cases} -X_1 + X_2 + 3X_{12} \geq 1 \\ +X_1 - X_2 - 3X_{12} \geq -2 \\ -X_2 + X_3 + 3X_{23} \geq 1 \\ X_2 - X_3 - 3X_{23} \geq -2 \\ X_1 - \partial \leq 0 \\ X_2 - \partial \\ X_3 - \partial \leq 0 \\ X_{12} \leq 1 \\ X_{23} \leq 1 \end{cases}$$

On se ramène à un programme linéaire en nombres entiers de 6 variables et 9 contraintes.

Une solution unique est la suivante :

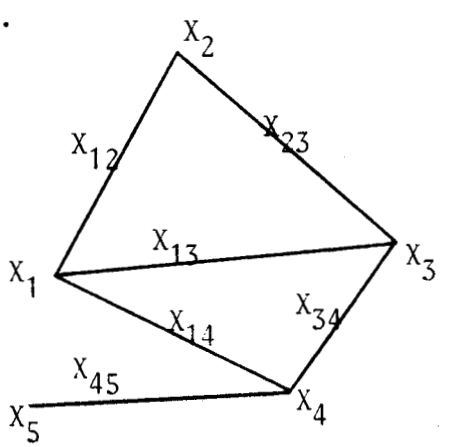
$\partial=1$, $X_1=0$, $X_2=1$ et $X_3=0$, $X_{12}=0$, $X_{23}=1$

Exemple N°2 :

Il s'agit d'un graphe de 5 sommets et 6 arcs.

$G_2 = \begin{cases} n = 5 \\ m = 6 \end{cases}$

La formulation est analogue à la précédente. On arrive à un programme de 12 variables et 23 contraintes.



Une solution entière optimale est :

$\partial = 2$

$X_1=2,\ X_2=0,\ X_3=1,\ X_4=0,\ X_5=1,\ X_{12}=1$

$X_{23}=0,\ X_{13}=1,\ X_{14}=1,\ X_{45}=1,\ X_{34}=0$

Exemple n°3 :

C'est un graphe de 7 sommets de 11 arcs :

$G_3 \begin{cases} n = 7 \\ m = 11 \end{cases}$

Le programme correspondant comprend : 19 variables et 40 contraintes.

Une solution étant :

$\partial = 2$

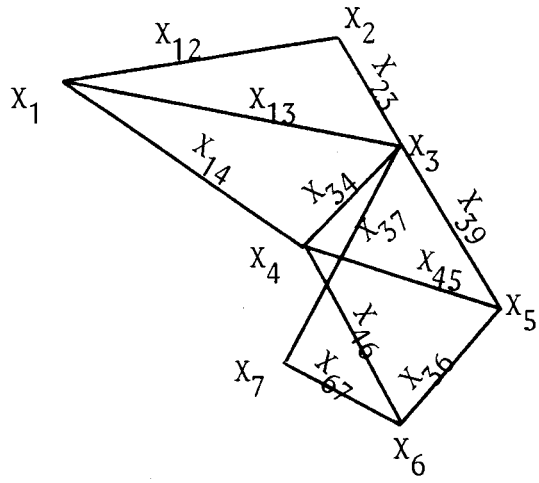
$X_1=0,\ X_2=2,\ X_3=1,\ X_4=2$

$X_5=0,\ X_6=1,\ X_7=0$

$X_{12}=0,\ X_{23}=1,\ X_{13}=0,\ X_{14}=0$

$X_{34}=0,\ X_{35}=1,\ X_{37}=1,\ X_{45}=1$

$X_{46}=0,\ X_{56}=0,\ X_{67}=1$



Exemple N°4 :

C'est un graphe de 6 sommets et 9 arcs.

Le programme linéaire correspondant fait 16 variables et 33 contraintes :

Une solution entière optimale est :

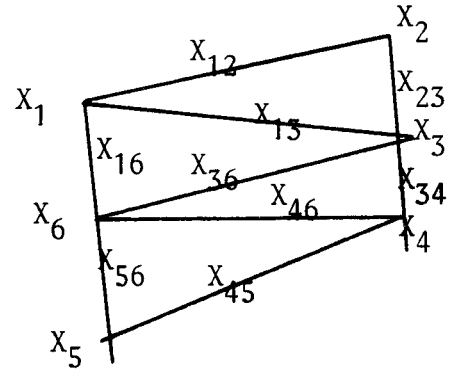
$\partial = 2$

$X_1=2,\ X_2=1,\ X_3=0$

$X_4=2,\ X_5=0,\ X_6=1$

$X_{12}=1,\ X_{13}=1,\ X_{16}=1,\ X_{23}=1,\ X_{34}=1$

$X_{36}=1,\ X_{45}=1,\ X_{46}=1,\ X_{56}=0$



CODES	C D C 6 6 0 0									
	TOKOSI 1		TOKOSI 2		HERVE		GEOFFRIAN		BALAS	
N°	γ	temps	γ	temps	γ	temps	γ	temps	γ	temps
G_1	17	.018	17	.211	11	6		2.	51	5.
G_2	90	2.996	88	2.661	83	586	4453	66.	884	20.
G_3	-	-	182	428	83	1200	-	-		240.
G_4	180	7.643	154	1037	117	340	-	-		35.

Tableau VIII.5

L'examen du tableau (VIII.5) montre que les temps mis par le code basé sur la méthode d'HERVE se chiffrent en minutes. Néanmoins c'est le seul des trois codes à base exploratoire qui a permis la résolution de l'exemple G_3. Les difficultés rencontrées par les codes TOKOSI lors de la résolution de l'exemple G_3 sont du type dépassement de capacité de certaines mémoires dès les premières itérations.

VIII.3 Conclusion générale :

Nous venons de tester les algorithmes de Gomory [22], [23] sur quelques quatre vingt exemples aussi variés que possible par la taille et l'origine. Les difficultés que nous avons rencontrées sont plutôt d'ordre technique : le dépassement de capacité de certaines mémoires notamment quand les calculs sont effectués en fixe. Ce danger n'est pas totalement écarté quand les calculs sont effectués en virgule flottante ; des calculs qui nécessitent beaucoup de précaution permettent d'éliminer tout pivotage autour d'un élément dont l'ordre de grandeur est assez petit. Dans les deux modes de calculs, on se trouve devant des obstacles identiques. Cependant les calculs en mode fixe semblent avoir un léger avantage sur l'autre mode de traitement ; en effet pour ce dernier, quel que soit le seuil d'erreur on est parfois conduit à accepter comme entier un nombre rationnel et à rejeter comme rationnel un entier.

Les exemples numériques qui viennent d'être traités ne permettent pas de donner des conclusions précises quant à l'efficacité des algorithmes de Gomory ou des algorithmes exploratoires. La comparaison entre les deux types d'algo-

rithmes paraît très difficile si l'on se base sur les quelques exemples traités, d'ailleurs les arguments et les bases de comparaison restent mal déterminés. Les algorithmes exploratoires se heurtent, sur le plan pratique, aux problèmes d'encombrement de mémoire (dépassement de capacité de la mémoire entière des calculateurs) et au facteur temps d'investigation des solutions réalisables entières.

Quant aux algorithmes de Gomory les difficultés sont dans l'ensemble différentes des précédentes. Les algorithmes dépendant notamment des règles de choix, des lignes génératrices de contraintes de troncature. La difficulté réside dans le choix de la règle convenable qui garantit la convergence de l'algorithme quels que soient la taille et le type du problème traité. Les résultats numériques précédents ont permis néanmoins de constater l'efficacité relative de la règle de choix du code TOKOSI 2 (Chapitre VI). Le code utilisant cette règle a permis la résolution de tous les programmes résolus par les autres codes TOKOSI et d'un grand nombre de ceux qui n'ont pu être résolus par ces derniers. La seule difficulté rencontrée par TOKOSI 2 dans la résolution de certains exemples est le dépassement de capacité de quelques mémoires. La convergence du code TOKOSI 2 paraît plus rapide que les autres mais il est à remarquer que dès que le nombre d'itérations dépasse 2000 il y a peu de chance d'atteindre la solution entière. Nous avons vu (§ VIII.2.2) quelques raisons de cette lenteur de convergence ; afin de palier à cette lenteur une amélioration technique dans la construction de la contrainte de Gomory a été tentée par la méthode de GLENNT MARTIN [43] . Les résultats obtenus quant à l'accélération de la convergence de l'algorithme n°1 de Gomory [22] par cette méthode sont encourageants,mais nous n'en savons rien quand les exemples sont de taille importante.

Notons au passage que les algorithmes exploratoires permettent d'avoir à toute "itération" une solution réalisable entière. Par contre le premier algorithme de Gomory [22] (le seul d'ailleurs) permet d'avoir une solution réalisable et optimale mais mixte à toute itération. En outre les algorithmes de Gomory [22] [23], [24] comme l'algorithme simplicial permettent d'avoir les valeurs des variables duales qui, selon les problèmes traités, sont intéressantes pour toute étude duale ou évaluation marginale [5].

A N N E X E

CODIFICATIONS ORGANIGRAMMES ET PROGRAMMATION

Dans cette section nous donnerons les modalités d'utilisation des codes TOKOSI.

Afin de faciliter l'exploitation de ces codes nous donnerons :

- Dans une première partie les différentes codifications des données et des résultats numériques.

- Dans une deuxième partie les principaux organigrammes des codes TOKOSI.

- Dans une troisième et dernière partie les textes symboliques des codes TOKOSI suivis de quelques exemples traités et codifiés.

PREMIERE PARTIE

Codification des données et des résultats numériques

A. Données principales des codes TOKOSI

L'introduction des données se fait de la même manière et suivant le même format pour tous les codes TOKOSI.

Les données comprennent d'une part les nombres de contraintes et de variables et d'autre part le tableau simplicial introduit ligne par ligne. La première ligne du tableau simplicial est formée par les coefficients de la fonction économique tandis que la première colonne représente le vecteur second membre.

Le tableau simplicial correspondant au programme mis sous la forme suivante :

$$\text{Min } fX$$
$$\text{tel que } AX \leqslant B$$

se présente ainsi

$$\begin{bmatrix} 0 & f \\ B & A \end{bmatrix}$$

Le format de lecture des données est (10I6). Les éléments du tableau simplicial sont supposés entiers.

B. Codification des variables

Toute variable active X_j est codée par le code (-J). Toute variable secondaire (ou d'écart) Y_j est représentée par le terme $10 \times J$. La fonction économique est codée 1111. Chaque variable d'écart associée à une contrainte additionnelle d'ordre k est représentée par le code correspondant à la valeur 1000+k.

Cette codification correspond aux programmes dont la somme des nombres de variables et de contraintes est inférieure à 100 et dont le nombre de variables est inférieur à 50.

Les exemples traités à la fin de l'annexe schématisent cette procédure de codification.

Les résultats numériques comportent :

- la liste des variables de base et leur valeur correspondante
- la liste des variables hors base
- les coefficients de substitution du vecteur ligne de la fonction économique
- la valeur de la fonction économique
- le nombre total d'itérations
- le nombre de troncations
- les temps d'exécution en secondes (Central Processor et Périphérique).

L'obtention à chaque itération de la valeur de la fonction économique ainsi que les noms de la variable quittant la base et de la variable entrant dans la base sont optionnelles.

. Liste des symboles utilisés dans les codes TOKOSI

M : nombre de contraintes

N : nombre de variables

IA : Tableau simplicial dimensionné à 100x50

IFA : Tableau de mêmes dimensions que IA destiné aux parties fractionnaires

IVLI : Vecteur liste des variables de base

JUCOL : Vecteur liste des variables hors base

IAIGUI : indicateur aiguillage :
IAIGUI = 1 programme dual réalisable
IAIGUI = 2 programme primal réalisable

DTP : durée en secondes d'utilisation des périphériques

DTC : durée d'exécution en secondes (temps Central Processor)

ID : pivot de l'avant-dernière itération

IS : rang de la colonne pivot

IR : rang de ligne pivot

DEUXIEME PARTIE

. TOKOSI 1

Le code TOKOSI 1 comprend 4 modules :

- TRANSFO
- PRIMPIV
- DUALEX 2
- MODTAB.

Le programme principal :

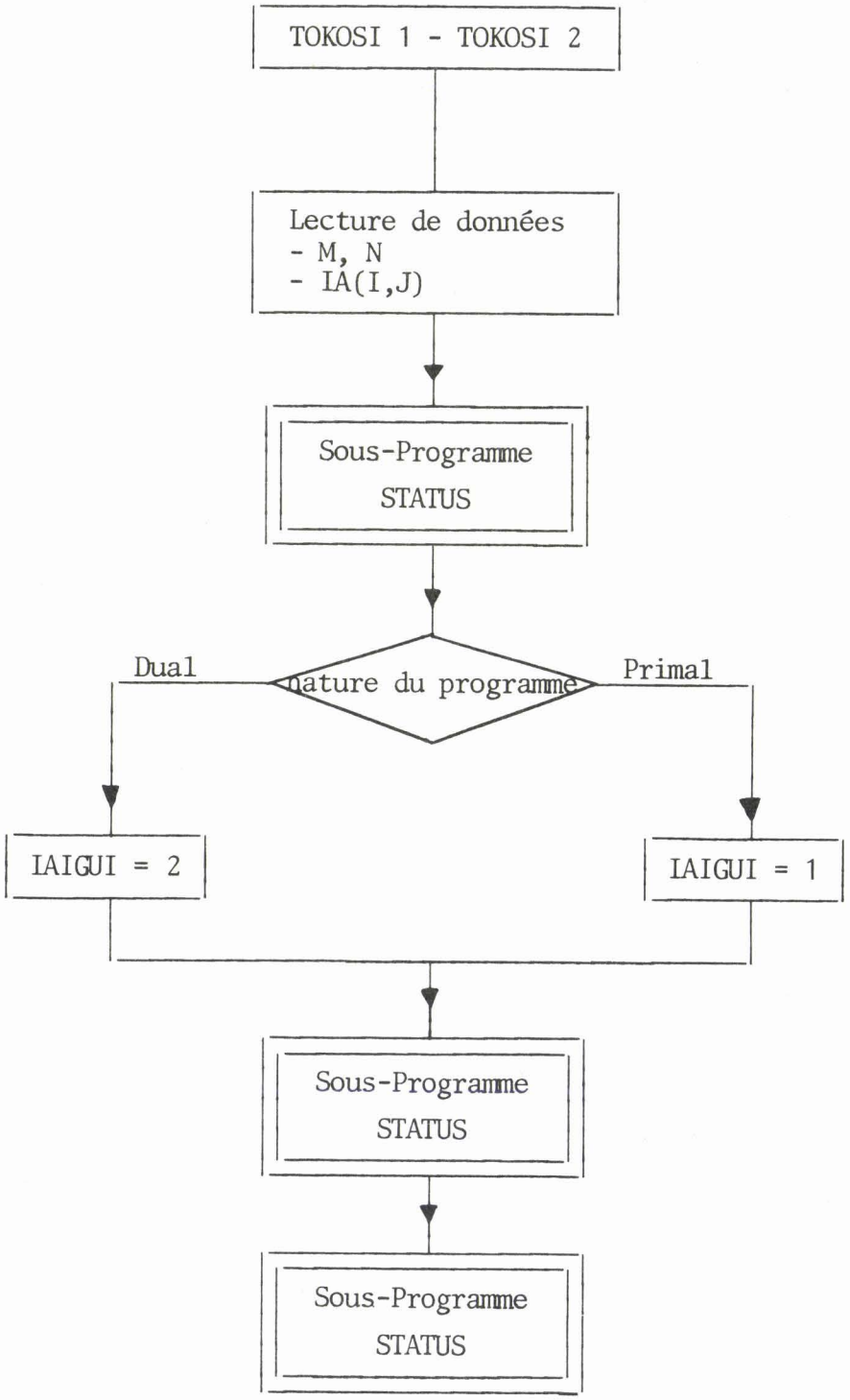
- permet la lecture des données
- détermine le type de programme (dual ou primal)
- attribue une valeur 1 ou 2 à l'indicateur d'aiguillage IAIGUI
- enregistre les temps de début et de fin d'exécution (les temps sont fournis par le programme système STATUS.

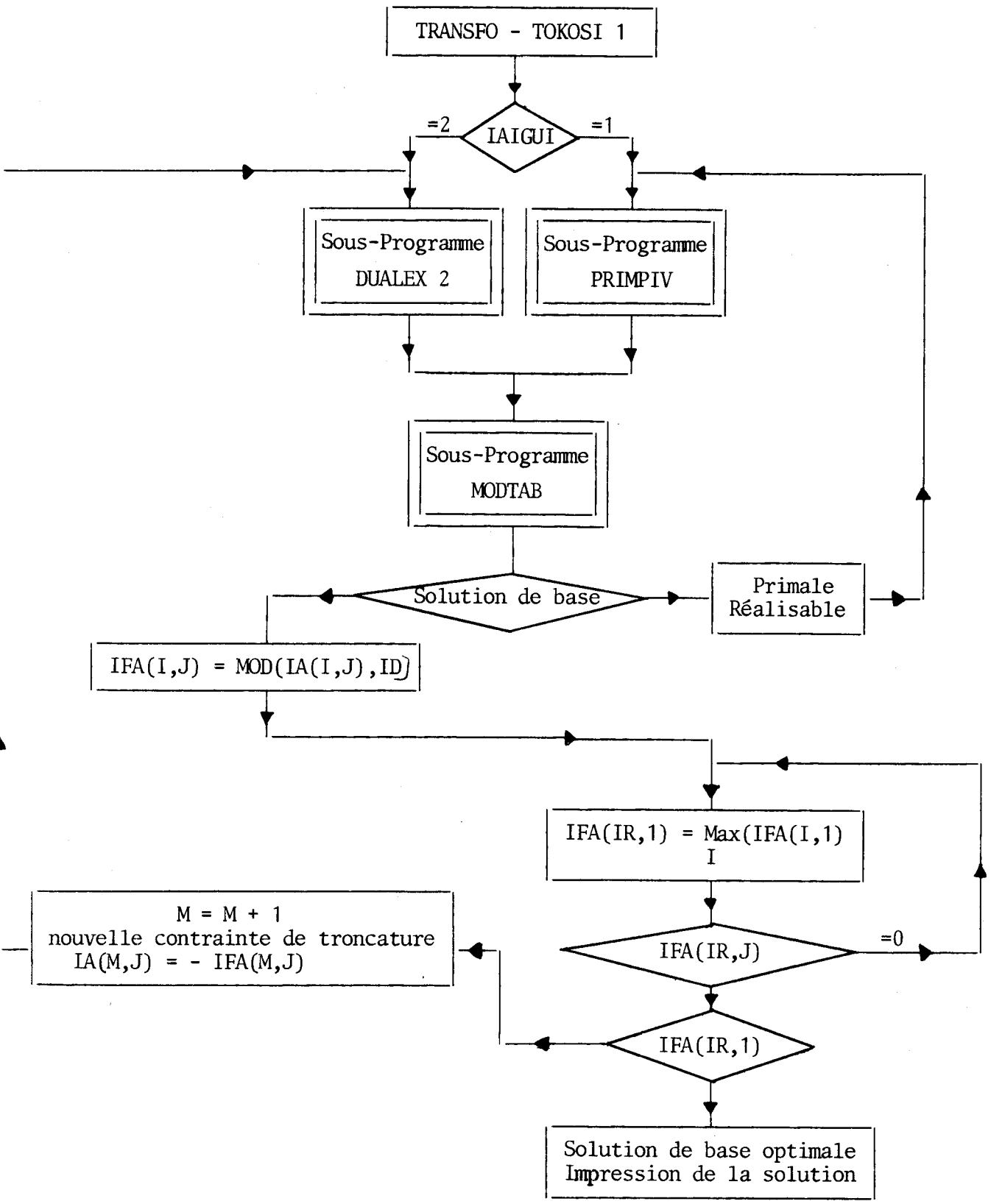
Le sous-programme TRANSFO traduit la règle de choix utilisée (règle n°1, § IV.4.1).

Lorsque le programme est primal, le sous-programme PRIMPIV permet la détermination de la colonne et de la ligne du pivot.

Le sous-programme DUALEX 2 détermine la colonne inflexicographique du tableau simplicial.

Le sous-programme MODTAB assure la transformation du tableau simplicial après chaque itération.

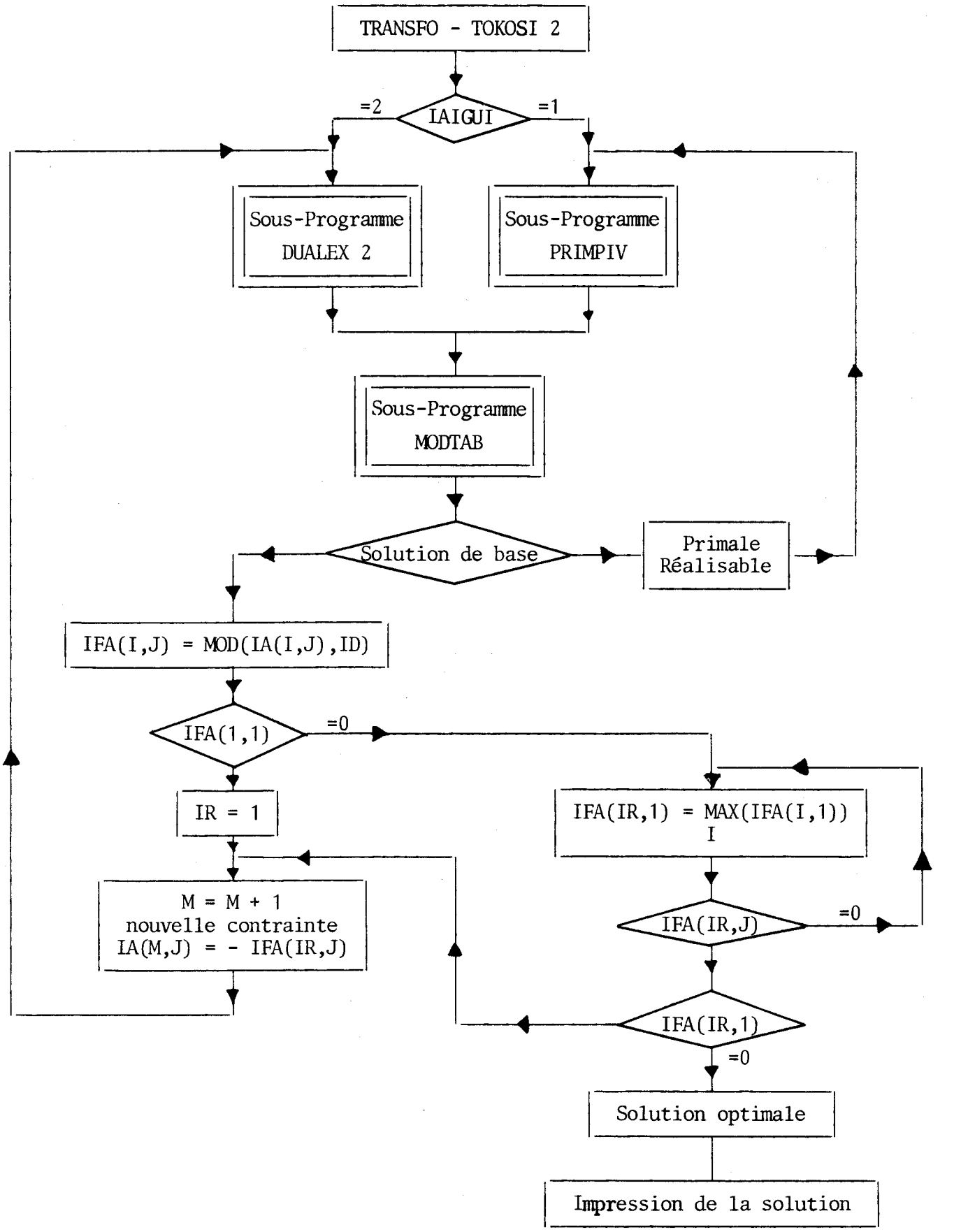

TOKOSI 1 - TOKOSI 2
Lecture de données
- M, N
- IA(I,J)
Sous-Programme
STATUS
Dual
Nature du programme
Primal
IAIGUI = 2
IAIGUI = 1
Sous-Programme
STATUS
Sous-Programme
STATUS


TRANSFO - TOKOSI 1
IAIGUI
=2
=1
Sous-Programme DUALEX 2
Sous-Programme PRIMPIV
Sous-Programme MODTAB
Solution de base
Primale Réalisable
IFA(I,J) = MOD(IA(I,J),ID)
IFA(IR,1) = Max(IFA(I,1)
I
IFA(IR,J)
=0
M = M + 1
nouvelle contrainte de troncature
IA(M,J) = - IFA(M,J)
IFA(IR,1)
Solution de base optimale
Impression de la solution

B. TOKOSI 2

Le code de TOKOSI 2 ne diffère de TOKOSI 1 que par le critère de choix de la règle de choix de la ligne génératrice de la contrainte de troncature. La règle de choix de TOKOSI 2 étant la règle n° 6 (§ IV.4.5).

Dans ce qui suit, nous ne donnerons pas un organigramme complet de TOKOSI 2,nous ne mentionnerons que l'organigramme du module TRANSFO.


TRANSFO - TOKOSI 2
IAIGUI
=2
=1
Sous-Programme DUALEX 2
Sous-Programme PRIMPIV
Sous-Programme MODTAB
Solution de base
Primale Réalisable
IFA(I,J) = MOD(IA(I,J),ID)
IFA(1,1)
=0
IR = 1
M = M + 1
nouvelle contrainte
IA(M,J) = - IFA(IR,J)
IFA(IR,1) = MAX(IFA(I,1))
I
IFA(IR,J)
=0
IFA(IR,1)
=0
Solution optimale
Impression de la solution

C. TOKOSI 3

Le code TOKOSI 3 ne concerne que les programmes dual-réalisables.

Le programme principal comprend 2 modules :

DUALEX 2
MODIF 1

Le module DUALEX 2 permet la détermination de la colonne inf-lexicographique et de la ligne génératrice de contrainte additionnelle. La règle de choix étant celle du "maximum lexicographique" (§ III.2.4).

Le module MODIF 1 permet la détermination du diviseur réel λ (chapitre II).

```
      PROGRAM TOKOSI2(INPUT,OUTPUT)
C     RESOLUTION DE PROGRAMMES LINEAIRES EN NOMBRES ENTIERS SUIVANT
C     LA PREMIERE METHODE DE GONORY
C
C
C
C     VERSION II
      COMMON IA(100,50),IFA(100,50)
      DIMENSION JVCOL(100),IVLI(100),IST(9)
C
      IC=0
      READ 2,ND
C
 1000 READ 2,M,N
      M=M+1
      N=N+1
      READ 2 ,((IA(I,J),J=1,N),I=1,M)
    2 FORMAT(10I6)
      CALL STATUS(IST)
      TEMDEBC=IST(1)
      TEMDEBP=IST(2)
      DO 3 I=1,M
      DO 3 J=1,N
    3 IFA(I,J)=0
C
      JVCOL(1)=11111
      DO 4 I=2,100
    4 JVCOL(I)=10*(I-1)
C
      DO 6 J=1,100
      IVLI(J)=1-J
    6 IF(J-N.GT.0)IVLI(J)=0
C
      PRINT 9
    9 FORMAT(1H1,36X,17HTABLEAU DE DEPART,//)
C
      PRINT 19,(IVLI(J),J=1,N)
   19 FORMAT(7X,18(I5,1X))
   10 FORMAT(1X,20(I5,1X))
      DO 11 I=1,M
   11 PRINT 10,JVCOL(I),(IA(I,J),J=1,N)
      I1=1
C
      DO 12 J=2,N
      IF(IA(1,J).GE.0)I1=I1+1
   12 CONTINUE
C
      IF(I1-N.EQ.0)GOTO 14
      PRINT 18
   18 FORMAT(///,37X,26HPROBLEME PRIMAL REALISABLE,//)
      IAIGUI=1
      GOTO 17
   14 PRINT 15
```

```
      15 FORMAT(///,37X,24HPROBLEME DUAL REALISABLE,///)
55       IAIGUI=2
C
      17 CALL TRANSFO(JVCOL,IVLI,M,N,IAIGUI)
       1 IC=IC+1
    1080 CALL STATUS(IST)
60       DTP=(IST(2)-TEMDEBP)/1000.
         DTC=(IST(1)-TEMDEBC)/1000.
         PRINT 1011,DTC,DTP
    1011 FORMAT(///,1X,17HTEMPS D EXECUTION,10X,3HU.C,F8.3,3HSEC,10X,3HU.P,
        1F8.3,3HSEC,//)
65       IF(IC-ND.NE.0)GOTO 1000
         STOP
         END
```

```
      SUBROUTINE TRANSFO(JVCOL,IVLI,M,N,IAIGUI)
      COMMON IA(100,50),IFA(100,50)
      DIMENSION JVCOL(1),IVLI(1)
C
      ID=1
      NNR=N-1
      IK=1
      K=0
      PRINT 1
    1 FORMAT(///,8X,5HF.ECO,6X,6H PIVOT,6X
     1,2HIR,7X,2HIS,7X,15HVARIABLES BASE,6X,16HVARIABLES H.BASE,7X
     2,10H ITERATION,//)
      GOTO(200,100)IAIGUI
  200 CALL PRIMPIV(IS,N,M,IR,IPIVOT)
      GOTO 250
  100 CALL DUALEX2(NNR,M,IS,IR,IPIVOT)
  250 CALL MODTAB(M,N,IR,IS,ID,IPIVOT)
C
      II=JVCOL(IR)
      JVCOL(IR)=IVLI(IS)
      IVLI(IS)=II
      PRINT 2,IA(1,1),IPIVOT,IR,IS,JVCOL(IR),IVLI(IS),IK
      IK=IK+1
  101 CONTINUE
    2 FORMAT(I12,1X,I12,3(1X,I8),2(15X,I3))
      NT=0
    3 FORMAT(16(1X,I3))
      DO 7 J=2,N
      IF(IA(1,J).LT.0)GOTO 200
    7 CONTINUE
      DO 18 I=2,M
      IF(JVCOL(I).LE.1000)GOTO 18
      IF(IA(I,1).LT.0)GOTO 18
      M=M-1
      I1=I
      DO 19 IS=I1,M
      JVCOL(IS)=JVCOL(IS+1)
      DO 20 J=1,N
   20 IA(IS,J)=IA(IS+1,J)
   19 CONTINUE
   18 CONTINUE
      DO 4 I=2,M
      IF(IA(I,1).GE.0)GOTO 4
      GOTO 100
    4 CONTINUE
C
      IF(IABS(ID)-1.EQ.0)GOTO 6
    9 FORMAT(//,31X,21HITERATION PHASE DUALE,//)
      K=K+1
      ID1=IABS(ID)
      DO 8 I=1,M
      DO 8 J=1,N
```

```
      IFA(I,J)=MOD(IA(I,J),ID1)
      IF(IFA(I,J).LT.0)IFA(I,J)=IFA(I,J)+ID1
    8 CONTINUE
      IR=1
      NK=-1
      DO 10 I=1,M
      IF(IFA(I,1).LT.NK)GOTO 10
      DO 103 J=2,N
      IF(IFA(I,J).EQ.0)GOTO 103
      GOTO 106
  103 CONTINUE
      GOTO 10
  106 CONTINUE
      NK=IFA(I,1)
      IR=I
   10 CONTINUE
      IF(NK.GT.0)GOTO 16
      DO 17 I=1,M
   17 IA(I,1)=IA(I,1)/ID1
      GO TO  6
   16 DO 11 J=1,N
   11 IA(M+1,J)=-IFA(IR,J)
      M=M+1
      JVCOL(M)=1000+K
      GOTO 100
    6 PRINT 12
   12 FORMAT(//,36X,17HSOLUTION OPTIMALE,///)
C
   13 FORMAT(5X,2HZ=,I10,//)
      PRINT 3,NT,(IVLI(I),I=1,N)
      PRINT 3,JVCOL(1),(IA(1,J),J=1,N)
      PRINT 13,IA(1,1)
      DO 15 I=2,M
   15 PRINT 14,JVCOL(I),IA(I,1)
   14 FORMAT(5X,I10,5X,I10)
  102 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE MODTAB(M,N,IR,IS,ID,IPIVOT)
      COMMON IA(100,50),IFA(100,50)
C
C
      MM=M
      ISI=IA(IR,IS)/IABS(IA(IR,IS))
      DO 1 I=1,MM
      IF(I.EQ.IR)GOTO 1
      DO 4 J=1,N
      IF(J.EQ.IS)GOTO 4
      IQIJ=IA(I,J)/ID
      IRIJ=IA(I,J)-ID*IQIJ
      IQI0=IA(I,IS)/ID
      IRI0=IA(I,IS)-IQI0*ID
      IQ0J=IA(IR,J)/ID
      IR0J=IA(IR,J)-ID*IQ0J
      IQ00=IPIVOT/ID
      IR00=IPIVOT-ID*IQ00
      IT1=ID*(IQIJ*IQ00-IQ0J*IQI0)
      IT2=IQIJ*IR00-IQ0J*IRI0+IQ00*IRIJ-IQI0*IR0J
      IT3=(IRIJ*IR00-IR0J*IRI0)/ID
      IA(I,J)=ISI*(IT1+IT2+IT3)
    4 CONTINUE
    1 CONTINUE
C
      DO 2 I=1,MM
      IF(I-IR.EQ.0)GOTO 2
      IA(I,IS)=-IA(I,IS)*ISI
    2 CONTINUE
C
      DO 3 J=1,N
      IF(J-IS.EQ.0)GOTO 3
      IA(IR,J)=IA(IR,J)*ISI
    3 CONTINUE
C
      IA(IR,IS)=ID*ISI
      ID=IABS(IPIVOT)
      RETURN
      END
```

```
      SUBROUTINE PRIMPIV(IS,N,M,IR,IPIVOT)
      COMMON IA(100,50),IFA(100,50)
       DIMENSION INTER(50),JRANCOL(50)
      JRANCOL(1)=0
      INTER(1)=0
      DO 4 I=1,49
      INTER(I+1)=0
      JRANCOL(I+1)=0
    4 IF(I.LT.N)INTER(I+1)=IA(1,I+1)
   10 FORMAT(20(1X,I5))
      DO 5 I=2,N
      ND=0
      DO 2 J=2,N
      IF(INTER(J).GE.0)GOTO 2
      NK=MIN0(ND,INTER(J))
      IF(ND-NK.EQ.0)GOTO 2
      ND=NK
      IS=J
    2 CONTINUE
      INTER(IS)=0
    5 JRANCOL(I)=IS
      DO 6 J=2,N
      IS=JRANCOL(J)
      IF(IS.EQ.0)GOTO 6
      PIVOT=1000000.
      DO 1 I=2,M
      IF(IA(I,1).LE.0)GO TO 1
      IF(IA(I,IS).LE.0)GOTO 1
      C=FLOAT(IA(I,1))/FLOAT(IA(I,IS))
      IF(PIVOT-C.LT.0.)GOTO 1
      PIVOT=C
      IR=I
    1 CONTINUE
      IF(IR.NE.0)GOTO 9
    6 CONTINUE
    3 FORMAT(20H PROBLEME IMPOSSIBLE)
      PRINT 3
      STOP
    9 IPIVOT=IA(IR,IS)
      RETURN
      END
```

```
      SUBROUTINE DUALEX2(N,M,IS,IR,IPIVOT)
      COMMON IA(100,50),IFA(100,50)
      DIMENSION INTER(50),ILIGNE(90),JRANCOL(50)
      DIMENSION XIND(100),YIND(100),IND(100)
C
      IS=1
      IR=1
      KP=0
      IMD=1
      KAIG=0
      DO 1 I=1,50
      JRANCOL(I)=0
      INTER(I)=0
      IF(I-N.GT.0)GOTO 1
      INTER(I)=IA(1,I+1)
    1 CONTINUE
C
      DO 4 I=1,N
      ND=1000000
      J1=0
      DO 2 J=1,N
      IF(INTER(J).LT.0)GOTO 2
      NB=MIN0(ND,INTER(J))
      IF(NB-ND.NE.0)GOTO 3
      IF(ND-INTER(J).NE.0)GOTO 2
      DO 5 L=2,M
      IF(IA(L,J1+1)-IA(L,J+1))2,5,32
    5 CONTINUE
      PRINT 100,J1,J
  100 FORMAT(1X,20H COLONNES IDENTIQUES,2(1X,I3))
      STOP
   32 IF(IA(L,J1+1)*IA(L,J+1).LT.0.AND.IA(L,J1+1).LT.0)GOTO 2
    3 ND=NB
      J1=J
      NB=0
    2 CONTINUE
      INTER(J1)=-10000
      IF(IA(1,J1+1).EQ.0)KP=KP+1
    4 JRANCOL(I)=J1
      RATIO=FLOAT(N-KP)/FLOAT(N)
   93 FORMAT(/,9H RATIO = ,F5.3,/)
      DO 24 I=2,M
      IF(IA(I,1).GE.0)GOTO 24
      DO 25 J=1,N
      IF(IA(I,J+1).GE.0)GOTO 25
      GOTO 26
   25 CONTINUE
   26 IF(KAIG-IA(I,1).LE.0)GOTO 24
      IR=I
      KAIG=IA(I,1)
   24 CONTINUE
      IF(IR.LE.1)GOTO 200
      GOTO 27
```

```
      300 PRINT 303
      303 FORMAT(20H COLONNES IDENTIQUES)
          STOP
C
       20 K=1
          MM=M-1
          DO 7 I=1,MM
          ILIGNE(K)=0
          IF(IA(I+1,1).GE.0)GOTO 7
          ND=0
          LR=10000
          DO 10 J=1,N
          JK=JRANCOL(J)
          IF(JK.EQ.0)GOTO 10
          IF(IA(I+1,JK+1).GE.0)GOTO 10
          ND=MIN0(J,LR)
          IF(ND-LR.EQ.0)GOTO 10
          LR=ND
          ND=0
       10 CONTINUE
          ILIGNE(K)=(I+1)*1000+LR
        7 K=K+1
C
          DO 11 J=K,50
       11 ILIGNE(J)=0
          ND=0
          DO 13 LI=1,M
          IF(ILIGNE(LI).EQ.0)GOTO 13
          NN=ILIGNE(LI)-(ILIGNE(LI)/1000)*1000
          IF(NN-ND.LT.0)GOTO 13
          IF(ND-NN.NE.0)GOTO 19
          II=ILIGNE(J1)/1000
          III=ILIGNE(LI)/1000
          IF(IABS(IA(II,1))-IABS(IA(III,1)).GE.0)GOTO 13
       19 ND=NN
          J1=LI
       13 CONTINUE
C
          IR=ILIGNE(J1)/1000
          JS=ILIGNE(J1)-IR*1000
          IS=JRANCOL(JS)+1
          IF(IA(IR,IS).LT.0)GOTO 102
       27 DO 12 J=1,N
          XIND(J)=1000000.
          IF(IA(IR,J+1).GE.0)GOTO 12
          XIND(J)=-IA(1,J+1)
          XIND(J)=XIND(J)/IA(IR,J+1)
       12 CONTINUE
          L=1
      103 FORMAT(10(1X,F10.3))
      204 DO 200 J=1,N
          PIVOT=1000000.
          XIND(1)=1.
```

```
      DO 201 I=1,N
      IF(XIND(I).GE.PIVOT)GOTO 201
      IS=I
110   PIVOT=XIND(I)
  201 CONTINUE
      YIND(J)=XIND(IS)
      XIND(IS)=1000000.
      IND(J)=IS+1
115   200 CONTINUE
  104 FORMAT(10(F8.3,1X,I3))
      IF(L.GT.M)GOTO 300
      L=L+1
      IF(YIND(1).NE.YIND(2))GOTO 301
120   I1=IND(1)-1
      XIND(I1)=-IA(L,I1+1)
      XIND(I1)=XIND(I1)/IA(IR,I1+1)
      I2=IND(2)-1
      XIND(I2)=-IA(L,I2+1)
125   XIND(I2)=XIND(I2)/IA(IR,I2+1)
      DO 202 J=3,N
      IF(YIND(J).NE.YIND(J-1))GOTO 204
      IF(YIND(J).LT.0.OR.YIND(J).NE.YIND(J-1))GOTO 204
      I2=IND(J)-1
130   XIND(I2)=-IA(L,I2+1)
      XIND(I2)=XIND(I2)/IA(IR,I2+1)
  202 CONTINUE
      GO TO 204
  301 IS=IND(1)
135   IF(IS.GT.1)GOTO 101
      K=0
   18 K=K+1
      LL=JRANCOL(K)
      IF(IA(IR,LL+1).LT.0)GOTO 15
140   GOTO 18
C
   15 IS=LL+1
  102 IF(IA(IMD,IS).GT.0)GOTO 101
      IMD=IMD+1
145   GOTO 102
  101 IPIVOT=IA(IR,IS)
      RETURN
      END
```

Le programme symbolique du Code TOKOSI 2 est identique à celui du code TOKOSI 1. La seule différence réside dans le critère de la règle de choix de la ligne génératrice de la contrainte de troncature. Cette règle est donnée par le sous-programme TRANSFO selon la version suivante :

```
      SUBROUTINE TRANSFO(JVCOL,IVLI,M,N,IAIGUT)
      COMMON IA(100,50),IFA(100,50)
      DIMENSION JVCOL(1),IVLI(1)
C
05    ID=1
      NNR=N-1
      IK=1
      K=0
      PRINT 1
10    1 FORMAT(///,8X,5HF.ECC,6X,6H PIVOT,6X
     1,2HIR,7X,2HIS,7X,15HVARIABLES BASE,6X,16HVARIABLES H.BASE,7X
     2,10H ITERATION,//)
      GOTO(200,100)IAIGUI
  200 CALL PRIMPIV(IS,N,M,IR,IPIVOT)
15    GOTO 250
  100 CALL DUALEX2(NNR,M,IS,IR,IPIVOT)
  250 CALL MODTAB(M,N,IR,IS,ID,IPIVOT)
C
      II=JVCOL(IR)
20    JVCOL(IR)=IVLI(IS)
      IVLI(IS)=II
      PRINT 2,IA(1,1),IPIVOT,IR,IS,JVCOL(IR),IVLI(IS),IK
      IK=IK+1
  101 CONTINUE
25    2 FORMAT(I12,1X,I12,3(1X,I8),2(15X,I8))
      NT=0
    3 FORMAT(14(1X,I8))
      DO 7 J=2,N
      IF(IA(1,J).LT.0)GOTO 200
30    7 CONTINUE
      DO 18 I=2,M
      IF(JVCOL(I).LE.1000)GOTO 18
      IF(IA(I,1).LT.0)GOTO 18
      M=M-1
35    I1=I
      DO 19 IS=I1,M
      JVCOL(IS)=JVCOL(IS+1)
      DO 20 J=1,N
   20 IA(IS,J)=IA(IS+1,J)
40   19 CONTINUE
   18 CONTINUE
      DO 4 I=2,M
      IF(IA(I,1).GE.0)GOTO 4
      GOTO 100
45    4 CONTINUE
C
      IF(IABS(ID)-1.EQ.0)GOTO 6
    9 FORMAT(//,31X,21HITERATION PHASE DUALE,//)
C
50    K=K+1
      ID1=IABS(ID)
      DO 8 I=1,M
      DO 8 J=1,N
```

```
      IFA(I,J)=MOD(IA(I,J),ID1)
55    IF(IFA(I,J).LT.0)IFA(I,J)=IFA(I,J)+ID1
    8 CONTINUE
      IR=1
      NK=-1
      IF(IFA(1,1).EQ.0)GOTO 105
60    DO 104 J=2,N
      IF(IFA(1,J).EQ.0)GOTO 104
      GOTO 16
  104 CONTINUE
  105 CONTINUE
65    DO 10 I=1,M
      IF(IFA(I,1).LT.NK)GOTO 10
      DO 103 J=2,N
      IF(IFA(I,J).EQ.0)GOTO 103
      GOTO 106
70    103 CONTINUE
      GOTO 10
  106 CONTINUE
      NK=IFA(I,1)
      IR=I
75    10 CONTINUE
      IF(NK.GT.0)GOTO 16
      DO 17 I=1,M
   17 IA(I,1)=IA(I,1)/ID1
      GO TO 6
80    16 DO 11 J=1,N
   11 IA(M+1,J)=-IFA(IR,J)
      M=M+1
      JVCOL(M)=1000+K
      GOTO 100
85    6 PRINT 12
   12 FORMAT(//,36X,17HSOLUTION OPTIMALE,///)
C
   13 FORMAT(5X,2HZ=,I10,//)
      PRINT 3,NT,(IVLI(I),I=1,N)
90    PRINT 3,JVCOL(1),(IA(1,J),J=1,N)
      PRINT 13,IA(1,1)
      DO 15 I=2,M
   15 PRINT 14,JVCOL(I),IA(I,1)
   14 FORMAT(5X,I10,5X,I10)
95    102 CONTINUE
      RETURN
      END
```

Algorithme TOKOSI 3

```
      PROGRAM TOKOSI3(INPUT,OUTPUT)
      COMMON IA(200,100)
      DIMENSION IA1(100),IST(9)
      DIMENSION JN(100),KM(100)
C       VERSION III
 1010 FORMAT(10I6)
    9 FORMAT(I6)
   10 FORMAT (1X,I6)
    4 FORMAT (1H1,25X,18H TABLEAU DE DEPART)
    6 FORMAT(1H0,19(1X,I6))
      NCOUNT=13
      ICOUNT=0
 2222 ICOUNT=ICOUNT+1
      CALL STATUS(IST)
      TEMDEBP=IST(2)
      TEMDEBC=IST(1)
      K=1
      READ 1010,M,N
      IDER=1
      M1=M+1
      N1=N+1
      IM= M1+N
      READ 1010,((IA(I,J),J=1,N1),I=1,M1)
      ICASE=0
      PRINT 4
      PRINT 6,(IA(1,J),J=1,N1)
      DO 661 I=1,M
  661 PRINT 6,(IA(I+1,J),J=1,N1)
   18 FORMAT (1H0,I5,13HEME ITERATION)
      DO 1005 I=1,N
      DO 1005 J=1,N1
   20 IA(I+M1,J) =0
      IF(I-J+1.EQ.0)IA(I+M1,J)=-1
 1005 CONTINUE
      DO 660 I=1,N
  660 PRINT 6,(IA(I+M1,J),J=1,N1)
    2 FORMAT(3X,9H LAMDA = ,F10.3)
   13 FORMAT(1H0,17HSOLUTION OPTIMALE//,1X,1HZ,4X,1H=,3X,I6)
   12 FORMAT(1H0,22H NOMBRE D ITERATION =,I3)
   14 FORMAT(1H0,1HX,I2,2X,1H=,3X,I6)
 9999 FORMAT(1H0,20X,24H PROGRAMME DUAL FAISABLE)
      I1=1
C
      DO 999 J=2,N1
      IF(IA(1,J).GE.0)I1=I1+1
  999 CONTINUE
      IF(I1-N1,EQ.0)GOTO 898
      GOTO 1000
  898 PRINT 9999
C
C     RECHERCHE DE LA LIGNE CANDIDATE PAR LA METHODE DUALE SIMPLITIALE
 7000 DO 1003 J=2,IM
      IF(IA(J,1).LT.0)GOTO 1004
```

```
PROGRAM    TOKOSI3  FORTRAN EXTENDED VERSION 2.0          20/09/70          02.09.55.
          1003 CONTINUE
55             GOTO 7 00
          1004 CALL DUALEX2(N,IN,IS,ID,IND)
               CALL MODIF1(N1,M1,IR,IS,XLAMAX,JM,KM,IER,IND)
               IF(IER.NE.0)GOTO 1000
               ICASE=ICASE+1
60             PRINT 18,ICASE
               PRINT 895,IA(1,1),IR,IS
           895 FORMAT(//,2X,9HFONCT.ECO,5X,13HLIGNE GENERAT,5X,14HCOLONNE PIVOT,/
              1/,3X,I6,1)X,I4,15X,I4,/)
               IF(ICASE-100*IDER.NE.0)GOTO 7000
65             IDER=IDER+1
               GOTO 7000
          C    SOLUTION OPTIMALE
           3000 PRINT 13,IA(1,1)
               PRINT 12,ICASE
70             DO 1070 I=2,M1
          1070 PRINT 15,I,IA(I,1)
             15 FORMAT(2X,1HY,I2,2X,1H=,3X,I6)
               DO 1071 I=1,N
          1071 PRINT 14,I,IA(I+M1,1)
75        1080 CALL STATUS(IST)
               DTC=(IST(1)-TEMDEBC)/1000.
               DTP=(IST(2)-TEMDEBP)/1000.
               PRINT 1011,DTC,DTP
          1011 FORMAT(//,1X,17HTEMPS D EXECUTION,10X,2HU,C,F8.3,3HSEC,10X,2HU,P,
80            1F8.3,3HSEC,//)
               IF(ICOUNT-NCOMPT,N5,0)GOTO 2222
               CALL EXIT
               END
```

```
      SUBROUTINE MODIF1(N,M,IR,IS,XLAMAX,JM1,KM,IER,IND)
      COMMON IX(200,100)
      DIMENSION IA1(100)
      IM=M+N-1
05    XLAMAX=0
  100 FORMAT(5X,I6,3X,I2)
      DO 31 J=2,N
      IF(IX(IND,J).LE.0)GOTO 31
      IF(IX(IR,J).GE.0)GOTO 31
10    I1=IX(IND,J)/IX(IND,IS)
      IF(I1.EQ.0)GOTO 31
      X1=-FLOAT(IX(IR,J))/FLOAT(I1)
      IF(X1-XLAMAX.LT.0)GOTO 31
      XLAMAX=X1
15    31 CONTINUE
  102 FORMAT(1X,F10.3)
      IF(XLAMAX.EQ.0)GOTO 15
      DO 13 JJ=2,N
      11 L=2
20    IF(IX(L,JJ))6,6,13
      6 CC=FLOAT(IX(IR,JJ))/XLAMAX
      ICC=IFIX(CC)
      IF(IC.GT.CC)IC=IC-1
      IF(JJ.EQ.IS)GOTO 13
25    IF(IX(1,JJ)+IC*IX(1,IS))8,8,13
      8 IF(IX(L,JJ)+IC*IX(L,IS))4,9,13
      9 L=L+1
      GOTO 8
      4 DO 10 IJ=2,N
30    IF(-IX(IR,IJ).LE.XLAMAX)GOTO 10
      XLAMAX=-IX(IR,IJ)
      GOTO 11
      10 CONTINUE
      13 CONTINUE
35    PRINT 100,L,JJ
      IF(XLAMAX.EQ.0)GOTO 15
C
C     TRANSFORMATION DU TABLEAU DUAL
C
40    C     EQUATION DE TRONCATURE
 1030 DO 50 J=1,N
      C=IX(IR,J)
      IA1(J)=C/XLAMAX
      IF(FLOAT(IA1(J))-C/XLAMAX.GT.0.)IA1(J)=IA1(J)-1
45    50 CONTINUE
      DO 1040 I=1,IM
      DO 1040 J=1,N
      IF(J-IS.EQ.0)GOTO 1040
      IX(I,J)=IX(I,J)+IX(I,IS)*IA1(J)
50    1040 CONTINUE
      DO 1301 I=1,IM
 1301 IX(I,IS)=IX(I,IS)
      IER=0
```

```
      20 RETURN
55    15 PRINT 14
      14 FORMAT(1X,13HLAMDA EST NUL)
         IER=2
         GOTO 20
         END
```

```
      SUBROUTINE DUALEX2(N,M,IS,IR,IND)
      COMMON IA(200,100)
      DIMENSION INTER(50),ILIGNE(90),JRANCOL(50)
C
05    IS=1
      IR=1
      KP=1
      IND=1
      KAIG=0
10    DO 1 I=1,50
      JRANCOL(I)=0
      INTER(I)=0
      IF(I-N.GT.0)GOTO 1
      INTER(I)=IA(1,I+1)
15  1 CONTINUE
C
      DO 4 I=1,N
      ND=1000000
      J1=0
20    DO 2 J=1,N
      IF(INTER(J).LT.0)GOTO 2
      NB=MIN0(ND,INTER(J))
      IF(NB-ND.NE.0)GOTO 3
      IF(ND-INTER(J).NE.0)GOTO 2
25    L=2
    6 IF(IA(L,J1+1)-IA(L,J+1))2,5,32
    5 L=L+1
      GOTO 6
   32 IF(IA(L,J1+1)*IA(L,J+1).LT.0)GOTO 33
30    GO TO 3
   33 IF(IA(L,J1+1).LT.0)GOTO 2
    3 ND=NB
      NB=0
      J1=J
35  2 CONTINUE
      INTER(J1)=-10000
      IF(IA(1,J1+1).EQ.0)KP=KP+1
    4 JRANCOL(I)=J1
      RATIO=FLOAT(N-KP+1)/FLOAT(N)
40 23 FORMAT(9H RATIO = ,F5.2)
      J1=JRANCOL(1)
      IF(IA(1,J1+1).EQ.0)GOTO 20
      DO 24 I=2,M
      IF(IA(I,1).GE.0)GOTO 24
45    DO 25 J=1,N
      IF(IA(I,J+1).GE.0)GOTO 25
      GOTO 26
   25 CONTINUE
   26 IF(KAIG-IA(I,1).LE.0)GOTO 24
50    IR=I
      KAIG=IA(I,1)
   24 CONTINUE
C
```

```
      20 K=1
      NM=M-1
      DO 7 I=1,MM
      ILIGNE(K)=0
      IF(IA(I+1,1).GE.0)GOTO 7
      ND=0
      LR=1000
      DO 10 J=1,N
      JK=JRANCOL(J)
      IF(JK.EQ.0)GOTO 10
      IF(IA(I+1,JK+1).GE.0)GOTO 10
      ND=MIN0(J,LR)
      IF(ND-LR.EQ.0)GOTO 10
      LR=ND
      ND=0
   10 CONTINUE
      ILIGNE(K)=(I+1)*1000+LR
    7 K=K+1
C
      DO 11 J=K,50
   11 ILIGNE(J)=0
      ND=0
      DO 13 LI=1,M
      IF(ILIGNE(LI).EQ.0)GOTO 13
      NN=ILIGNE(LI)-(ILIGNE(LI)/1000)*1000
      IF(NN-ND.LT.0)GOTO 13
      IF(ND-NN.NE.0)GOTO 19
      II=ILIGNE(J1)/1000
      II1=ILIGNE(LI)/1000
      IF(IABS(IA(II,1))-IABS(IA(II1,1)).GE.0)GOTO 13
   19 ND=NN
      J1=LI
   13 CONTINUE
C
      IR=ILIGNE(J1)/1000
      JS=ILIGNE(J1)-IR*1000
      IS=JRANCOL(JS)+1
      IF(IA(IR,IS).LT.0)GOTO 102
   27 K=0
   18 K=K+1
      LL=JRANCOL(K)
      IF(IA(IR,LL+1).LT.0)GOTO 15
      GOTO 18
C
   15 IS=LL+1
  102 IF(IA(IND,IS).GT.0)GOTO 101
      IND=IND+1
      GOTO 102
   16 FORMAT(//,10X,5HPIVOT,5X,2HIR,5X,2HIS,//,5X,I10,2X,I5,2X,I5,//)
  101 IPIVOT=IA(IR,IS)
      RETURN
      END
```

Editions des résultats selon les codes TOKOSI 1 et TOKOSI 2.

TABLEAU DE DEPART

	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11
11111	-0	1	1	1	1	1	-0	-0	-0	-0	-0	-0
10	-1	-1	1	-0	-0	-0	5	-0	-0	-0	-0	-0
20	4	1	-1	-0	-0	-0	-5	-0	-0	-0	-0	-0
30	-1	-0	-1	1	-0	-0	-0	5	-0	-0	-0	-0
40	4	-0	1	-1	-0	-0	-0	-5	-0	-0	-0	-0
50	-1	-0	-0	-1	1	-0	-0	-0	5	-0	-0	-0
60	4	-0	-0	1	-1	-0	-0	-0	-5	-0	-0	-0
70	-1	-1	-0	1	-0	-0	-0	-0	-0	5	-0	-0
80	4	1	-0	-1	-0	-0	-0	-0	-0	-5	-0	-0
90	-1	-1	-0	-0	1	-0	-0	-0	-0	-0	5	-0
100	4	1	-0	-0	-1	-0	-0	-0	-0	-0	-5	-0
110	-1	-0	-0	-0	-1	1	-0	-0	-0	-0	-0	5
120	4	-0	-0	-0	1	-1	-0	-0	-0	-0	-0	-0

PROBLEME DUAL REALISABLE

F.ECO	PIVOT	IR	IS	VARIABLES BASE,	VARIABLES H.BASE	ITERATION
-1	-1	2	2	-1	10	1
-3	-1	4	3	-2	30	2
-6	-1	6	4	-3	50	3
-10	-1	12	5	-4	110	4

SOLUTION OPTIMALE

0	0	10	30	50	110	-5	-6	-7	-8	-9	-10	-11
11111	-10	1	2	3	4	5	5	10	15	-0	-0	-0
Z=	-10											

-1	4
20	3
-2	3
40	3
-3	2
60	3
70	1
80	2
90	2
100	1
-4	1
120	3

TEMPS D EXECUTION U.C .136SEC U.P .016SEC

TABLEAU DE DEPART

	0	-1	-2	-3	-4	-5	-6
11111	-0	-0	-0	-0	-0	-0	1
10	2	-1	1	-0	3	-0	-0
20	-1	1	-1	-0	-3	-0	-0
30	2	-0	-1	1	-0	3	-0
40	-1	-0	1	-1	-0	-3	-0
50	-0	1	-0	-0	-0	-0	-1
60	-0	-0	1	-0	-0	-0	-1
70	-0	-0	-0	1	-0	-0	-1
80	1	-0	-0	-0	1	-0	-0
90	1	-0	-0	-0	-0	1	-0

6 var
9 cont

PROBLEME DUAL REALISABLE

F.ECO	PIVOT	IR	IS	VARIABLES BASE,	VARIABLES H,BASE	ITERATION
-0	-1	3	3	-2	20	1
-0	-3	5	6	-5	60	2
-0	-9	7	5	-4	60	3
-0	-6	11	4	-3	1001	4
-0	-3	8	6	40	70	5
-0	-6	4	5	60	50	6
-3	-15	3	7	-5	-2	7
-9	-9	12	4	1001	1002	8
-3	-3	12	6	70	1003	9
-5	-6	11	5	30	-3	10
-2	-2	13	2	-1	1004	11
-4	-4	6	7	-2	50	12
-2	-2	14	5	-3	1003	13
-2	-2	12	6	1003	70	14
-5	-5	8	7	50	40	15
-2	-2	14	6	70	1005	16
-2	-2	13	7	60	-3	17

SOLUTION OPTIMALE

0	0	1004	20	1002	1005	1006	-3
11111	-1	-0	-0	2	-0	-0	-0
Z=	-1						

10	1
-6	1
60	-0
-3	1
-2	1
-4	1
80	1
90	1

```
 30          =0
 =1          =0
 40           1
 70           1

TEMPS D EXECUTION      U.C    .018SEC        U.P    .373SEC
```

B I B L I O G R A P H I E

***

[1] BALAS E. *"Un algorithme additif pour la résolution des programmes linéaires en variables bivalentes"* Comptes rendus de l'Académie des Sciences de Paris. Vol.258, pp. 3817-3820, (1964)

[2] BALAS E. *"Extension de l'Algorithme additif à la programmation en nombres entiers et à la programmation non linéaire"* Comptes rendus de l'Académie des Sciences de Paris. Vol.258, pp. 5136-5139, (1964)

[3] BALAS E. *"On additive algorithm for solving linear programs with 0-1 variables"* Operations Research 13 (4), 1965, pp. 517-688

[4] BALINSKI M. *"Integer Programming : Methods, Uses, Computation"* Management Science : Vol. 12, n°3, Novembre 1965

[5] BAUMEL W.J. *"Théorie Economique et Analyse Opérationnelle"* Dunod 1963

[6] BEALE E.M.L. *"A method of solving linear programming problems when some but not all of the variables must take Integral values"* Statistical Techniques Research Group, Technical Report n°19, Princeton University, Juillet 1958.

[7] BEN ISRAËL A. et CHARMY A. *"On some problems of diaphantine programming"* Cahier du centre d'études de R.O. 1962, pp. 219-280

[8] BERGE C. *"Théorie des graphes et ses applications"* Dunod. Paris 1967

[9] BERTIER P. ROY et NGHIEM
"Résolution de problèmes en variables bivalentes. Algorithme de Balas et procédures S.E.P."
Note de travail n°33 Janvier 1965. SEMA

[10] BERTIER P. ROY et NGHIEM
"Procédures S.E.P. trois exemples numériques"
Notes de travail n°32 Janvier 1965. SEMA

[11] BOUVIER B. et MESSOUMIAM
"Programmes linéaires en variables bivalentes : Algorithme de E. Balas"
Thèse Grenoble 1965

[12] CAMION P.
"Caractérisation des matrices unimodulaires"
Cahiers du Centre d'Etudes de Recherche Opérationnelle.
Vol. 5 (1963)

[13] CHARNES. A., COOPER W.W. and G.L. THOMPSON
"Contrained generalized medians and L.P. under uncertainly"
Evanston North Western University, the technological institute and Pittsburg Carnegie institute of technology July 1961)

[14] DANTZIG G.B.
"Applications et prolongements de la programmation linéaire"
Dunod 1966

[15] DANTZIG G.B.
"On the significance of solving L.P. problems with some integral variables"
Econometrica Vol. 28 n°1 Janvier 1960
Rand paper pp.1486, Avril 1959

[16] DANTZIG G.B.
"Note in solving L.P. in integers"
Naval research logistic quaterly Vol. 6.75.76, 1950

[17] DODU, LUDOT, POUGET *"Sur le regroupement optimal des sommets dans un réseau électrique"*
Electricité de France : Direction des Etudes et Recherches
S.E.R.C.A. Décembre 1965

8] FAURE R. et MALGRANGE
"Une méthode booléenne pour la résolution des programmes linéaires en nombres entiers"
Gestion : numéro spécial, avril 1963

9] FORTET R.
"Mathématiques des programmes économiques"
Monographie Dunod

0] FORTET R.
"L'algèbre de Boole et ses applications en recherche opérationnelle"
Cahiers du Centre d'Etudes de Recherche Opérationnelle.
Nov. 1959

1] GEOFFRIAN
"Integer Programming by implicit enumeration and Balas Method"
The rand Corporation, RM 4783, PR février 1966

2] GOMORY R.E.
"An algorithm for integer solutions to linear programs"
Princeton I.B.M. Math Research project technic report n°1,
17 novembre 1958

3] GOMORY R.E.
"An integer programming algorithm"
I.B.M. research report R.C. 189, 29 Janvier 1950

4] GOMORY R.E.
"An algorithm for the mixed integer problem"
R.M. 2597 Rand Corporation, 7 Juillet 1960

5] GOMORY R.E.
"On the relations between integer and noninteger solutions to linear programs"
Proceeding of the National Academy of Sciences, Vol. 53 (1965)
pp. 260-265

6] GOMORY R.E. et BAUMOL
"Integer Programming and Pricing"
Econometrica. Vol. 28 (1960) pp. 521-550

7] GOMORY R.E.
"On the relation between integer and noninteger solutions to linear programs"
Thomas J. Watson research center, Yorktown Heights. Newyork.
Communicated by K. Commant. December 22, 1964

8] GOMORY R.E.
"Outlines of an Algorithm for integer solutions to linear programs"
Communicated by A.W. Tucker, May 3, 1958

[29] GOMORY R.E. *"Faces of an Integer Polyhedron"*
Thomas J. Watson Research Center Yorktown Heights New York

[30] HALDI J. ISAACSON, LEONARD M
"A computer code for Integer Solutions to linear programs"
Operations Research, Vol. 13, n°6. November - December 1965
pp. 946-959

[31] HALDI J. *"25 Integer Programming. Test Problems"*
Working paper n°43, Graduate School of Business, Standford University

[32] HERVE P. *"Resolution des programmes linéaires invariables mixtes par la procédure S.E.P."*
Metra, Vol.6 n°1 1967

[33] HERVE P. *"Les procédures arborescentes d'optimisation"*
Revue française d'information et de recherche opérationnelle n°14, Vol. 3 Novembre 1967

[34] HEURGON E. *"Programmation linéaire en nombres entiers"*
Thèse, Paris 1967

[35] HUARD P. BROISE P. SENTENAN J.
"Décomposition des programmes mathématiques"
Monographie de Recherche opérationnelle n°6, 1968 Dunod Paris

[36] HUARD P. *"Résolution des programmes mathématiques en variables bivalentes, méthode CBV"*
Note EDF HR 7. 029, mai 1966

[37] KOUDRINE J. SIROT M. TOURNOY B.
"Expérience comparative des méthodes de BALAS, HERVE, GEOFFRION"
Communication privée 1969 (Thèse à paraître)

[38] LAMBERT E. *"Programmes en nombres entiers et programmes mixtes"*
Metra. Vol. 1 n°1, 1962

[39] LAND et DOIG *"An automatic method of solving discrete programming problems"*
Econometrica, Vol 28 (1960)

0] LEMKE C.E. *"The Dual method for solving linear programming problem"* Naval Research logistics Quaterly 1, 1957, p.36

1] LUDOT J.P. *"Méthode heuristique de résolution des programmes mathématiques en nombres entiers. Recherche d'une solution entière appartenant à un domaine convex de R.M."* Bulletin de la Direction des Etudes et Recherches E.D.F.

2] LUDOT J.P. *"Méthode heuristique de résolution des programmes mathématiques en nombres entiers. Expériences numériques"* Bulletin de la Direction des Etudes et Recherches E.D.F. série C n°2 196

3] MARTIN G. *"An accelerated euclidean algorithm for integer linear programming"* Recent Advances in Math Programming R. GRAVES et P. WOLFE, 1963

4] OUYAHIA AIT *"Programmes linéaires à variables discrètes"* Revue Française de Recherche Opérationnelle, Vol.6, 1962 n°22 pp. 55-75

5] ROY B. *"Procedure d'exploration par séparation et évaluation (P.S,E.P. et P.S.E.S)"* R.I. R.O. Série verte n°18, 1969

6] SIMONARD M. *"Programmation linéaire"* Dunod, Paris (1962)

7] TRAUTH JR. and R.E. WOOLSEY *"Integer linear programming : A study in computational efficiency"* Management Sciences Vol 15 n°9 May 1969

8] WADE C.S. and GOMORY R.E. *"I.P.M. 1 SHARE"* Distribution Number 1192 September 1962

9] WADE C.S. and GOMORY R.E. *"I.P.M. 2 SHARE"* Distribution 1191 September 1962

0] WOOLSEY *"An integer linear programming in comdinatorial Analysis"* Sandia Laboratories Reprint 54 . R . 65 August 1965