THÈSE

présentée à

L'UNIVERSITE DES SCIENCES ET TECHNIQUES DE LILLE

pour obtenir le titre de

DOCTEUR DE 3ème CYCLE

par

Arnaud FREVILLE

HEURISTIQUES ET REDUCTION POUR LES PROGRAMMES MATHEMATIQUES EN VARIABLES 0-1 A CONTRAINTES D'INEGALITE

Soutenue le 14 Octobre 1983 devant la Commission d'Examen

MEMBRES DU JURY :

Président	C.	BREZINSKI
Rapporteur	G.	PLATEAU
Examinateurs	C.	CARREZ
	J.C.	FIOROT
	P.	HUARD
	M.	MINOUX

PROFESSEURS 1ère CLASSE

M.	BACCHUS Pierre	Mathématiques
M.	BEAUFILS Jean-Pierre (dét.)	Chimie
M.	BIAYS Pierre	G.A.S.
M.	BILLARD Jean	Physique
M.	BONNOT Ernest	Biologie
M.	BOUGHON Pierre	Mathématiques
M.	BOURIQUET Robert	Biologie
M.	CELET Paul	Sciences de la Terre
M.	COEURE Gérard	Mathématiques
M.	CONSTANT Eugène	I.E.E.A.
M.	CORDONNIER Vincent	I.E.E.A.
M.	DEBOURSE Jean-Pierre	S.E.S.
M.	DELATTRE Charles	Sciences de la Terre
M.	DURCHON Maurice	Biologie
M.	ESCAIG Bertrand	Physique
M.	FAURE Robert	Mathématiques
M.	FOURET René	Physique
M.	GABILLARD Robert	I.E.E.A.
M.	GRANELLE Jean-Jacques	S.E.S.
M.	GRUSON Laurent	Mathématiques
M.	GUILLAUME Jean	Biologie
M.	HECTOR Joseph	Mathématiques
M.	HEUBEL Joseph	Chimie
M.	LABLACHE COMBIER Alain	Chimie
M.	LACOSTE Louis	Biologie
M.	LANSRAUX Guy	Physique
M.	LAVEINE Jean-Pierre	Sciences de la Terre
M.	LEBRUN André	C.U.E.E.P.
M.	LEHMANN Daniel	Mathématiques
Mme	LENOBLE Jacqueline	Physique
M.	LHOMME Jean	Chimie
M.	LOMBARD Jacques	S.E.S.
M.	LOUCHEUX Claude	Chimie
M.	LUCQUIN Michel	Chimie

M.	MAILLET Pierre	S.E.S.
M.	MONTREUIL Jean	Biologie
M.	PAQUET Jacques	Sciences de la Terre
M.	PARREAU Michel	Mathématiques
M.	PROUVOST Jean	Sciences de la Terre
M.	SALMER Georges	I.E.E.A.
Mme	SCHWARTZ Marie-Hélène	Mathématiques
M.	SEGUIER Guy	I.E.E.A.
M.	STANKIEWICZ François	Sciences Economiques
M.	TILLIEU Jacques	Physique
M.	TRIDOT Gabriel	Chimie
M.	VIDAL Pierre	I.E.E.A.
M.	VIVIER Emile	Biologie
M.	WERTHEIMER Raymond	Physique
M.	ZEYTOUNIAN Radyadour	Mathématiques

PROFESSEURS 2ème CLASSE

M.	AL FAKIR Sabah	Mathématiques
M.	ANTOINE Philippe	Mathématiques
M.	BART André	Biologie
Mme	BATTIAU Yvonne	Géographie
M.	BEGUIN Paul	Mathématiques
M.	BELLET Jean	Physique
M.	BKOUCHE Rudolphe	Mathématiques
M.	BOBE Bernard	S.E.S.
M.	BODART Marcel	Biologie
M.	BOILLY Bénoni	Biologie
M.	BONNELLE Jean-Pierre	Chimie
M.	BOSQ Denis	Mathématiques
M.	BREZINSKI Claude	I.E.E.A.
M.	BRUYELLE Pierre (Chargé d'enseignement)	Géographie
M.	CAPURON Alfred	Biologie
M.	CARREZ Christian	I.E.E.A.
M.	CHAMLEY Hervé	E.U.D.I.L.

M. CHAPOTON Alain	C.U.E.E.P.
M. COQUERY Jean-Marie	Biologie
Mme CORSIN Paule	Sciences de la Terre
M. CORTOIS Jean	Physique
M. COUTURIER Daniel	Chimie
Mle DACHARRY Monique	Géographie
M. DEBRABANT Pierre	E.U.D.I.L.
M. DEGAUQUE Pierre	I.E.E.A.
M. DELORME Pierre	Biologie
M. DEMUNTER Paul	C.U.E.E.P.
M. DE PARIS Jean-Claude	Mathématiques
M. DEVRAINNE Pierre	Chimie
M. DHAINAUT André	Biologie
M. DORMARD Serge	S.E.S.
M. DOUKHAN Jean-Claude	E.U.D.I.L.
M. DUBOIS Henri	Physique
M. DUBRULLE Alain	Physique
M. DUEE Gérard	Sciences de la Terre
M. DYMENT Arthur	Mathématiques
M. FLAMME Jean-Marie	E.U.D.I.L.
M. FONTAINE Hubert	Physique
M. GERVAIS Michel	S.E.S.
M. GOBLOT Rémi	Mathématiques
M. GOSSELIN Gabriel	S.E.S.
M. GOUDMAND Pierre	Chimie
M. GREVET Patrice	S.E.S.
M. GUILBAULT Pierre	Biologie
M. HANGAN Théodore	Mathématiques
M. HERMAN Maurice	Physique
M. JACOB Gérard	I.E.E.A.
M. JACOB Pierre	Mathématiques
M. JOURNEL Gérard	E.U.D.I.L.
M. KREMBEL Jean	Biologie
M. LAURENT François	I.E.E.A.
Mle LEGRAND Denise	Mathématiques
Mle LEGRAND Solange	Mathématiques (Calais)
Mme LEHMANN Josiane	Mathématiques

M. LEMAIRE Jean	Physique
M. LENTACKER Firmin	G.A.S.
M. LEVASSEUR Michel	I.P.A.
M. LHENAFF René	G.A.S.
M. LOCQUENEUX Robert	Physique
M. LOSFELD Joseph	I.E.E.A.
M. LOUAGE Francis	E.U.D.I.L.
M. MACKE Bruno	Physique
M. MAIZIERES Christian	I.E.E.A.
Mle MARQUET Simone	Mathématiques
M. MESSELYN Jean	Physique
M. MIGEON Michel	E.U.D.I.L.
M. MIGNOT Fulbert	Mathématiques
M. MONTEL Marc	Physique
Mme NGUYEN VAN CHI Régine	G.A.S.
M. PARSY Fernand	Mathématiques
Mle PAUPARDIN Colette	Biologie
M. PERROT Pierre	Chimie
M. PERTUZON Emile	Biologie
M. PONSOLLE Louis	Chimie
M. PORCHET Maurice	Biologie
M. POVY Lucien	E.U.D.I.L.
M. RACZY Ladislas	I.E.E.A.
M. RICHARD Alain	Biologie
M. RIETSCH François	E.U.D.I.L.
M. ROGALSKI Marc	M.P.A.
M. ROUSSEAU Jean-Paul	Biologie
M. ROY Jean-Claude	Biologie
M. SALAMA Pierre	S.E.S.
Mme SCHWARZBACH Yvette (CCP)	M.P.A.
M. SCHAMPS Joël	Physique
M. SIMON Michel	S.E.S.
M. SLIWA Henri	Chimie
M. SOMME Jean	G.A.S.
Mle SPIK Geneviève	Biologie
M. STERBOUL François	E.U.D.I.L.
M. TAILLIEZ Roger	Institut Agricole

M. TOULOTTE Jean-Marc	I.E.E.A.
M. VANDORPE Bernard	E.U.D.I.L.
M. WALLART Francis	Chimie
M. WATERLOT Michel	Sciences de la Terre
Mme ZINN JUSTIN Nicole	M.P.A.

CHARGES DE COURS

M. TOP Gérard	S.E.S.
M. ADAM Michel	S.E.S.

CHARGES DE CONFERENCES

M. DUVEAU Jacques	S.E.S.
M. HOFLACK Jacques	I.P..A
M. LATOUCHE Serge	S.E.S.
M. MALAUSSENA DE PERNO Jean-Louis	S.E.S.
M. OPIGEZ Philippe	S.E.S.

Que Monsieur le Professeur BREZINSKI trouve ici l'expression de ma profonde reconnaissance pour m'avoir fait l'honneur d'être le Président du jury de cette thèse.

Je tiens à témoigner toute ma gratitude et mon amitié à Gérard PLATEAU qui est l'instigateur de ce travail. Avec ses nombreuses suggestions et ses remarques pertinentes, sa compétence et sa gentillesse ont été des arguments décisifs dans l'élaboration de cette thèse.

Je tiens à remercier très vivement

Monsieur le Professeur Jean-Charles FIOROT, qui par sa confiance et sa disponibilité, a été pour moi un soutien constant.

Monsieur Michel MINOUX pour ses critiques constructives et l'intérêt très vif qu'il a manifesté pour cette étude.

Messieurs les Professeurs Christian CARREZ et Pierre HUARD de m'avoir fait l'honneur de participer à ce jury.

Je remercie les membres de l'équipe d'Analyse Numérique et d'Optimisation qui m'ont entouré de leur sollicitude et de leurs encouragements.

Je tiens enfin à remercier Patricia CARON pour le soin et la diligence apportés à la dactylographie de cette thèse, ainsi que Madame DEBOCK pour son excellent travail de reproduction.

A Isabel
Axelle
Camille
Pierre-Olivier

Table des Matières

INTRODUCTION

Ce travail s'ajoute aux nombreuses études consacrées à la programmation en nombres entiers et tout spécialement aux programmes en variables bivalentes dont l'importance dans les domaines économiques et techniques (problèmes d'investissement, planification de production, dans les sociétés de transport ou de service, chargement de navires, etc ...) n'est plus à démontrer.

Son but essentiel est la construction d'un code de calcul pour la réduction de la taille du problème traité, rejoignant ainsi les recherches très actuelles de groupes industriels tels que IBM. Etant donné que les temps de calcul de toute procédure de résolution exacte dépendent crucialement du nombre de variables, toute tentative raisonnable de diminution de la taille du problème peut s'avérer fructueuse, le mot raisonnable signifiant l'utilisation d'algorithmes de complexité linéaire en moyenne.

Le concept de la contraction de contraintes est à la base de cette réalisation qui s'articule dans le cas d'un problème de maximisation autour des trois points suivants :

- calcul d'un majorant par la résolution de relaxations
- calcul d'un minorant par la construction d'heuristiques
- mise en oeuvre de tests spécifiques d'élimination de variables

et de contraintes faisant intervenir explicitement le majorant et le minorant précités.

Au vu des expériences numériques menées avec ce code automatique de réduction (environ 3000 instructions FORTRAN) sur une cinquantaine de problèmes tests, notre objectif semble atteint.

Cet exposé est scindé en cinq parties : le chapitre 1 fait une synthèse des propriétés relatives aux fonctions duales Lagrangiennes et composites et à l'existence des écarts de dualité dans le cas Lagrangien et composite pour les programmes linéaires (cas convexe) et les programmes linéaires en variables bivalentes (cas non convexe).

Le chapitre 2 est consacré à l'étude des principales méthodes de résolution des différentes relaxations envisageables pour le calcul d'un majorant. Deux algorithmes de type sous-gradient sont présentés en détail pour la résolution des relaxations Lagrangiennes et composites. Dans ce

dernier cas de nombreuses expériences numériques ont permis de déterminer une régle pratique pour le choix des coefficients de relaxation.

Le chapitre 3 présente deux heuristiques de complexité linéaire en moyenne (conjecturé pour l'une, démontré pour l'autre). Leurs performances, aussi bien en temps calcul qu'en précision, sont supérieures à celles de toutes les autres méthodes répertoriées dans la littérature.

Le chapitre 4 concerne la présentation et la mise en oeuvre du code pour la réduction en nombre de variables et de contraintes de la taille du problème traité. Prolongeant un premier travail expérimental dû à P. Hansen et G. Plateau, il est proposé un certain nombre de solutions originales pour le choix des relaxations, pour l'ordre de priorité des contraintes à éliminer ainsi que pour l'ordre d'utilisation des tests d'élimination. Les résultats numériques démontrent l'efficacité de cette procédure dont l'utilisation avant toute méthode de résolution doit permettre en général un gain en temps de calcul important. De plus la complexité linéaire en moyenne constatée numériquement ouvre des perspectives intéressantes pour les problèmes de moyenne et grande taille.

Le chapitre 5 est consacré à l'étude d'un algorithme engendrant une contrainte supplémentaire permettant de réduire l'ensemble des solutions réalisables du problème. Cette nouvelle contrainte consiste en un encadrement de la somme des variables égales à 1 à l'optimum du problème. Basée sur certaines des propriétés de la fonction duale composite, cette procédure améliore très sensiblement une première approche due à F. Glover.

NOTATIONS

Les notations utilisées tout au long de l'exposé sont les suivantes :

$\lfloor\lambda\rfloor$: partie entière d'un réel λ ; $\lceil\lambda\rceil = -\lfloor-\lambda\rfloor$

étant donné un ensemble J :

$[J]$: enveloppe convexe de J

$|J|$: cardinal de J

$J\backslash U$: complémentaire d'un sous-ensemble U de J

étant donné A *une matrice de format* $(I\times J)$ *où* I *et* J *sont des ensembles finis :*

a_{ij} : élément $(i, j) \in I \times J$ de A

A^j : colonne $j \in J$ de A

A_i : ligne $i \in I$ de A

$A^{J'}$: sous-matrice de A composée de colonnes A^j $j \in J' \subset J$

$A_{I'}$: sous-matrice de A composée de lignes A_i $i \in I' \subset I$

étant donné y *un vecteur (ligne ou colonne) indicé par un ensemble* J :

y_j : élément $j \in J$ de y

$y_{J'}$: sous-vecteur de y composé des éléments y_j $j \in J' \subset J$

étant donné un problème d'optimisation (P) :

$v(P)$: valeur optimale de (P)

$\bar{v}(P)$: (resp. $\underline{v}(P)$) borne supérieure (resp. inférieure) de $v(P)$

$F(P)$: domaine de (P)

$\Omega(P)$: ensemble des solutions optimales de (P)

$(P \mid x \in X)$: lorsque (P) est résolu avec les conditions supplémentaires $x \in X$

[i.e. $(\exists\, j : x_j = \varepsilon)$ ou $(\exists\, j_1, j_2 : x_{j_1} = \varepsilon_1\ x_{j_2} = \varepsilon_2)$ ou $(\sum_j x_j \geq \ell)$ avec $\varepsilon, \varepsilon_1, \varepsilon_2 \in \{0, 1\}$, $\ell \in \mathbb{N}$]

$x_j \leftarrow \varepsilon$: x_j est fixée à la valeur $\varepsilon \in \{0, 1\}$

lorsque (P) est un problème à n variables bivalentes :

x^* : solution optimale de (P)

$V = \{x \in \mathbb{R}^n \mid x_j = 0 \text{ ou } 1,\ j = 1, \ldots, n\}$

$(\bar{P})$: problème (P) dans lequel [V] remplace V

$\bar{x}$: solution optimale de $(\bar{P})$

étant donné $x \in [V]$:

$[x]$: élément de V tel que $[x]_j = \lfloor x_j \rfloor$ $j = 1, \ldots, n$

$||x||_1$: norme L_1 du vecteur x ($= \sum_{j=1}^{n} x_j$)

$||x||_2$: norme L_2 du vecteur x ($= (\sum_{j=1}^{n} x_j^2)^{1/2}$)

$c\ell(X)$: fermeture de l'ensemble X

GENERALITES SUR LA DUALITE EN PROGRAMMATION MATHEMATIQUE EN VARIABLES BIVALENTES

CHAPITRE 1

INTRODUCTION

Ce chapitre est consacré à l'énoncé d'une part, des propriétés permettant de comparer les deux types de dualité envisagés, lagrangienne et composite et d'autre part des différentes techniques de résolution de ces duaux.

Après le rappel (§ 1.1) des définitions de base, une deuxième partie (§ 1.2) compare les deux dualités utilisées et rappelle les principales propriétés des fonctions duales Lagrangienne et composite, nécessaires à la bonne compréhension de la suite de l'exposé. Nous présentons en parallèle les théorèmes d'existence des écarts de dualité, basés sur la nature convexe (resp. non convexe) d'un programme linéaire en variables réelles (resp. en variables entières ou bivalentes). Enfin nous précisons les liens existants entre multiplicateurs Lagrangiens et composites.

1.1 - Définitions

1.1.1 - Position du problème

Le problème s'énonce comme suit :

$$(B)\quad \left[\begin{array}{ll} \max & c\,x \\ \text{s.c.} & A\,x \leq b \\ & C\,x \leq d \\ & x \in V = \{x \in \mathbb{R}^n \mid x_j = 0 \text{ ou } 1,\ j = 1, 2, \ldots, n\} \end{array}\right.$$

où $c \in \mathbb{R}^n_+$, $b \in \mathbb{R}^{m_1}$, $d \in \mathbb{R}^{m_2}$, $A \in \mathbb{R}^{m_1 \times n}$, $C \in \mathbb{R}^{m_2 \times n}$, les matrices des contraintes A et C se différenciant par leur structure :

- A matrice dense : $\rho(A) \geq 50$
- C matrice creuse: $\rho(C) \geq 5$

ρ étant le coefficient de densité défini pour une matrice de taille $m \times n$ par $p = \frac{100}{m \times n} \times$ nombre total des éléments non nuls de la matrice.

Ce problème appartient à la classe suivante des problèmes de maximisation avec contraintes :

$$(P) \quad \begin{cases} \max \ f(x) \\ \text{s.c. } a(x) \leq 0 \\ x \in X \subseteq \mathbb{R}^n \end{cases}$$

avec
- $f : \mathbb{R}^n \to R$ fonction économique linéaire
  $x \to f(x) = cx$
- $a : \mathbb{R}^n \to \mathbb{R}^{m_1}$ contraintes affines
  $x \to a(x)$ tel que : $\forall\, i \in \{1,2,\ldots,m_1\}\ a_i(x) = A_i\, x - b_i$
- $X = \{x \in V \mid Cx \leq d\}$ sous-ensemble compact, fini, non convexe de V et supposé non vide.

1.1.2 - Exemples

Exemple 1.1 : $m_1 \geq 0$, $m_2 > 0$, m_2 très grand devant m_1

$m_1 \neq 0$: ce type de problème est en général de grande taille et correspond par exemple à des problèmes d'affectation (Guignard, [25]). Dans ce cas les contraintes correspondant à la matrice C sont très structurées et conduisent à des méthodes de résolution qui essayent d'utiliser au mieux ces structures particulières.

$m_1 = 0$: suite aux travaux de M. et K. Spielberg ([63]), Crowder, Johnson et Padberg ([5]) présentent un code de calcul qui résout exactement 10 problèmes de ce type ($33 \leq n \leq 2756$, $16 \leq m_2 \leq 756$) provenant directement du monde industriel.

Exemple 1.2 :

$m_2 = 0$: ce type de programme se caractérise par une matrice A de contraintes très dense. Cette modélisation se rencontre par exemple dans les problèmes d'investissement. C'est sur ces problèmes que porte l'essentiel de notre étude. Dans ce cas, X = V.

1.1.3 - Relaxation et dualité

Définition 1.1

Un problème de maximisation (Q) est appelé ***relaxation*** *d'un problème de maximisation (P) si*

(i) $F(Q) \supseteq F(P)$

(ii) la fonction économique de (Q) majore celle de (P) sur $F(P)$.

De manière très classique, une relaxation (qui fournit une borne supérieure de la valeur du problème d'origine) sera un nouveau problème plus facile à résoudre que le problème initial.

Dans la littérature, on rencontre essentiellement trois types de relaxation obtenus par :

a) relachement des conditions d'intégralité

on obtient le programme linéaire $(\bar{B})$ suivant

$$(\bar{B}) \quad \begin{cases} \max \ cx \\ \text{s.c. } Ax \leq b \\ \qquad x \in \bar{X} = \{x \in \mathbb{R}^n \mid Cx \leq d,\ 0 \leq x \leq e\} = [V] \cap \{x \in \mathbb{R}^n \mid Cx \leq d\} \\ \qquad\quad (e \text{ est le vecteur unité de } \mathbb{R}^n) \end{cases}$$

note : Comme [V] est un sous-ensemble compact et convexe de $\mathbb{R}^n$, $\bar{X}$ posséde la même propriété. □

b) suppression des m_1 contraintes associées à la matrice A en les introduisant par combinaison linéaire dans la fonction économique :

Définition 1.2

On appelle ***relaxation lagrangienne*** *de (B) associée aux contraintes* $Ax \leq b$ *et au vecteur* $w \in \mathbb{R}_+^{m_1}$ *le problème*

$$(BL(w)) \quad \begin{cases} \max \; cx + w\,(b - Ax) \\ \text{s.c. } x \in X \end{cases}$$

c) <u>combinaison linéaire des m_1 contraintes</u> associées à la matrice A pour aboutir à une seule contrainte, dite composite (Glover, [18]) :

<u>Définition 1.3</u>

On appelle ***relaxation composite*** *de (B) associée aux contraintes* $Ax \leq b$ *et au vecteur* $w \in \mathbb{R}_+^{m_1}$ *le problème*

$$(BS(w)) \quad \begin{cases} \max \; c\,x \\ \text{s.c. } wAx \leq wb \quad \textit{(contrainte composite)} \\ x \in X \end{cases}$$

<u>note</u> : il est immédiat que les problèmes $(\bar{B})$, (BL(w)) et (BS(w)) sont des relaxations du problème (B) au sens de la définition 1.1. □

De façon naturelle, nous définissons les problèmes duaux associés.

<u>Définition 1.4</u>

Le ***dual lagrangien*** *de (B) est le problème de minimisation*

$$(D_L) \quad \begin{cases} \inf \; v(BL(w)) \\ \text{s.c. } w \in \mathbb{R}_+^{m_1} \end{cases}$$

<u>Définition 1.5</u>

Le ***dual composite*** *de (B) est le problème de minimisation*

$$(D_S) \quad \begin{cases} \inf \; v(BS(w)) \\ \text{s.c. } w \in \mathbb{R}_+^{m_1} \end{cases}$$

<u>Remarque 1</u> : On montrera (proposition 1.5) que la fonction $w \to v(BL(w))$ atteint son minimum sur $\mathbb{R}_+^{m_1}$ et par conséquent, on peut remplacer "inf" par "min". De même (proposition 1.10), on montrera que les fonctions

$w \to v(BS(w))$ et $w \to v(\overline{BS}(w))$ atteignent leurs minimums sur le compact $B_1 = \{w \in \mathbb{R}_+^{m_1} \mid ||w||_1 = 1\}$.

□

Remarque 2 : Le programme linéaire $(\bar{B})$ est un programme primal appartenant à la catégorie plus générale des programmes convexes suivants :

$$(P) \quad \begin{cases} \max \ f(x) \\ \text{s.c. } a(x) \leq 0 \\ x \in C \end{cases}$$

avec $f : \mathbb{R}^n \to \mathbb{R}$ concave

$a : \mathbb{R}^n \to \mathbb{R}^m$ convexe

C convexe non vide de $\mathbb{R}^n$

le programme dual (D) associé à (P) est défini par :

$$(D) \quad \begin{cases} \min \ \theta(w) \\ \text{s.c. } w \in \mathbb{R}_+^m \end{cases}$$

avec $\theta(w) = \max_{x \in C} \{f(x) - w\, a(x)\}$

Le théorème fondamental de la dualité pour les programmes convexes nous assure, sous certaines conditions de régularité, que $v(P) = v(D)$. Dans le cas plus particulier du programme linéaire $(\bar{B})$ (f : linéaire, a affine, $C = \bar{X}$ convexe) ces conditions se ramènent à la non-vacuité des domaines réalisables $F(\bar{B})$ et $F(D)$. Comme le dual (D) est équivalent au dual lagrangien (D_L), il en résulte que $v(\bar{B}) = v(D_L)$.

□

1.2 - Principaux résultats théoriques

1.2.1 - Comparaison des dualités

Nous allons rappeler dans ce paragraphe les propriétés relatives aux connexions entre le problème (B), ses relaxations et les problèmes duaux associés définis ci-dessus. On pourra se reporter pour une étude

plus approfondie aux articles suivants : Geoffrion ([17]), Glover ([19]), Greenberg-Pierskalla ([27], [29]) et Karwan-Rardin ([38]).

Notons $\Omega(.)$ l'ensemble éventuellement vide des solutions optimales du problème (.), $\bar{w}$ le *multiplicateur associé aux contraintes* $Ax \leq b$ *à l'optimum de* $(\bar{B})$.

Théorème 1.1

(i) $v(B) \leq v(D_L) \leq v(BL(\bar{w})) \leq v(\bar{D}_L) = v(\bar{B})$

(ii) si pour un multiplicateur $w^* \in \mathbb{R}_+^{m_1}$, x^* *vérifie les conditions*

(a) $x^* \in \Omega(BL(w^*))$

(b) $Ax^* \leq b$

(c) $w^* (Ax^* - b) = 0$

alors x^* *est une solution optimale de* (B).

Démonstration

(i) . $F(B) \subseteq F(\bar{B}) \Rightarrow v(B) \leq v(\bar{B})$

. $F(B) \subseteq F(BL(w)) \quad \forall\, w \in \mathbb{R}_+^{m_1}$

. $cx \leq cx + w(b - Ax), \ \forall\, (w,x) \in \mathbb{R}_+^{m_1} \times \{x \mid Ax \leq b\}$

$\Rightarrow v(B) \leq v(BL(w)) \quad \forall\, w \in \mathbb{R}_+^{m_1}$

. $v(\bar{B}) = v(\bar{D}_L)$ (voir remarque 2)

$= \min_{w \geq 0} \{\max_{x \in \bar{X}} \{cx + w(b - Ax)\}}$ (définition du dual)

$= \min_{w \geq 0} v(\overline{BL(w)})$

$= v(\overline{BL(\bar{w})})$ ($\bar{w}$ est une solution optimale du dual $(\bar{D}_L)$)

$\geq v(BL(\bar{w}))$ car $F(BL(w)) \subseteq F(\overline{BL(w)}), \ \forall\, w \in \mathbb{R}_+^{m_1}$

$\geq \min_{w \geq 0} v(BL(w)) = v(DL)$

(ii) (a) $\Rightarrow x^* \in X$ et $cx^* + w^*(b - Ax^*) \geq cx + w^*(b - Ax)$, $\forall\, x \in X$

(b) et $x^* \in X \Rightarrow x^* \in F(B)$

(c) $\Rightarrow cx^* \geq cx + w^*(b - Ax) \quad \forall\, x \in X$

$$\left.\begin{array}{l} cx^* \geq cx \quad \forall\, x \in F(B) \\ x^* \in F(B) \end{array}\right\} \Rightarrow x^* \in \Omega(B)$$

□

Remarque : La partie (ii) fournit une CNS pour que $v(B) = v(D_L)$, ou en d'autres termes, pour que l'écart de dualité lagrangienne $\tau_L = v(D_L) - v(B)$ soit nul.

□

Le (i) du théorème 1.1 peut faire espérer un écart de dualité lagrangienne strictement plus petit que l'écart de dualité linéaire, soit $\tau_L < \tau = v(\bar{B}) - v(B)$. Geoffrion a donné dans [17] la condition suffisante suivante pour que $\tau_L = \tau$.

Définition 1.6

La relaxation lagrangienne du problème de maximisation (B) possède la ***propriété d'intégralité*** *si* $v(BL(w)) = v(\overline{BL(w)})$, $\forall\, w \geq 0$.

Remarques

(i) Cette propriété est vérifiée pour l'exemple 1.2

(ii) Plus généralement, cette propriété sera vérifiée pour tout couple (C, d) tel que C soit une matrice totalement unimodulaire et d un vecteur entier.

Proposition 1.1

Si la relaxation lagrangienne de (B) possède la propriété d'intégralité alors $v(D_L) = v(\bar{B})$.

Démonstration

Immédiat à partir de celle du théorème 1.1.

□

Cette situation où $\tau = \tau_L$ se rencontre souvent en programmation mathématique, en particulier lorsqu'on dualise toutes les contraintes ($m = m_1$, $m_2 = 0$).

Donnons maintenant dans le cas de la relaxation Lagrangienne un résultat parallèle au théorème 1.1.

Théorème 1.2

(i) $v(B) \leq v(D_S) \leq v(BS(\bar{w})) \leq v(\bar{D}_S) = v(\bar{B})$

(ii) si pour un multiplicateur $w^* \in \mathbb{R}_+^{m_1}$, x^* *vérifie les conditions*

(a) $x^* \in \Omega(BS(w^*))$

(b) $Ax^* \leq b$

alors x^* *est une solution optimale de (B).*

Démonstration

(i) . $F(B) \subseteq F(BS(w))\ \forall\ w \in \mathbb{R}_+^{m_1} \Rightarrow v(B) \leq v(BS(w)),\ \forall\ w \in \mathbb{R}_+^{m_1}$

. $v(\bar{D}_S) = \min\limits_{w \geq o} \{\max\limits_{x \in \bar{X}} \{cx \mid w\,Ax \leq wb\}\}$; or $(\max\limits_{x \in \bar{X}} cx$ s.c. $w\,Ax \leq wb)$

est un programme convexe et par conséquent

$$\max_{x \in \bar{X}} \{cx \mid w\,Ax \leq wb\} = \min_{\alpha \in \mathbb{R}_+} \{\max_{x \in \bar{X}} \{cx + \alpha(wb - w\,Ax)\}\}.$$

En posant $u = \alpha\ w \in \mathbb{R}_+^{m_1}$ on a alors

$v(\bar{D}_S) = \min\limits_{u \geq o} \{\max\limits_{x \in \bar{X}} \{cx + u(b - Ax\}\} = v(\bar{D}_L) = v(\bar{B})$ (Théorème 1.1)

. $v(\bar{D}_S) = \min\limits_{w \geq o} v(\overline{BS(w)})$

$= v(\overline{BS(\bar{w})})$ ($\bar{w}$ est une solution optimale du dual $(\bar{D}_S)$)

$$\geq v(BS(\bar{w})) \quad \text{car } F(BS(w)) \subseteq F(\overline{BS(w)}),\ \forall\, w \in \mathbb{R}_+^{m_1}$$

$$\geq \min_{w \geq o} v(BS(w)) = v(D_S)$$

(ii) (a) $\Rightarrow x^* \in X$ et $c x^* \geq cx \quad \forall\, x \in X \cap \{x \mid w^* Ax \leq w^* b\}$

(b) et $x^* \in X \Rightarrow x^* \in F(B)$

$\{x \mid Ax \leq b\} \subset \{x \mid w^* Ax \leq w^* b\}$

$\Rightarrow x^* \in \Omega(B)$

□

Remarque :

La partie (ii) fournit une CNS pour que $v(D_S) = v(B)$, ou en d'autres termes, pour que l'écart de dualité composite $\tau_S = v(D_S) - v(B)$ soit nul. A la différence du cas lagrangien, la *condition de complémentarité* $w^* (Ax^* - b) = 0$ n'est plus exigée.

□

Dans le cas composite, on peut définir aussi une propriété d'intégralité :

Définition 1.7

La relaxation composite du problème de maximisation (B) possède la ***propriété d'intégralité*** *si* $v(BS(w)) = v(\overline{BS(w)}),\ \forall\, w \geq 0$

Proposition 1.2

Si la relaxation composite (B) possède la propriété d'intégralité alors $v(D_S) = v(\bar{B})$.

Démonstration

Immédiate à partir de celle du théorème 1.2.

□

Au contraire de la dualité Lagrangienne, la situation où $\tau = \tau_S$ se rencontre très rarement en programmation mathématique.

Le résultat suivant précise les relations entre les valeurs des duaux Lagrangiens et composites.

Théorème 1.3

(i) $v(D_S) \leq v(D_L)$

(ii) si $v(D_S) = v(D_L)$ *alors* $\forall\, w^* \in \Omega(D_L)$ *on a :*

$$w^* \in \Omega(D_S) \text{ et } \Omega(BS(w^*)) = \Omega(BL(w^*)) \cap \{x \mid w^*(Ax - b) = 0\}$$

démonstration

(i) $\forall\, w \in \mathbb{R}_+^{m_1}$, (BL(w)) est la relaxation lagrangienne de (BS(w)) associée au multiplicateur 1, donc $v(BS(w)) \leq v(BL(w))$ et par suite $\min_{w \geq o} v(BS(w)) \leq \min_{w \geq o} v(BL(w))$

(ii) - $\forall\, x \in X$, $cx > v(D_L) = \max_{x \in X} \{cx + w^*(b - Ax)\}$

$\Rightarrow w^*(b - Ax) < 0$, soit $x \notin F(BS(w^*))$

donc si $x \in F(BS(w^*))$ alors $cx \leq v(D_L)$, soit $v(BS(w^*)) \leq v(D_L)$

mais alors $v(D_L) \geq v(BS(w^*)) \geq v(D_S) = v(D_L)$ donc $w^* \in \Omega(D_S)$.

- soit $x \in \Omega(BS(w^*))$: d'une part, $w^*(Ax - b) \leq 0$ et d'autre part, puisque $w^* \in \Omega(D_L)$, $v(D_L) \geq cx + w^*(b - Ax) \geq cx \geq v(D_S) = v(D_L)$

donc $w^*(b - Ax) = 0$ et $x \in \Omega(BL(w^*))$

- réciproquement, si $x \in \Omega(BL(w^*))$ et $w^*(b - Ax) = 0$ alors a fortiori $w^*(Ax - b) \leq 0$ et $cx = v(BL(w^*)) = v(D_L) = v(D_S) = v(BS(w^*))$

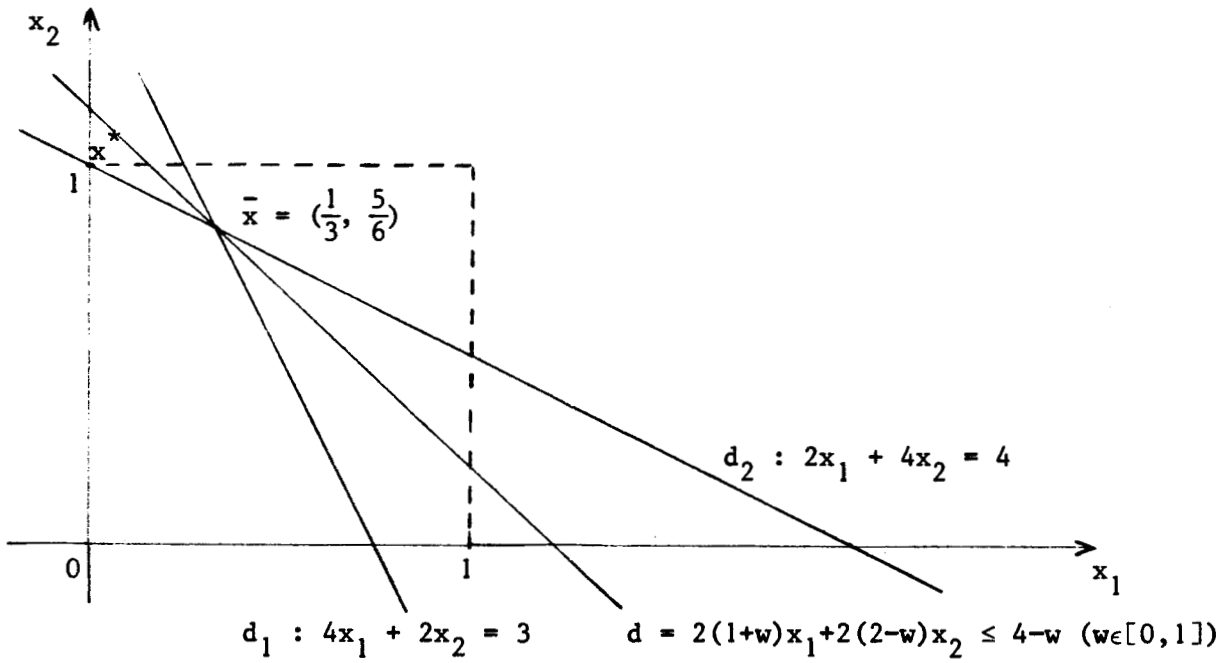
donc $x \in \Omega(BS(w^*))$.

□

Au regard des théorèmes précédents on s'aperçoit que la condition de complémentarité joue un rôle primordial pour l'existence et la valeur des écarts de dualité τ, τ_L et τ_S. En particulier Glover a développé dans [20] une théorie sur la dualité composite et ses extensions dont la pierre angulaire est justement cette condition de complémentarité.

Exemple 1.3

$$(B)\quad \begin{cases} \max \quad x_1 + x_2 \\ \text{s.c. } 4x_1 + 2x_2 \le 3 \\ \qquad 2x_1 + 4x_2 \le 4 \\ \qquad x_1,\ x_2 \in V \end{cases} \quad Ax \le b$$



La solution optimale x^* de (B) est (0, 1), d'où $v(B) = 1$. La solution optimale $\bar{x}$ de $(\bar{B})$ est $(\frac{1}{3}, \frac{5}{6})$, d'où $v(\bar{B}) = \frac{7}{6}$. D'autre part, $v(D_L) = v(\bar{B}) = \frac{7}{6}$ (propriété d'intégralité) et $\Omega(D_L) = \{(\frac{1}{6}, \frac{1}{6})\}$. Pour $w^* = (\frac{1}{6}, \frac{1}{6})$, la condition de complémentarité s'écrit :

$$6x_1 + 6x_2 - 7 = 0 \quad (1)$$

(1) étant vérifiée par aucun élément $x = (x_1, x_2) \in V$, il est sûr que

$\tau_S < \tau_L = \frac{1}{6}$, ce qui est confirmé par un calcul direct puisque $v(D_S) = v(B) = 1$ ($\forall\, w \in \mathbb{R}^2_+$, $(0, 1) \in F(BS(w))$ mais $(1, 1) \notin F(BS(w))$).

□

1.2.2 - Propriétés de la fonction duale lagrangienne

Notons ℓ la fonction duale lagrangienne définie comme suit :

$$\ell : \mathbb{R}^{m_1}_+ \to \mathbb{R}$$

$$w \to \ell(w) = v(BL(w)) = \max_{x \in X} \{cx + w\,(b - Ax)\}$$

Posons $|X| = N$ et $I_X = \{1, 2, \ldots, N\}$. En numérotant les éléments de X, on peut écrire $\ell(w) = \max_{k \in I_X} \{c_k + w\, g_k\}$

avec $c_k = c\, x^k$ et $g_k = b - A\, x^k$

Puisque (B) est supposé réalisable ($N > 0$), par conséquent $\Omega(BL(w)) \neq \emptyset$. Notons $I^*(w) \subset I_X$ le sous-ensemble non vide d'indices, associé à $\Omega(BL(w))$:

$$I^*(w) = \{k \in I_X \mid \ell(w) = c_k + w\, g_k\}$$

Posons enfin

$$L(w) = \{g_k \in \mathbb{R}^{m_1} \mid k \in I^*(w)\}$$

Proposition 1.3

ℓ est une fonction continue, convexe et linéaire par morceaux sur le demi-espace $\mathbb{R}^{m_1}_+$ (figure 1.1).

Démonstration

$\ell(w) = \max \{f_k(w) \mid k \in I_X\}$ où $f_k : w \to c_k + w\, g_k$ est une fonction affine, $\forall\, k \in I_X$. L'enveloppe supérieure d'une famille finie de fonctions affines définies sur le convexe fermé $\mathbb{R}^{m_1}_+$ est une fonction s.c.i., convexe, linéaire par morceaux. Enfin ℓ est s.c.s. car convexe et bornée sur le cône polyèdrique $\mathbb{R}^{m_1}_+$ ([58], Théorème 10.2).

□

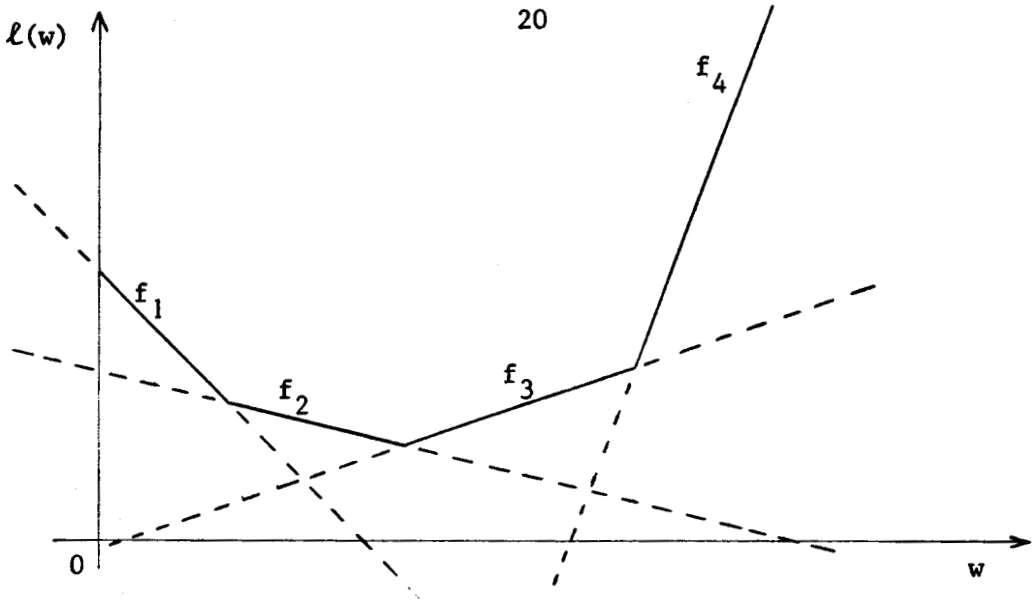


Figure 1.1 : ($m_1 = 1$ N = 4)

Proposition 1.4

Le sous-différentiel $\partial\ell(w)$ coïncide avec l'enveloppe convexe de L(w) *pour* $w \in \mathbb{R}_+^{m_1 *}$

Démonstration ([24])

Procédons en 2 étapes

- $\forall\, w \geq 0,\ co(L(w)) \subset \partial\ell(w)$

soit $k^* \in I^*(w)$: $\forall\, w' \geq 0,\ \ell(w') = \max \{c_k + w'\, g_k \mid k \in I_X\}$

$$\geq c_k^* + w'\, g_k^*$$

$$\geq \ell(w) - w\, g_k^* + w'\, g_k^*$$

$$\text{d'où } \ell(w') - \ell(w) \geq g_k^*\,(w' - w)$$

équivalent à dire que g_k^* est un sous-gradient de la fonction convexe ℓ en w.

Alors, de façon immédiate, pour $g = \sum_{k \in I_X} \alpha_k\, g_k$ avec $\sum_{k \in I_X} \alpha_k = 1$, on a

$$\forall\, w' \geq 0,\ g\,(w'-w) = \sum_{k \in I_X} \alpha_k\, g_k\,(w'-w) \leq \sum_{k \in I_X} \alpha_k (\ell(w')-\ell(w)) = \ell(w') - \ell(w)$$

- $\forall\, w > 0,\ \partial\ell(w) \subset co(L(w))$

L'épigraphe de la fonction convexe ℓ est le convexe :

$$epi(\ell) = \{(z, w) \in \mathbb{R} \times \mathbb{R}_+^{m_1} \mid \ell(w) \leq \tau\}$$

Il est défini par le système d'inéquations linéaires

$$\begin{cases} c_k + w\, g_k \le z, \quad k \in I_X \\ w \ge 0, \quad (z) \end{cases}$$

Pout tout g_o sous-gradient de ℓ en w_o on a $[\ell(w_o)-g_o\, w_o \le \ell(w)-g_o\, w$, $\forall\, w \in \mathbb{R}_+^{m_1}]$, ce qui signifie que le couple $(\ell(w_o),\, w_o)$ est une solution optimale du programme linéaire convexe consistant à minimiser la fonction $f(z, w) = z-g_o\, w$ sur le convexe épi(ℓ) défini par les contraintes $a_k(z, w) = z-g_k\, w-c_k \ge 0$, $k \in I_X$.

Ecrivons les conditions d'optimalité associées à ce programme :

programme	conditions d'optimalités	
min $f(z, w)$	$\exists\, u \in \mathbb{R}^N$	
s.c. $a_k(z, w) \ge 0$, $k \quad I_X$	$u \ge 0$	
$w \ge 0$	$u\, \nabla_w a(\ell(w_o),w_o) - \nabla_w f(\ell(w_o),w_o) \le 0$	
(z)	$u\, \nabla_z a(\ell(w_o),w_o) - \nabla_z f(\ell(w_o),w_o) = 0$	(2)
	$u\, a(\ell(w_o),w_o) = 0$	(1)
	$[u\, \nabla_w a(\ell(w_o),w_o) - \nabla_w f(\ell(w_o),w_o)]\, w_o = 0$	(3)

avec $\nabla a(z, w) = (e, -g_1, \ldots, -g_N)$, $\nabla f(z, w) = (1, -g_o)$

La relation d'exclusion (1) s'écrit $\sum_{k\in I_X} u_k\, (\ell(w_o) - g_k\, w_o - c_k) = 0$

d'où $u_k > 0 \Rightarrow \ell(w_o) = c_k + w_o\, g_k$, c'est à dire que $g_k \in L(w_o)$

(2) donne : $u\, e - 1 = 0$ soit $\sum_{k\in I_X} u_k = 1$

(3) donne : $(-\sum_{k\in I_X} u_k\, g_k + g_o)\, w_o = 0$

si $w_o > 0$ alors $g_o = \sum_{k\in I^*(w_o)} u_k\, g_k$, combinaison linéaire convexe d'éléments de $L(w_o)$.

□

Remarques

(i) Une autre démonstration est donnée dans [30] pour le programme plus général

$$\begin{cases} \max \ f(x) \\ \text{s.c. } a(x) \leq 0 \\ \qquad x \in X \subset \mathbb{R}^n \end{cases}$$

sous les hypothèses f s.c.s., a continu, X compact non vide.

(ii) Les propriétés rappelées en Annexe 1 s'appliquent à la fonction convexe ℓ. En particulier ℓ est presque partout différentiable et w sera un minimum global ($\in \Omega(D_L)$) si $0 \in \partial\ell(w)$. En pratique l'éventualité d'avoir $0 \in \partial\ell(w)$ ne se présente pratiquement jamais. Pour un couple (A, b), (A matrice entière, b vecteur entier), la probabilité de trouver un vecteur binaire x^k, avec $k \in I^*(w)$, tel que $A x^k - b = 0$ est presque nulle.

Proposition 1.5

$\Omega(D_L)$ *est un convexe fermé non vide de* $\mathbb{R}_+^{m_1}$

Démonstration

On a $\Omega(D_L) = \{w \in \mathbb{R}_+^{m_1} \mid \ell(w) = v(D_L)\}$

et $v(D_L) = \inf \{v(BL(w)) \mid w \in \mathbb{R}_+^{m_1}\} = \inf \{\ell(w) \mid w \in \mathbb{R}_+^{m_1}\}$

alors la proposition 1.5 découle des propriétés de ℓ (proposition 1.3)

ℓ linéaire par morceaux sur $\mathbb{R}_+^{m_1}$ => $\Omega(D_L) \neq \emptyset$

ℓ convexe => $\Omega(D_L)$ convexe

ℓ continu => $\Omega(D_L)$ fermé (image réciproque d'un fermé de $\mathbb{R}$).

□

1.2.3 - *Propriétés de la fonction duale composite*

Notons s la fonction duale composite définie comme suit :

$$s : \mathbb{R}_+^{m_1} \to \mathbb{R}$$

$$w \to s(w) = v(BS(w)) = \max_{x \in X} \{cx \mid w\,Ax \leq wb\}$$

$\bar{s}$ la fonction duale composite "relachée" définie comme suit :

$$\bar{s} : \mathbb{R}_+^{m_1} \to \mathbb{R}$$

$$w \to \bar{s}(w) = v(\overline{BS(w)}) = \max_{x \in \bar{X}} \{cx \mid w\,Ax \leq wb\}$$

Les ensembles de solutions réalisables $F(BS(w)) = \{x \in X \mid w\,Ax \leq wb\}$ et $F(\overline{BS(w)}) = \{x \in \bar{X} \mid w\,Ax \leq wb\}$ ont les propriétés suivantes :

Proposition 1.6

$\forall\, w \in \mathbb{R}_+^{m_1}$

(i) $F(BS(w))$ *est un sous-ensemble compact, fini, non convexe de* $\mathbb{R}^n$

(ii) $F(\overline{BS(w)})$ *est un sous-ensemble compact, convexe de* $\mathbb{R}^n$

Démonstration

Immédiat à partir des propriétés de X et $\bar{X}$.

□

Il est immédiat de constater que pour tout réel $\lambda > 0$, $s(\lambda w) = s(w)$ (idem pour $\bar{s}$). Ceci va permettre par le biais d'une normalisation de restreindre le domaine de définition de s et $\bar{s}$ à un sous-ensemble compact $Y \subset \mathbb{R}_+^{m_1}$ pour lequel on a l'équivalence.

$$\left[\begin{array}{l}\max \; s(w) \\ \text{s.c. } w \in \mathbb{R}_+^{m_1}\end{array}\right] \equiv \left[\begin{array}{l}\max \; s(w) \\ \text{s.c. } w \in Y\end{array}\right] \quad \text{(idem pour } \bar{s})$$

Les normes utilisées sont usuellement les normes L_1 et L_2. On posera en conséquence

$B_1 = \{w \geq 0,\ ||w||_1 = 1\}$ compact, convexe de $\mathbb{R}_+^{m_1}$

$B_2 = \{w \geq 0,\ ||w||_2 = 1\}$ compact de $\mathbb{R}_+^{m_1}$.

Précisons maintenant les propriétés essentielles des fonctions duales composites s et $\bar{s}$:

Proposition 1.7

(i) s *est une fonction quasi-convexe sur* $\mathbb{R}_+^{m_1}$, *semi-continue supérieurement sur* B_1.

(ii) $\bar{s}$ *est une fonction quasi-convexe sur* $\mathbb{R}_+^{m_1}$, *continue sur* B_1.

Démonstration ([27], [47])

(i) - s quasi-convexe sur $\mathbb{R}_+^{m_1}$: $w = \alpha w_1 + (1-\alpha)\, w_2 \in \mathbb{R}_+^{m_1}$, $\forall\, w_1, w_2 \in \mathbb{R}_+^{m_1}$, $\alpha \in [0, 1]$ puisque $\mathbb{R}_+^{m_1}$ est convexe. Soient $x_1 \in \Omega(BS(w_1))$, $x_2 \in \Omega(BS(w_2))$, $x \in \Omega(BS(w))$; supposons que $x \notin F(BS(w_1)) \cup F(BS(w_2))$, soit $w_1 (Ax-b) > 0$ et $w_2 (Ax-b) > 0$, alors $\alpha w_1 (Ax-b) + (1-\alpha) w_2 (Ax-b) = w (Ax-b) > 0$, ce qui est contradictoire avec l'hypothèse $x \in \Omega(BS(w))$. Donc $\forall\, \alpha \in [0, 1]$, x est solution réalisable d'au moins l'un des problèmes $(BS(w_i))$, $i = 1, 2$: par conséquent $cx_1 \geq cx$ ou $cx_2 \geq c\,x$ et par suite

$$s(w) = c\,x \leq \max \{cx_1, cx_2\} = \max \{s(w_1), s(w_2)\}.$$

Il s'ensuit que s est quasi- convexe sur le convexe $B_1 \subset \mathbb{R}_+^{m_1}$.

- s semi-continue supérieurement sur B_1 : soit une suite $\{w^k\}_{k\in\mathbb{N}} \subset B_1$ convergeant vers $w^* \in B_1$ (compact). $\forall\, k \in \mathbb{N}$, $x^k \in \Omega(BS(w^k))$ et $x^k \in Y(w^k) \subset X$ (compact). Par conséquent, on peut extraire une sous-suite $\{x^k\}_{k\in\mathbb{N}'\subset\mathbb{N}}$ convergeant vers $x^* \in X$. $\forall\, k \in \mathbb{N}'$, $w^k (A\,x^k - b) \leq 0$ et par passage à la limite $w^* (Ax^* - b) \leq 0$, soit $x^* \in F(BS(w^*))$ donc $s(w^*) \geq c\,x^*$. Par suite,

$$\lim_{k\to+\infty,\, k\in\mathbb{N}'} s(w^k) = \lim_{k\to+\infty,\, k\in\mathbb{N}'} c\,x^k = cx^* \leq s(w^*)$$

□

(ii) - $\bar{s}$ quasi-convexe sur $\mathbb{R}_+^{m_1}$: même démonstration qu'en (i) puisque la nature de $F(\overline{BS(w)})$ ne transparait pas

- $\bar{s}$ continue sur $\mathbb{R}_+^{m_1}$:

. $\bar{s}$ est semi-continue supérieurement sur B_1 : voir (i)

. $\bar{s}$ est semi-continue inférieurement sur $\mathbb{R}_+^{m_1}$: la démonstration de ce point nécessite au préalable l'énoncé de plusieurs lemmes.

Considérons le problème perturbé $(\overline{B(y)})$ et la fonction de perturbation associée $v(\overline{B(y)})$ définie par :

$$v(\overline{B(y)}) = \max \{cx \mid Ax - b \leq y,\ x \in \bar{X}\}.$$

Notons $\Gamma = \{y \in \mathbb{R}^{m_1} \mid F(\overline{B(y)}) \neq \emptyset\}$. Par commodité, posons $\bar{\phi}(y) = v(\overline{B(y)})$

Lemme 1

Γ est un convexe non vide.

Démonstration

Par hypothèse $F(\overline{B(0)}) = \emptyset$ donc $0 \in \Gamma$. Soient $y_1, y_2 \in \Gamma$, $\alpha \in [0, 1]$, $x_1 \in F(\overline{B(y_1)})$, $x_2 \in F(\overline{B(y_2)})$: d'une part $A(\alpha x_1+(1-\alpha)x_2)-b = \alpha(Ax_1-b) + (1-\alpha)(Ax_2-b) \leq y_1+(1-\alpha)y_2$, d'autre part $\alpha x_1+(1-\alpha)x_2 \in \bar{X}$ (convexe). Donc $\alpha x_1 + (1-\alpha)x_2 \in F(\overline{B(\alpha y_1+(1-\alpha)y_2)})$, ie $\alpha y_1+(1-\alpha)y_2 \in \Gamma$.

□

Lemme 2

La fonction $\bar{\phi}$ est concave sur Γ.

Démonstration

Soient $y_1, y_2 \in \Gamma$, $\alpha \in [0, 1]$, $y = \alpha y_1+(1-\alpha)y_2$. $\forall\ i \in \{1, 2\}$, $\exists\ x_i \in F(\overline{B(y_i)})$ tel que $\bar{\phi}(y_i) = c\,x_i$. Comme $x = \alpha x_1 + (1-\alpha)x_2 \in F(\overline{B(y)})$ (lemme 1), $\bar{\phi}(y) \geq c\,x = \alpha\, cx_1+(1-\alpha)c\,x_2 = \alpha\bar{\phi}(y_1) + (1-\alpha)\,\bar{\phi}(y_2)$.

□

Lemme 3

La fonction multivoque $y \to F(\overline{B(y)})$ est continue sur Γ.

Démonstration

Ce lemme est une application directe du corollaire 5.8.2 de [34] appliqué à l'intersection $F(\overline{B(y)}) = \bar{X} \cap \{x \mid Ax - b \leq y\}$, avec $\bar{X}$ enveloppe convexe d'un nombre fini de points et $\{x \mid Ax - b \leq y\}$ fermé, convexe, non vide si $y \in \Gamma$.

□

Lemme 4 :

La fonction $\bar{\phi}$ est continue sur Γ.

Démonstration

Montrons que $\bar{\phi}$ est d'abord semi-continue supérieurement puis semi-continue inférieurement sur Γ. Considérons une suite quelconque d'éléments de Γ, $\{y_k\}_{k\in\mathbb{N}}$, convergeant vers $y^* \in \Gamma$.

- $\bar{\phi}$ s.c.s. en y^* : soit une sous-suite $\{y_k\}_{k\in\mathbb{N}'\subset\mathbb{N}}$ telle que $\lim\limits_{\substack{k\to+\infty \\ k\in\mathbb{N}'}} \phi(y_k)=\alpha$.

$\forall\, k \in \mathbb{N}'$, $F(\overline{B(y_k)}) = \{x \in \bar{X} \mid Ax - b \leq y_k\}$ est un compact non vide : par suite, $\exists\, x_k \in F(\overline{B(y_k)})$ tel que $\bar{\phi}(y_k) = c\, x_k$ (1). La suite $\{x_k\}_{k\in\mathbb{N}'}$ appartient au compact $\bar{X}$, on peut donc en extraire une sous-suite $\{x_k\}_{k\in\mathbb{N}''\subset\mathbb{N}'}$ convergeant vers $x^* \in \bar{X}$. D'autre part $\lim\limits_{\substack{k\to\infty \\ k\in\mathbb{N}''}} y_k = y^*$ et comme la fonction multivoque $y \to F(\overline{B(y)})$ est sup-continue sur Γ, $x^* \in F(\overline{B(y^*)})$ donc $\bar{\phi}(y^*) \geq c\, x^*$ (2).

Enfin, d'après (1) et (2), $\bar{\phi}(y^*) \geq c\, x^* = \lim\limits_{\substack{k\to\infty \\ k\in\mathbb{N}''}} c\, x_k = \lim\limits_{\substack{k\to\infty \\ k\in\mathbb{N}''}} \bar{\phi}(y_k) = \alpha$

- $\bar{\phi}$ s.c.i. en y^* : soit une sous-suite $\{y_k\}_{k\in\mathbb{N}'\subset\mathbb{N}}$ telle que $\lim\limits_{\substack{k\to+\infty \\ k\in\mathbb{N}'}} \bar{\phi}(y_k) = \alpha$. $\exists\, x^* \in F(\overline{B(y^*)})$ tel que $\bar{\phi}(y^*) = c\, x^*$ (3). Comme la fonction multivoque $y \to F(\overline{B(y)})$ est inf-continue sur Γ, il existe une suite $\{x_k\}_{k\in\mathbb{N}'}$ telle que, $\forall\, k \in \mathbb{N}'$ $x_k \in F(\overline{B(y_k)})$, et $\lim\limits_{\substack{k\to\infty \\ k\in\mathbb{N}'}} x_k = x^*$: Enfin,

$x_k \in F(\overline{B(y_k)}) \Rightarrow \bar{\phi}(y_k) \geq c\, x_k$, et d'après (3), $\bar{\phi}(y^*) = c\, x^* = \lim_{\substack{k\to\infty \\ k\in\mathbb{N}'}} c\, x_k \leq \lim_{\substack{k\to\infty \\ k\in\mathbb{N}'}} \bar{\phi}(y_k) = \alpha$.

□

Lemme 5

$\forall\, w \in \mathbb{R}_+^{m_1}$, $\bar{s}(w) = \sup \{\bar{\phi}(y) \mid y \in \Gamma(w)\}$ *où* $\Gamma(w) = \{y \in \Gamma \mid w\, y \leq 0\}$

Démonstration

$$\forall\, w \in \mathbb{R}_+^{m_1},\ \bar{s}(w) = \max_{x\in\bar{X}} \{c\, x \mid w\,(Ax-b) \leq 0\}$$

$$= \sup_{y\in\Gamma,\, w\, y\leq 0} \{\max_{x\in\bar{X}} \{cx \mid Ax-b \leq y\}\}$$

$$= \sup_{y\in\Gamma(w)} \bar{\phi}(y)$$

si $\Gamma(w)$ désigne l'intersection du convexe Γ et du demi-espace fermé orthogonal à w.

□

Considérons pour $a \in R$, les ensembles de niveau associés respectivement à $\bar{s}$ et $\bar{\phi}$

$$L_a\bar{s} = \{w \in \mathbb{R}_+^{m_1} \mid \bar{s}(w) \leq a\},\ L_a\bar{\phi} = \{y \in \Gamma \mid \bar{\phi}(y) > a\}$$

ainsi que K intersection du cône polaire positif de $L_a\bar{\phi}$ et de l'orthant positif de $\mathbb{R}^{m_1}$, $K = \{w \in \mathbb{R}_+^{m_1} \mid w\, y \geq 0,\ \forall\, y \in L_a\bar{\phi}\}$.

Lemme 6

Soit $w \in K$: $\forall\, y_1 \in \Gamma$ *tel que* $w\, y_1 = 0$, alors $y_1 \notin L_a\bar{\phi}$.

Démonstration

$\forall\, w \in K$ le convexe $L_a\bar{\phi}$ est contenu dans l'un des demi-espaces de frontière l'hyperplan H d'équation $w\, y = 0$. $\Gamma \cap H \neq \emptyset$ car $0 \in \Gamma \cap H$; soit

$y_1 \in \Gamma \cap H$ et $x_1 \in F(\overline{B(y_1)})$. La continuité de l'application $x \to Ax-b$ sur le compact $\bar{X}$ assure l'existence d'un couple $(x_2, y_2) \in F(B(y_2) \times \Gamma$ tel que $Ax_2 - b = y_2 < 0$. $\forall\ \alpha \in]0, 1]$, considérons $y_\alpha = (1-\alpha)y_1+\alpha y_2$; $y \in \Gamma$ (lemme 1) mais $y_\alpha \notin L_a\bar{\phi}$ car $w\ y_\alpha = \alpha w\ y_2 < 0$ et par conséquent $\bar{\phi}(y_\alpha) \leq a$. Comme $\bar{\phi}$ est continue sur Γ (lemme 4), $\lim_{\alpha \to o^+} \bar{\phi}(y_\alpha) = \bar{\phi}(y_1) \leq a$ et $y_1 \notin L_a\bar{\phi}$.

□

Pour montrer que $\bar{s}$ est s.c.i sur $\mathbb{R}_+^{m_1}$, il suffit de montrer que $L_a\bar{s} = K$, $\forall\ a \in \mathbb{R}$. Comme K est un convexe ferme non vide, $L_a\bar{s}$ sera fermé pour tout réel a, condition équivalente à $\bar{s}$ s.c.i sur $\mathbb{R}_+^{m_1}$ (Théorème 7.1, [57]).

(i) $L_a\bar{s} \subset K$: soit $w \in L_a\bar{s}$ et supposons $\exists\ y \in L_a\bar{\phi}$ tel que $w\ y < 0$.

Alors $\bar{\phi}(y) > a$

$\bar{s}(w) \geq \bar{\phi}(y)$ (lemme 5)

$\Rightarrow \bar{s}(w) > a$ en contradiction avec l'hypothèse sur w.

(ii) $K \subset L_a\bar{s}$: soit $w \in K$

le lemme 6 implique en fait que $K = \{w \in \mathbb{R}_+^{m_1} \mid w\ y > 0, \forall\ y \in L_a\bar{\phi}\}$ et par conséquent $\{y \mid w\ y \leq 0\} \cap \Gamma \cap L_a\bar{\phi} = \emptyset$. Il s'ensuit que, puisque d'après le lemme 6, $\bar{s}(w) = \sup\{\bar{\phi}(y) \mid w\ y \leq 0 \text{ et } y \in \Gamma\}$, la borne supérieure peut être prise sur $\{y \mid w\ y \leq 0\} \cap \Gamma \cap \{y \mid \bar{\phi}(y) \leq a\}$. Par conséquent $\bar{s}(w) \leq a$.

□

Remarque :

Cette démonstration prouve aussi la quasi-convexité de $\bar{s}$ sur $\mathbb{R}_+^{m_1}$ puisque les ensembles de niveau $L_a\bar{s}$ sont convexes, $\forall\ a \in \mathbb{R}$.

□

Pour $w \in B_1$, considérons les ensembles des solutions optimales $\Omega(BS(w))$ et $\Omega(\overline{BS(w)})$ des problèmes respectifs $(BS(w))$ et $(\overline{BS(w)})$. Les ensembles des solutions réalisables $F(BS(w))$ et $F(\overline{BS(w)})$ étant compacts (proposition 1.6), la borne supérieure de la fonction linéaire $x \to cx$ est atteinte donc $\Omega(\overline{BS(w)})$ et $\Omega(\overline{BS(w)})$ sont tous deux non vides.

La notion de *quasi-sous-gradient* d'une fonction quasi-convexe, analogue à la notion de *sous-gradient* pour une fonction convexe, a été introduite en premier lieu par Greenberg et Pierskalla ([29]) en 1973. Ces deux notions sont en fait des cas particuliers du concept plus général de *gradient généralisé*, développé par Clarke ([4]) en 1975. La notion de quasi-sous-gradient nous permet de définir des directions de déplacement dans des méthodes de résolution du dual composite.

Posons $g(x) = b-Ax$ et considérons les ensembles suivants :
$S(w) = \{g(x) \mid x \in \Omega(BS(w))\}$ (fini) et $\bar{S}(w) = \{g(x) \mid x \in \Omega(BS(w))\}$.
La proposition suivante permet de caractériser les vecteurs $g(x)$ comme des quasi-sous-gradients respectivement des fonctions s et $\bar{s}$.

Proposition 1.8

Soit $w_o \in B_1$, *alors* $S(w_o) \subset \partial^* s(w_o)$ *et* $\bar{S}(w_o) \subset \partial^* \bar{s}(w_o)$.

Démonstration :

Utilisons la caractérisation (iv) de la proposition 13 (Annexe 1).
Soit $g_o \in S(w_o)$: $\exists\, x_o \in \Omega(BS(w_o))$ tel que $g_o = b-Ax_o$ et $S(w_o) = cx_o$.
$\forall\, w \in B_1$ tel que $wg_o \geq w_o g_o$ on a $w(Ax_o-b) \leq w_o(Ax_o-b) \leq 0$ donc $cx_o \leq s(w)$ et par conséquent $s(w) \geq s(w_o)$. Idem pour $\bar{s}(w_o)$.

□

Remarque

La proposition 1.8 est plus faible que la proposition 1.4. Une question ouverte serait de savoir sous quelles conditions l'enveloppe convexe de $S(w_o)$ (resp. $\bar{S}(w_o)$) coïncide avec le quasi-sous différentiel $\partial^* s(w_o)$ (resp. $\partial^* \bar{s}(w_o)$).

□

La proposition suivante caractérise géométriquement les quasi-sous-gradients.

Proposition 1.9

Soit H *l'hyperplan d'équation* $w \to g(w-w_o)$

(i) si $L^{o}_{s(w_o)}s \neq \emptyset$ *et si* $L^{o}_{s(w_o)}s = cl(L_{s(w_o)}s)$ *alors*

$(g \in \partial^* s(w_o) \Rightarrow$ H *est un hyperplan d'appui de* $L_{s(w_o)}s$ *en* $w_o)$

(ii) si $L^{-o}_{\bar{s}(w_o)}\bar{s} \neq \emptyset$ *alors*

$(g \in \partial^* \bar{s}(w_o) \Leftrightarrow$ H *est un hyperplan d'appui de* $L^{-}_{\bar{s}(w_o)}\bar{s}$ *en* $w_o)$

Démonstration

Vérifions les hypothèses de la proposition 9 de l'annexe 1.

$L_{s(w_o)}s$ est convexe car s est quasi-convexe en w_o. Comme $L_{s(w_o)}s \neq \emptyset$, d'après un résultat classique (proposition 2.17 bis [36])
$L^{o}_{s(w_o)}s = L_{s(w_o)}s$ (idem pour $\bar{s}$). De plus $\bar{s}$ étant s.c.i en w_o, $L^{-}_{\bar{s}(w_o)}\bar{s}$ est un fermé et par conséquent $L^{-o}_{\bar{s}(w_o)}\bar{s} = cl(L^{-}_{\bar{s}(w_o)}\bar{s})$. Enfin la s.c.i de $\bar{s}$ en w_o assure la condition suffisante en (ii).

□

Remarques

(i) Chaque élément de S(w) et de $\bar{S}(w)$ peut donc se caractériser géométriquement comme un vecteur normal à un hyperplan de support d'un ensemble de niveau approprié.

(ii) $\bar{s}$ étant continue, $L^{-o}_{\bar{s}(w_o)}\bar{s}$ est l'ouvert $\{w \in B_1 \mid \bar{s}(w) < \bar{s}(w_o)\}$
B_1 étant compact, cet ouvert sera assurément non vide si $\bar{s}(w_o) > v(\bar{D}_S)$, c'est à dire si $w_o \notin \Omega(\bar{D}_S)$.

□

Proposition 1.10

(i) $\Omega(\bar{D}_S)$ *est un convexe fermé non vide de* B_1.

(ii) $\Omega(D_S)$ *est un convexe non vide de* B_1

(iii) $\exists\ \varepsilon_o > 0$ *tel que* $\Omega(D_S) = W(\varepsilon) = \{w \in B_1 \mid s(w) < v(D_S) + \varepsilon\},\ \forall\ \varepsilon \leq \varepsilon_o$.

Démonstration

$\Omega(D_S) = \{w \in B_1 \mid s(w) = v(D_S)\}$, $\Omega(\bar{D}_S) = \{w \in B_1 \mid \bar{s}(w) = v(\bar{D}_S)\}$

- (i) découle des propriétés de $\bar{s}$ (proposition I.7)

. $\bar{s}$ s.c.i sur le compact B_1 => $\Omega(\bar{D}_S) \neq \emptyset$ (toute fonction s.c.i sur un compact y atteint sa borne inférieure)

. $\bar{s}$ quasi-convexe sur $\mathbb{R}_+^{m_1}$ => $\Omega(\bar{D}_S)$ convexe

. $\bar{s}$ continue sur B_1 => $\Omega(\bar{D}_S)$ fermé

- (ii)

. $F(B) \neq \emptyset$ donc $-\infty < v(B) \leq v(D_S)$; supposons que $v(D_S)$ ne soit pas atteint. $\{s(w) \mid w \in B_1\}$ est un sous-ensemble borné inférieurement de $\mathbb{R}$, on peut en extraire une suite décroissante $\{s(w^k)\}_{k\geq 0}$ convergeant vers $v(D_S)$, vérifiant, sous l'hypothèse faite, pour tout k $s(w^k) > v(D_S)$. Or $s(w^k) = c\ x^k$ pour au moins un $x^k \in \Omega(BS(w^k))$. Par conséquent, $v(D_S)$ serait un point d'accumulation du sous-ensemble *discret* $\{c\ x \mid x \in X\}$ ($c\ x \in \mathbb{N}$), ce qui est impossible. Donc $v(D_S)$ est atteint.

. s quasi-convexe sur $\mathbb{R}_+^{m_1}$ => $\Omega(D_S)$ convexe

- (iii) . $\Omega(D_S) \subset W(\varepsilon)$, $\forall\ \varepsilon > 0$

. $v(D_S)$ n'étant pas un point d'accumulation de $\{c\ x \mid x \in X\}$, il existe un voisinage "ouvert" de $v(D_S)$ ne contenant aucun point de $\{cx \mid x \in X\}$ distinct de $v(D_S)$, c'est à dire, $\exists\ \varepsilon_o > 0$ tel que $v(D_S) \leq c\ x < v(D_S) + \varepsilon$ => $v(D_S) = c\ x$.

Soit $w \in W(\varepsilon)$; $\exists\ x \in \Omega(BS(w))$ tel que $v(D_S) \leq s(w) = c\ x < v(D_S) + \varepsilon$ et d'après ce qui précède $v(D_S) = s(w)$ donc $w \in \Omega(D_S)$. Par conséquent $W(\varepsilon_o) \subset \Omega(D_S)$.

□

Remarque

s n'est pas s.c.i sur B_1 au contraire de $\bar{s}$ (proprosition 1.7).

1.2.4 - *Existence et comparaison des écarts de dualité*

Au paragraphe 1.2.1, nous avons rappelé à travers un certain nombre de théorèmes des conditions suffisantes pour que la valeur du problème primal soit égale à la valeur du problème dual (lagrangien ou composite). Ces conditions sont liées à la connaissance des solutions optimales des relaxations (lagrangiennes ou composites) associées au problème dual. Parallèlement à ce qui a pu être fait pour les programmes convexes, nous allons essayer de synthétiser les résultats essentiels, relatifs aux programmes non convexes, concernant l'existence et la valeur de l'écart de dualité. Ces propriétés seront liées essentiellement aux structures des programmes en question et des programmes perturbés associés. Elles nous permettront sous d'autres hypothèses de retrouver les conclusions du paragraphe 1.2.1.

Associés au problème *primal*

$$(P) \quad \begin{cases} \sup \ f(x) \\ \text{s.c. } a(x) \leq 0 \\ \quad x \in X \subset \mathbb{R}^n \end{cases}$$

on considérera le problème *perturbé*

$$(P(y)) \quad \begin{cases} \sup \ f(x) \\ \text{s.c. } a(x) \leq y \\ \quad x \in X \subset \mathbb{R}^n, \ y \in \mathbb{R}^m \end{cases}$$

et la fonction de perturbation associée $\phi(y) = v(P(y))$

le dual *lagrangien*

$$(D_L) \quad \begin{cases} \inf \ \ell(w) \\ \text{s.c. } w \in \mathbb{R}^m_+ \end{cases}$$

avec $\ell(w) = v(PL(w))$, où

$$(PL(w)) \quad \begin{cases} \sup \ f(x) - wa(x) \\ \text{s.c. } x \in X \end{cases}$$

le dual *composite*

$$(D_S) \quad \begin{cases} \inf \ s(w) \\ \text{s.c. } w \in B_1 \end{cases}$$

avec $s(w) = v(PS(w))$, où

$$(PS(w)) \quad \begin{cases} \sup f(x) \\ \text{s.c. } wa(x) \leq 0 \\ \qquad x \in X \end{cases}$$

Cas Convexe

a) dual lagrangien (voir Geoffrion [16])

Théorème 1.4

Si les conditions suivantes sont remplies

(i) f *concave*

(ii) a *convexe*

(iii) X *convexe non vide*

(iv) $\Omega(P) \neq \emptyset$

(v) $\exists\ x_o \in X,\ a(x_o) < 0$ *(condition de Slater ou contrainte de qualification)*

alors $\Omega(D_L) \neq \emptyset$ et $v(P) = v(D_L)$.

Démonstration

Soit $x^* \in \Omega(P)$ et posons $h(x) = f(x^*) - f(x)$, $\forall\ x \in X$

Le système $\begin{cases} h(x) < 0 \\ a(x) \leq 0 \\ x \in X \end{cases}$ est contradictoire puisque x^* est une solution optimale de (P)

Sous l'hypothèse (v), on peut appliquer le théorème de Farkas-Minkowski :

$$\exists\ w^* \in \mathbb{R}^m_+,\ h(x) + w^* a(x) \geq 0,\ \forall\ x \in X$$

soit $\quad f(x^*) \geq f(x) - w^* a(x) \quad \forall\ x \in X \quad (1)$

en particulier, pour $x = x^*$, on obtient $w^* a(x^*) \geq 0$ et comme $w^* \geq 0$ et $a(x^*) \leq 0$, il s'ensuit $w^* a(x^*) = 0$ (condition de complémentarité).

$\ell(w^*) = \sup \{f(x) - w^* a(x) \mid x \in X\} \leq f(x^*)$ d'après (1)

$\leq f(x^*) - w\, a(x^*) \quad \forall\, w \geq 0$ car $a(x^*) \leq 0$

$\leq \sup\{f(x) - w\, a(x) \mid x \in X\} = \ell(w), \forall\, w \geq 0$

donc $w^* \in \Omega(D_L)$.

Enfin $\ell(w^*) = \sup\{f(x) - w^* a(x) \mid x \in X\} \geq f(x^*) - w^* a(x^*)$

$\geq f(x^*)$ puisque $w^* a(x^*) = 0$

donc $\ell(w^*) = f(x^*)$, soit $v(P) = v(D_L)$.

□

Remarque

Sous l'hypothèse affaiblie "$\phi(0)$ fini" à la place des conditions (iv) et (v) Geoffrion a donné deux autres caractérisations intéressantes

P_1 : $w \in \Omega(D_L)$ et $v(D_L) = v(P) \iff w \in \partial\phi(0)$

P_2 : $v(P) = v(D_L) \iff \phi$ est s.c.s en 0

En particulier, la semi-continuité supérieure de ϕ en 0 est assurée si X est fermé, f et a continues sur X, $\phi(0)$ fini, F(P) borné.

□

Le programme linéaire $(\bar{B})$ vérifie les conditions ci-dessus et on retrouve bien sûr le résultat classique : $v(\bar{B}) = v(\bar{D}_L)$ (théorème 1.1). Or comme le programme (D_L) est équivalent à $(\bar{D}_L)$, $v(\bar{B}) = v(D_L)$.

b) dual composite

Théorème 1.5

Si les conditions suivantes sont remplies

(i) f *quasi-concave, s.c.i par ligne (*)*

(ii) a *convexe (ou quasi-convexe)*

(iii) X *convexe non vide*

(*) f *est* s.c.i par ligne *si* $\forall (x_1, x_2) \in X^2$ *la fonction* $\alpha \to f(\alpha x_1 + (1-\alpha)x_2)$ *est s.c.i sur* [0, 1]. *Toute fonction concave est s.c.i par ligne et quasi-concave.*

(iv) $\Omega(P) \neq \emptyset$

(v) $\exists\ x_o \in X,\ a(x_o) < 0$

alors $\Omega(D_S) \neq \emptyset$ *et* $v(P) = v(D_S)$.

Démonstration (d'après [47])

Soit $x^* \in \Omega(P)$; a étant quasi-convexe, $Y = \{y \mid \exists\ x \in X,\ a(x) \leq y\}$ est convexe ; f étant quasi-concave, $Z = \{y \mid \exists\ x \in X,\ a(x) \leq y,\ f(x) > v(P)\}$ est convexe Y est non vide, Z peut être supposé non vide sinon $\Omega(D_S) = \mathbb{R}^m_+$ et $Z \cap \mathbb{R}^m_- = \emptyset$.

Il existe un hyperplan séparateur $y \to w^* y$ $(w^* \neq 0)$ tel que

$$w^* y \geq 0 \quad \forall\ y \in Z \quad (1) \quad , \quad w^* y \leq 0 \quad \forall\ y \in \mathbb{R}^m_- \quad (2)$$

Une première conséquence directe de (2) est que $w^* \geq 0$.

Supposons qu'il existe $y_1 \in Y$ tel que $w^* y_1 = 0$ et soit $x_1 \in X$ tel que $a(x_1) \leq y_1$.

Posons $y_o = a(x_o) < 0$; comme $y_o, y_1 \in Y$ convexe, $y(\alpha) = \alpha y_o + (1-\alpha)y_1 \in Y$, $\forall\ \alpha \in [0, 1]$. $w^* y(\alpha) = \alpha\ w^* y_o < 0$ si $\alpha \neq 0$ donc $y(\alpha) \notin Z$ d'après (1) et (2). Puisque a est convexe, $a(\alpha x_o + (1-\alpha)x_1) \leq \alpha a(x_o) + (1-\alpha)\ a(x_1) = y(\alpha)$ et d'après la définition de Z, pour $\alpha \in\ ]0, 1]$, $f(\alpha x_o + (1-\alpha)x_1) \leq v(P)$. Par s.c.i de f sur $[x_o, x_1]$, $f(x_1) \leq \lim_{\alpha \to o} f(\alpha x_o + (1-\alpha)x_1) \leq v(P)$. Ainsi x_1 étant quelconque, $y_1 \notin Z$, et on a donc montré que $w^* y = 0 \Rightarrow y \notin Z$. Ajouté à (1) il s'ensuit que $w^* y \leq 0 \Rightarrow y \notin Z$.

Par conséquent $x \in X$, $w^* a(x) \leq 0 \Rightarrow f(x) \leq v(P)$ donc $s(w^*) \leq v(P)$.

Or $x^* \in X$ et $w^* a(x^*) \leq 0$ donc $f(x^*) \leq s(w^*) \leq v(P) = f(x^*)$, soit $v(P) = v(D_S)$. Enfin, $v(P) \leq s(w)$, $\forall\ w \geq 0$ donc $w^* \in \Omega(D_S)$.

□

Remarques

(i) La condition (i) du théorème 1.5 est plus faible que celle du théorème 1.4. Les conditions étant moins restrictives, on peut penser avoir plus souvent un écart de dualité nul dant le cas composite que dans le cas lagrangien.

(ii) La condition de complémentarité $w^* a(x^*) = 0$ n'est pas en général vérifiée au contraire du cas lagrangien.

Le programme linéaire $(\bar{B})$ vérifiant les conditions ci-dessus, on retrouve le résultat $v(\bar{B}) = v(\bar{D}_S)$ (théorème 1.2).

Cas non convexe

a) dual lagrangien

Théorème 1.6

Si les conditions suivantes sont remplies

(i) f *continue (ou s.c.s)*

(ii) a *continue*

(iii) X *compact (non vide)*

si ϕ *est concave alors* $v(P) = v(D_L)$.

Démonstration

Notons $[f, a] = \{(z, y) \in \mathbb{R} \times \mathbb{R}^m \mid \exists\, x \in X,\ f(x) \geq z \text{ et } a(x) \leq z\}$ et $co([f, a])$ son enveloppe convexe. On a les égalités suivantes

$$epi(\phi) = \{(z, y) \in \mathbb{R} \times \mathbb{R}^m \mid z \leq \phi(y)\}$$

$$= \{(z, y) \mid \exists\, x \in X,\ f(x) \geq z \text{ et } a(x) \leq y\} = [f, a]$$

et

$$\phi(y) = \sup \{z \mid \in x \in X,\ f(x) \geq z \text{ et } a(x) \leq y\}$$

$$= \sup \{z \mid (z, y) \in [f, a]\}$$

La fonction de perturbation convexifiée $\Psi(y) = \sup\{z \mid (z,y) \in co[f,a]\}$ se caractérise comme l'enveloppe convexe supérieure de ϕ.

Sous les hypothèses f, a continues et X compact, un résultat classique (Shapiro [60]) établit l'équivalence entre le problème (P) convexifié et le dual lagrangien (D_L) et par conséquent $v(D_L) = \Psi(0)$.

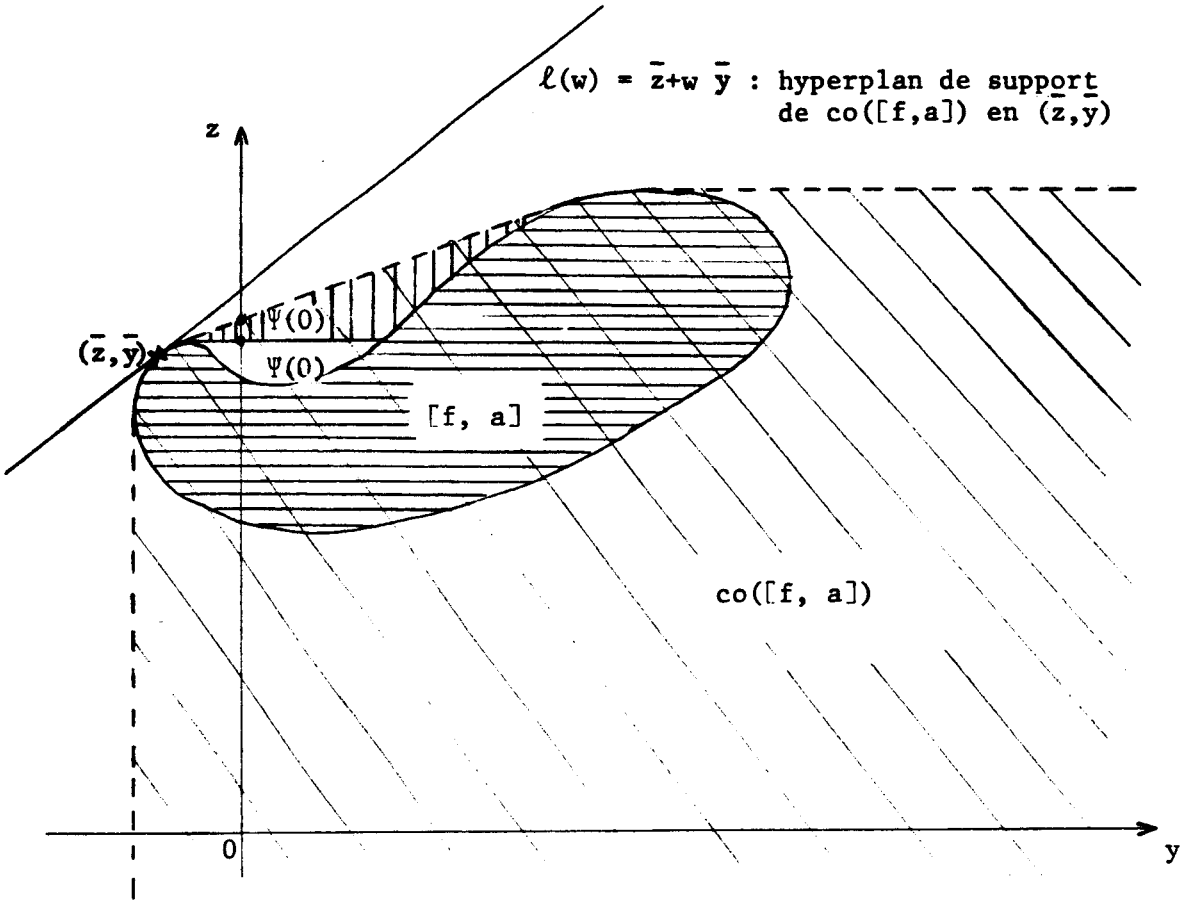


Figure 1.2 : représentation de [f, a] et co([f, a])

Par conséquent, si ϕ est concave, ϕ coïncide avec son enveloppe convexe supérieure et donc $v(B) = \phi(0) = \Psi(0) = v(D_L)$.

□

Dans le cas où $f(x) = cx$ et $a(x) = Ax-b$, Ψ n'est autre que la fonction de perturbation $\tilde{\phi}$ associée au programme convexe pertrubé $\tilde{P}(y)$ défini par

$$(\tilde{P}(y)) \quad \begin{cases} \max \ cx \\ \text{s.c. } Ax-b \leq y \\ x \in co(X) \end{cases}$$

Ici l'écart de dualité est précisément la différence évaluée à l'origine entre ϕ et $\tilde{\phi}$ (Geoffrion donne une démonstration directe dans [17]). D'autre part dans le cas particulier des programmes en variables bivalentes il est

bien connu que ϕ n'est pas concave bien que non décroissante, s.c.s et affine par morceaux. Avec l'exemple 1.4 ci-dessous, on peut suivre l'évolution comparée de ϕ et $\overset{\vee}{\phi}$ sur la figure 1.3 ainsi que les connexions entre les ensembles X, [f, a] et co([f, a]) sur la figure 1.4.

Exemple 4

$$P(y)\begin{cases} \max \; 3x_1 + 7x_2 + 10x_3 \\ \text{s.c.} \; x_1 + 3x_2 + 5x_3 - 7 \leq y \\ x_1 \, , \; x_2 \, , \; x_3 \in V \end{cases}$$

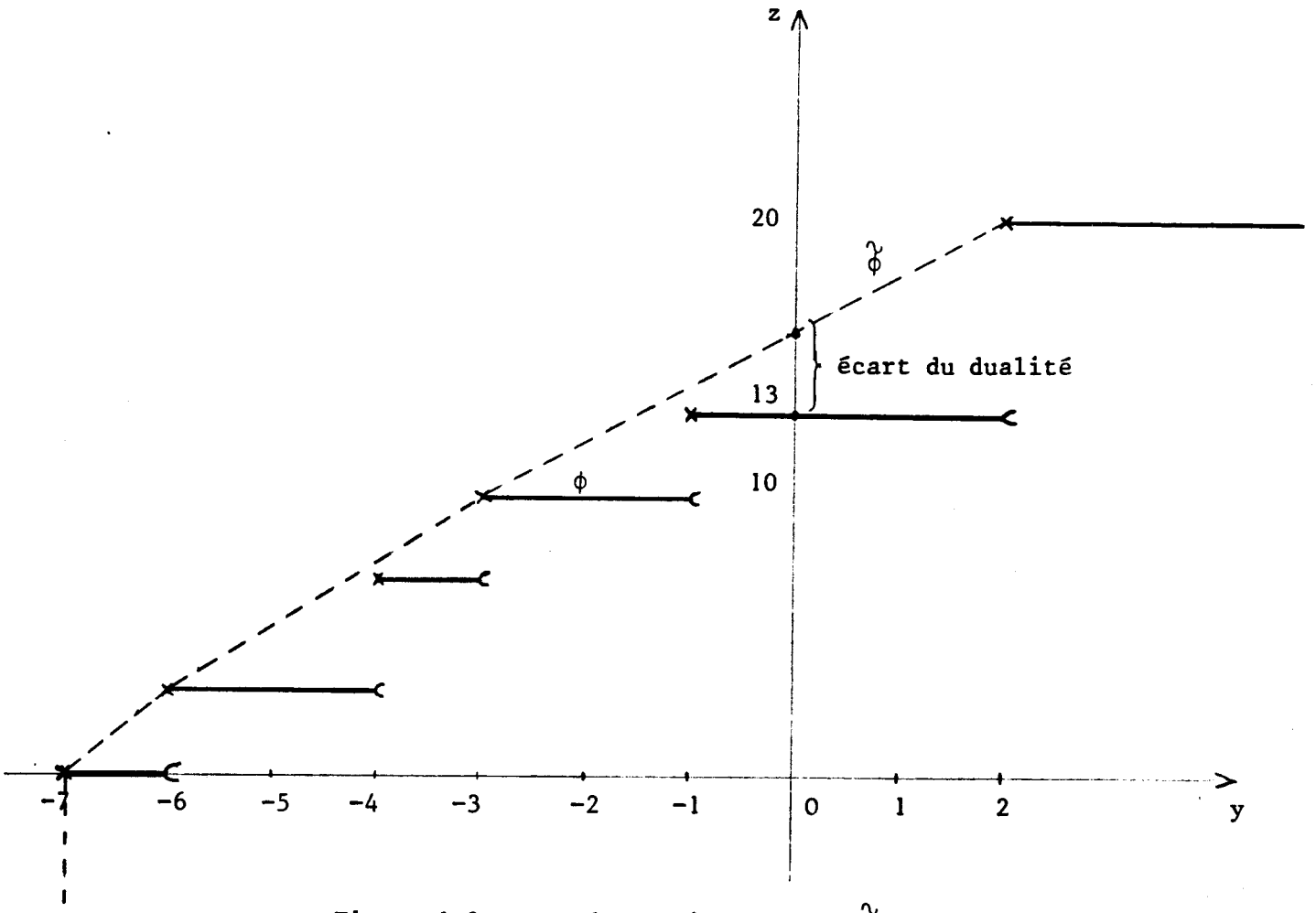


Figure 1.3 : Représentation de ϕ et $\overset{\vee}{\phi}$

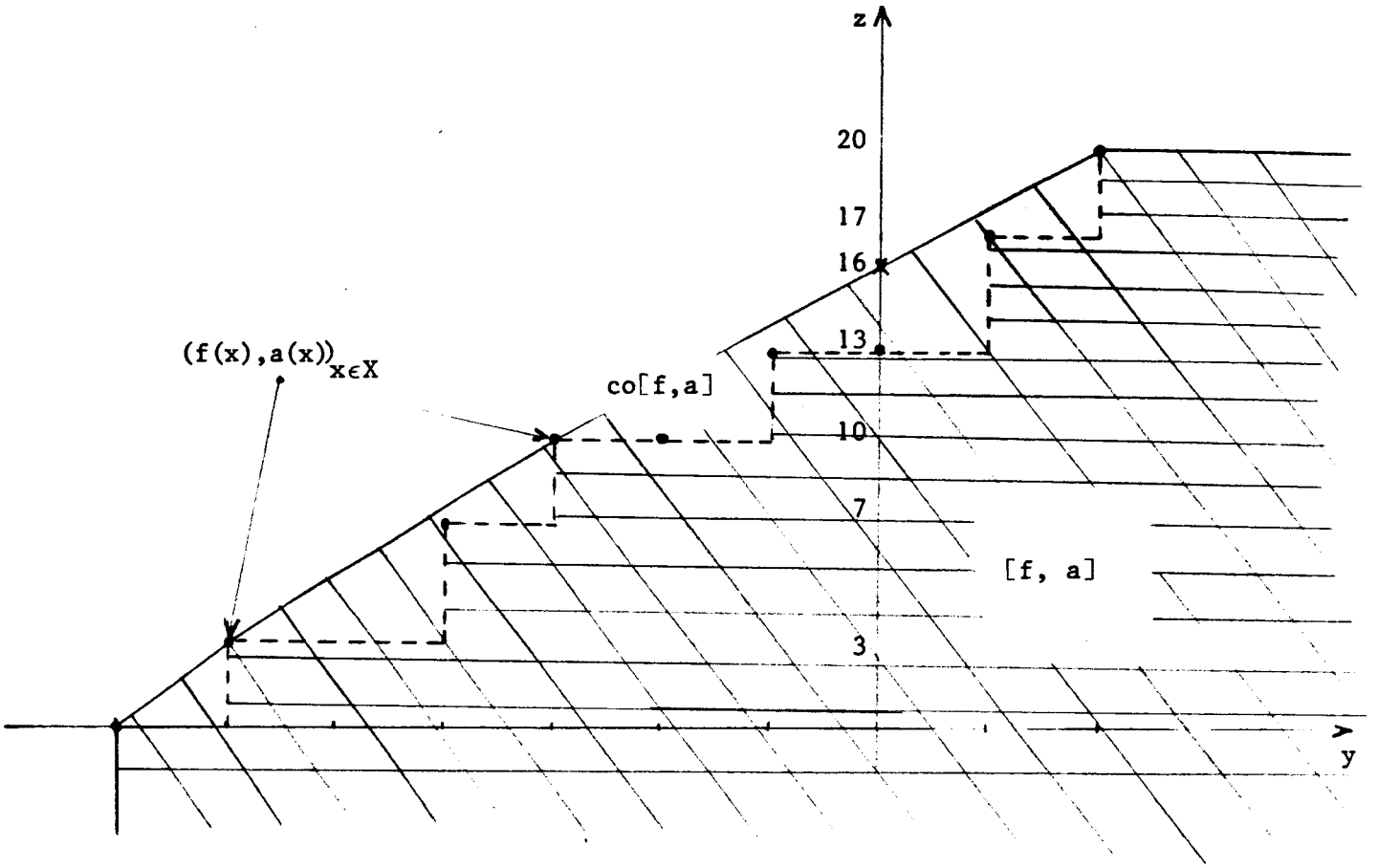


Figure 1.4 : X, [f, a], co([f, a])

b) dual composite

Posons $y(w) = \{y \mid w\,y \leq 0 \text{ et } \exists\, x \in X : a(x) \leq y\}$ pour $w \in B_1$.

Notons C(w) l'enveloppe convexe de $\{a(x) \mid x \in \Omega(PS(w))\}$.

Théorème 1.7

Si les conditions suivantes sont remplies

(i) f *continue (ou s.c.s)*

(ii) a *continue*

(iii) X *compact*

(iv) $0 \in C(w^*)$ *où* $w^* \in \Omega(D_S) \neq \emptyset$

(v) ϕ *s.c.s et quasi-concave*

alors $v(P) = v(D_S)$.

Démonstration (corollaire 2.1, [27])

Donnons simplement les lignes essentielles de la démonstration :

$\forall\ w \in B_1$, l'ensemble $\phi(w) = \{y \mid \phi(y) \leq \phi(y'),\ \forall\ y' \in Y(w)\} \cap Y(w)$ est fermé et convexe. D'autre part $\forall\ x^* \in \Omega(PS(w^*))$, $a(x^*) \in \phi(w^*)$. Si $0 \in C(w^*)$ alors $0 \in \phi(w^*) \subset Y(w^*)$ et par conséquent $v(P) = v(D_S)$.

□

1.2.5 - Relations entre multiplicateurs lagrangiens et composites

Si l'essentiel est de réduire le gap lagrangien $\tau_L = v(D_L) - v(B)$; un multiplicateur optimal de (D_L) peut être un bon multiplicateur pour démarrer une recherche d'un multiplicateur optimal de (D_S) comme le suggère la proposition ci-dessous (Dyer, [6]). Elle justifie a posteriori l'emploi des multiplicateurs optimaux de $(\bar{B})$ comme multiplicateurs composites en programmation en nombres entiers (Glover, [18]).

Proposition 1.11

Soit $w^* \in \Omega(D_L)$ *et* $\tilde{w} = w^* / ||w^*||_1 \in B_1$.

Alors $v(BS(\tilde{w})) \leq v(D_L)$.

De plus, exactement une seule des propositions suivantes est vraie

(i) $v(BS(\tilde{w})) < v(D_L)$

(ii) $v(BS(\tilde{w})) = v(D_L)$ *mais chaque voisinage de* $\tilde{w}$ *dans* B_1 *contient un multiplicateur* w *tel que* $v(BS(w)) < v(D_L)$

(iii) $v(BS(\tilde{w})) = v(D_L) = v(D_S)$.

Démonstration

Proposition 11 dans [6].

□

Remarque

$\tilde{w}$ peut très bien n'avoir aucun rapport avec l'ensemble des meilleurs multiplicateurs composites (voir exemple 3 dans [6]).

□

METHODES DE RESOLUTION DES RELAXATIONS

CHAPITRE 2

INTRODUCTION

Ce chapitre est consacré à l'étude des principales méthodes de résolution des relaxations du problème initial que l'on peut mettre en oeuvre pour engendrer des multiplicateurs indispensables à l'élaboration du code de réduction présenté au chapitre 4.

Après avoir fait le tour des méthodes attachées à chaque type de relaxation (§ 2.1), la deuxième partie (§ 2.2) présente de manière détaillée deux procédures similaires de type sous-gradient qui résolvent les duaux lagrangiens et composites. Les nombreuses expériences numériques nous ont permis d'une part, dans le cas lagrangien de comparer et de vérifier le bon comportement numérique des règles bien connues pour le choix des coefficients de relaxation, et d'autre part d'établir des règles semblables dans le cas composite pour lequel aucune expérience numérique n'avait été menée jusqu'à ce jour, du moins à notre connaissance. Enfin quelques remarques permettent de dégager l'intérêt d'utiliser ce type de méthode pour des problèmes de grande taille.

2.1 - Généralités

Considérons le problème en variables bivalentes suivant :

$$(B)\quad \begin{cases} \max\ cx & (1) \\ \text{s.c. } Ax \leq b & (2) \\ x \in V & (3) \end{cases}$$

avec $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times n}$.

2.1.1 - Résolution du primal

2.1.1.1 - Méthode du simplex

La voie classique pour résoudre le programme linéaire $(\bar{B})$ est la méthode du simplex énoncée en 1947 par DANTZIG et depuis lors largement

développée et améliorée. Cette méthode construit un chemin le long des arêtes du polyèdre convexe définie par le système (2) et (3), de telle façon que la suite des valeurs de la fonction économique (1) attachées à chaque point extrême rencontré soit non décroissante. A chaque itération, la méthode examine la possibilité d'améliorer la valeur de la fonction économique en se déplaçant vers un des voisins de point extrême courant. L'optimum du programme linéaire est atteint en un sommet du polyèdre convexe, sauf lorsque celui-ci est vide ou lorsque la solution de (P) est non-bornée. Il faut donc dans le plus mauvais des cas $\binom{n+m}{m} \equiv 0\left(\binom{m}{(n+m)}\right)$ itérations pour atteindre cet optimum. Par conséquent, la complexité théorique de la méthode simpliciale n'est pas polynômiale. En pratique, cependant, la méthode s'avère beaucoup plus efficace que ne le laisse supposer sa borne supérieure. En effet, le nombre moyen d'itérations requises varie entre m et 3m, et à chaque itération le nombre d'opérations arithmétiques est borné par une fonciton polynômiale des paramètres n et m. De fait la complexité pratique de la méthode est polynômiale bien que sur l'exemple ci-dessous, le nombre d'itérations soit exponentiel.

Exemple 2.1 : (Jens Clausen [49])

$$\begin{cases} \max \sum_{j=1}^{n} c_j x_j \\ \text{x.c.} \sum_{j=1}^{n} a_{ij} x_j \leq 5^{i-2}, \ \ldots \ i \in \{1, \ldots, n\} \\ x \geq 0 \end{cases}$$

avec

$$\begin{cases} c_j = \left(\frac{4}{5}\right)^{j-1} \\ a_{ii} = 1 \\ a_{ij} = \begin{cases} 2\left(\frac{5}{4}\right)^{i-j} & \text{pour } i < j \\ 0 & \text{sinon} \end{cases} \end{cases}$$

2.1.1.2 - Méthodes de relaxation et de Khachyan

Le dual (D) du programme linéaire (P) défini au § 2.1.1.1 est le programme linéaire

$$(D)\quad \begin{cases} \min\ ub \\ \text{s.c. } uA \geq c \\ \qquad u \geq 0 \end{cases}$$

et si $F(P) \neq \emptyset$ alors $v(P) = v(D)$.

Le système (S) défini par

$$(S)\quad \begin{cases} cx \geq ub \\ Ax \geq b \\ x \geq 0 \\ uA \geq c \\ u \geq 0 \end{cases}$$

admet une solution (x^*, u^*) si $F(P) \neq \emptyset$ et alors $cx^* = v(P) = v(D) = u^* b$. On peut donc en déduire que résoudre un programme linéaire (P) revient à résoudre un système associé (S) d'inéquations au sens large, dans le cas présent au nombre de $2(m+n)+1$.

Une première classe de méthodes pour résoudre le problème

[Trouver une solution réalisable du système d'inégalités : (S) : $A_i x + b_i \geq 0$ $\quad i \in I$ fini, $b_i \in \mathbb{R}$, $A_i \in \mathbb{R}^n$]

est celle des relaxations. Les articles de base relatifs à leur étude sont dus à Agmon (1954, [1]) et Motzkin-Schoenberg (1954, [52]) et on trouvera une récente généralisation de ces recherches dans les travaux de Goffin (1980, [23]).

Désignons par $F(S) = \{x \in \mathbb{R}^n \mid A_i x + b_i \geq 0,\ i \in I\}$ l'ensemble supposé non vide des solutions réalisables du système d'inégalités et par H^i l'hyperplan d'équations $A_i x + b_i = 0$.

Le schéma général d'une méthode de relaxation se traduit sous la forme de l'algorithme suivant :

Algorithme de relaxation

1 *Choisir* $x^0 \in \mathbb{R}^n$; $q \leftarrow 0$

2 *si* $x^q \in F(S)$ *alors* *STOP*

sinon $$x^{q+1} \leftarrow s^q + \lambda q \times d(x^q, H^{i_q}) \times \frac{\alpha_{i_q}}{||\alpha_{i_q}||} \quad \text{où } \lambda q \in]0, 2]$$

$q \leftarrow q+1$; *aller en 2.*

notes

- $\alpha_{i_q} / ||\alpha_{i_q}||$ désigne un vecteur unitaire normal à l'hyperplan H^{i_q} (figure 2.1)
- en général la suite $\{\lambda_q\}_{q \geq o}$ est constante.

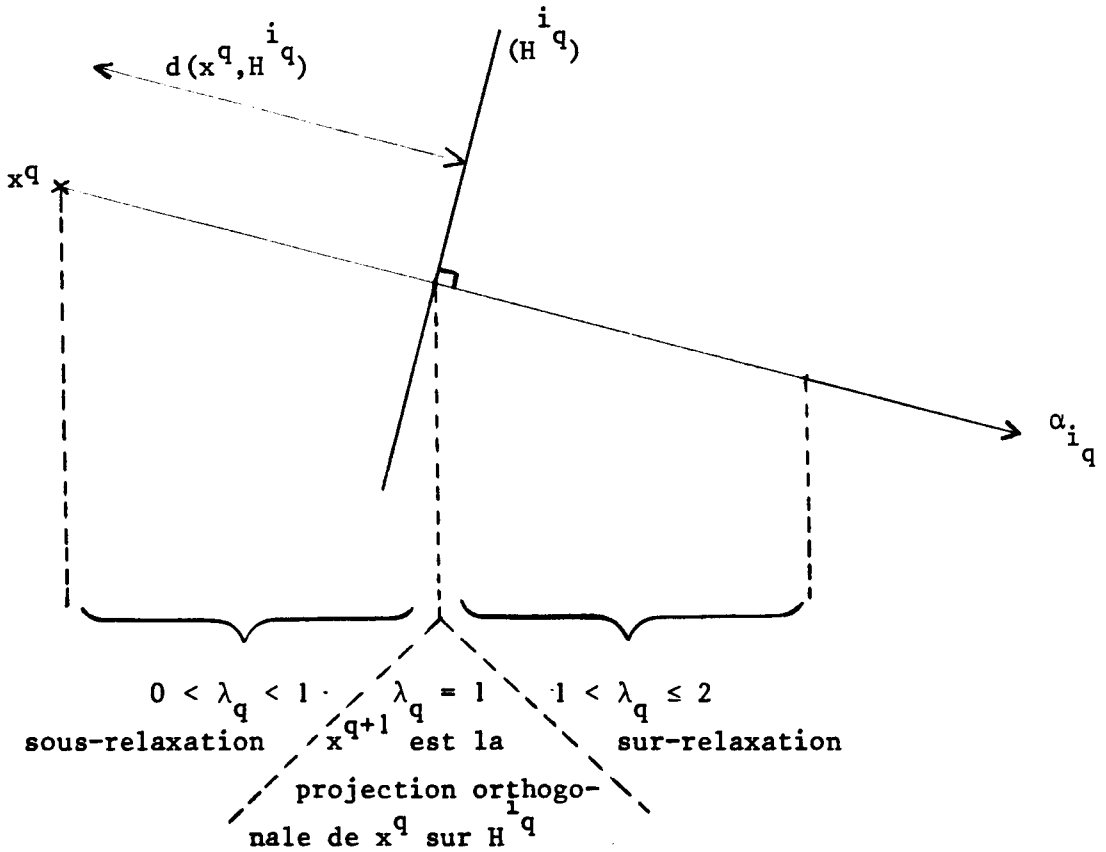


Figure 2.1

Agmon a proposé trois choix pour i_q

(i) distance maximale : $d(x^q, H^{i_q}) \geq d(x^q, H^i)\ \forall\ i \in I$

(ii) résidu maximum : $A_{i_q} x^q + b_{i_q} \leq A_i x^q + b_i$ (contrainte la plus violée)

(iii) méthode cyclique : $i_q = q \pmod m + 1$

Selon les hypothèses sur F(S) et λ, l'algorithme est fini ou convergent. Un des intérêts essentiels de ces méthodes est d'avoir donné naissance à des méthodes de type sous-gradient utilisées avec succès pour certains programmes d'optimisations convexes non différentiables (Held et Karp, [31]).

Une autre classe de méthodes beaucoup plus récentes est due à Khachyan (1979, [39]). Elle permet de déterminer la vacuité d'un système d'inégalités strictes. Or comme on a l'équivalence suivante entre système d'inégalités au sens large et système d'inégalités au sens strict :

soit $L = \sum_{i,j} \log_2(|a_{ij}|+1) + \sum_i \log_2(|b_i|+1) + \log_2 mn + 1$

alors le système d'inégalités linéaires

(S) $A_i x \leq b_i$, $i = 1, 2, \ldots, m$, $x \in \mathbb{R}^n$

a une solution si et seulement si le système d'inégalités strictes

(S') $A_i x < b_i + 2^{-L}$ $i = 1, 2, \ldots, m$

a une solution

La méthode de Khachyan (encore appelée méthode ellipsoïdale) permet, en résolvant le système (S'), de traiter le système (S).

Algorithme

1 *Poser* $x^o = 0$; $Q^o = 2^{2L} U$; $k = 0$ (U *matrice unité)*

2 *(critère d'arrêt)*

si x^k *est solution du système* (S') *alors STOP*

si $k < 4(n+1)^2 L$ *alors aller en 3*

sinon (S') *ne possède pas de solutions réalisables : STOP*

3 *choisir une inégalité de* (S') *violée par* x^k : $A_{i_k} x^k \geq b_{i_k}$

calculer

$$x^{k+1} = x^k - \frac{1}{n+1} \frac{Q^k A_{i_k}}{\sqrt{{}^t A_{i_k} Q^k A_{i_k}}}$$

$$Q^{k+1} = \frac{n^2}{n^2-1} \left[Q^k - \frac{2}{n+1} \left[\frac{(Q^k A_{i_k})\, {}^t(Q^k A_{i_k})}{{}^t A_{i_k} Q^k A_{i_k}} \right] \right]$$

$k \leftarrow k+1$ *et* *aller en 2*

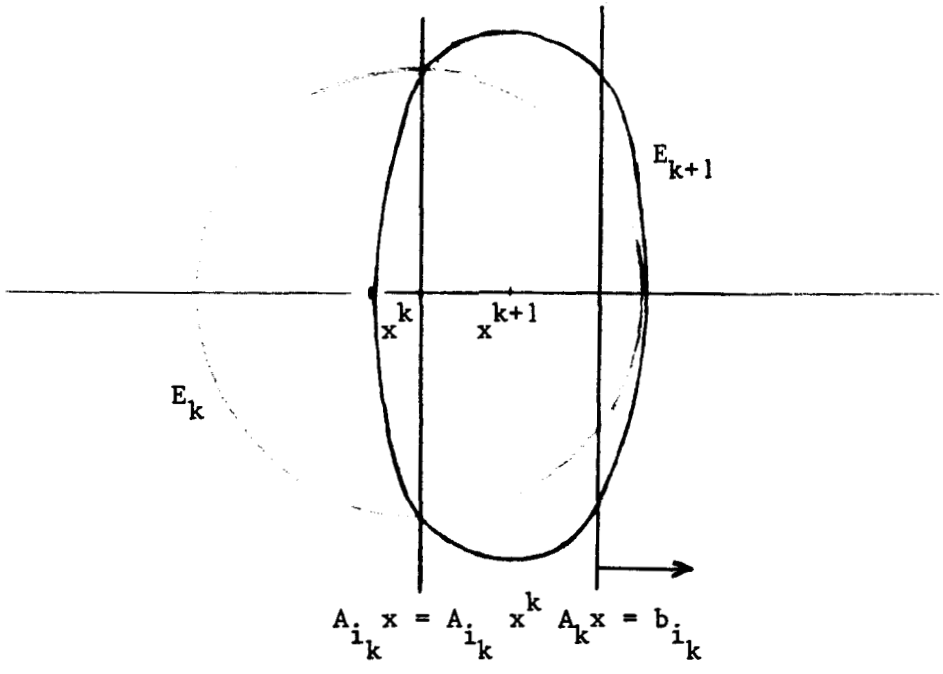


Figure 2.2 : Interprétation géométrique de l'algorithme

A le $k^{ième}$ itération, l'ellipsoide $E_k = \{x \mid {}^t(x-x^k)\ (Q^k)^{-1}\ (x-x^k) \leq 1\}$ est tel que, si il existe une solution réalisable pour (S'), elle ext contenue dans E_k. Le test de réalisibilité est fait sur x^k centre de E_k. Dans le cas où x^k est non réalisable, on construit un hyperplan passant par x^k et parallèle à l'hyperplan violé. Cet hyperplan partage E_k en deux demi-ellipsoïdes, dont l'un ne contient certainement pas de solution réalisable. On construit alors un nouvel ellipsoïde E_{k+1} circonscrit à l'autre demi-ellipsoïde, soit l'itération (k+1).

On trouvera dans ([39], [40], [41]) des études détaillées de cet algorithme.

2.1.1.3 - Remarques sur la complexité

Dès sa parution cet algorithme a suscité un intérêt assez extraordinaire car c'est en théorie un algorithme de complexité polynomiale. Jusqu'à cette date la complexité exponentielle de la méthode simpliciale induisait l'appartenance de la programmation linéaire à la classe *NP* (voir note ci-dessous) si l'on convient d'exprimer la taille d'un programme linéaire à partir du nombre de variables et de contraintes. Par contre il faut au plus $4\ (n+1)^2\ L$ itérations pour réduire le volume de l'hypersphère initiale E_o à moins de la valeur $2^{-(n+1)L}$, majorant du volume de l'ensemble des solutions appartenant à l'hypercube $\{x \mid |x_2| \leq 2^L, \forall\ i\}$ dont on est sûr qu'il contient au moins une solution du système (S'), s'il en existe.

Dès lors l'algorithme de Khachyan plaçait la programmation linéaire dans la classe *P* si l'on prend en compte, dans la taille du programme, la longueur L du codage des données. C'est là que réside l'avantage théorique de la méthode de Khachyan sur l'algorithme du simplex.

Malheureusement comme la programmation linéaire n'appartenait pas à la classe des problèmes *NP*-complet, le problème fondamental de la théorie de la complexité - a t' on classe *NP* = classe *P* ? - reste ouvert.

De plus l'algorithme de Khachyan peut difficilement dans l'état actuel des choses rivaliser sur le plan pratique avec l'algorithme du simplex. En particulier le nombre très important d'itérations, même pour des problèmes de très petite taille, entraine de très facheux problèmes d'arrondi (voir [41]).

Note : Rappel des définitions essentielles relatives à la théorie de la complexité. Cook et Karp (voir [9]) sont à la base de la théorie de la complexité et plus particulièrement de la définition des classes de problèmes permettant d'appréhender a priori la difficulté de résoudre ces problèmes par des algorithmes dits déterministes, c'est à dire les algorithmes classiques pour lesquels le résultat de chaque opération est défini de manière unique. On ne donnera ci-dessous que les quelques notions utilisées dans l'exposé, le lecteur intéressé pouvant trouver dans le livre de Garey et Johnson (1979) une analyse plus complète de cette théorie.

Définition 2.1

Soit un problème (P) *de taille* t *(nombre de variables, de contraintes, longueur d'encodage des données).*

Si pour un algorithme, il existe $k \in \mathbb{R}$, $f : \mathbb{N} \to \mathbb{N}$, $to \in \mathbb{N}$ *tels que*

$\forall\ t \geq to$ *Temps de calcul* $\leq k\ f(t)$

alors cet algorithme est dit de complexité en $O(f(t))$.

par exemple si

f ≡ polynôme => algorithme de complexité polynôniale

f ≡ exponentielle => algorithme de complexité exponentielle

Définition 2.2

La classe P *est celle des problèmes pour lesquels il existe un algorithme (déterministe) de complexité polynômiale.*

Définition 2.3

La classe NP *est celle des problèmes pour lesquels il existe un algorithme non déterministe de complexité polynômiale.*

La notion d'algorithme non déterministe est équivalente à la notion d'algorithme déterministe sous l'hypothèse (non réalisée jusqu'ici !) d'utiliser un ordinateur capable de traiter en parallèle un nombre de calculs aussi grand que l'on veut.

Définition 2.4

L'ensemble des problèmes NP-complet est une sous-classe de la classe NP telle que : $\forall$ (Q) $\in$ NP-*complet, s'il existait un algorithme déterministe de complexité polynomiale pour* (Q), *alors il en existerait un pour tout problème de la classe NP.*

La conjecture actuelle est que P est strictement inclus dans NP. Pour avoir l'égalité il faudrait trouver, si c'est possible, un algorithme déterministe de complexité polynômiale résolvant un problème NP-complet.

Les problèmes NP-complet sont donc théoriquement les plus difficiles à résoudre. Les programmes d'optimisation en nombres entiers et plus particulièrement ceux de la sous-classe faisant l'objet de notre étude, appartiennent à la classe des problèmes NP-complet.

□

2.1.2 - Résolution de la relaxation lagrangienne

Cette partie est consacrée à une synthèse des principales méthodes utilisées pour résoudre le programme dual

$$(D_L) \quad \begin{cases} \min \ \ell(w) \\ \text{s.c. } w \geq 0 \end{cases}$$

dans le cas où ℓ est une fonction convexe, linéaire par morceaux. Ce type de problème se rencontre souvent en programmation mathématique et correspond en particulier à celui de notre étude. La non-différentiabilité de la

fonction lagrangienne ℓ (bien que presque partout différentiable) ne permettant pas l'utilisation des méthodes de descente classiques dans le cas différentiable (plus forte pente, gradient conjugué, métrique variable, etc ...), nous exposerons les principales caractéristiques des méthodes, de descente ou non, valides dans le cas non différentiable. Toutefois nous mettrons l'accent plus particulièrement sur les méthodes de sous-gradient car ce sont deux algorithmes de ce type qui seront étudiés en détail au § 2.2.

L'essentiel des méthodes proposées dans la littérature peut se scinder en deux classes :

(1) Les méthodes de sous-gradients

Supposons que w_o ne minimise pas ℓ et soit $\bar{w}$ un moint meilleur que w_o, c'est à dire vérifiant $\ell(w_o) > \ell(\bar{w})$.
Si g est un sous-gradient de ℓ en w_o, on a : $\ell(\bar{w}) - \ell(w_o) \geq g\ (\bar{w}-w_o)$.
On en déduit que $g\ (\bar{w}-w_o) < 0$; c'est à dire que l'ensemble des points meilleurs se situe dans le demi-espace ouvert de frontière l'hyperplan $g\ (w-w_o) = 0$, en fait dans un certain cône car la relation est vraie $\forall\ g \in \partial\ell(w_o)$. Sur la figure 2.3, on voit que la direction -g pointe nécessairement à l'intérieur de la sphère de centre $\bar{w}$ et passant par w_o ; c'est dire que $w_1 = w_o - \lambda g$ sera plus près de $\bar{w}$ que w_o pour λ appartenant à un certain voisinage de 0. On a ainsi le germe géométrique des méthodes de Polyak (1967, [55]) que l'on peut statuer sous la forme suivante :

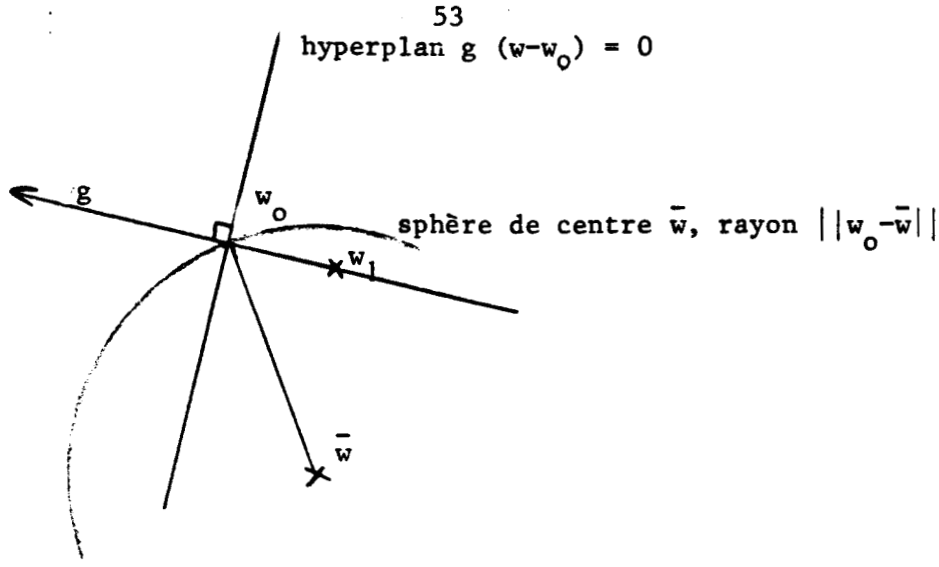


Figure 2.3 : Interprétation géométrique de la méthode de Polyak

Algorithme de sous-gradient

1 *Choisir* $w_o \in \mathbb{R}^m_+$ *et une suite* $\{\lambda_k\}_{k\geq 1}$ *de réels positifs ;* $k \leftarrow 0$

2 *Calculer un sous-gradient* g_k *de* ℓ *en* w_k

si $g_k = 0$ *alors* w_k *est optimal : STOP*

3 $w_{k+1} \leftarrow w_k - \lambda_k g_k$; $k \leftarrow k+1$

si $w_{k+1} \notin \mathbb{R}^m_+$ *alors projeter* w_{k+1} *sur* $\mathbb{R}^m_+$; *aller en 2*

Remarque

Théoriquement l'introduction d'un opérateur de projection dans le cas avec contraintes ne modifie pas la méthode. Par contre si le convexe relatif aux solutions admissibles n'est pas l'orthant positif $\mathbb{R}^m_+$, le calcul du projeté peut s'avérer un problème difficile.

□

L'interprétation géométrique indique que la convergence vers un point optimal w^* de la suite $\{w_k\}_{k\geq o}$ sera assurée si λ_k n'est pas trop grand mais aussi pas trop petit pour pouvoir se rapprocher suffisamment du point w^*.

Le premier choix proposé par Polyak [55] est le suivant :

$$\sum_{k=o}^{\infty} \lambda_k = +\infty \ , \quad \lim_{k\to+\infty} \lambda_k = 0$$

Pour remédier au problème d'une faible vitesse de convergence, d'autres choix pratiques et théoriques ont été proposés :

$$\lambda_k = \rho_k \times \frac{\ell(w^k) - \gamma}{||g_k||^2} \times g_k \quad \text{où } \gamma \text{ est une estimation de } v(D_L)$$

c1 valeur exacte : $\gamma = v(D_L)$ ([1] ; [52])

Polyak (1969, [56]) a démontré pour le choix suivant des coefficients de relaxation $\varepsilon_1 \leq \rho_k \leq 2-\varepsilon_2$ (ε_1, $\varepsilon_2 > 0$), la convergence géométrique de la suite $\{w_k\}$, résultat essentiellement d'intérêt théorique puisque $v(D_L)$ n'est en général pas connu.

c2 sur-estimation : $\gamma > v(D_L)$

En particulier γ peut être égal à la valeur d'une relaxation lagrangienne du problèmme primal (P). Des résultats de convergence ont été établis dans ce cas par Polyak (1969, [56]).

c3 sous-estimation : $\gamma > v(D_L)$

On prend pour γ la valeur $\underline{v}(P)$ correspondant à la meilleure solution réalisable connue de (P) (obtenue par exemple par heuristique). Ce choix a été validé par les travaux de Held, Wolfe et Crowder (1974, [32]).

Ces méthodes ont pour elles leur grande simplicité d'implantation lorsque à chaque itération le calcul de $\ell(w_k)$ est facile et elles ont donné pour un certain nombre de problèmes d'excellents résultats pratiques.
Par contre, elles présentent un certain nombre d'inconvénients :

- le sous-gradient-g_k ne définit pas une direction de descente pour la fonction lagrangienne ℓ et par conséquent la suite $\{\ell(w_k)\}_{k\geq o}$

n'est pas monotone décroissante. Le choix de la direction $-g_k$ se justifie par le fait que $-g_k$ est une direction de descente pour la fonction $\frac{1}{2}\delta^2(w)$, où $\delta(w)$ est la distance du point w à l'ensemble convexe $\Omega(D_L)$ des points optimaux. En effet, si $\tilde{w}$ désigne la projection de w sur $\Omega(D_L)$, on a

$$0 > \ell(w) - \ell(w) \geq g\ (\tilde{w}-w)$$

$$\nabla(\tfrac{1}{2}\delta^2(w)) = \nabla(\tfrac{1}{2}\ ||w-\tilde{w}||^2) = w-\tilde{w}$$

par conséquent $-g\ \nabla(\frac{1}{2}\delta^2(w))^2 < 0$.

et $\delta(w)$, bien que non calculable, a le même ensemble de points optimaux que ℓ.

- la direction $-g_k$ n'étant pas de descente, il n'y a pas lieu de faire de recherche linéaire. Le choix des pas de déplacement λ_k (ou ρ_k) se fait généralement suivant des régles a priori difficiles à justifier en théorie. Toutefois Held-Karp ([31]) et Legendre-Minoux ([43]) ont proposé dans le cas (c3) des régles expérimentalement très satisfaisantes. (Ces techniques seront précisées lors de la mise en oeuvre numérique de l'algorithme de sous-gradients, détaillé au § 2.2).

- impossibilité de vérifier que le point w^* obtenu satisfait la condition d'optimalité $0 \in \partial\ell(w^*)$ en exhibant une combinaison convexe nulle d'un certain nombre de sous-gradients (proposition 1.4).

(2) Les méthodes de descente

Contrairement aux méthodes de type sous-gradient, elles génèrent une suite $\{\ell(w^k)\}$ finie ou infinie qui est monotone décroissante, et nécessitent à chaque itération

- une phase de recherche de la direction de descente
- une phase de minimisation, exacte ou approchée, dans la direction définie précédemment.

Peuvent se rattacher à cette classe :

en premier les méthodes dites de ε-sous-gradients développés par Lemarechal (1974, [45]) et Wolfe (1975, [67]). Pour ces dernières, la détermination de la direction de descente en un point w nécessite la recherche d'un vecteur de norme euclidienne minimale dans la fermeture de l'enveloppe convexe d'un certain ensemble fini de sous-gradients ou d'ε-sous-gradients en w : ceci nécessite la résolution d'un programme quadratique. Minoux propose dans [51] de remplacer ce problème difficile par une procédure itérative. Mis à part les problèmes de convergence liés au conditionnement des problèmes, ces méthodes sont en général plus délicates à implanter que celles de sous-gradients. Toutefois les travaux de Minoux vont dans le sens d'une mise en oeuvre simplifiée et se veulent particulièrement bien adaptés à l'optimisation des grands systèmes par l'utilisation de relaxations lagrangiennes.

Un autre type de méthodes mettant en oeuvre des algorithmes de type simplex est particulièrement bien adapté aux problèmes issus de relaxations lagrangiennes. Nous citerons principalement les travaux de Fisher, Northup, Shapiro (1975, [11]) et de Marsten (1975, [48]).

Pour compléter ce recensement partiel, nous citerons l'article de Shor (1982, [62]) dans lequel on pourra trouver une bibliographie de base sur toutes les méthodes d'optimisation dans le cas non différentiable. Rappelons que Shor est à l'origine de la méthode dite de sous-gradients avec dilatation de l'espace qui se voulait être au départ une accélération de la méthode de sous-gradient classique. L'importance de cette méthode est qu'à travers des extensions (par exemple le r-algorithme), qui s'avèrent dans certains cas pratiques très performantes, Shor ait pu considérer comme un cas particulier la méthode de Khachyan (à l'intérêt théorique indéniable).

De fait, si le calcul d'un majorant du problème (B), obtenu soit par la résolution de $(\bar{B})$ soit par la résolution de (D_L), est l'objectif essentiel, il pourrait être intéressant de comparer les performances des méthodes suivantes - simplex, sous-gradient, ε-sous-gradient, ellipsoïde, r-algorithme - en fonction de la taille des problèmes et de la précision désirée.

2.1.3 - *Résolution de la relaxation composite*

La littérature est beaucoup moins abondante en méthodes de résolution du dual composite : On peut essayer d'en donner quelques débuts d'explication. Ce n'est qu'à partir des succès enregistrès par l'emploi de la dualité lagrangienne et des procédures de résolution utilisées dans un certain nombre de programmes en nombres entiers (notamment Held et Karp pour le problème du voyageur de commerce (1970), Geoffrion pour le problème de localisation (1972), Fisher pour le problème d'affectation avec contraintes (1973), et avec l'idée force que l'écart de dualité peut être réduit dans le cas composite (Greenberg-Pierskalla, [27]), que quelques chercheurs se sont lancés dans l'utilisation de la dualité composite et l'étude des procédures de résolution adaptées. De plus la fonction duale dans le cas composite n'est pas convexe mais seulement quasi-convexe, cet appauvrissement n'étant pas de nature à faciliter la mise au point de procédures de résolution simples et performantes en comparaison du cas lagrangien.

De fait, les algorithmes de résolution envisagés pour le dual composite sont très souvent similaires à ceux utilisés pour le dual lagrangien. Avant d'entreprendre l'étude au § 2.2 d'un algorithme de type quasi-sous-gradient (généralisation d'un algorithme de sous-gradient) donnons à titre d'exemple deux algorithmes semblables, du même type que la procédure de Benders, pour la résolution respective des duaux lagrangien et composite.

Exemple 2.2 : Karwan, Rardin ([38])

Les notations sont celles définies au paragraphe 1.1.3.

$$(D_L) \quad \begin{cases} \min \ v(BL(w)) \\ \text{s.c. } w \in \mathbb{R}^m_+ \end{cases}$$

avec $v(BL(w)) = \max_{x \in V} \{cx + w\,(b-Ax)\}$

$$(D_S) \quad \begin{cases} \min \ v(BS(w)) \\ \text{s.c. } w \in B_1 = \{w \in \mathbb{R}^m_+ \mid ||w||_1 = 1\} \end{cases}$$

avec $v(BS(w)) = \max_{x \in V} \{cx \mid w(Ax-b) \leq 0\}$

Algorithme 1 (résolution du dual lagrangien (D_L))

0 $k \leftarrow 1$; $w^k \leftarrow 0$,

1 *résolution de la relaxation lagrangienne*

1.1 *déterminer une solution optimale* x^k *du problème* $(BL(w^k))$

1.2 *poser* $g^k = Ax^k - b$; *si* $g^k \leq 0$ *et* $w^k g^k = 0$ *alors* x^k *est solution optimale du primal (B) ; STOP*

2 *passage à l'itéré suivant*

2.1 *déterminer une solution optimale* (z^{k+1}, w^{k+1}) *du programme linéaire :*

$$(PL_k) \quad \begin{cases} \min \ z \\ \text{s.c. } c\,x^j - w\,g^j \leq z \quad j \in \{1, 2, \ldots, k\} \\ \qquad w \geq 0 \end{cases}$$

2.2 *si* $z^{k+1} = v(BL(w^{k+1}))$ *alors* (w^{k+1}) *est une solution optimale de* (D_L) *; STOP*

2.3 $k \leftarrow k+1$; *aller en 1*

Algorithme 2 (résolution du dual composite (D_S))

0 $k \leftarrow 1$; $w^k \leftarrow 0$; $\bar{w}^k \leftarrow 0$; $\alpha \leftarrow +\infty$

1 *résolution de la relaxation composite*

1.1 *déterminer une solution optimale* x^k *du problème* $(BS(w^k))$.

1.2 *poser* $g^k = A x^k - b$; *si* $g^k \leq 0$ *alors* x^k *est solution optimale du primal (B) ; STOP*

1.3 *si* $c x^k < \alpha$ *alors* $\alpha \leftarrow c x^k$, $\bar{w}^k \leftarrow w^k$

2 *passage à l'itéré suivant*

2.1 *déterminer une solution optimale* (z^{k+1}, w^{k+1}) *du programme linéaire :*

$$(PS_k) \quad \begin{cases} \min \ z \\ \text{s.c. } w g^j \leq z \qquad j \in \{1, 2, \ldots, k\} \\ \qquad w \in B_1 \end{cases}$$

2.2 *si* $z^{k+1} \leq 0$ *alors* $(\bar{w}^k, x^k, \alpha)$ *est une solution optimale de* (D_S) *: STOP*

2.3 $k \leftarrow k+1$; *aller en 1*

Quelques remarques sont à faire sur ces deux algorithmes.

1) dual lagrangien (D_L)

Le problème dual (D_L) est équivalent au programme linéaire

$$(PL) \quad \begin{cases} \min \ z \\ \text{s.c. } c x^j - w (Ax^j - b) \leq z, \ j \in \{1, 2, \ldots, N\} \\ \qquad w \geq 0 \end{cases}$$

N désignant le cardinal de $V = \{x^1, x^2, \ldots, x^N\}$.

Supposons qu'à une étape k quelconque, on ait calculé v(BL(w)) en un certain nombre de points $w^1, w^2, \ldots, w^k$ $(k << N)$:

$$v(BL(w^j)) = c x^j - w^j g^j \qquad \text{pour } j \in \{1, 2, \ldots, k\}$$

où $g^j = Ax^j - b$ est un sous-gradient de $w \to v(BL(w))$ en w^j.

On résoud alors à l'étape k le programme restreint (PL_k)

$$(PL_k) \quad \begin{cases} \min \ z \\ \text{s.c. } cx^j - w\, g^j \le z & j \in \{1, 2, \ldots, k\} \\ w \ge 0 \end{cases}$$

Soit (z^{k+1}, w^{k+1}) la solution optimale obtenue. Calculons $v(BL(w^{k+1}))$; deux cas peuvent se présenter :

a) <u>si</u> $z^{k+1} = v(BL(w^{k+1}))$ <u>alors</u> (w^{k+1}) est une solution optimale du dual (D_L). En effet, d'une part $z^{k+1} = v(PL_k) \le v(PL) = v(D_L)$ et d'autre part (z^{k+1}, w^{k+1}) est une solution réalisable de (PL) puisque, par définition de $(BL(w^{k+1}))$, $z^{k+1} \ge c\, x^j - w^{k+1}\, g^j \ \forall\, j \in \{1, 2, \ldots, N\}$.

b) <u>si</u> $z^{k+1} < v(BL(w^{k+1}))$ <u>alors</u> on ajoute à (PL_k) la contrainte $cx^{k+1} - w\, g^{k+1} \le z$.

Cette méthode consiste donc en fait à linéairiser la fonction duale. En général on s'arrête après l'obtention d'une solution approchée à ε près de (D_L), soit $|z^{k+1} - v(BL(w^{k+1}))| < \varepsilon$.

2) <u>Dual composite (D_S)</u>

L'algorithme de résolution du dual composite recherche à chaque itération une direction de déplacement qui n'est pas une direction de descente. Il s'appuie sur la remarque suivante : soit w_k un point courant ; une condition nécessaire (mais non suffisante) pour que w soit une direction de descente pour le dual (D_S) à partir du point w_k est que $w\,(Ax-b) > 0$ pour toute x solution optimale de $(BS(w_k))$.

Si $z \le 0$ alors la solution courante est optimale car il n'existe aucun multiplicateur $w \in B_1$ éliminant les solutions optimales $x^1, \ldots, x^k$ des problèmes $BS(w^1), \ldots, BS(w^k)$ correspondants. D'autre part, il est nécessaire de garder en mémoire la meilleure solution courante $(\alpha, \bar{w})$ car la suite finie $\{v(BS(w^k))\}_{k\ge1}$ n'est pas monotone décroissante.

2.2 - Méthodes de résolution de type sous-gradient

2.2.1 - Méthode de sous-gradients

2.2.1.1 - Algorithme SG

Cet algorithme est une application directe des articles suivants : Held et Karp ([31], 1970) Crowder, Held et Woffe ([32], 1973), Camerini, Fratta et Maffioli ([3], 1975). Il détermine une solution optimale w^* du dual lagrangien (D_L) associe au problème primal (B) :

$$(D_L) \quad \min_{w \geq o} \ \max_{x \in V} \ c\,x + w\,(b - Ax)$$

Algorithme SG :

but : calculer une valeur approchée ℓ de $v(D_L)$

étape 0 : initialisation

0.1 choisir N_{iter}, N_{am}, *entiers positifs et* ε, γ, ρ *réels positifs*

0.2 $w^1 \leftarrow 0$, $t_1 \leftarrow 2$, $d^o \leftarrow 0$, $\ell \leftarrow +\infty$, $k \leftarrow 1$

0.3 lire α, β, $\underline{v}(B)$

étape 1 : résolution de la relaxation lagrangienne

1.1 : déterminer une solution optimale x^k *du problème*

$$\left[\begin{array}{l} \max \ w^k\, b + (c - w^k\, A)\, x \\ \text{s.c. } \alpha \leq \sum_{j=1}^{n} x_j \leq \beta \\ \qquad x \in V \end{array}\right.$$

notons $J^+ = \{j \mid x_j^k = 1\}$

1.2 Calculer $g^k = b - A\,x^k$; $\ell(w^k) = w^k\, b + \sum_{j \in J^+} (c_j - w^k\, A^j)$

1.3 si $\ell(w^k) < \ell$ *alors* $\ell \leftarrow \ell(w^k)$, $t_{k+1} \leftarrow t_k$

sinon $t_{k+1} \leftarrow \rho\, t_k$

1.4 si $g^k = 0$ *alors* x^k *solution optimale de (B) : STOP*

1.5 si $g^k \geq 0$ *alors* x^k *est solution réalisable de (B) ;*

si $\underline{v}(B) < c\,x^k$ *alors* $\underline{v}(B) \leftarrow c\,x^k$

.../...

étape 2 : construction du successeur

2.1 si $k \leftarrow N_{am}$ *alors* $d^k \leftarrow g^k$; *aller en 2.2*

2.1.1 si $d^{k-1}\, g^k < 0$ *alors calculer* $\mu = -\gamma \dfrac{d^{k-1}\, g^k}{||d^{k-1}||_2^2}$

sinon $\mu = 0$

2.1.2 calculer $d^k = g^k + \mu\, d^{k-1}$

2.2 calculer $w^{k+1} = w^k - t_{k+1} \times \dfrac{\ell(w^k) - \underline{v}(B)}{||d^k||_2^2} \times d^k$

2.3 $\forall\, i : w_i^{k+1} < 0$: *faire* $w_i^{k+1} = 0$

étape 3 : critères d'arrêt

3.1 si $||w^{k+1} - w^k||_2 \leq \varepsilon$ *alors STOP*

sinon $k \leftarrow k+1$

3.2 si $k > N_{iter}$ *alors STOP*

sinon aller en 1

Remarques

(i) α et β sont un encadrement de la somme des variables, calculé au préalable et qui peut encore être peaufiné par la procédure décrite au chapitre 5. L'utilisation de cet encadrement évite une dispersion des valeurs $\ell(w^k)$ dans les premières itérations et peut s'avérer être un procédé d'accélération.

si $J^+ < \alpha$, on met à 1 les variables correspondants aux coûts réduits négatifs les plus petits en valeur absolue.

si $J^+ > \beta$, on met à 0 les variables correspondants aux coûts réduits positifs les plus petits.

(ii) N_{iter} et ε définissent les critères d'arrêt de l'algorithme ; ε est pris égal à 10^{-3} ou 10^{-4}, N_{iter} est le nombre d'itérations choisi par l'utilisateur.

(iii) $\underline{v}(B)$ est une estimation par défaut de la valeur $v(B)$; elle peut être prise égale à 0 et incrémentée au fur et à mesure par l'étape 1.5, mais en pratique, $\underline{v}(B)$ sera fournie par une des heuristiques décrites au chapitre 3.

(iv) la détermination du *pas de déplacement* t_k, encore appelé *coefficient de relaxation*, se fait conformément à la stratégie de Legendre et Minoux (1977, [43]). Les coefficients t_k sont calculés de façon dynamique en tenant compte de la progression de la valeur de la fonction duale ℓ (étapes 0.2 et 1.3) avec $\rho = 0.87$. Cette valeur correspond à la règle de Crowder, Held et Wolfe ([32]) consistant à diviser par 2 le coefficient t_k toutes les 5 itérations en fin de convergence.

(v) N_{am} est le nombre d'itérations à partir duquel on enclenche la procédure de Camerini, Fratta et Maffioli ([3]). Elle se révèle pour la plupart des problèmes traités être effectivement une procédure d'accélération. Le paramètre γ est fixé à la valeur 1.5, la meilleure valeur expérimentale obtenue par les auteurs.

De nombreuses expériences numériques mettant en jeu les différents paramètres évoqués pendant ces remarques, sont rapportées au § 2.3.

2.2.1.2 - Convergence

a) : $\boxed{N_{am} = N_{iter}}$ (la procédure de Camerini, Fratta, Maffioli n'est pas mise en jeu)

. La convergence de l'algorithme SG peut s'étudier sur le schéma itératif suivant :

$$(I) \quad \begin{cases} w^0 \in \mathbb{R}_+^m \\ w^{k+1} = \Pi_{\mathbb{R}_+^m}(w^k - \lambda_{k+1}\, g^k) \qquad k = 0, 1, 2, \ldots \end{cases}$$

où - $\Pi_{\mathbb{R}^m_+}$ désigne l'opérateur de projection de $\mathbb{R}^m$ sur le convexe fermé $\mathbb{R}^m_+$

- $\lambda_{k+1} = t_{k+1} \times \dfrac{\ell(w^k) - \gamma\!\!\!/}{||g^k||_2^2} \times g^k$ étant un sous-gradient de la fonction lagrangienne $\ell(w) = \max_{x \in V} \{cx + w\,(b-Ax)\}$ au point w^k (proposition 1.4) et $\gamma\!\!\!/$ une sous-estimation de $v(D_L)$.

Soit $M(\bar{w})$ l'ensemble des points "meilleurs" que $\bar{w} \in \mathbb{R}^m_+$ défini par $M(\bar{w}) = \{w \in \mathbb{R}^m_+ \mid \ell(w) \leq \ell(\bar{w})\}$

De l'inégalité suivante (proposition 1.4), où $g(\bar{w})$ désigne un sous-gradient de la fonction lagrangienne ℓ au point $\bar{w}$;

$$\forall\, w \in M(\bar{w}) \quad (w-\bar{w})\, g(\bar{w}) \leq \ell(w) - \ell(\bar{w}) \leq 0 \qquad (1)$$

on peut déduire que l'ensemble $M(\bar{w})$ est contenu dans un demi-espace, de frontière l'hyperplan passant par $\bar{w}$, de vecteur normal $g(\bar{w})$, et ne contenant pas $g(\bar{w})$ (figure 2.4).

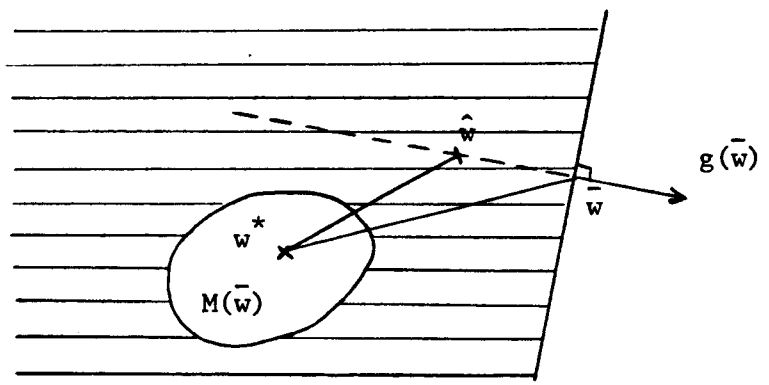


Figure 2.4

Soit $w^* \in \Omega(D_L) \subset M(\bar{w})$ un point optimal ; le principe de la méthode est que pour $\lambda > 0$ suffisamment petit, le point $\hat{w} = \bar{w} - \lambda\, g(\bar{w})$ sera plus proche de w^* que $\bar{w}$ (figure 2.4). La proposition suivante précise l'intervalle auquel doit appartenir λ pour qu'il en soit ainsi.

Proposition 2.1

$\forall\ w \in M(\bar{w})$

si $0 < \lambda \leq 2 \times \dfrac{\ell(\bar{w}) - \ell(w)}{||g(\bar{w})||^2}$ (2)

alors $||w - \Pi_{\mathbb{R}^m_+}(\bar{w} - \lambda\, g(\bar{w}))|| \leq ||w - \bar{w}||$ (3)

Démonstration

$$||w - \Pi_{\mathbb{R}^m_+}(\bar{w} - \lambda\, g(\bar{w}))||^2 \leq ||w - (\bar{w} - \lambda\, g(\bar{w}))||^2$$

$$= ||w - \bar{w}||^2 + \lambda[\lambda\, ||g(\bar{w})||^2 + 2\,(w-\bar{w})\, g(\bar{w})]$$

$$\leq ||w - \bar{w}||^2 + \lambda[\lambda\, ||g(\bar{w})||^2 + 2(\ell(w) - \ell(\bar{w}))] \quad \text{(d'après (1))}$$

$$\leq ||w - \bar{w}||^2 \quad \text{(d'après (2))}.$$

□

On peut énoncer le théorème de convergence suivant (lemme 3, [31]) faisant intervenir l'ensemble M_γ défini par :

$$M_\gamma = \{w \in \mathbb{R}^m_+ \mid \ell(w) \leq \gamma\} \quad \text{où } \gamma > v(D_L)$$

Théorème de convergence 2.1

Soit la suite $\{w^k\}_{k\geq o}$ *définie par le schéma itératif (I)*

alors deux cas peut se présenter :

Cas 1 : si $0 < t_k \leq 2$ *pour tout* k *alors*

- *soit* $\exists\ k_o$ *tel que* $w^{k_o} \in M_\gamma$
- *soit la suite* $\{w^k\}_{k\geq o}$ *converge vers un point de la frontière de* M_γ

Cas 2 : si $t_k = 2$ *pour tout* k *alors* $\exists\ k_o$ *tel que* $w^{k_o} \in M_\gamma$

Notes

(i) M_γ est l'intersection de l'orthant positif $\mathbb{R}^m_+$ et de l'ensemble de niveau $L_\gamma \ell$ de la fonction ℓ associé au réel γ.

(ii) Le résultat à la base du théorème 2.1 est un théorème de convergence énoncé par Motzkin et Schoenberg ([52], théorème 1) au sujet d'une méthode de relaxation pour un système d'inégalités linéaires. La différence essentielle étant que le coefficient de relaxation t_k était une constante $\bar{t}$.

Démonstration

(i) Le cas 2 est une application directe de la partie 2 du théorème 1 de [52].

(ii) La démonstration du cas 1 nécessite le concept de "suite Fejer-monotone" introduit dans [52].

Définition 2.5

Soit G *une partie non vide d'un espace vectoriel normé* E *de dimension finie* n. *Une suite* $\{x^j\}_{j\geq o}$ *est dite* ***Fejer-monotone relative à*** G *si* $\forall\, x \in G$, *la suite* $\{||x^j - x||\}_{j\geq o}$ *est monotone non croissante.*

Lemme 2.1

Si G *est une partie de dimension pleine (la plus petite variété linéaire contenant* G *est de dimension* n*), alors toute suite* $\{x^j\}_{j\geq o}$ *Fejer-monotone relative à* G *est convergente.*

Démonstration du lemme 2.1

L'existence d'un point d'accumulation de la suite $\{x^j\}_{j\geq o}$ est assurée par le fait que cette suite appartient à la boule fermée compacte $B(\bar{x}, \rho)$, $\bar{x} \in G$, $\rho = ||x^o - \bar{x}||$.

Supposons que la suite $\{x^j\}_{j\geq o}$ possède deux points d'accumulation distincts x_1 et x_2. Soit H l'hyperplan médiateur du segment $[x_1, x_2]$; $\exists\ \hat{x} \in G$ tel que $\hat{x} \notin H$ car G est une partie de dimension pleine. Par conséquent, en posant $d_1 = ||x_1-\hat{x}||$ et $d_2 = ||x_2-\hat{x}||$, on peut supposer sans restreindre la généralité qu'il existe $\varepsilon > 0$ tel que : $0 < 2\varepsilon < d_1-d_2$. (1)

On peut extraire de la suite $\{x^j\}_{j\geq o}$ deux sous-suites infinies $\{x^j\}_{j\in\mathbb{N}_1}$ et $\{x^j\}_{j\in\mathbb{N}_2}$ avec $\mathbb{N}_1 \subset \mathbb{N}$, $\mathbb{N}_2 \subset \mathbb{N}$ convergeant respectivement vers x_1 et x_2.

Pour ε vérifiant (1)

$$\exists\ p_o,\ \forall\ p \in \mathbb{N}_1,\ p \geq p_o \quad ||x^p - x_1|| \leq \varepsilon \qquad (2)$$

$$\exists\ q_o,\ \forall\ q \in \mathbb{N}_2,\ q \geq q_o \quad ||x^q - x_2|| \leq \varepsilon \qquad (3)$$

d'où, en conjuguant (1), (2) et (3)

$$\forall\ p \in \mathbb{N}_1,\ \forall\ q \in \mathbb{N}_2,\ p \geq p_o,\ q \geq p_o$$

$$d_2 - \varepsilon \leq ||x^q - \hat{x}|| \leq d_2 + \varepsilon < d_1 - \varepsilon \leq ||x^p - \hat{x}|| \leq d_1 + \varepsilon$$

et par conséquent, $\exists\ (\bar{p}, \bar{q}) \in \mathbb{N}_1 \times \mathbb{N}_2$, $\bar{p} > \bar{q}$ tel que $||x^{\bar{p}}-\hat{x}|| > ||x^{\bar{q}}-\hat{x}||$, ce qui est en contradiction avec l'hypothèse "suite $\{x^j\}_{j\geq o}$ Fejer-monotone relative à G". Il y a donc unicité du point d'accumulation et convergence de la suite $\{x^j\}_{j\geq o}$.

□

Remarque

Si G n'est pas de dimension pleine, une suite Fejer-monotone n'est pas nécessairement convergente. Considérons par exemple la partie $G =]-a, +a[\subset \mathbb{R}^2$ $(a > 0)$ et la suite $\{x^j\}_{j\geq o}$ définie par le couple de ses coordonnées (x_1^j, x_2^j) :

$$x_1^j = 0 \qquad x_2 = (-1)^j \left[1 + \frac{1}{1+\ell}\right]$$

cette suite admet de façon évidente 2 points d'accumulation (0, 1) et (0, -1) en étant Fejer-monotone relative à G puisque $\forall\ j \geq 0$, $\forall\ \hat{x}_1 \in]-a,+a[$

$$||x^{j+1} - \hat{x}||^2 - ||x^j - \hat{x}||^2 = \hat{x}_1^2 + (1 + \frac{1}{2+j}) - \hat{x}_1^2 - (1 + \frac{1}{1+j})^2 < 0$$

□

Lemme 2.2

M_ℓ *est une partie de dimension pleine de* $\mathbb{R}^m$.

Démonstration du lemme 2.2

Soit $\bar{w} \in \mathbb{R}_+^m$ tel que $v(D_L) \leq \ell(\bar{w}) < \ell$.

$\forall\, w \in \mathbb{R}_+^m$, $c\,x + w\,(b-Ax) = cx + \bar{w}\,(b-Ax) + (w-\bar{w})\,(b-Ax) \quad \forall\, x \in V$

$$\leq \ell(\bar{w}) + (w-\bar{w})\,(b-Ax) \qquad \forall\, x \in V$$

$$\leq \ell(\bar{w}) + ||w-\bar{w}||\;||b-Ax|| \qquad \forall\, x \in V$$

d'où $\ell(w) = \max\limits_{x \in V} cx + w\,(b-Ax) \leq \ell(\bar{w}) + ||w-\bar{w}||\,K$ (1)

avec $K = \max\limits_{x \in V} ||b-Ax|| < +\infty$

On déduit de (1) que $\forall\, w \in B(\bar{w}, \rho) \cap \mathbb{R}_+^m$, où $B(\bar{w}, \rho)$ est la boule de centre $\bar{w}$ et de rayon $\rho = \dfrac{\ell - \ell(\bar{w})}{K}$, $\ell(w) < \ell$, soit $w \in \overset{\circ}{M}_\ell$.

□

La démonstration du théorème 2.1 Cas 1 s'articule comme celle du lemme 3 de [31].

En résumé

Supposons que la suite $\{w^k\}_{k \geq o}$ ne contienne aucun point de M_ℓ alors

$$\forall\, k \geq 1 \quad 0 < \lambda_k \leq 2 \times \frac{\ell(w^k) - \ell}{||g^k||^2} \leq 2 \times \frac{\ell(w^k) - \ell(w)}{||g^k||^2} \quad , \forall\, w \in M_\ell.$$

d'où (proposition 2.1) $||w - \Pi_{\mathbb{R}_+^m}(w^k - \lambda_{k+1}\, g^k)|| \leq ||w - w^k||$

soit $||w - w^{k+1}|| \leq ||w - w^k||$. La suite $\{w^k\}_{k \geq o}$ étant Fejer-monotone relative à M_ℓ est convergente (lemmes 2.1 et 2.2)

Par conséquent $\lim_{k\to+\infty} ||w^{k+1} - w^k|| = 0$ et $\lim_{k\to+\infty} \ell(w^k) = \gamma$ puisque $t_{k+1} \times \frac{\ell(w^k) - \ell}{||g^k||} = ||\Pi_{\mathbb{R}^m_+}(w^{k+1} - w^k)|| \leq ||w^{k+1} - w^k||$. Puisque ℓ est continue, $\ell(\lim_{k\to+\infty} w^k) = \gamma$ donc le point de convergence appartient à M_γ, en fait à sa frontière puisque aucun point de $\{w^k\}_{k\geq o}$ n'appartient à M_γ.

□

b) : $\boxed{0 \leq N_{am} < N_{iter}}$ (la procédure de Camerini, Fratta et Maffioli est mise en jeu après N_{am} itérations)

Le principe de cette méthode est de remplacer à l'itération k la direction de descente g^k par une combinaison linéaire d^k des sous-gradients g^k et g^{k-1} de la forme $d^k = g^k + \mu\, g^{k-1}$ (Algorithme SG, étape 2.1).

Le germe géométrique de la méthode est que pour un scalaire μ bien choisi, la direction d^k formera avec $w - w^k$ ($w \in M(w^k)$) un angle plus aigu que la direction g^k et sera pour un petit déplacement une meilleure direction de descente (figure 2.5).

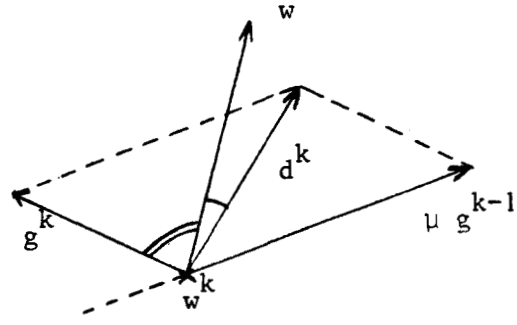


Figure 2.5

En utilisant le schéma itératif suivant

$$(II) \quad \begin{cases} w^o \in \mathbb{R}^m_+ \\ w^{k+1} = \Pi_{\mathbb{R}^m_+}(w^k - \lambda_{k+1}\, d^k) \quad k = 0, 1, 2, \ldots \end{cases}$$

où d^k est la direction définie à l'étape 2.1 de l'algorithme SG.

On peut énoncer un résultat de convergence identique au théorème 2.1 Cas 1 sous l'hypothèse plus restrictive :

$$\exists\ \varepsilon > 0 \qquad \varepsilon \times \frac{\ell(w^k) - \gamma\!\!\!/}{||d^k||^2} \leq t_k \leq 1 \qquad \text{pour tout } k.$$

Démonstration :

Voir [3].

□

Remarque

Dans les expériences numériques, la sur-estimation $\gamma\!\!\!/$ a été remplacée par une sous-estimation de $v(D_L)$ - un minorant $\underline{v}(B)$ - plus facile à calculer Toutefois les résultats ne semblent pas dépendre de façon critique de la valeur $\gamma\!\!\!/$ (voir § 2.2.3.1).

2.2.2 - Méthodes de quasi-sous-gradients

2.2.2.1 - Algorithme QSG

Cet algorithme est une adaptation de celui proposé par Dyer dans [6]. Il détermine une solution optimale w^* du dual composite en continu $(\bar{D}_S)$ associé au problème primal (B) :

$$(\bar{D}_S) : \min_{w \geq o} \max\ c\ x \text{ s.c. } wAx \leq wb,\ x \in [V]$$

Algorithme QSG

but : Calculer une valeur approchée $\bar{s}$ *de* $v(\bar{D}_S)$

étape 0 : initialisation

0.1 *Choisir* w^1 *tel que* $||w^1||_2 = 1$; *choisir* N_{iter} *entier positif et* ε *réel positif*

0.2 $t_1 \leftarrow 1$; $\bar{s} \leftarrow +\infty$; $k \leftarrow 1$

.../...

étape 1 : résolution "en continu" du problème contracté

1.1 *calculer* $a_s = w^k A$, $b_s = w^k b$

1.2 *déterminer une solution optimale* x^k *du problème contracté*

$$\left[\begin{array}{l} \max \ c.x \\ \text{s.c. } a_s\, x \le b_s \qquad\qquad (K) \\ \quad 0 \le x \le e \end{array}\right.$$

notons i^* *l'indice de la variable en base.*

1.3 *calculer* $g^k = A\, x^k - b$, $\bar{s}(w^k) = c\, x^k$

1.4 *si* $\bar{s}(w^k) < \bar{s}$ *alors* $\bar{s} \leftarrow \bar{s}(w^k)$; $t_{k+1} \leftarrow t_k$

sinon $t_{k+1} \leftarrow \rho\, t_k$

1.5 *si* $g^k \le 0$

si $x^k_{i^*} \in \{0, 1\}$ *alors* x^k *est solution optimale de (B)*

alors STOP

étape 2 : Construction du successeur

2.1 $d^k \leftarrow g^k - (w^k g^k)\, w^k$; $d^k \leftarrow d^k / \|d^k\|_2$

2.2 $w^{k+1} \leftarrow w^k + t_{k+1}\, d^k$

2.3 $\forall\, i : w_i^{k+1} < 0$ *faire* $w_i^{k+1} \leftarrow - w_i^{k+1}$

2.4 $w^{k+1} \leftarrow w^{k+1} / \|w^{k+1}\|_2$

étape 3 : critère d'arrêt

3.1 *si* $\|w^{k+1} - w^{k-1}\|_2 \le \varepsilon$ *alors STOP*

sinon $k \leftarrow k+1$

3.2 *si* $k < N_{iter}$ *alors STOP*

sinon aller en 1

Remarques

(i) N_{iter} et ε définissent les critères d'arrêt de l'algorithme ; ε est pris égal à 10^{-2} ou 10^{-3}, N_{iter} est le nombre d'itérations choisi par l'utilisateur.

(ii) w^1 est soit le vecteur unitaire (1)

soit la solution "optimale" du dual lagrangien (2)

(iii) $\bar{s} = +\infty$ si (1)

= meilleure valeur entière $\lfloor \ell(w^k) \rfloor$ obtenue au cours de la résolution du dual lagrangien si (2).

(iv) Aucune expérience numérique n'est rapportée dans [6]. En particulier le pas de déplacement doit obéir théoriquement aux conditions classiques de Polyak [55] :

$$\lim_{k \to +\infty} t_k = 0 \text{ et } \sum_{k=o}^{\infty} t_k = +\infty$$

En pratique les coefficients t_k sont calculés de façon dynamique en tenant compte de la progression de la valeur de la fonction composite $\bar{s}$ (étape 1.4) avec $\rho = 0.95$. D'autres choix seront comparés numériquement au § 2.2.3.2.

(v) $\lfloor \bar{s} \rfloor$ pourra être au mieux égal à $\lfloor v(B) \rfloor$ puisque l'on utilise que des relaxations du type (K) du problème $(\bar{B})$:

$$\forall\, k \geq 1 \quad v(\bar{B}) \leq \bar{s}(w^k)$$

(chaque problème (K) est résolu par l'algorithme NKR).
On peut espérer beaucoup mieux comme majorant en utilisant une résolution du problème contracté bivalent (dans (K), [V] est remplacé par V et le nouveau problème est résolu par l'algorithme FPK 79), malheureusement plus coûteuse en temps calcul. Toutefois la mise en oeuvre de l'algorithme QSG sur le problème de Fleisher de taille 10 × 20 ([13], Annexe 1) a permis de vérifier que l'écart de dualité composite pouvait être strictement inférieur à l'écart de dualité lagrangien (résultat fondamental du § 1.2.1 : $v(B) \leq v(D_S) \leq v(\bar{B}) = v(D_L) = v(\bar{D}_S)$)

Pour (B) ≡ (Fleisher) :

$$\begin{cases} v(B) = 2139 \\ v(D_S) = 2219 \\ v(\bar{B}) = 2221.82 \\ \ell = 2224.30 \\ \bar{s} = 2221.95 \end{cases}$$

(vi) g^k (étape 1.3) est un quasi-sous-gradient de la fonction composite $\bar{s}$ au point w^k (proposition 1.9). La fonction quasi-convexe $\bar{s}$ n'admet pas de sous-gradient en chaque point w^k.

(vii) La nature géométrique du calcul du successeur w^{k+1} (étape 2) peut se préciser comme suit : la direction de pseudo-descente d^k est obtenue par projection de g^k sur l'hyperplan orthogonal au vecteur w^k (étape 2.1) ; en fait d^k n'est autre qu'une combinaison linéaire de g^k et de w^k (étape 2.2) ; les étapes 2.3 et 2.4 consistent en une sorte de projection sur l'ensemble non convexe $B_2 = \{w \in \mathbb{R}^m_+ \mid ||w||_2 = 1\}$.

2.2.2.2 - Convergence

Les propriétés de convergence de l'algorithme QSG font intervenir l'ensemble M^* suivant :

$\forall\ \varepsilon > 0$, posons $M(\varepsilon) = \{w \in \mathbb{R}^m_+ \mid ||w||_2 \leq 1,\ \bar{s}(w) < v(\bar{D}_S) + \varepsilon\}$

alors M^* est l'ensemble non vide défini par

$$M^* = B_2 \cap \{ \bigcap_{\varepsilon>0} c\ell(M(\varepsilon))\}, \text{ où } c\ell(M(\varepsilon)) \text{ est la fermeture de } M(\varepsilon).$$

Théorème de convergence 2.2

Sous les hypothèses : $\sum_{k=0}^{\infty} t_k = +\infty$ *et* $\lim_{k\to+\infty} t_k = 0$, *l'algorithme QSG génère une suite* $\{w_k\}_{k\geq 0}$ *telle que :*

(i) $\lim_{k\to+\infty} \inf\ \bar{s}(w^k) = v(\bar{D}_S)$

(ii) tous les points d'accumulation de la suite $\{w^k\}_{k\geq 0}$ *sont dans* M^*.

Démonstration :

voir [6], propositions 21 et 22.

□

Remarques

(i) si $\bar{s}$ est continue sur l'ensemble des points optimaux $\{w \in B_2 \mid \bar{s}(w) = v(\bar{D}_S)\}$, alors M^* est justement l'ensemble des points optimaux ([6], corollaire 7).

(ii) Un lien important entre les méthodes de sous-gradients et de quasi-sous-gradients semble être le théorème de Polyak [55], dont l'énoncé bien connu est rappelé ci-dessous :

"Soit le programme suivant :

$$\begin{cases} \inf \ f(x) \\ \text{s.c. } x \in Q \subset E \end{cases}$$

où f est une fonction quasi-convexe et continue sur l'espace de Hilbert E, Q un sous-ensemble convexe et fermé de E.

Considérons le schéma itératif

$$\begin{cases} x^0 \quad E \\ x^{k+1} = \Pi_Q(x^k + c^k) \qquad k = 0, 1, 2, \ldots \end{cases}$$

avec - c_k une fonctionnelle de support arbitraire de l'ensemble de niveau $L_{f(x^k)} f = \{x \mid f(x) \leq f(x^k)\}$ au point x^k $(\forall\, x \in L_{f(x^k)} f,\ c^k x^k \leq c^k x)$.

- Π_Q un opérateur de projection de E sur Q.

Alors sous les hypothèses $\lim_{k \to +\infty} ||c^k|| = 0$ et $\sum_{k=0}^{\infty} ||c^k|| = +\infty$ (1)

la suite $\{x^k\}_{k \geq 0}$ possède un point d'accumulation x^* tel que $f(x^*) = \inf_{x \in Q} f(x)$"

Remarquons tout d'abord que la quasi-convexité de f suffit puisqu'elle assure la convexité des ensembles de niveau $L_{f(x^k)}$ f. D'autre part on peut prendre $c^k = \lambda_k \, g^k / ||g^k||$, la suite de scalaires $\{\lambda_k\}_{k\geq o}$ vérifiant (1), avec g^k sous-gradient de f en x^k si f est convexe (proposition 1.4) ou g^k quasi-sous-gradient de f en x^k si f est quasi-convexe (proposition 1.10).

Par conséquent ce théorème, à la base des démonstrations de convergence des méthodes de sous-gradients pour la résolution du dual lagrangien, peut être aussi un outil intéressant pour une généralisation de ces méthodes dans le cadre de la résolution du dual composite.

□

Exemple 2.3

Sur pratiquement tous les problèmes tests nous avons pu constater que l'algorithme QSG engendrait deux sous-suites convergentes $\{w^{2k}\}_{k\geq o}$ et $\{w^{2k+1}\}_{k\geq o}$ associées aux deux sous-suites $\{\bar{s}(w^{2k})\}_{k\geq o}$ et $\{\bar{s}(w^{2k+1})\}_{k\geq o}$, conformément à la convergence en points d'accumulation assurée par le théorème 2.2.

Nous exhibons sur le problème 6 de Petersen ([53]) cette propriété pendant 10 itérations ($v(B) = 10618$; $v(\bar{B}) = 10672.3$).

Notons ici que c'est essentiellement la quatrième composante w_4^k qui différencie les deux points d'accumulation (les 10 itérations sont les dernières du processus arrêté à l'itération 150).

k	$\bar{s}(w^k)$	w_1^k	w_2^k	w_3^k	w_4^k	w_5^k
140	10672.51953	.68824917	.29614317	.17122012	.00152683	.63976091
141	—73.48437	.—41791	.—605347	.—39471	.—000056	.—57614
142	—72.45312	.—33089	.—592240	.—29976	.—153845	.—75424
143	—73.48828	.—22205	.—614407	.—54616	.—009881	.—70464
144	—72.51953	.—16465	.—608184	.—19098	.—124850	.—88900
145	—73.28125	.—30895	.—600465	.—34053	.—006105	.—73039
146	—72.47266	.—23797	.—589784	.—26316	.—119248	.—87535
147	—73.25000	.—14963	.—607832	.—46385	.—001984	.—833530
148	—72.44141	.—08210	.—597682	.—39030	.—117102	.—997298
149	—73.21094	.—799812	.—614818	.—58091	.—005700	.—993490

2.2.3 - *Expériences numériques*

De nombreuses expériences nuémriques nous ont permis, tout en validant les algorithmes SG et QSG de comparer l'efficacité de ces méthodes en fonction des valeurs prises par les différents paramètres et de faire ainsi un choix "optimal" de ces dits-paramètres conduisant aux formulations finales de ces algorithmes (§ 2.2.1 et 2.2.2) pour leur utilisation dans un code de réduction (chapitre 4).

2.2.3.1 - Algorithme SG

Le choix des coefficients de relaxation t_k est déterminant pour obtenir une convergence satisfaisante. Deux règles pratiques peuvent être utilisées :

Re 1 (Held, Karp [31]) : $t_k = 2$ pendant 2m itérations, puis diviser par 2 la valeur de t_k et le nombre d'itérations de validité de t_k et ainsi de suite jusqu'à un seuil minimum q (généralement égal à 5) du nombre d'itérations ;

enfin diviser par 2 la valeur de t_k toutes les q itérations jusqu'à ce que les λ_k soient suffisamment petits (< précision fixée).

Re 2 (Legendre, Minoux [43]) : $t_1 = 2$: si $\ell(w^{k+1}) < \min_{j=1,\ldots,k} \ell(w^j)$ alors $t_{k+1} = t_k$ sinon $t_{k+1} = \rho\, t_k$ ($\rho < 1$). (Plus précisément, il est indiqué dans [24] que la règle Re 1 correspond à un coefficient $\rho \simeq 0.87$, valeur pour laquelle des résultats expérimentaux très convaincants ont été obtenus sur certains problèmes discrets de grande dimension).

Le tableau 2.1 compare les règles 1 et 2 sur le problème 6 de Petersen. D'autres tests effectués accordent à la règle 2 des performances sensiblement meilleures par rapport à la valeur de la fonction ℓ. En conséquence, c'est la règle 2 qui a été retenue pour toutes les expériences numériques avec $\rho = 0.87$.

Les résultats rapportés dans le tableau 2.2 permettent de juger de l'apport d'un encadrement de la somme des variables lors du calcul de $\ell(w^k)$ (voir chapitre 5). De façon générale les expériences ont pu mesurer une légère amélioration du majorant liée à une moins grande grande oscillation des valeurs $\ell(w^k)$, ceci ne justifient pas toutefois une utilisation systématique d'un encadrement, d'autant plus cher en temps calcul qu'il sera fin.

Le tableau 2.3 justifie la modification de la méthode classique de sous-gradients par l'utilisation, après N_{am} itérations, de la procédure de Camerini, Fratta, Maffioli [3]. Cette méthode utilisée seule ($N_{am} = 0$) donne des résultats très irréguliers (des cas de divergence ont été constatés) et il semble qu'une politique raisonnable soit de prendre N_{am} égal à 5 ou 10.

	Re 1		Re 2	
nombre itérations k	$\min_{j=1,k} \ell(w^j)$	w^k	$\min_{j=1,k} \ell(w^j)$	w^k
10	10788.777	(6.78,4.42,0.89,0.64,2.80)	10761.168	(6.57,4.16,1.43,0.00,3.86)
20	10700.523	(6.24,3.74,0.52,0.00,3.86)	10693.313	(6.25,3.14,0.96,0.00,4.25)
30	10693.125	(6.18,3.62,0.51,0.00,4.04)	10680.785	(6.13,3.07,0.96,0.00,4.78)
40	10691.945	(6.16,3.59,0.50,0.00,4.08)	10678.332	(6.10,3.01,0.91,0.00,4.94)
50	10691.293	(6.15,3.59,0.51,0.00,4.11)	10677.551	(6.09,2.99,0.89,0.00,4.99)

Tableau 2.1

($m = 5$, $n = 39$, $\alpha = 10$, $\beta = 36$, $N_{iter} = N_{am}$, $\hat{\ell} = v(B)$)

	$\ell(w^k)$				
itération k	$0 \le c\ x \le n$	$\alpha \le c\ x \le \beta$	itération k	$0 \le c\ x \le n$	$\alpha \le c\ x \le \beta$
1	14723.000	14650.000	10	10755.066	10773.680
2	16553.234	16406.715	15	10693.863	10794.148
3	12821.441	12766.844	20	10688.973	10742.355
4	11916.125	11898.266	25	10685.457	10685.926
5	11197.996	11174.621	30	10683.328	10685.414
6	11027.441	11036.059	35	10680.375	10680.992
7	11914.508	11858.363	40	10679.340	10679.980
8	11050.473	11021.547	45	10679.105	10678.176
9	10842.574	10998.305	50	10678.570	10677.840

Tableau 2.2

($m = 5$, $n = 39$, $\alpha = 10$, $\beta = 36$, $N_{iter} = N_{am}$, $\hat{\ell} = v(B)$)

Les tests de réduction décrits au chapitre 4 utilisent la partie entière du majorant $\ell(w^k)$. Le tableau 2.4 prouve l'inutilité de pousser très loin la convergence de l'algorithme SG, 50 à 80 itérations suffisent pratiquement dans tous les cas à obtenir la partie entière de la valeur optimale $\min_k \ell(w^k)$.

Le tableau 2.5 montre de façon assez surprenante que le fait de prendre pour ℓ un minorant $\underline{v}(B)$ à la place de la valeur exacte $v(B)$ (ou d'un majorant de $v(D_L)$) n'handicape pas la méthode de sous-gradients et qu'au contraire le majorant obtenu après 50 itérations est souvent meilleur.

2.2.3.2 - Algorithme QSG

Pour le choix des coefficients de relaxation t_k, nous nous sommes basés de façon naturelle, à cause de la structure très voisine des deux algorithmes, sur les règles expérimentalement satisfaisantes pour les méthodes de sous-gradients.

Le tableau 2.6 compare la règle Re 2 (§ 2.2.3.1) pour différentes valeurs du paramètre ρ avec la suite classique $\{t_k = 1/k\}_{k\geq 1}$ qui vérifie bien les conditions de Polyak. La règle Re 2 avec ρ=0.95 a donné sur tous les exemples testés, si ce n'est la meilleure valeur, du moins une valeur convenable très proche.

Le tableau 2.7 confirme le résultat théorique de la proposition 1.16 comme quoi un multiplicateur optimal du dual lagrangien est en général une bonne solution de départ pour démarrer une méthode de résolution du dual composite. De plus on constate (voir aussi le tableau 2.8) que la convergence de la fonction $\bar{s}$ est moins rapide que celle de la fonction ℓ ; par conséquent un plus grand nombre d'itérations est nécessaire

	$\min \ell(w^k)$, k = 1, ..., 100				
Problème	$N_{am} = 0$	$N_{am} = 10$	$N_{am} = 20$	$N_{am} = 50$	$N_{am} = 100$
P1	4152.2734	4155.4219	4156.6484	4157.3516	4157.4414
P2	9354.0062	9336.8812	9345.8375	9357.0000	9357.3062
P3	4148.1641	4162.9414	4161.8320	4162.6445	4162.6602
P4	6180.2109	6172.7891	6172.7227	6172.7227	6172.7227
P5	12507.7891	12493.4766	12493.4766	12493.4766	12493.4766
P6	10779.2773	10675.1211	10676.3672	10676.4219	10676.4453
P7	16613.1250	16614.2227	16614.7031	16614.7852	16614.8125

Tableau 2.3

($N_{iter} = 100$, $\mathcal{L} = v(B)$)

Problèmes	$\ell^* = \min_k \ell(w^k)$	numéro itération où ℓ^* est atteint	numéro itération où $\lfloor \ell^* \rfloor$ est atteint
P1	4155.4219	98	48
P2	9336.8812	96	90
P3	4162.9414	89	68
P4	6172.7891	89	37
P5	12493.4766	99	55
P6	10675.1211	99	56
P7	16614.2227	86	38

Tableau 2.4

($N_{iter} = 100$, $N_{am} = 10$, $\mathcal{L} = v(B)$)

Problèmes	$\mathcal{L} = v(B)$	$\min_k \ell(w^k)$	$\mathcal{L} = v(B)$	$\min_k \ell(w^k)$
P6	10552	10673.9648	10618	10674.8828
P7	16470	16613.9453	16537	16614.8750
W1	140778	142018.6875	141278	142037.6875
W6	130233	131336.8750	130623	131338.4375
W8	620060	628985.6250	624319	628783.8750

Tableau 2.5

(N_{iter} = 80, N_{am} = 5, les minorants $\underline{v}(B)$ sont calculés grâce aux heuristiques définies au chapitre 3, les valeurs exactes v(B) sont données par les auteurs : [53], [66]).

	$\min \bar{s}(w^k)$, k = 1, ..., N_{iter}				
Problèmes	$t_k = 1/k$	règle Re 2			
		ρ = 0.80	ρ = 0.87	ρ = 0.90	ρ = 0.95
P1	4154.1641	*4155.4219	*4155.4219	*4155.4219	*4155.4219
P2	9301.3000	9319.3625	9304.1187	9297.8312	9301.6250
P3	4147.4609	4153.1172	4149.0195	4144.4297	4135.7031
P4	6169.8594	6170.6523	6169.3984	6167.8789	6164.7930
P5	12486.7891	12488.4062	12486.1836	12484.5156	12484.9414
P6	10672.9570	10675.1953	10673.1602	10672.5781	10673.1562
P7	16612.9023	16613.4727	16612.9375	16612.8789	16613.0820

Tableau 2.6

(initialisation algorithme SG, N_{iter} = 100)

(*) l'algorithme QSG n'a pu améliorer après 100 itérations la valeur initiale fournie par l'algorithme SG.

si l'on veut obtenir la partie entière de la valeur optimale (en pratique ce seuil est pris égal à 100).

Problèmes	sans initialisation	avec initialisation		
	$\bar{s}^* = \min \bar{s}(w^k)$	$\bar{s}^* = \min \bar{s}(w^k)$	numéro k où $\bar{s}^*$ est atteint	numéro k où $\lvert\bar{s}^*\rvert$ est atteint
P1	4150.6172	4154.1641	100	64
P2	9324.5062	9301.3000	100	100
P3	4156.4844	4147.4609	100	94
P4	6166.5547	6169.8594	100	97
P5	12492.9258	12486.7891	100	96
P6	10673.1758	10672.9570	100	76
P7	16612.9102	16612.9023	92	68

Tableau 2.7

$(N_{iter} = 100,\ t_k = 1/k)$

2.2.3.3 - Comparaison de la précision et du coût des algorithmes SG et QSG

Le tableau 2.8 récapitule sur 18 problèmes tests réels tirés de la littérature ([13], Annexe 1) les résultats de convergence des algorithmes SG et QSG le test d'arrêt étant la précision ε sur la norme de la différence de 2 multiplicateurs consécutifs. Si théoriquement nous avons l'égalité $v(\bar{B}) = v(D_L) = v(\bar{D}_S)$, nous pouvons constater que l'algorithme QSG permet d'approcher beaucoup mieux la valeur $v(\bar{B})$ que ne le fait l'algorithme SG, malheureusement avec un coût en temps calcul par itération 1.8 fois plus élevé en moyenne (tableau 2.9).

Problème et taille	v(B)	$v(\bar{B})$	Algorithme SG			Algorithme QSG		
			$\ell^* = \min \ell(w^k) \simeq v(D_L)$	numéro itération où ℓ atteint	numéro itération arrêt	$\bar{s} = \min \bar{s}(w^k) \simeq v(\bar{D}_S)$	numéro itération où $\bar{s}^*$ atteint	numéro itération arrêt
P1 6×10	3800	4134	4156.55	102	109	4155.42	1	89
P2 10×10	8706.1	9297.7	9354.06	119	138	9297.76	191	200*
P3 10×15	4015	4127.	4158.37	93	110	4134.63	200	200*
P4 10×20	6120	6155.	6172.69	88	100	6161.87	197	200*
P5 10×28	12400	12465.	12491.51	121	126	12478.44	197	200*
P6 5×39	10618	10672.3	10674.77	113	126	10672.36	199	200*
P7 5×50	16537	16612.8	16615.35	109	120	10612.84	193	200*
W1 2×28	141278	142019.0	142037.68	7	123	142018.93	142	200*
W2 2×28	130883	131637.5	131647.62	113	144	131637.50	61	194
W3 2×28	95677	99647.0	99812.68	22	141	99647.06	86	189
W4 2×28	119337	122505.2	122519.87	41	126	122505.18	119	193
W5 2×28	98796	100433.1	100432.93	72	101	100433.12	1	90
W6 2×28	130623	131335.0	131338.43	12	110	131334.93	1	90
W7 2×105	1095445	1095721.2	1095709.00	18	124	1095721.00	3	181
W8 2×105	624319	628773.7	628783.75	125	129	628774.06	164	188
F 10×20	2139	2221.8	2224.30	107	110	2221.95	197	200*
ST1 30×60	7772	7839.	7854.78	73	76	7846.42	197	200*
ST2 30×60	8722	8773.	8774.10	89	89	8773.30	199	200*

Tableau 2.8

Notes : caractéristiques des algorithmes SG et QSG

SG : $N_{iter} = 150$, $N_{am} = 5$, Re 2 avec $\rho = 0.87$, $\varepsilon = 10^{-5}$

QSG : $N_{iter} = 200$, Re 2 avec $\rho = 0.95$, $\varepsilon = 10^{-4}$

* la précision ε sur la norme $||w^{k+1}-w^k||$ n'est pas atteinte après N_{iter} itérations.

Taille	10×6	10×10	10×15	10×20	10×28	5×39	5×50	2×28	2×105	30×60
temps/itération Algorit. SG	55	72	90	133	174	126	166	55	169	1018
temps/itération Algorit. QSG	89	139	180	232	310	260	275	103	327	1700
rapport QSG/SG	1.62	1.80	1.74	1.78	2.06	1.65	1.87	1.93	1.93	1.67

Tableau 2.9

(les temps moyens par itération sont en dixmillièmes de seconde)

Aucune expérience comparative n'a été menée avec une méthode simplex, de toute évidence très à son avantage sur ces problèmes de petite et moyenne taille. Toutefois pour les problèmes de grande taille les méthodes de sous-gradients peuvent devenir très concurrentielles si l'on se fie aux remarques suivantes :

- les principaux inconvénients des méthodes du simplex sont : stockage et inversion de matrices de grandes tailles ; accumulation des erreurs d'arrondis ; cas de dégénerescences. Pour remédier à ces difficultés des méthodes sont été mises au point. La méthode des paramètres (avec résolution directe des sous-systèmes linéaires à chaque itération) rentre parfaitement dans ce cadre (Huard, [34], [35]).

- le nombre d'itérations nécessaire pour atteindre n'importe quel degré de précision ne semble pas croître de façon significative avec la taille du problème pour les méthodes de sous-gradients, ce qui n'est pas le cas de la méthode simplex (suite aux conclusions des expériences numériques menées dans [7] et [50]).

- la méthode simplex semble dominer tout algorithme de sous-gradients si un très haut degré de précision est désiré. Or ce ne sera pas le cas dans le cadre du code de réduction développé au chapitre 4.

HEURISTIQUES

CHAPITRE 3

INTRODUCTION

L'objet de ce chapitre est la construction de deux heuristiques simples et efficaces, appelées AGNES 1 et 2, fournissant des minorants de la valeur du programme linéaire en variables bivalentes (B) suivant :

$$(B) \quad \begin{cases} \max \ cx \\ \text{s.c.} \ \ A \leq b \\ \qquad x \in V \end{cases}$$

La matrice A de taille $m \times n$ est à coefficients non négatifs et a priori ne contient pas de contraintes à structure particulière.

Ce travail a déjà été l'objet d'une première publication ([13]) dans laquelle on pourra trouver un catalogue des principales heuristiques rencontrées dans la littérature ainsi que les données de tous les problèmes tests utilisés.

La première partie (§ 3.1) consiste dans le rappel des points essentiels relatifs à ces deux heuristiques : description, complexités théorique et pratique, performances en précision et en temps calcul.

Une deuxième partie (§ 3.2) met l'accent sur l'utilisation de ces heuristiques dans le cadre spécifique de la réduction.

Dans une dernière partie (§ 3.3) nous soulignons les difficultés rencontrées lorsque l'hypothèse "A matrice non négative" est levée et nous précisons les modifications à apporter pour pouvoir exploiter dans ce cas-là les heuristiques définies précédemment.

3.1 - Heuristique AGNES 1 et 2

Les deux heuristiques AGNES 1 et AGNES 2 ne sont qu'une adaptation dans le cadre d'un code de réduction des méthodes SURRO 1 et 2 décrites en détail dans [13].

L'idée directrice de ces méthodes est d'exploiter le concept classique de la contraction de contraintes dû à Glover (1965).

Au programme linéaire en variables bivalentes à m contraintes

(B) $\max cx \quad s.c. \; Ax \leq b \; ; \; x \in V$

sont associés, pour chaque multiplicateur $w \in \mathbb{R}_+^m$, le programme linéaire en variable bivalentes à une seule contrainte dite *composite*

(B(w)) $\max cx \quad s.c. \; wAx \leq wb \; ; \; x \in V$

et sa relaxation

$(\overline{B(w)})$ $\max cx \quad s.c. \; wAx \leq wb \; ; \; x \in [V]$

Les deux méthodes heuristiques utilisent systèmatiquement les informations et propriétés fournies par la relaxation $(\overline{B(w)})$. Ce choix de relaxation est motivé par le fait que

- d'une part, la contrainte composite permet d'engranger un maximum d'informations provenant de chacune des contraintes initiales, ce qui est plus riche (et à la fois plus rapide) que d'examiner chaque contrainte séparément.

- d'une part, nous possédons un algorithme performant et d'implantation facile pour la résolution du problème $(\overline{B(w)})$: l'algorithme NKR (Fayard, Plateau, [8]) de complexité temporelle linéaire en moyenne.

3.1.1 - Caractéristiques essentielles

AGNES 1 et AGNES 2 sont adaptés aux problèmes en variables bivalentes dont la structure est la suivante

$$(B) \quad \begin{cases} \max \ cx \\ \text{s.c. } Ax \leq b \\ x \in V \end{cases}$$

avec $\begin{cases} c \in \mathbb{R}^n \\ b \in \mathbb{R}^m \\ A \in \mathbb{R}_+^{m \times n} \text{ (A matrice non négative)} \end{cases}$

On peut sans restreindre la généralité supposer que

$$c \in \mathbb{R}_+^n, \quad b \in \mathbb{R}_+^m$$

En outre, on supposera que le problème (B) est "bien posé", c'est à dire qu'après une éventuelle réduction triviale (fixation de variables et/ou élimination de contraintes), les données possèdent les propriétés suivantes :

$$(H) \quad \begin{cases} \text{a) } \forall\, i \in \{1, 2, \ldots, m\} \quad \max\limits_{j=1,2,\ldots,n} a_{ij} \leq b_i \text{ (sinon } x_j^* = 0 \text{ si } \exists\, i : a_{ij} > b_i) \\ \text{b) } \forall\, i \in \{1, 2, \ldots, m\} \quad b_i < \sum\limits_{j=1}^{n} a_{ij} \text{ (sinon la contrainte i est éliminée)} \\ \text{c) } \forall\, j \in \{1, 2, \ldots, n\}, \ A^j \neq 0 \text{ (sinon } x_j^* = 1 \text{ lorsque } A^j = 0) \end{cases}$$

a) Algorithme AGNES 1

L'algorithme glouton AGNES 1 génère, en plusieurs itérations d'un même processus, un ensemble de solutions réalisables pour le problème (B). Le cardinal de cet ensemble est fixé a priori par l'utilisateur qui choisira la meilleure solution notée $\underline{x}^*$ (de valeur $\underline{v}(B)$).

Une solution réalisable $\underline{x}$ fournie au cours d'une itération du processus est construite en un nombre fini d'étapes qui se décomposent toutes cmme suit :

étape $k \geq 1$

Etant donnés

(i) les ensembles des indices des *variables fixées* (à 1 ou à 0) au cours des étapes 1, 2, ..., k-1 : (pour $k = 1$, $X_o = X_1 = \emptyset$)

$$X_o = \{j \in \{1, \ldots, n\} \mid x_j = 0\} \; ; \; X_1 = \{j \in \{1, \ldots, n\} \mid x_j = 1\}$$

(ii) l'ensemble M des numéros des contraintes *non redondantes* compte tenu de la donnée de X_o et de X_1 :

i.e. en posant $X_2 = \{1, \ldots, n\} \setminus (X_o \cup X_1)$

$$M = \{i \in \{1, 2, \ldots, m\} \mid \sum_{j \in X_2} a_{ij} > b_i - \sum_{j \in X_1} a_{ij}\}$$

(pour $k = 1$, $M = \{1, 2, \ldots, m\}$)

on considère le sous-problème (B^k) de (B) (à $n_k = |X_2|$ variables et à $m_k = |M|$ contraintes), supposé "bien posé" (voir hypothèse (H)) :

$$(B^k) \quad \left[\begin{array}{l} e\, c_{X_1} + \max \; \bar{c}x \\ \qquad \text{s.c. } \bar{A}x \leq \bar{b} \\ \qquad\qquad x \in \bar{V} \equiv \{x \in \mathbb{R}^{n_k} \mid x_j = 0 \text{ ou } 1 \; \forall \, j \in X_2\} \end{array} \right.$$

où $e_j = 1 \; \forall \, j \in X_1$; $\bar{c} = c_{X_2}$; $\bar{A} = A_M^{X_2}$; $\bar{b} = b_M - \sum_{j \in X_1} A_M^j$.

Dans le cas où n_k est suffisamment petit (inférieur à 10 par exemple), cette étape k (la dernière de l'itération en cours) consiste à résoudre exactement (B^k) par une procédure simple d'énumération implicite ([13], Annexe 2).

Dans le cas général, l'étape se déroule en 3 phases :

Phase 1 : Associé à un paramètre de perturbation μ voisin de 1, le calcul d'un vecteur "multiplicateur" w de $\mathbb{N}^{m_k}$ permet la construction d'une relaxation "composite" de (B^k)

$$(B^k(w, \mu)) \quad \begin{cases} e\, c_{X_1} + \max \ \bar{c}x \\ \text{s.c. } w\bar{A}x \leq \mu w\bar{b} \\ x \in \bar{V} \end{cases}$$

Phase 2 : L'algorithme NKR de complexité linéaire en moyenne (voir [8],[10]) fournit une solution optimale $\bar{x}$ du programme linéaire associé à $(B^k(w, \mu))$ dans le but de déterminer un sous-ensemble U^* de

$$U = \{j \in X_2 \mid \bar{x}_j = 1\}$$

tel que $\sum_{j \in U^*} \bar{A}^j \leq \bar{b}$

Les variables d'indices appartenant à U^* sont alors fixées à 1, ce qui implique l'augmentation (resp. la diminution) de X_1 (resp. de X_2).

Phase 3 : Le sous-problème ainsi obtenu est éventuellement réduit en nombre de variables et de contraintes de manière à satisfaire l'hypothèse (H) (ce qui implique une éventuelle modification de X_o, X_1, X_2 et M).

Note :
Le problème "bien posé" ainsi construit n'est autre que le problème (B^{k+1}) considéré à l'étape suivante.

□

L'itération s'achève lorsque toutes les variables sont fixées à 1 ou 0 (i.e. $X_2 = \emptyset$) ; la solution approchée $\underline{x}$ obtenue est donc définie par :

$$\underline{x}_j \quad \begin{cases} 1 \quad \forall\, j \in X_1 \\ 0 \quad \forall\, j \in X_o \end{cases}$$

(avec $X_o \cup X_1 = \{1, 2, \ldots, n\}$).

L'organigramme de l'algorithme AGNES 1 est donné par la figure 3.1. Tous les détails des différents modules figurant dans cet organigramme sont explicités dans ([13], p. 25).

b) Algorithme AGNES 2

L'algorithme glouton AGNES 2 engendre en un nombre fini d'étapes une seule solution approchée $\underline{x}$ associée à un minorant $\underline{V}(B)$ de $V(B)$.

A la différence de la méthode précédente, l'algorithme AGNES 2 contrôle au cours de chaque étape grâce au paramètre entier α le nombre de variables fixées à 1 ou à 0 ; en conséquence cela permet de contrôler également le nombre d'étapes dans le but d'obtenir une complexité théorique (et naturellement pratique) linéaire en moyenne. Ce nombre d'étapes (qui croit de manière logarithmique avec la taille n) est en général bien supérieur à celui (2 ou 3 étapes quelque soit la taille n) d'une itération de l'algorithme AGNES 1 ; c'est ce qui explique le choix de ne construire pour chaque valeur du paramètre entier α qu'une seule solution approchée par l'algorithme AGNES 2.

Chaque étape du processus se décompose comme suit :

étape $k \geq 1$

A l'instar de l'algorithme AGNES 2, étant donnés

$X_\varepsilon = \{j \in \{1, 2, \ldots, n\} \mid x_j$ fixée à la valeur ε au cours des étapes $1, 2, \ldots, k-1\}$ où $\varepsilon \in \{0, 1\}$

$X_2 = \{1, 2, \ldots, n\} \setminus (X_o \cup X_1)$; $n_k = |X_2|$

$M = \{i \in \{1, 2, \ldots, m\} \mid$ contrainte i non redondante$\}$; $m_k = |M|$

on considère le sous-problème de (B) suivant :

$$(B^k) \quad \left[\begin{array}{l} e\, c_{X_1} + \max \ \bar{c}x \\ \quad \text{s.c.} \ \bar{A}x \leq \bar{b} \\ \quad x \in \bar{V} = \{x \in \mathbb{R}^{n_k} \mid x_j = 0 \text{ ou } 1 \ \forall j \in X_2\} \end{array} \right.$$

où $e_j = 1 \ \forall j \in X_1$; $\bar{c} = c_{X_2}$; $\bar{A} = A_M^{X_2}$; $\bar{b} = b_M - \sum_{j \in X_1} A_M^j$.

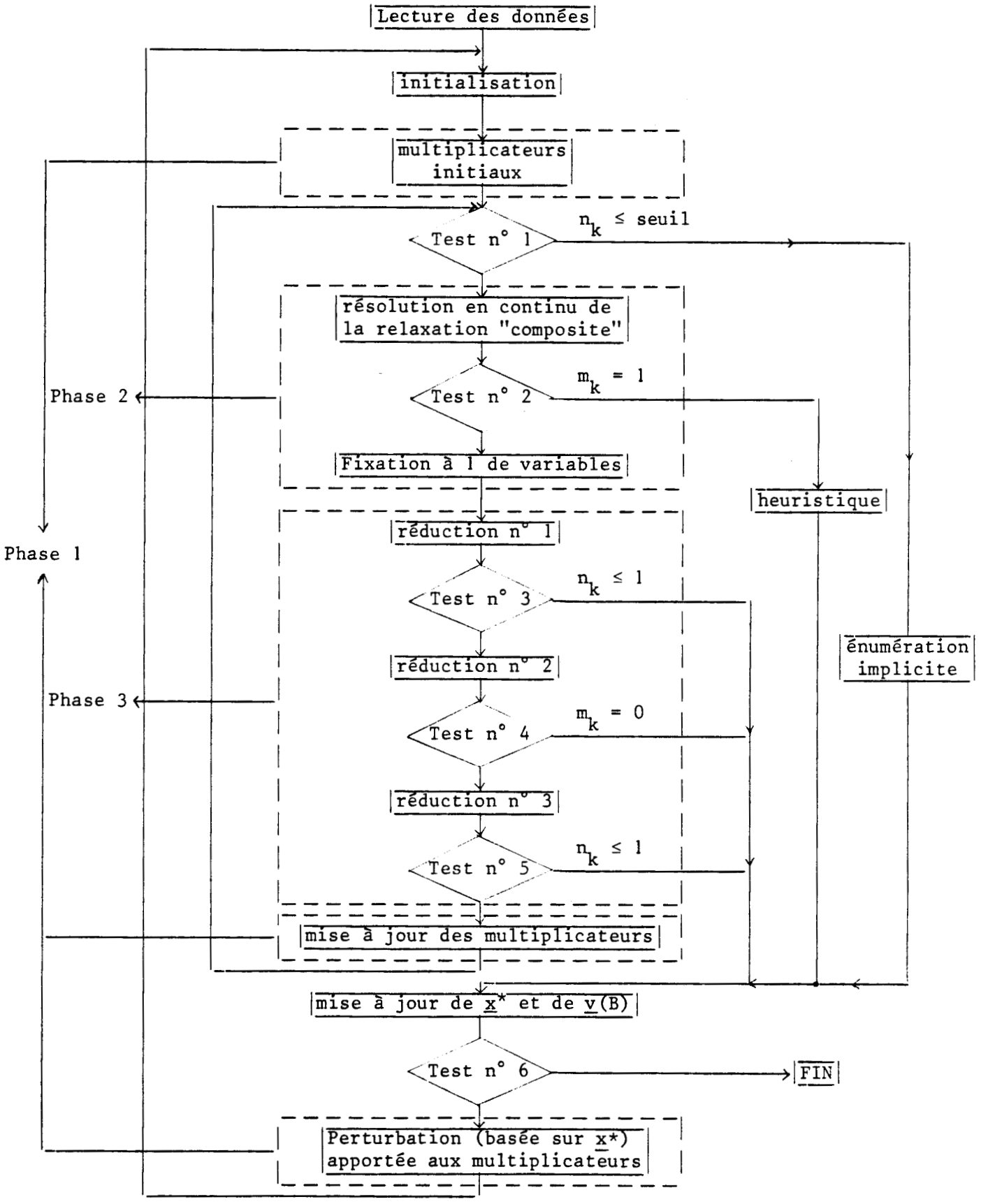


Figure 3.1 : Organigramme de AGNES 1

Dans le cas particulier où n_k est suffisamment petit (inférieur à 10), cette étape k (la dernière de l'algorithme) consiste à résoudre exactement (B^k) par une procédure simple d'énumération implicite ([13], Annexe 2).

Dans le cas général, l'étape k se déroule également en trois phases : seule la seconde d'entre elles diffère de la phase 2 de l'algorithme AGNES 1 :

Phase 2 : une solution optimale $\bar{x}$ du programme linéaire associé à

$$(B^k(w)) \quad \begin{cases} e\, c_{X_1} + \max \ \bar{c}x \\ \text{s.c. } w\bar{A}x \leq w\bar{b} \quad \text{où } w \in \mathbb{N}^{m_k} \\ x \in \bar{V} \end{cases}$$

(obtenue par l'algorithme NKR de complexité linéaire en moyenne) est exploitée de la façon suivante : deux sous-ensembles de X_2, $U^* \subset U = \{j \in X_2 \mid \bar{x}_j = 1\}$ et $L^* \subset L = \{j \in X_2 \mid \bar{x}_j = 0\}$, sont déterminés de sorte que :

(i) $|U^* \cup L^*| = \lfloor n_k/\alpha \rfloor$ où α est un paramètre entier donné a priori (constant pour chaque étape)

(ii) $\sum_{j \in U^*} \bar{A}^j \leq \bar{b}$

(iii) Les valeurs absolues des coûts réduits à l'optimum de $\overline{(B^k(w))}$ d'indices appartenant à $U^* \cup L^*$ sont supérieures à celles d'indices appartenant à $X_2 \setminus (U^* \cup L^*)$.

Les variables d'indices appartenant à U^* sont fixées à 1 (X_1 est augmenté de U^*) et celles d'indices appartenant à L^* sont fixées à 0 (X_o est augmenté de L^*) ; ce qui implique la diminution de X_2 de $U^* \cup L^*$.

L'organigramme de l'algorithme AGNES 2 est donné par la figure 3.2. Tous les détails des différents modules figurant dans cet organigramme sont explicités dans ([13], p. 25).

3.1.2 - Complexités théorique et pratique

De par la construction même de ces deux heuristiques, seule l'étude de la complexité théorique pour l'algorithme AGNES 2 est envisageable.

Toutefois, deux points nous amènent à conjecturer que AGNES 1 est de complexité linéaire en moyenne.

(i) Au vu des expériences numériques, pour chaque itération très peu d'étapes sont nécessaires, ceci indépendamment de la taille n initiale. En fait, la première itération permet de fixer à leur valeur finale un grand nombre de variables (une interprétation géométrique est que la solution réalisable obtenue à la première itération est très proche des facettes du polyèdre définissant le domaine des solutions réalisables) ([13], exemples p. 26).

(ii) En dehors des modules de complexité linéaire (réduction, calcul des multiplicateurs, construction du knapsack), chaque étape exige

- une résolution en continu du knapsack, de complexité linéaire en moyenne (algorithme NKR de Fayard, Plateau)
- une procédure de libération (c'est à dire la construction de U^* à partir de U : [13], p. 22, 24) comportant un classement de complexité $O(\xi(\gamma-\xi))$ avec $\gamma = |U|$ et $\xi = |U^*|$. Les expériences numériques montrent que :
  - quelle que soit l'étape, $\gamma-\xi$ est très petit devant n
  - γ est proche de n à la première étape
  - γ est petit devant n pour les autres étapes

Ces remarques permettent d'espérer pour la procédure de libération une complexité linéaire.

Pour l'algorithme AGNES 2, le nombre d'étapes est connu et géré par le choix du paramètre entier α ce qui nous a permis d'établir le résultat suivant :

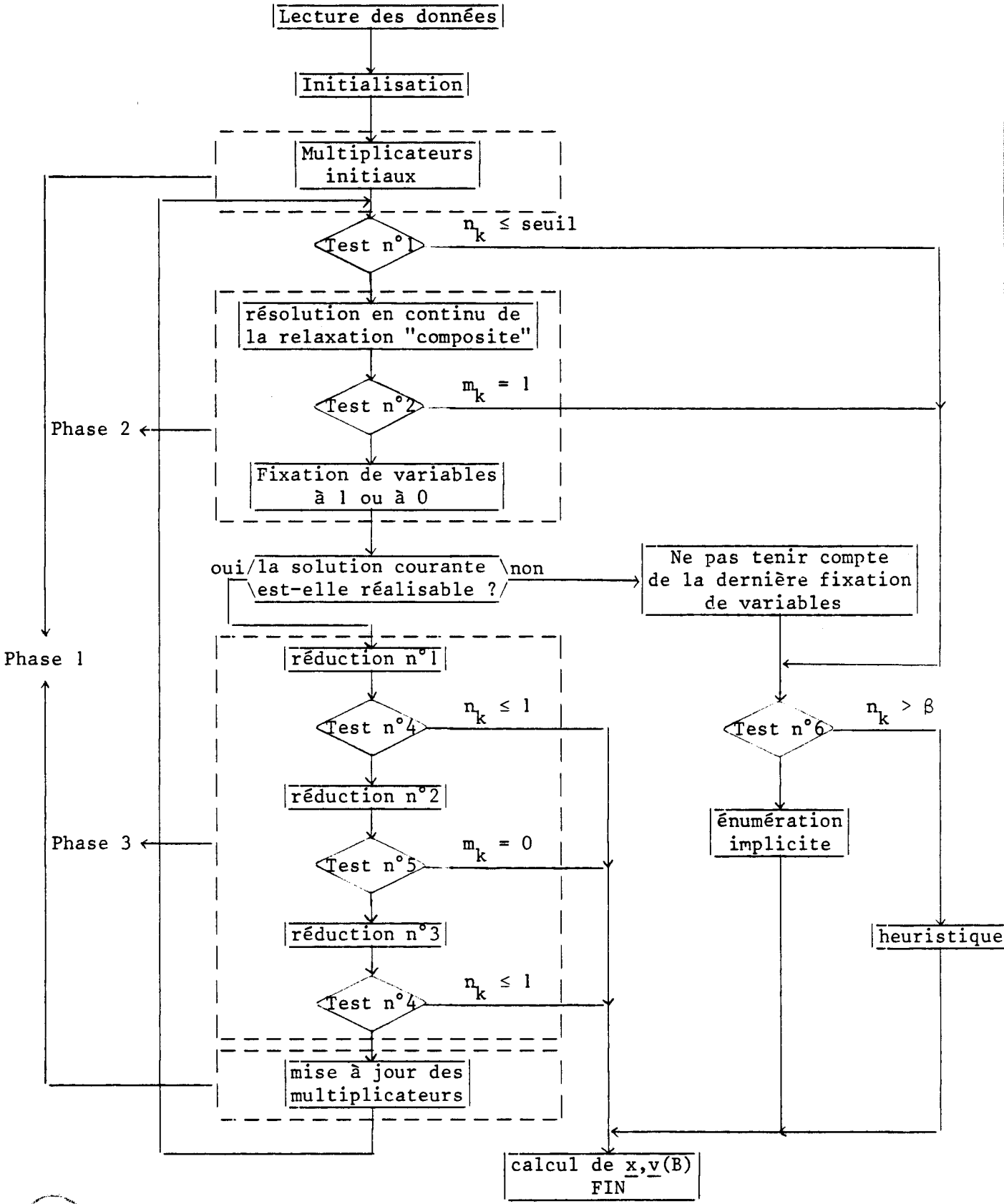


Figure 3.2 : Organigramme de AGNES 2

Théorème 3.1

L'algorithme AGNES 2 est de **complexité** $O(n)$ ***en moyenne*** *(*) et si* $T(n)$ *désigne le nombre d'opérations effectuées en moyenne par AGNES 2 on a*

$$T(n) \leq K(m) \times (\alpha-1)\ (1 - \frac{\beta}{n}\ (1 - \frac{1}{\alpha}))\ n + K'(m, \beta)$$

K, K' *étant des constantes entières dépendant de* m *et des règles adoptées pour le dénombrement dans le cas le plus défavorable,* β *un seuil fixé à partir duquel le sous-problème résiduel est résolu par une procédure énumérative.*

Démonstration

On prend en compte, comme opérations, les opérations arithmétiques (opa), les comparaisons (comp) et les affectations (aff). A l'itération k, soient n_k le nombre de variables libres (= $|X_2|$) et m_k le nombre de contraintes effectives (= $|M|$) ; alors le nombre d'opérations $T(n_k)$ à cette itération peut se décomposer comme suit :

. multiplicateurs : $n_k \times (5m_k$ opa)

. contraction : $M_k \times (2m_k$ opa + m_k aff) + ($2m_k$ opa + 1 aff)

. réduction n° 1 : $M_k \times (m_k$ comp)

. coûts réduits : $n_k \times (4$ opa)

. fixation : $\lfloor n_k/\alpha \rfloor \times (1$ comp + $3m_k$ aff)

. mise à jour : $2\lfloor n_k/\alpha \rfloor$ (opa)

. réduction n° 2 : $(n_k - \lfloor n_k/\alpha \rfloor) \times (m_k$ comp)

. réduction n° 3 : $(n_k - \lfloor n_k/\alpha \rfloor) \times (m_k$ opa + m_k comp)

. résolution en continu : $\leq 4\eta\ n_k$ }

. sélection des meilleurs coûts réduits : $4\eta\ n_k$ }

(*) Sous l'hypothèse que les coefficients de la fonction économique et de la contrainte soient tirés au hasard suivant la loi uniforme.

sont des adaptations directes de l'algorithme NKR de *complexité* $O(n)$ *en moyenne* (*) (voir [8]), η étant un entier dépendant des règles adoptées pour le dénombrement des opérations dans le cas le plus défavorable.

En résumé, $T(n_k) \leq K(m)\ n_k \quad \forall\ k \geq 1$ (1)

où K est un entier dépendant de m et de η, cette majoration étant faite en supposant qu'aucune élimination de contraintes n'est effectuée

i.e. $m_k = m, \quad \forall\ k \geq 1.$

Déterminons le nombre maximum NI d'itérations effectuées par AGNES 2 nécessaires pour que n_k soit inférieur ou égal à un seuil β fixé, à partir duquel le sous-problème résiduel est résolu par la procédure L.I.F.O en un nombre maximum d'opérations $K'(m, \beta)$, indépendant de n.

NI sera calculé sous l'hypothèse qu'aucune des phases 1 et 2 de la réduction ne fixe de variables à 0 ou à 1 (diminution stricte de n_k).

Par récurrence : $n_1 = n$

$$n_2 = n_1 - \lfloor n_1/\alpha \rfloor \simeq n(1 - \frac{1}{\alpha})$$

(en fait, si on veut avoir explicitement une inégalité en $\leq$, on prend $\lceil n_1/\alpha \rceil$)

$$n_{k+1} = n_k(1 - \frac{1}{\alpha}) = n(1 - \frac{1}{\alpha})^k \qquad (k \geq 1)$$

$\alpha > 1 \Rightarrow \lim_{k\to\infty} n_k = 0$

NI sera le plus petit entier k tel que $n(1 - \frac{1}{\alpha})^k \leq \beta$.

$$\text{soit } k \times \text{Log } (1 - \frac{1}{\alpha}) \leq \text{Log } (\beta/n)$$

$$\text{soit } k \geq \text{Log } (\beta/n) \ / \ \text{Log } (1 - \frac{1}{\alpha})$$

$$\text{d'où NI} = \left\lceil \frac{\text{Log } n \ - \ \text{Log } \beta}{\text{Log } \alpha \ - \ \text{Log}(\alpha-1)} \right\rceil \qquad (2)$$

(*) Sous l'hypothèse que les coefficients de la fonction économique et de la contrainte soient tirés au hasard suivant la loi uniforme.

De (1) et (2), nous pouvons déduire immédiatement que

$$T(n) \leq \sum_{k=1}^{NI} K(m)\ n\ (1 - \frac{1}{\alpha})^{k} + K'(m, \beta)$$

$$\text{or} \quad \sum_{k=1}^{NI} (1 - \frac{1}{\alpha})^{k} = \frac{(1 - \frac{1}{\alpha}) - (1 - \frac{1}{\alpha})^{NI+1}}{1 - (1 - \frac{1}{\alpha})}$$

$$= (\alpha-1)\ (1 - (1 - \frac{1}{\alpha})^{NI}) \quad \text{(après simplifications).}$$

Posons $u = \text{Log}\ (\frac{\alpha}{\alpha-1})$, $(u > 0)$

on peut écrire : $NI = \left\lceil \frac{\text{Log } n - \text{Log } \beta}{\text{Log } \alpha - \text{Log}(\alpha-1)} \right\rceil = \left\lceil \frac{\text{Log } n - \text{Log } \beta}{u} \right\rceil$

$$= \frac{1}{u}\ \text{Log}\ (\frac{n}{\beta}) + \varepsilon \text{ où } \varepsilon \in [0, 1[$$

d'où

$$1 - (1 - \frac{1}{\alpha})^{NI} = 1 - (\frac{\alpha}{\alpha-1})^{-NI} = 1 - e^{-NI\ u}$$

$$= 1 - \frac{\beta}{n}\ e^{-\varepsilon\ u}$$

$$\left.\begin{array}{l} u > 0 \\ \varepsilon \in [0, 1[\end{array}\right\} \Rightarrow \varepsilon\ u < u \Rightarrow e^{-\varepsilon\ u} > e^{-u}$$

$$\Rightarrow 1 - \frac{\beta}{n}\ e^{-\varepsilon\ u} < 1 - \frac{\beta}{n}\ e^{-u} = 1 - \frac{\beta}{n} \times (1 - \frac{1}{\alpha})$$

en définitive, nous obtenons la majoration suivante

$$\underline{T(n) \leq K(m) \times (\ -1)}\ (1 - \frac{\beta}{n}\ (1 - \frac{1}{\alpha}))\ \underline{n + K'(\beta, m)} \qquad (3)$$

□

Remarque

$K'(\beta, m) \leq C\ m\ 2^{\beta}$ où C est une constante. Pour α et n fixés, les deux termes de la somme dans (3) sont en opposition : quand β varie, l'un croit et l'autre décroit. En pratique, la majoration sera d'autant meilleur que β est près de 0, ceci au détriment de la précision désirée. Dans les applications β est égal à 10.

□

En pratique nous avons pu calculer les droites de régression linéaire du temps en fonction de la taille n, sur 30 problèmes tirés au hasard (Shih, [61]). Les caractéristiques de ces problèmes sont les suivantes ; le nombre de contraintes est 5 tandis que le nombre de variables varie de 30 à 90. Les a_{ij} sont générés de manière aléatoire à partir d'une loi de distribution uniforme discrète sur [0,99], tandis que les c_j sont distribués entre 1 et 999. Le choix des seconds membres b_i est destiné à faire varier dans un large éventail le degré de relachement des contraintes : $s_i = (\sum_{j=1}^{n} a_{ij} - b_i) / \sum_{j=1}^{n} a_{ij}$, i = 1, 2, ..., m. Si t est le temps en dixième de seconde (sur CII HB IRIS 80) et en posant n' = n/10, nous obtenons respectivement :

pour AGNES 1 : $t(n') = 0.5\ n' + 0.2$

pour AGNES 2 : $t(n') = 0.16n' + 0.5$.

3.1.3 - *Résultats numériques*

Les expériences numériques ont été menées sur cinquante problèmes tests de la littérature dont les données sont intégralement dans [13], Annexe 1.

Tous possèdent les caractéristiques suivantes :

(i) l'hypothèse (H) est vérifiée

(ii) la matrice A est non négative

(iii) la matrice A est dense : $\rho > 67$

où $\rho = \frac{100}{m \times n} \times$ nombre total des a_{ij} non nuls, i = 1, 2, ..., m,

j = 1, 2, ..., n.

La liste de ces problèmes se décompose comme suit :

- Petersen [53] : 7 P
- Hansen, Plateau [54] : 2 HP
- Weingartner, Ness [66] : 8 W
- Shih [61] : 30 WS
- Senju, Toyoda [59] : 2 ST
- Fleisher [12] : 1 F

Tous les problèmes sont d'origine concrète sauf ceux de W. Shih dont les caractéristiques sont données au § 3.1.2.

Les tableaux 3.1 et 3.2 donnent respectivement les performances de AGNES 1 et AGNES 2 sur les 50 problèmes tests, avec les multiplicateurs calculés à partir des structures. (Précisons que dans ce cas les multiplicateurs sont calculés de telle façon à accorder plus de poids aux contraintes qui risquent a priori d'être saturées plus rapidement que les autres ; pour cela on calcule pour chaque contrainte son degré de relachement :

$s_i = \left(\sum_{j=1}^{n} a_{ij} - b_i \right) / \sum_{j=1}^{n} a_{ij}$ $\quad i = 1, 2, \ldots, m$ pour définir les multiplicateurs initiaux à valeurs entières positives : $w_i = \lfloor 100 \times s_i \rfloor$).

Les tableaux 3.3 et 3.4 comparent respectivement sur 20 problèmes réels l'apport des multiplicateurs structurels, lagrangiens et composites. Ces deux derniers types, multiplicateurs optimaux (en pratique presque optimaux) respectifs des duaux lagrangien et composite, sont générés respectivement par un algorithme de type sous-gradient et par un algorithme de type quasi-sous-gradient décrits dans le chapitre 2.

Problème	$\underline{v}(B)$	déviation (**)	temps (*)	Problème	$\underline{v}(B)$	déviation	temps
P1	3800	0.	0	WS 6	5542	0.	2
P2	8336.9	4.24	1	WS 7	5567	0.	2
P3	3245	19.18	1	WS 8	5603	0.03	2
P4	6010	1.79	1	WS 9	5246	0.	1
P5	12400	0.	2	WS10	6338	0.01	3
P6	10415	1.91	1	WS11	5624	0.33	2
P7	16376	0.97	3	WS12	6338	0.01	2
HP1	3288	3.80	1	WS13	6159	0.	2
HP2	3153	1.03	1	WS14	6954	0.	3
W1	140128	0.81	0	WS15	7442	0.58	3
W2	130723	0.12	1	WS16	7287	0.02	2
W3	95627	0.05	0	WS17	8609	0.28	2
W4	119337	0.	1	WS18	9580	0.	4
W5	98796	0.	1	WS19	7698	0.	3
W6	130233	0.29	0	WS20	9445	0.05	3
W7	1095445	0.	3	WS21	9074	0.	3
W8	620060	0.68	4	WS22	8908	0.43	5
F	2085	2.52	1	WS23	8341	0.43	4
ST1	7761	0.14	19	WS24	10202	0.17	4
ST2	8685	0.42	14	WS25	9939	0.	4
WS1	4554	0.	1	WS26	9532	0.54	5
WS2	4506	0.66	2	WS27	9816	0.03	4
WS3	4080	0.85	1	WS28	9492	0.	4
WS4	4561	0.	1	WS29	9410	0.	4
WS5	4514	0.	1	WS30	11187	0.03	4

Tableau 3.1 : AGNES 1

(*) le temps est donné en dixièmes de seconde (avec troncature)

(**) la déviation est le rapport $100 \times \left(\frac{v(B) - \underline{v}(B)}{v(B)}\right)$

La déviation moyenne pour les 30 problèmes de Wei Shih est de 0.15 %.

Problème	$\underline{v}(B)$	Déviation (**)	Temps (*)	α	Problème	$\underline{v}(B)$	Déviation	Temps	α
P1	3800	0.	.0	3	WS 6	5539	0.32	3	7
P2	8706.1	0.	.0	3	WS 7	5503	1.14	1	3
P3	4015	0.	4	3	WS 8	5605	0.	3	4
P4	6120	0.	0	3	WS 9	5212	0.64	1	3
P5	12400	0.	2	3	WS10	6292	0.74	3	7
P6	10618	0.	1	5	WS11	5538	1.86	2	5
P7	16447	0.54	3	8	WS12	6268	1.12	2	5
HP1	3418	0.	4	5	WS13	6159	0.	1	3
SP2	3157	0.91	2	5	WS14	6923	0.44	2	4
W1	140778	0.35	0	3	WS15	7486	0.	2	4
W2	130723	0.12	0	3	WS16	7289	0.	1	3
W3	95517	0.16	0	3	WS17	8633	0.	1	3
W4	119337	0.	1	6	WS18	9540	0.41	3	5
W5	98526	0.27	0	3	WS19	7698	0.	2	4
W6	130233	0.29	0	3	WS20	9400	0.52	3	5
W7	1095264	0.01	8	6	WS21	9074	0.	4	7
W8	624319	0.	8	5	WS22	8947	0.	4	7
F	2076	2.94	3	3	WS23	8341	0.	3	3
ST1	7675	1.24	7	3	WS24	10172	0.03	4	6
ST2	8721	0.01	7	3	WS25	9922	0.46	3	3
WS1	4554	0.	0	3	WS26	9584	0.	2	3
WS2	4531	0.11	0	3	WS27	9819	0.	2	3
WS3	4106	0.21	2	6	WS28	9436	0.58	3	4
WS4	4561	0.	1	3	WS29	9345	0.69	3	4
WS5	4514	0.	1	3	WS30	11191	0.	2	3

Tableau 3.2 : AGNES 2

(*) et (**) : se référer au tableau 3.1

La déviation moyenne pour les 30 problèmes de Wei Shih est de 0.31 %

Problème	Structurel	Lagrangien	Composite
P1	3800	3800	3800
P2	8336.9	8336.9	8577.8
P3	3245	3900	3900
P4	6010	6010	6010
P5	12400	12220	12220
P6	10415	10516	10513
P7	16376	16468	16468
HP1	3280	3276	3276
HP2	3153	3057	3117
W1	140128	140477	140477
W2	130723	130723	130723
W3	95627	94908	94908
W4	119337	119337	119337
W5	98796	98796	98796
W6	130233	130233	130233
W7	1095445	1095352	1095352
W8	620060	620060	615853
F	2085	1988	1816
ST1	7761	7761	7761
ST2	8685	8685	8685

Tableau 3.3 : AGNES 1

La déviation moyenne pour les 20 problèmes réels testés est respectivement pour chaque type de multiplicateur envisagé :

- structurel : 1.88
- lagrangien : 1.46
- composite : 1.76

Pour AGNES 1, des bons multiplicateurs au sens de la valeur du dual ne donnent pas nécessairement les meilleurs minorants.

Problème	Structure		Lagrangien		Composite	
	Heuristique	α	Heuristique	α	Heuristique	α
P1	3800	3	3800	3	3800	3
P2	8706.1	3	8706.1	3	8706.1	3
P3	4015	3	4015	3	4015	3
P4	6120	3	6120	3	6120	3
P5	12400	3	12400	3	12400	3
P6	10618	5	10618	7	10618	4
P7	16447	8	16447	8	16447	8
HP1	3418	5	3418	7	3418	7
HP2	3157	5	3157	4	3157	6
W1	140778	3	141278	3	140728	3
W2	130723	3	130723	3	130723	3
W3	95517	3	95517	3	95517	3
W4	119337	6	119337	6	119337	6
W5	98526	3	98526	3	98526	3
W6	130233	3	130233	3	130233	3
W7	1095264	6	1095264	6	1095264	6
W8	624319	5	624319	5	624319	5
F	2076	3	2085	3	2139	3
ST1	7675	3	7675	3	7675	3
ST2	8721	3	8721	5	8721	5

Tableau 3.4 : AGNES 2

La déviation moyenne pour les 20 problèmes réels testés est respectivement pour chaque type de multiplicateur envisagé :

- structurel : 0.34
- lagrangien : 0.30
- composite : 0.20

Au contraire de AGNES 1, des "bons" multiplicateurs au sens du dual donnent toujours un minorant au moins aussi bon que dans le cas structurel.

Dans le tableau 3.5 sont rassemblés quelques résultats comparatifs par rapport aux 7 méthodes suivantes (présentées dans [11]).

Balas, Martin ([2], 1980) : UNIVAC 1108 (*)

Senju, Toyoda ([59], 1968) : IBM 360/75 (*)

Kochenberger, Mc Karl, Wyman ([42], 1974) : "

Hillier ([33], 1969) : "

Toyoda ([65], 1975) } temps de calcul

Loulou, Michaelides ([46], 1978) } non précisé

Guignard ([25], 1972) } par les auteurs

Lorsqu'ils sont précisés, les temps de calcul sont donnés en secondes par rapport aux calculateurs cités ci-dessus. Sont soulignés les meilleures valeurs obtenues

Problèmes / Heuristiques	P6		P7		W1		W7	
AGNES 1	10415	.1	16376	.2	140128	.0	<u>1095445</u>	.3
AGNES 2	<u>10618</u>	.4	16447	.3	<u>141278</u>	.0	1094806	.4
BALAS, MARTIN	10588	.5	<u>16499</u>	.8				
SENJU, TOYODA	10516	.18	16337	.22	139878	.05	1094611	.22
KOCHENBERGER	10313	.10	16031	.13	140618	.05	1095332	.37
HILLIER	8255	7.65	<u>16499</u>	13.05	127244	2.95	1088177	65.23
TOYODA	9888		15897		139278		990337	
LOULOU	10199		15897		<u>141278</u>		991921	
GUIGNARD	10427		16274					

Tableau 3.5

(*) *Les calculateurs UNIVAC 1108 et IBM 360/75 sont au moins aussi rapides que le CII HB IRIS 80.*

Les performances des algorithmes AGNES 1 et AGNES 2 sont pour la précision au moins aussi bonnes que celles de la méthode BALAS-MARTIN, qui était sans aucun doute la meilleure de toutes celles existantes pour le type de problème traité . Par contre, on peut espérer faire beaucoup mieux au niveau du temps de calcul, surtout pour des problèmes de grande taille. Alors que nos deux algorithmes sont de complexité linéaire en moyenne, la méthode de BALAS-MARTIN est de complexité $O((m+n)^4)$ et, en pratique, demande en moyenne deux à trois fois le temps de calcul nécessaire à la résolution du problème par la méthode simplex en variables bornées. Par exemple, les temps moyens sur dix problèmes de taille 5×100 sont respectivement en secondes ; AGNES 1 : .53(*), AGNES 2 : .33(*), BALAS-MARTIN : 1.41(**).

3.1.4 - *Influence des paramètres μ et α*

Dans la phase 1 de l'algorithme AGNES 1, la perturbation du second membre grâce à l'introduction du paramètre μ permet dans une grande majorité des cas d'améliorer la solution heuristique initiale.

La relaxation "composite" du sous-problème (B^k), que l'on résout par l'algorithme NKR est le problème $(\bar{B}^k(w, \mu))$ suivant :

$$(\bar{B}^k(w, \mu)) \quad \begin{cases} e.c_{X_1} + \max \ \bar{c}x \\ \text{s.c. } w\bar{A}x \leq \mu \, w \, \bar{b} \\ x \in \bar{V} \end{cases}$$

où $0 \leq 1 - \eta \leq \mu \leq 1 + \eta$, η étant le coefficient de perturbation maximum

- si $\mu = 1$: $(\bar{B}^k(w, 1)) \equiv (\bar{B}^k(w))$
- si $\mu \in [1-\eta, 1]$, cette perturbation favorise l'aspect primal en diminuant le domaine du problème ($|U-U^*|$ est presque nul). En effet, la

(*) temps obtenus sur CII HB IRIS 80
(**) temps obtenus sur UNIVAC 1108

solution $\underline{x}$ sera construite à chaque étape k à partir de solutions "presque réalisables" de (B^k).

- si $\mu \in [1, 1+\eta]$, la perturbation favorise l'aspect dual ($|U-U^*|$ est plus grand que dans le cas $\mu = 1$). La solution $\underline{x}$ sera construite à chaque étape k à partir de solutions non réalisables de (B^k).

Un échantillonnage de valeurs de μ permet d'envisager les différents comportements de l'algorithme AGNES 1. Le tableau 3.6 rapporte les expériences numériques menées sur 9 problèmes concrets pour lesquels la valeur $\mu = 1$ du paramètre ne donnait pas la valeur optimale ; sont soulignés les minorants obtenus meilleurs que le minorant initial.

	F	P6	P7	HP1	HP2	W1	W3	W7	W8
μ=0.8	<u>2025</u>	10377	16468	<u>3384</u>	<u>3117</u>	<u>140568</u>	94908	<u>1095382</u>	<u>620060</u>
μ=1	1988	10516	16468	3276	3057	140477	94908	1095352	615853
μ=1.2	<u>2068</u>	<u>10552</u>	<u>16499</u>	<u>3300</u>	3057	<u>140568</u>	<u>95627</u>	1082542	610759

Tableau 3.6 : AGNES 1

Pour l'algorithme AGNES 2 l'influence du paramètre entier α (qui donne le nombre de variables fixées à 0 ou à 1 à chaque étape) sur la précision des valeurs obtenues est plus difficile à cerner. Toutefois on peut dire de manière très générale que la précision des solutions est une fonction non décroissante de α. En contre-partie, le nombre d'itérations et le temps augmentent bien sûr en fonction de α. Une politique raisonnable est de démarrer avec $\alpha = 3$ et d'augmenter α lorsque le sous-problème courant est contradictoire.

Le tableau 3.7 donne l'évolution des minorants quand α varie de 3 à 6 : sont indiqués le temps en dixième de secondes et le nombre d'étapes pour 7 problèmes concrets.

Problème	α = 3			α = 4			α = 5			α = 6		
	Minorant	tps	étape	Minorant	tps	étape	Minorant	tps	étape	Minorant	tps	étape
P6	10530	1	4	10436*	2	4	10618	4	6	10618	4	8
P7	16388	2	4	16390	2	6	16390	2	8	16141	2	7
HP1	3385	0	3	3386	1	4	3418	4	5	3418	4	6
HP2	3057	0	4	3098	1	5	3157	2	7	3108	2	7
W1	140778	0	2	140778	0	3	140778	0	4	140778	0	5
W4	115831*	0	1	115831*	0	1	115831*	0	2	119337	1	6
W8	620872	4	6	623952	7	9	624319	4	11	624319	6	13

Tableau 3.7 : AGNES 2

(*) La taille n_k du problème (B_k) étant supérieure au seuil (10), le problème est résolu de manière heuristique et non de manière exacte.

3.2 - Heuristiques dans le cadre de la réduction

3.2.1 - *Insertion dans le code de réduction*

Bien que toute méthode heuristique puisse être utilisée en lieu et place de toute méthode exacte avec les inconvénients et avantages que ce choix comporte, les deux heuristiques proposées ont été conçues dans notre esprit pour être insérées dans un algorithme de réduction. Leur finalité est donc de produire le meilleur minorant possible à la valeur d'un problème donné de maximisation en un temps de calcul pratiquement négligeable en comparaison de celui demandé par la résolution exacte de ce même problème. Dans ce contexte, le problème "temps de calcul" est essentiel car ces deux heuristiques seront amenées à être appelées très souvent en tant que sous-programmes d'un code de réduction.

En regard des meilleurs résultats obtenus avec AGNES 2 sur les problèmes concrets nous avons utilisé de façon systématique cet algorithme

sur chaque sous-problème traité pendant le déroulement de l'algorithme général de réduction. Par contre, l'algorithme AGNES 1 est utilisé en deux occasions précises : d'une part, en tant que phase préliminaire pour le calcul d'un premier minorant (dont on a besoin pour la mise en oeuvre d'un algorithme de sous-gradient qui va engendrer des multiplicateurs lagrangiens), d'autre part lorsque AGNES 2 s'achève par la considération d'un sous-problème contradictoire pour toutes les valeurs envisagées du paramètre α (appelée situation d'échec dans la figure 3.3).

La figure 3.3 précise l'enchaînement de ces deux algorithmes dans le cadre du code de réduction. Le module 1 n'est autre qu'un des algorithmes de résolution d'un dual associé au problème courant, évoqués dans le chapitre précédent. Le module 2 qui met en oeuvre tous les tests de réduction, fera l'objet du chapitre suivant.

Chaque appel du module AGNES 1 prend en compte systématiquement et successivement trois valeurs du paramètre μ : 0.8, 1., 1.2. De même chaque appel du module AGNES 2 prend en compte trois valeurs du paramètre α : 3, 6, 9 (ceci bien entendu tant que le nombre de variables du problème courant est supérieur ou égal à un seuil minimal fixé à la valeur 2α, sinon on sort du module).

3.2.2 - *Résultats numériques*

Le tableau 3.8 fait le bilan sur 18 problèmes concrets (7P, 8W, 2ST, 1F) et 30 problèmes générés de manière aléatoire (30WS), de l'efficacité des heuristiques dans le cadre de la réduction.

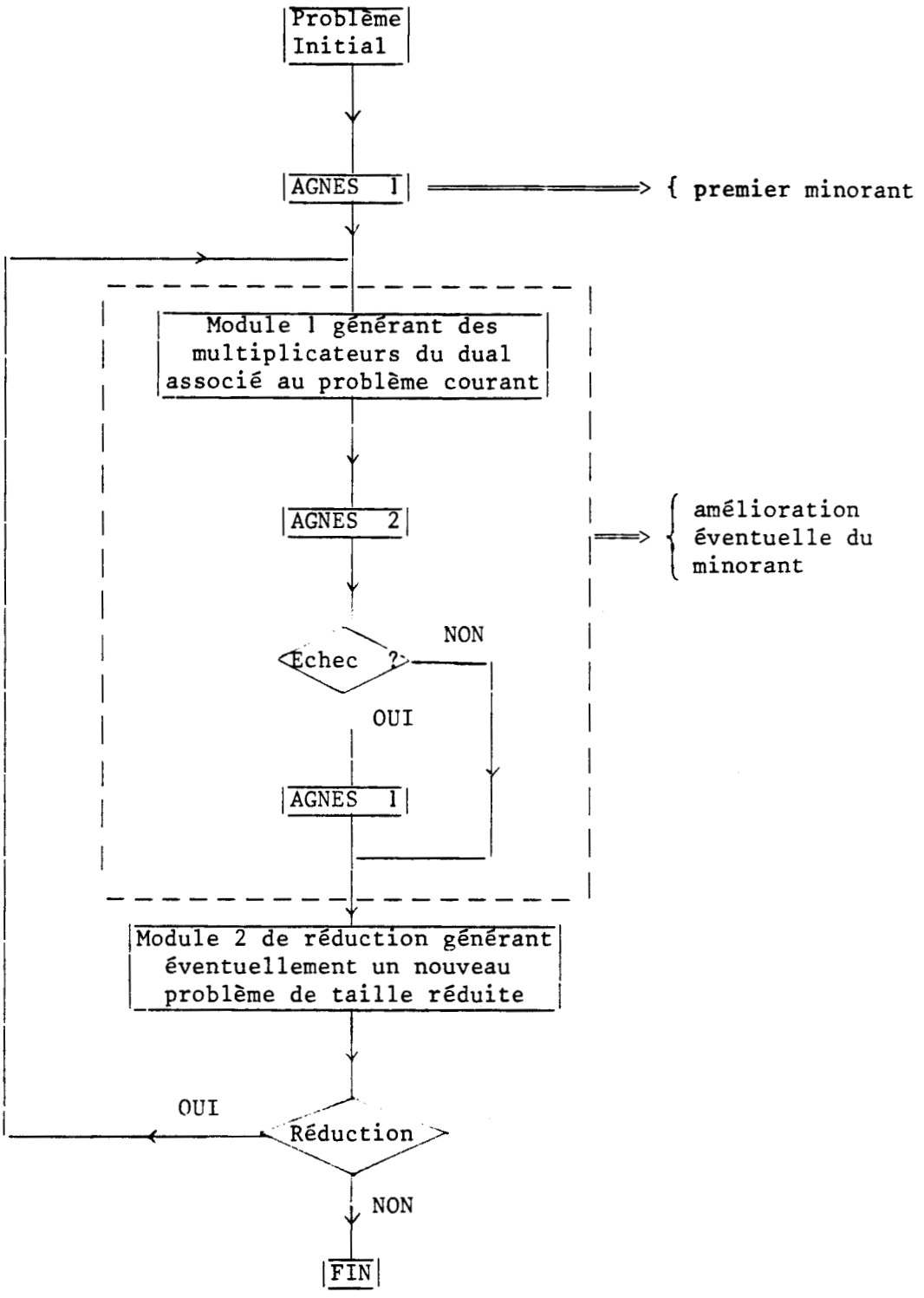


Figure 3.3

	Nbre de fois où la valeur optimale est obtenue		Nbre de fois où la valeur optimale n'est pas atteinte
	dès le premier minorant	en cours de réduction	
18 problèmes concrets	13	2	3
30 problèmes aléatoires	23	3	4

Tableau 3.8

3.3 - Cas de signes quelconques : bases d'une nouvelle heuristique

Considérons le problème en variables bivalentes

$$(B) \quad \begin{cases} \max \ cx \\ \text{s.c. } Ax \leq b \\ x \in V \end{cases}$$

avec $\begin{cases} c \in \mathbb{R}^n_+ \\ b \in \mathbb{R}^m \\ A \in \mathbb{R}^{n \times m}, \text{ matrice sans condition de signe} \end{cases}$

note : On peut toujours se ramener au cas d'un vecteur c positif ou nul par un changement de variables éventuel du type $x'_j = 1 - x_j$.

□

Pour passer au cas de signes quelconques, l'idée de départ était de voir dans quelle mesure on pouvait adapter les algorithmes AGNES 1 et 2. Si une telle extension ne pose pas de grosses difficultés pour AGNES 2, il n'en est pas de même pour AGNES 1. Avant de préciser les modifications à apporter à l'heuristique AGNES 2 permettant de jeter les bases d'une nouvelle heuristique, indiquons la difficulté rencontrée pour que l'on puisse en faire de même avec AGNES 1.

Rappelons que pour construire une solution réalisable du problème courant

$$(B^k) \qquad e\, c_{X_1} + \max \bar{c}x \text{ s.c. } \bar{A}x \leq \bar{b},\ x \in \bar{V} \equiv \{x \in \mathbb{R}^{n_k} \mid x_j = 0 \text{ ou } 1,\ \forall\, j \in X_2\}$$

où $\quad e_j = 1 \quad \forall\, j \in X_1 \ ;\ \bar{c} = c_{X_2} \ ;\ \bar{A} = A_M^{X_2} \ ;\ \bar{b} = b_M - \sum_{j \in X_1} A_M^j$

on part de la solution bivalente $[\bar{x}]$ obtenue en annulant la variable de base de $\bar{x}$, solution optimale du programme linéaire $(\bar{B}^k(w))$ défini par :

$$(\bar{B}^k(w)) \quad e\, c_{X_1} + \max \bar{c}x \quad \text{s.c. } w\bar{A}x \leq w\bar{b},\ x \in [\bar{V}].$$

La solution $[\bar{x}]$ n'étant pas en général réalisable pour (B^k), un processus dit "de retrait" détermine un sous-ensemble $U^* \subset U = \{j \in X_2 \mid \bar{x}_j = 1\}$ tel que :

$$\sum_{j \in U^*} \bar{A}^j \leq \bar{b}.$$

Ce processus, inspiré de l'algorithme proposé par Senju-Toyoda ([59]), consiste à mettre à 0 dans $[\bar{x}]$ la variable correspondant au "meilleur" rapport "plus petite diminution de la fonction économique / plus grand déplacement sur la meilleure direction de retrait vers le domaine des solutions réalisables", c'est à dire au plus petit rapport c_i/v_i.

avec $\quad v_i = \sum_{i \in M} \bar{a}_{ij} * \max \{0,\ \sum_{j \in U} \bar{a}_{ij} - \bar{b}_i\}$

ce qui correspond géométriquement (à un coefficient de proportionnalité près) à la norme de la projection orthogonale du vecteur non négatif $\bar{A}^j$ sur la direction définie par le plus court chemin du point $[\bar{x}]$ à l'enveloppe convexe de l'ensemble des solutions réalisables de (B^k).

Cette procédure simple et efficace dans le cas non négatif ne nous semble pas pouvoir s'étendre de façon acceptable au cas de signe quelconque.

Par contre on peut définir les bases d'une nouvelle heuristique à partir de l'algorithme AGNES 2 en adoptant ses différents modules (voir organigramme, figure 3.2) au cas de signe quelconque.

a) Module de réduction

Considérons, pour chaque contrainte i, $i \in \{1, 2, \ldots, m\}$, les sous-ensembles de $\{1, 2, \ldots, n\}$ définis par : $N_i^+ = \{j \mid a_{ij} \geq 0\}$ et $N_i^- = \{j \mid a_{ij} < 0\}$, alors les tests ci-dessous, introduits par Spielberg ([63], 1979) et repris entre autres par Crowder, Johnson et Padberg ([5], 1982) permettent successivement de :

1 Détecter la vacuité d'un problème

si $\exists\ i \in \{1,2,\ldots,m\}$ tel que $b_i - \sum_{j \in N_i^-} a_{ij} < 0$ alors le domaine des solutions réalisables du problème est vide

2 Fixer des variables à la valeur 0 ou 1

(i)

si $\exists\ i \in \{1, 2, \ldots, m\}$ et $\exists\ j \in N_i^+$ tels que $a_{ij} > b_i - \sum_{k \in N_i^-} a_{ik}$ alors $x_j \leftarrow 0$

(généralisation du test réduction 1)

(ii)

si $\exists\ i \in \{1, 2, \ldots, m\}$ et $\exists\ j \in N_i^-$ tels que $-a_{ij} > b_i - \sum_{k\ N_i^-} a_{ik}$ alors $x_j \leftarrow 1$

(iii)

si $\exists\ j \in \{1, 2, \ldots, n\}$ tel que $A^j = 0$ alors $x_j \leftarrow 1$

(ceci peut arriver après élimination de contraintes ; idem que le test réduction 3).

3 Eliminer des contraintes

si $\exists\ i \in \{1, 2, \ldots, m\}$ tel que $b_i \geq \sum_{j \in N_i^+} a_{ij}$

alors la contrainte i est redondante

(test réduction 2)

4 Réduire la taille des coefficients

si $\exists\ i \in \{1, 2, \ldots, m\}$ et $\exists\ j \in \{1, 2, \ldots, n\}$ tels que

$|a_{ij}| > \sum_{k \in N_i^+} a_{ik} - b_i$ alors

$a_{ij} \leftarrow \sum_{k \in N_i^+} a_{ik} - b_i$ si $a_{ij} > 0$

$a_{ij} \leftarrow b_i - \sum_{k \in N_i^+} a_{ik}$ si $a_{ij} < 0$

(la réduction de la taille des coefficients ne modifie pas l'ensemble des solutions bivalentes mais diminue par contre l'ensemble des solutions appartenant au cube unité - motivation des techniques de rotation de contraintes - : ceci permet de diminuer la borne supérieure $v(\bar{B})$ et par conséquent de rendre plus performantes les techniques de réduction de la taille du problème développées dans le prochain chapitre).

note : la démonstration de ces différents points est immédiate.

□

b) Algorithme NKR

L'algorithme NKR résout le programme linéaire associé au problème du knapsack :

$$(K) \quad \begin{cases} \max \ cx \\ \text{s.c. } ax \leq b \\ x \in V \end{cases}$$

avec les hypothèses : $\begin{cases} c \in \mathbb{R}_+^n \\ a \in \mathbb{R}_+^n \\ b \in \mathbb{R}_+^* \end{cases}$

Si on suppose que $a \in \mathbb{R}^n$ et $b \in \mathbb{R}$ il est immédiat que toutes les variables x_j correspondent à un $a_j \leq 0$ doivent être fixées à la valeur 1 puisque les c_j restent positifs.

En posant $N_- = \{j \mid a_j \leq 0\}$ et $N_+ = \{j \mid a_j > 0\}$, le problème (K) est équivalent au problème suivant :

$$(K') \quad \begin{cases} e\, c_{N_-} + \max \ \bar{c}\, x \\ \text{s.c. } \bar{a}\, x \leq b - \sum_{j \in N_-} a_j \\ x \in \bar{V} = \{x \in \mathbb{R}^{N_+} \mid x_j = 0 \text{ ou } 1, \ \forall\, j \in N_+\} \end{cases}$$

où $e_j = 1$, $\forall\, j \in N_-$; $\bar{c} = c_{N_+}$; $\bar{a} = a_{N_+} > 0$.

Le problème initial de taille n est donc remplacé par un problème de taille $|N_+|$ de type knapsack dont la relaxation en continu est résoluble par l'algorithme NKR.

c) Procédure énumérative (L.I.F.O.)

L'algorithme simple d'énumération implicite de type L.I.F.O. (last in first out) développé dans ([13], Annexe 2) s'adapte sans beaucoup de difficultés au cas de signes quelconques, les principales modifications portant sur la partie réduction (se référer au a) ci-dessus).

d) Heuristique

Comme dans le cas non négatif, on peut déterminer un minorant de $v(B^k)$, lorsque la taille du dernier sous-problème (B^k) est supérieure au seuil 10, en classant dans l'ordre décroissant les c_j pour fixer en priorité à 1 les variables x_j de plus grands c_j. Une autre solution, surement plus performante en précision mais aussi beaucoup plus coûteuse en temps calcul, consiste à utiliser en tant que sous-programme de résolution une procédure semblable à l'algorithme de Balas-Martin.

Procédure

1 *Utiliser la méthode du simplex à variables bornées pour déterminer la solution optimale* $\bar{x}$ *du programme linéaire associé à* (B^k) :

$(\overline{B^k})$ max cx s.c. $Ax + y = b$, $0 \leq x \leq e$, $y \geq 0$

(sous l'hypothèse $F(B^k) \neq \emptyset$)

2 *Un procédé dit "de complémentation" consistant à explorer des voisinages de* $[\bar{x}]$ *génère une solution réalisable* $\underline{x}$ *de* (B^k).

(voir Hillier, [33]).

REDUCTION DE LA TAILLE D'UN PROBLEME

CHAPITRE 4

INTRODUCTION

Le but de ce chapitre est de reprendre dans l'optique de la contraction de contraintes les travaux de Hansen-Plateau (1976, [54]) sur la réduction (à la fois en nombre de variables et en nombre de contraintes) d'un problème linéaire en variables 0-1. La conjonction de tests simples, appliqués à différentes relaxations (de type knapsack) du problème initial, s'est averée particulièrement efficace sur une cinquantaine de problèmes tests considérés (références du chapitre 3). Ces tests, liés avec les algorithmes de résolution de duaux développés au chapitre 2 et avec les méthodes heuristiques proposées au chapitre 3, ont permis la construction d'un code automatique de réduction. Pour une partie des problèmes tests, celui-ci a généré des problèmes dits "difficiles", c'est à dire des problèmes pour lesquels on peut s'attendre à ce que toute tentative de réduction de la taille soit vaine et que toute résolution, par n'importe quelle procédure énumérative existante, soit chère en temps calcul (voir Annexe 2). Enfin le code a résolu le restant des problèmes tests, au sens où il les a réduit à des problèmes dont le nombre de variables n'excède pas 10, donc résolubles en un temps négligeable par n'importe quelle procédure énumérative (par exemple la procédure L.I.F.O., [11] Annexe 2).

Après un rappel des principes essentiels de l'algorithme de Hansen et Plateau, la première partie (§.4.1) développe l'apport de la notion de contraction de contraintes à travers l'utilisation des multiplicateurs générés par les algorithmes étudiés au chapitre 2.

L'énoncé des tests d'élimination définitive de variables et de contraintes (§ 4.2) précède la présentation de notre nouvel algorithme de réduction (§ 4.3) ainsi que les nombreuses expériences numériques réalisées sur le calculateur CII HB IRIS 80 (§ 4.4).

4.1 - Motivation et caractéristique essentielles des algorithmes de réduction

4.1.1 - *Algorithme de Hansen-Plateau* ([9], chapitre 3)

Les tests d'élimination définitive de variables et de contraintes présentés au paragraphe 4.2 nécessitent la résolution par l'algorithme NKR de la relaxation en continu d'un problème de type knapsack . Les auteurs ont choisi les problèmes du knapsack extraits de (B), notés (B_p^o) et (B_p^k) comme suit pour k, p $\in \{1, 2, \ldots, m\}$ donnés :

$$(B) \quad \begin{cases} \max \ cx \\ \text{s.c. } Ax \leq b \\ x \in V \end{cases} \qquad \text{où} \quad \begin{cases} c \in \mathbb{N}^n \\ b \in \mathbb{N}^m \\ A \in \mathbb{N}^{n \times m} \end{cases}$$

$$(B_p^o) \quad \begin{cases} \max \ cx \\ \text{s.c. } A_p x \leq b_p \\ x \in V \end{cases} \quad ; \quad (B_p^k) \quad \begin{cases} \max \ A_k x \\ \text{s.c. } A_p x \leq b_p \\ x \in V \end{cases}$$

Le *principe* de l'algorithme est le suivant

Phase 1 : élimination évidente de contraintes et de variables

Phase 2 : "préparation" de l'élimination de variables : calcul des valeurs de chacun des knapsacks $(\bar{B}_k^o)$, k = 1, 2, ..., m. *Classement des contraintes de telle sorte que :* $v(\bar{B}_1^o) \leq v(\bar{B}_2^o) \leq \ldots \leq v(\bar{B}_m^o)$.

Phase 3 : élimination de variables par l'intermédiaire d'algorithmes de réduction appliqués tour à tour sur (B_1^o), (B_2^o), ..., (B_m^o).

Phase 4 : élimination de contraintes : tests appliqués sur (B_1^k), k = 2, 3,...m

Phase 5 : élimination de variables en utilisant :

- *les contraintes composites*
- *le programme linéaire associé au problème (B) réduit.*

A travers ce travail, l'objectif majeur des auteurs était de mettre en évidence l'efficacité des tests simples utilisés, et d'ailleurs l'algorithme a donné des résultats très satisfaisants sur les problèmes de Petersen. Par contre leur souci n'a pas été de mettre au point un outil opérationnel et de ce fait le travail est resté purement expérimental. En conséquence des choix ne sont pas conformes à une possible mise en exploitation et certains critères n'ont pas été pris en compte. Plus précisément, les tests d'élimination de variables en phases 3 et 5 nécessitant la connaissance d'un minorant de la valeur du problème ; les auteurs ont choisi comme minorant la valeur optimale du problème. D'autre part, en phases 3 et 4, les tests d'élimination sont conduits systématiquement sur les 2m-1 problèmes du knapsack (B_k^o), k = 1, 2, ..., m et (B_1^k), k = 2, 3, ..., m ; il s'ensuit que les temps de calcul sont proportionnels au nombre de contraintes m, ce qui peut s'avérer un facteur défavorable pour les problèmes ayant un grand nombre de contraintes (d'ailleurs les temps de calcul ne sont pas connus). Enfin en phase 5, ce sont des contraintes composites construites empiriquement en fonction des renseignements acquis qui ont permis dans certains cas l'élimination d'un nombre respectable de variables ; à titre d'exemple pour les deux derniers problèmes de Petersen de 39 et 50 variables, respectivement 7 et 11 variables ont pu être fixées définitivement à 0 ou à 1.

4.1.2 - Caractéristiques du nouvel algorithme

En conséquence des remarques du paragraphe 4.1.1, la construction d'un code de réduction opérationnel implique

- le calcul d'un minorant de la valeur du problème,

obtenu par la mise en oeuvre des heuristiques AGNES 1 et 2 (chapitre 3)

- la prise en compte du temps de calcul

ceci s'effectuant par la conjonction des quatre principes suivants :

(i) l'utilisation de la notion de pénalité (borne inférieure de la diminution apportée à la valeur d'une relaxation d'un problème en variables bivalentes de maximisation lorsqu'une contrainte supplémentaire est imposée : dans ce cas précis, du type fixation d'une variable x_k à la valeur ε, $\varepsilon \in \{0, 1\}$, $k \in \{1, 2, \ldots, n\}$).

Dans le même esprit que l'algorithme de Hansen-Plateau, l'introduction des pénalités tend à minimiser le nombre d'appels de l'algorithme NKR dans les tests d'élimination de variables (voir tests V2 et V4, Théorème 4.3, § 4.2.1.1).

(ii) l'utilisation systématique de contraintes composites en lieu et place des m contraintes du problème initial ; ajoutons que la "meilleure" contrainte au sens de la réduction ne sera plus celle correspondant à la plus petite valeur $\overline{v(B_k^o)}$, $k = 1, 2, \ldots, m$, mais celle attachée à la plus grande composante w_i, $i = 1, 2, \ldots, m$ d'un multiplicateur optimal w associé au dual du problème initial.

Soit $w \in \mathbb{R}_+^m$ un multiplicateur, en notant ℓ l'indice tel que,

$$w_\ell = \max_{i=1,\ldots,m} w_i$$

nous considérons en priorité les problèmes du knapsack suivants :

$$(B^o(w)) \quad \begin{cases} \max \ cx \\ \text{s.c. } wAx \le wb \\ x \in V \end{cases} \qquad (B^k(w)) \quad \begin{cases} \max \ A_k x \\ \text{s.c. } wAx \le wb \\ x \in V \end{cases}$$

pour $k = 1, 2, \ldots, m$

$$(B_\ell^o) \quad \begin{cases} \max \ cx \\ \text{s.c. } A_\ell x \le b_\ell \\ x \in V \end{cases} \qquad (B_\ell^k) \quad \begin{cases} \max \ A_k x \\ \text{s.c. } A_\ell x \le b_\ell \\ x \in V \end{cases}$$

pour $k = 1, 2, \ldots, \ell-1, \ell+1, \ldots, m$.

(iii) la détection des contraintes ayant le plus de chances d'être éliminées par les tests d'élimination de contraintes, ce qui dégagera des priorités dans le processus de réduction.

On a constaté numériquement que la condition $w_i \simeq 0$ avec $i \in \{1, 2, \ldots, m\}$ était une condition presque suffisante pour pouvoir éliminer la contrainte i. Ceci peut s'expliquer à partir du résultat suivant.

Théorème 4.1

Soit w^* *le multiplicateur optimal du dual lagrangien associé au problème* (B). *Soit le sous-ensemble* $I \subset \{1, 2, \ldots, m\}$, *supposé non vide, défini par :*

$I = \{i \mid w_i^* = 0\}$.

Considérons les deux relaxations suivantes du problème (B) :

(B_1) max cx s.c. $w_{\bar{I}}^* A_{\bar{I}} x \leq w_{\bar{I}}^* b_{\bar{I}}$; $A_I x \leq b_I$; $x \in V$

(B_2) max cx s.c. $w_{\bar{I}}^* A_{\bar{I}} x \leq w_{\bar{I}}^* b_{\bar{I}}$; $x \in V$

Alors $v(\bar{B}_1) = v(\bar{B}_2)$, *c'est à dire qu'à l'optimum de* $(\bar{B}_1)$ *et* $(\bar{B}_2)$ *les contraintes d'indice* $i \in I$ *sont redondantes.*

Démonstration

D'une part $F(\bar{B}_1) \subseteq F(\bar{B}_2)$ donc $v(\bar{B}_1) \leq v(\bar{B}_2)$

D'autre part soit $\bar{x} \in \Omega(\bar{B}_2)$ une solution optimale de $(\bar{B}_2)$; (B_1) étant une relaxation de (B), nous avons $v(\bar{B}) \leq v(\bar{B}_1) \leq v(\bar{B}_2) = c\,\bar{x}$ (0)

D'après le théorème fondamental de la dualité

$$c\,\bar{x} = \min_{\lambda \geq 0} \max_{x \in [V]} cx + \lambda(w_{\bar{I}}^* b_{\bar{I}} - w_{\bar{I}}^* A_{\bar{I}} x)$$

$$= \min_{\lambda \geq 0} \max_{x \in [V]} cx + \lambda w^* (b - Ax)$$

$$\leq \max_{x \in [V]} cx + w^* (b - Ax)$$

$$= v(\bar{B}) \text{ (théorème fondamental de la dualité).}$$

Associé à (0), il s'ensuit que $v(\bar{B}_1) = v(\bar{B}_2)$.

□

(B_2) n'est autre que la relaxation composite de (B) associée au multiplicateur w^*, problème que l'on note encore $(B^o(w^*))$.

Le théorème 4.1 signifie en fait que, pour tout $i \in I$,

$$(1) \qquad \boxed{w_I^* A_I x \leq w_I^* b \Rightarrow A_i x \leq b_i \quad \forall\, x \in \overline{\Omega(B^o(w^*))}}$$

Or le test d'élimination de contrainte utilisé peut s'énoncer comme suit : "la contrainte i est éliminée si $\underline{v}(\overline{B^i(w^*)}) \geq b_i$" (voir paragraphe 4.2) où, ce qui est équivalent, si la condition suivante est vraie :

$$(2) \qquad \boxed{w_I^* A_I x \leq w_I^* b \Rightarrow A_i x \leq b_i \quad \forall\, x \in [V]}$$

La condition (2) est plus forte que la condition (1) car elle ne fait pas intervenir la fonction économique c x, mise à contribution par l'ensemble $\overline{\Omega(B^o(w^*))}$.

Toutefois, l'utilisation de la condition (2) dans les expériences numériques s'est avérée très efficace dans une très grande majorité de cas.

(iv) la mise en place de tests de décision permettant de ne pas prendre en compte des tests d'élimination trop sophistiqués (donc plus chers en temps calcul) à partir du moment où ils ont de fortes chances de ne pas être efficaces. Ces tests de décision sont décrits au paragraphe 4.2 à la suite des tests d'élimination.

4.2 - Tests de réduction et de décision

Tous les tests détaillés ci-dessous sont énoncés sous l'hypothèse supplémentaire $c \in \mathbb{N}^n$ et $a_{ij} \in \mathbb{N}$, $\forall\, i\ \forall\, j$; lorsque celle-ci n'est pas vérifiée, il suffit de supprimer la notion de partie entière dans chaque expression concernée.

4.2.1 - Tests de réduction

4.2.1.1 - Elimination de variables

Considérons les problèmes du knapsack suivants, associés à un multiplicateur $w \in \mathbb{R}_+^m$ donné :

$$(BL(w)) \quad \begin{cases} \max \; cx + w(b-Ax) \\ \text{s.c.} \; x \in V \end{cases}$$

$$(B^o(w)) \quad \begin{cases} \max \; cx \\ \text{s.c.} \; wAx \leq wb \\ x \in V \end{cases} \qquad (B_i^o) \quad \begin{cases} \max \; cx \\ \text{s.c.} \; A_i x \leq b_i \\ x \in V \end{cases} \quad i \in \{1,2,\ldots,m\}$$

Théorème 4.2

Etant donné $\varepsilon \in \{0, 1\}$,

s'il existe $j \in \{1, 2, \ldots, n\}$ *tel que*

(V1) $\quad \lfloor v(BL(w)) - |d_j^\varepsilon| \rfloor \leq \underline{v}(B) \qquad (*)$

avec $c_j - w A^j = d_j^o$ *(resp.* d_j^1*) si* $c_j - wA^j$ *est positif (resp. négatif) alors la variable d'indice* j *doit être fixée à la valeur* $1-\varepsilon$.

Démonstration

(BL(w)) étant une relaxation du problème (B) à valeur entière v(B), nous avons pour tout $j \in \{1, 2, \ldots, n\}$

$$v(B) \leq \max(\lfloor (v(BL(w) \mid x_j = \varepsilon) \rfloor, \lfloor (v(BL(w) \mid x_j = 1-\varepsilon) \rfloor). \quad (1)$$

En posant

$$J^+(w) = \{j \mid c_j - wA^j = d_j^o > 0\} \text{ et } J^-(w) = \{j \mid c_j - wA^j = d_j^1 < 0\}$$

on a :

$$v(BL(w)) = w\,b + \sum_{j \in J^+(w)} c_j - w\,A^j \quad \text{d'où}$$

$v(BL(w) \mid x_j = \varepsilon) = v(BL(w))$ *si* $(j \in J^+(w)$ et $\varepsilon = 1)$ ou $(j \in J^-(w)$ et $\varepsilon = 0)$

$= v(BL(w)) - |c_j - wA^j|$ si $(j \in J^+(w)$ et $\varepsilon = 0)$ ou $(j \in J^-(w)$ et $\varepsilon = 1)$

et par conséquent, si $\exists\ j \in \{1, 2, \ldots, n\}$ et $\varepsilon \in \{0, 1\}$ tels que

$\lfloor v(BL(w)) - |d_j^\varepsilon| \rfloor \leq \underline{v}(B) \leq v(B)$, alors d'après (1), x_j doit être fixée à la valeur complémentaire $1-\varepsilon$.

□

Les tests énoncés ci-dessous peuvent être conduits aussi bien sur $(B^o(w))$ que sur (B^o_ℓ) où ℓ est tel que : $w_\ell = \max_{i=1,\ldots,m} w_i$. Pour éviter les répétitions, les énoncés seront donnés pour $(B^o(w))$.

Théorème 4.3

Etant donné $\varepsilon \in \{0, 1\}$

(V2) *s'il existe* $j \in \{1, 2, \ldots, n\}$ *tel que*
$\lfloor \overline{v(B^o(w))} - |d_j^\varepsilon| \rfloor \leq \underline{v}(B)$
avec $c_j - \lambda w A^j = d_j^o$ *(resp.* d_j^1*) si* $c_j - \lambda w A^j$ *est positif (resp. négatif)*
et $\lambda = c_{i^*}/w\ A^{i^*,i^*}$ *étant l'indice de la variable de base optimale de* $\overline{(B^o(w))}$.

ou

(V4) *si* $\lfloor \overline{v(B^o(w) \mid x_i^* = \eta)} \rfloor > \underline{v}(B)$ *pour* $\eta \in \{0, 1\}$ *et*
s'il existe $j \in \{1, 2, \ldots, n\}$ *tel que*
$\lfloor \max \{ \overline{(v(B^o(w) \mid x_i^* = 0)} - |d^\varepsilon_{o,j}|), \overline{(v(B^o(w) \mid x_i^* = 1)} - |d^\varepsilon_{1,j}| \} \rfloor \leq \underline{v}(B)$
avec $c_j - \lambda\ w A^j = d^o_{\eta,j}$ *(resp.* $d^1_{\eta,j}$*) si* $c_j - \lambda_\eta w A^j$ *est positif*
(resp. négatif) et $\lambda_\eta = c_{i^*(\eta)}/w\ A^{i^*(\eta)}$, $i^*(\eta)$ *étant l'indice de la variable de base optimale de* $\overline{(B^o(w) \mid x_i^* = \eta)}$ *avec* $\eta \in \{0, 1\}$

alors la variable d'indice doit être fixée à la valeur $1-\varepsilon$.

Démonstration

La démonstration est semblable à celle du théorème 4.2 puisque dans chaque cas nous avons les majorations suivantes :

(V2) : $v(B) \leq \max \{ \lfloor \overline{v(B^o(w) \mid x_j = \varepsilon)} \rfloor , \lfloor \overline{v(B^o(w) \mid x_j = 1-\varepsilon)} \rfloor \}$

(V4) : $v(B) \leq \max_{\varepsilon \in \{0,1\}} \{ \lfloor \overline{v(B^o(w) \mid x_j = \varepsilon, x_i^* = 0)} \rfloor , \lfloor \overline{v(B^o(w) \mid x_j = \varepsilon, x_i^* = 1)} \rfloor \}$

(avec x_i^* variable de base optimale de $\overline{(B^o(w))}$

□

Théorème 4.4

Etant donnés $\varepsilon \in \{0, 1\}$ *et* $i \in \{1, 2, \ldots, m\}$

(V5) *s'il existe* $j \in \{1, 2, \ldots, n\}$ *tel que* $\lfloor v(B_i^o \mid x_j = \varepsilon) \rfloor \leq \underline{v}(B)$
alors la variable d'indice j *doit être fixée à la valeur* $1-\varepsilon$.

Démonstration

Comme précédemment, ceci découle du fait que

$v(B) \leq \max \{ \lfloor v(B_i^o) \mid x_j = \varepsilon) \rfloor , \lfloor v(B_i^o \mid x_j = 1-\varepsilon) \rfloor \}$

□

Remarque

Le test V5 sera appelé test V3 lorsque $i = \ell$ et (B_ℓ^o) est remplacé par $(B^o(w))$. Cette notation permet de mieux comprendre l'enchaînement des tests car le test V3 est en fait la première partie du test V4.

Remarques

(i) On peut faire une estimation comparative du coût de ces différents tests en se référant au nombre d'appels de l'algorithme NKR nécessaires pour résoudre les relaxations en continu des différents problèmes du knapsck envisagés :

V1 : 0 ; V2 : 1 ; V3, V4 : 2 ; V5 : 2

(ii) le test (V5) n'est utilisé au plus m fois que sur les variables x_j correspondant au plus grand a_{ij} de chaque contrainte i, $i \in \{1, 2, \ldots, m\}$.

(iii) ces 5 tests sont parties intégrantes d'un module de réduction de la taille de (B). Il faut leur ajouter deux tests d'élimination triviale, déjà évoqués lors de la construction des heuristiques (voir hypothèses (H) chapitre 3).

(R1) si il existe $j \in \{1, 2, \ldots, n\}$ et $i \in \{1, 2, \ldots, m\}$ tels que $a_{ij} > b_i$ alors la variable x_j est fixée à 0.

(R3) si il existe $j \in \{1, \ldots, n\}$ tel que $A^j = 0$ alors la variable x_j est fixée à 1.

4.2.1.2 - Elimination de contraintes

Considérons les problèmes du knapsack suivants, associés à un multiplicateur $w \in \mathbb{R}_+^m$ donné :

$$(B^k(w)) \quad \begin{cases} \max \; A_k x \\ \text{s.c. } wAx \leq wb \\ \qquad x \in V \end{cases} \qquad \text{pour } k = 1, 2, \ldots, m$$

$$(B_\ell^k) \quad \begin{cases} \max \; A_k x \\ \text{s.c. } A_\ell x \leq b_\ell \\ \qquad x \in V \end{cases} \qquad \text{pour } k = 1, 2, \ldots, \ell-1, \ell+1, \ldots, m \text{ et } \ell \text{ tel que : } w_\ell = \max_{i=1,2,\ldots,m} w_i$$

Les tests énoncés ci-dessous sont conduits à la fois sur $(B^k(w))$ et (B_ℓ^k). Pour éviter une répétition inutile, les énoncés seront donnés pour (B_ℓ^k).

Théorème 4.5

La contrainte $A_k x \leq b_k$, $k \in \{1, 2, \ldots, m\}$ *peut être éliminée du problème* (B) *sous l'une des hypothèses suivantes :*

(C1) $\lfloor v(\bar{B}^k) \rfloor \leq b_k$.

(C2) $q = \lfloor \max \{ v(\overline{B^k \mid x_{i(\ell)} = 0}),\ v(\overline{B^k \mid x_{i(\ell)} = 1}) \} \rfloor \leq b_k$

avec $i(\ell)$ *indice de la variable de base optimale de* $(\bar{B}^k_\ell)$.

si $\exists\, \varepsilon \in \{0, 1\}$; $q_\ell > b_k$ *et* $\lfloor v(\overline{B^k_\ell \mid x_{i(\ell)} = 1-\varepsilon}) \rfloor \leq b_k$:

(C3) $\lfloor \max \{ v(\bar{B}^k_\ell | x_{i(\ell)}=\varepsilon,\ x_{i(\ell,\varepsilon)}=0),\ v(\bar{B}^k_\ell | x_{i(\ell)}=\varepsilon,\ x_{i(\ell,\varepsilon)}=1) \} \rfloor \leq b_k$

avec $i(\ell,\varepsilon)$ *indice de la variable de base optimale de* $(\overline{B^k_\ell | x_{i(\ell)}=\varepsilon})$

Démonstration

$\forall\, x \in V : A_\ell x \leq b_\ell \Rightarrow A_k x \leq v(B^k_\ell) \leq \lfloor v(\bar{B}^k_\ell) \rfloor \leq b_k$, ce qui montre (C1).
Les tests (C2) et (C3) aboutissent par séparation à des meilleurs majorants de $v(B^k_\ell)$ inférieurs à b_k.

□

Remarques

(i) le coût en nombre d'appels de l'algorithme NKR est respectivement

(C1) : 1 ; (C2) : 2 ; (C3) : 2

(ii) A ces tests, il faut ajouter un test d'élimination triviale, composant avec les tests (R1) et (R3), un module de réduction triviale.

(R2) si il existe $i \in \{1, 2, \ldots, m\}$ tel que $\sum_{j=1}^{n} a_{ij} \leq b_i$

alors la contrainte i est éliminée.

4.2.2 - *Test de décision*

Les tests (V3, V3) et (C2, C3) nécessitent la résolution en continu de deux problèmes du knapsack, obtenus en fixant tour à tour à 0 et 1 la variable x_i^* de base optimale d'un problème (K). Ce problème (K) est respectivement, pour (V3, V4) le problème $(B^o(w))$ ou (B^o_ℓ) (voir théorème 4.3), pour (C2, C3) le problème $(B^k(w))$ ou (B^k_ℓ) (voir théorème 4.5)

Ces tests ont pour but de raffiner le majorant v(K) de la valeur du problème (B) par séparation sur la variable de base x_i^*. Toutefois ce raffinement s'avère inutile dans beaucoup de cas et de plus coûte cher en temps calcul.

Des majorants respectifs de la valeur des problèmes $(K \mid x_i^* = 0)$ et $(K \mid x_i^* = 1)$ vont nous permettre de décider s'il faut, oui ou non, mettre en oeuvre les tests (V3, V4) et (C2, C3).

Ces majorants ont été donnés par Toth (1976) ou Fayard-Plateau ([9]). Rappelons que pour la résolution de $(\bar{K})$, l'algorithme NKR détermine un sous-ensemble $U \subset \{1, 2, \ldots, n\}$ et un élément i^* de $\{1, 2, \ldots, n\} \setminus U$ tels que :

$$\text{(i)} \quad \forall\, j \in U \quad c_j / a_j \geq c_{i^*}/a_{i^*} \geq c_{\ell}/a_{\ell} \quad \forall\, \ell \notin U \cup \{i^*\}$$

$$\text{(ii)} \quad \sum_{j \in U} a_j \leq b < \sum_{j \in U \cup \{i^*\}} a_j$$

ceci lorsque $(\bar{K})$ s'énonce sous la forme

$$(\bar{K}) \quad \max cx \text{ s.c. } ax \leq b,\ x \in [V].$$

Posons $L = \{1, 2, \ldots, n\} \setminus (U \cup \{i^*\})$ et définissons les indices $\bar{j}$ et $\underline{j}$ par :

$$\bar{j} : c_{\bar{j}}/a_{\bar{j}} = \max \{c_j/a_j \mid j \in L\} \qquad \underline{j} : c_{\underline{j}}/a_{\underline{j}} = \min \{c_j/a_j \mid j \in U\}$$

Théorème 4.6

Avec les notations ci-dessus, nous avons les majorations suivantes :

$$v(\overline{K \mid x_i^* = 0}) \leq \alpha_o = \sum_{j \in U} c_j + c_{\bar{j}}/a_{\bar{j}} \times (b - \sum_{j \in U} a_j)$$

$$v(\overline{K \mid x_i^* = 1}) \leq \alpha_1 = \sum_{j \in U} c_j + c_{i^*} - c_{\underline{j}}/a_{\underline{j}} \times (\sum_{j \in U \cup \{i^*\}} a_j - b)$$

Démonstration [9], p 90-91.

□

Il s'ensuit que :

$\beta = \lfloor \max \{ v(\overline{K} | x_i^* = 0), v(\overline{K} | x_i^* = 1) \} \rfloor \leq \alpha = \lfloor \max \{ \alpha_o, \alpha_1 \} \rfloor \leq \lfloor v(\bar{K}) \rfloor$.

En comparant $\lfloor v(\bar{K}) - \alpha \rfloor$ et $\lfloor v(\bar{K}) - \underline{v}(B) \rfloor$ (voir figure 4.1) on peut faire un pari sur l'intérêt de calculer β.

Ces tests de décision gérés directement par l'utilisateur lui permettent

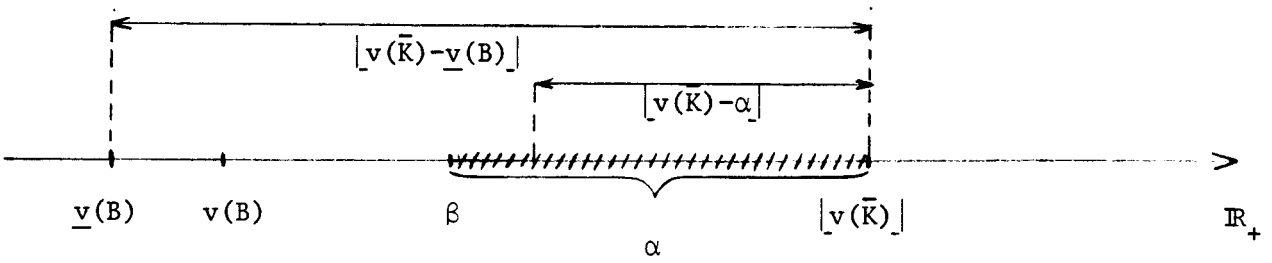


Figure 4.1

de faire l'équilibre entre les performances en temps calcul et celles en nombres de variables et contraintes éliminées (ce sont ces dernières qui seront privilégiées lors des expériences numériques) :

4.3 - Code automatique de réduction

4.3.1 - Organigramme général

Donnons le détail des différentes phases relatives à l'organigramme du code automatique de réduction (voir figure 4.2)

- Phase de "réduction triviale" : ce module met en jeu les tests R1, R2, R3 d'élimination triviale de variables et/ou de contraintes. Ces tests ont pour but de ramener si nécessaire le problème courant à un problème "bien posé", comme il a été défini par les hypothèses (H) (chapitre 3, paragraphe 3.1.1). Notons que ce module sera mis en oeuvre à chaque fois qu'il y aura eu réduction effective par action des phases de réduction 1 et 2.

- Préparation de la réduction : la mise en mémoire et la gestion des données suivantes liées aux structures du problème

- nombre d'éléments non nuls par colonne

- somme des éléments par contrainte

- numéro et valeur du plus grand élément par contrainte

évitent des calculs inutiles car répétitifs à chaque fois qu'il est fait appel au module de réduction triviale (Tests R1, R2, R3).

- Algorithmes de résolution des duaux lagrangiens et composites : ce sont les algorithmes décrits au chapitre 2, notons simplement que le critère d'arrêt est le nombre d'itérations, fixé a priori par l'utilisateur.

- Calcul d'un premier minorant et Essais d'amélioration du minorant : ces deux phases réalisent l'insertion des méthodes heuristiques AGNES 1 et 2 dans le code de réduction (chapitre 3, paragraphe 3.2.1)

- Phases de réduction 1 et 2 : la description détaillée de ces deux phases fait l'objet du paragraphe suivant.

Remarques

(i) Les modules de réduction lagrangienne et de réduction composite sont quasiment identiques, mis à part la technique de générations des multiplicateurs. Par conséquent il est possible de créer d'autres modules de réduction de dual (par exemple la méthode simplex, la méthode de sous-gradients avec dilatation de l'espace, etc ... voir chapitre 2).

(ii) Deux paramètres entiers p et q gèrent respectivement le nombre maximum d'appels des modules de réduction lagrangienne et composite

- si le nombre p d'appels du module de réduction lagrangienne est atteint (donc à chaque appel il y a eu réduction effective de la taille du problème) on force le passage au module de réduction composite.

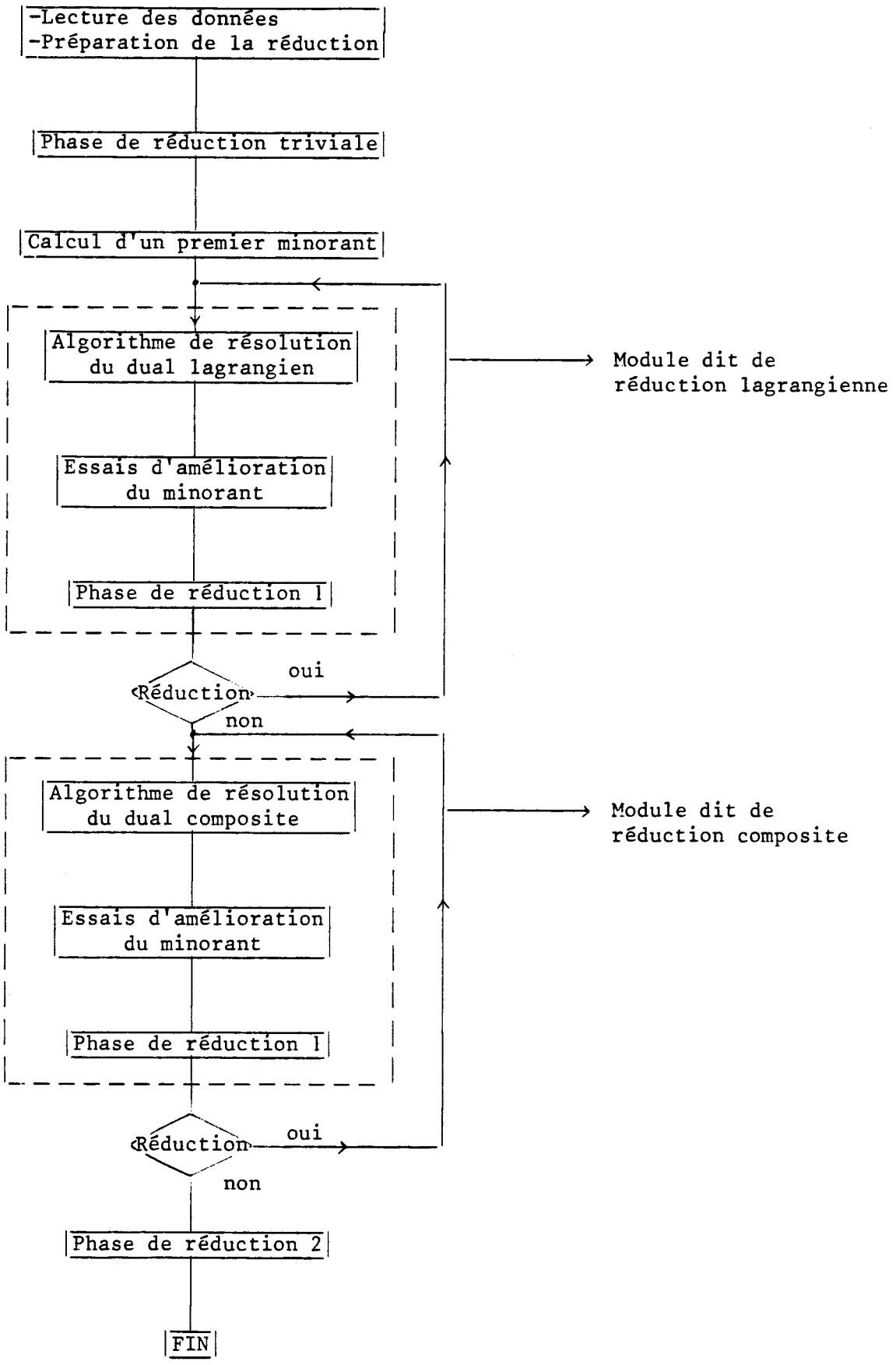


Figure 4.2

- si le nombre q d'appels du module de réduction composite est atteint, q est augmenté d'une unité tant qu'il y a réduction.

Lors des expériences numériques, nous avons choisi p = 2 et q = 1

(iii) un booléen gère la mise en oeuvre éventuelle de la phase de réduction 2, qui est à considérer comme une phase complémentaire de réduction, son rapport "coût en temps de calcul/efficacité" étant assez élevé.

4.3.2 - *Algorithmes détaillés de réduction*

Les algorithmes de réduction sont la traduction d'un enchaînement des tests d'élimination de variables et de contraintes exposés ci-dessus. Cet enchaînement a été mis au point après un grand nombre d'expériences numériques.

En posant

X_o (resp. X_1) l'ensemble d'indices des variables fixées à 0 (resp. 1)

E l'ensemble d'indices des contraintes éliminées

$X_2 = \{1, 2, \ldots, n\} \setminus (X_o \cup X_1)$ et $\bar{E} = \{1, 2, \ldots, m\} \setminus E$

Les différents tests seront appliqués, pour X_o, X_1, E donnés, au problème

$$\sum_{j \in X_1} c_j + \max \sum_{j \in X_2} c_j x_j$$

$$\text{s.c.} \sum_{j \in X_2} a_{ij} x_j \leq b_i - \sum_{j \in X_1} a_{ij} \qquad \forall\, i \in \bar{E}$$

$$x_j = 0 \text{ ou } 1 \qquad \forall\, j \in X_2$$

Algorithme de réduction 1

0 *Etant donnés* X_o, X_1, E ; $\underline{v}(B)$; $w \in \mathbb{R}_+^m$; μ *entier*

1 *Tous les tests ci-dessous sont appliqués à* $(B^o(w))$

1.1 *Faire* (V1) ; *si réduction* (*modification de* X_o, X_1)

alors faire tests R1, R2, R3 (*modif.* X_o, X_1, E)

1.2 *Faire* (V2) ; *si réduction* (*modif.* X_o, X_1)

alors faire tests R1, R2, R3 (*modif.* X_o, X_1, E)

1.3 *Test décision : si* $|v(\bar{B}^o(w)) - \underline{v}(B)| \geq \mu\ |v(\bar{B}^o(w)) - \alpha|$

alors aller en 2

1.4 *Faire* (V3) *et* (V4) ; *si réduction* (*modif* X_o, X_1)

alors faire tests R1, R2, R3 (*modif.* X_o, X_1, x_2)

2 *Tous les tests ci-dessous sont appliqués à* (B^o_ℓ), *avec* ℓ : $w_\ell = \max_{i \in \bar{E}} w_i$

2.1 *Faire* (V2) ; *si réduction* (*modif.* X_o, X_1)

alors faire tests R1, R2, R3 (*modif.* X_o, X_1, E)

2.2 *Test décision : si* $|v(B^o) - \underline{v}(B)| \geq \mu\ |v(\bar{B}^o) - \alpha|$

alors aller en 3

2.3 *Faire* (V3) *et* (V4) ; *si réduction* (*modif.* X_o, X_1)

alors faire tests R1, R2, R3 (*modif.* X_o, X_1, E)

3 *Tous les tests ci-dessous sont appliqués à* (B^k_ℓ), *avec* ℓ : $w_\ell = \max_{i \in \bar{E}} w_i$
et pour chaque $k \in \bar{E}$: $w_k \simeq 0$

3.1 *Faire* (C1)

3.2 *Test décision : si* $|v(\bar{B}^k) - b_k| \leq \mu |v(\bar{B}^k) - \alpha|$

alors aller en 3.5

3.3 *Faire* (C2)

3.4 *Si condition test* (C3) *vraie alors faire* (C3) *sinon aller en 3.5*

3.5 *si réduction* (*modif* E) *alors faire test* R3 (*modif* X_1)

4 *Tous les tests ci-dessous sont appliqués à* $(B^k(w))$ *pour chaque* $k \in \bar{E}$: $w_k \simeq 0$

4.1 *Faire* (C1)

4.2 *Test décision : si* $|v(\bar{B}^k(w)) - b_k| \geq \mu\ |v(\bar{B}^k(w)) - \alpha|$

alors aller en 4.5

4.3 Faire (C2)

4.4 Si condition test (C3) vraie alors faire (C3) sinon aller en 4.5

4.5 si réduction (modif E) alors faire test R3 (modif X_1).

5 Appel algorithme de réduction 2 : si oui aller en 6 sinon stop

Algorithme de réduction 2

6 Tous les tests ci-dessous sont appliqués à (B_k^o) pour chaque $k \in \bar{E}$

6.1 Faire (V5) ; si réduction (modif X_o, X_1)

alors aller en 6.2 sinon stop

6.2 Faire tests R1, R2, R3 (modif. X_o, X_1, E)

7 Tous les tests ci-dessous sont appliqués à (B_ℓ^k), avec ℓ : $w_\ell = \max_{i \in \bar{E}} w_i$ et pour chaque $k \in \bar{E} - \ell$

7.1 Faire (C1)

7.2 Test décision : si $|v(\bar{B}_\ell^k) - b_k| \geq \mu |v(\bar{B}^k) - \alpha|$

alors aller en 7.5

7.3 Faire (C2)

7.4 si condition test (C3) vraie alors faire (C3)

7.5 si réduction (modif. E) alors faire test R3 (modif. X_1)

8 Stop (branchement vers une méthode de résolution)

Remarques

(i) si $|\bar{E}| = 1$ alors l'algorithme de réduction est terminé : le dernier problème est résolu par l'algorithme FPK 79 ([10]).

(ii) si $|\bar{E}| = 0$ ou si $|X_2| = 0$ alors le problème est en fait résolu car totalement réduit.

(iii) lors du dernier appel de l'algorithme de réduction 1 (rappelons que le nombre d'appel est fixé par l'utilisateur), les parties 3 et 4 sont modifiées comme suit : les tests d'élimination de contraintes sont envisagés

sur toutes les contraintes k, $k \in \bar{E}$, et non pas seulement sur les contraintes k, $k \in \bar{E}$ et telles que $w_k \simeq 0$.

(iv) α est la valeur déduite des résultats du théorème 4.6 (paragraphe 4.2.2).

4.4 - Expériences numériques

Le code de réduction (appelé FPR 83) a été implanté sur CII HB IRIS 80.

Le tableau 4.1 compare les performances, en nombre de variables et de contraintes éliminées, du code FPR 83 et de l'algorithme de Hansen-Plateau (appelé HPR 76) sur les sept problèmes de Petersen. Pour chacun d'eux, la première ligne donne les résultats relatifs à l'algorithme HPR 76 avec comme minorant la valeur optimale. La deuxième ligne indique les différences avec l'algorithme FPR 83. Dans le cas où le minorant initial n'est pas la valeur optimale requise par HPR 76 , une troisième ligne donne les modifications éventuelles.

Au vu de ce tableau, l'algorithme FPR 83 obtient des résultats pratiquement identiques à ceux de l'algorithme HPR 83, mais avec un coût en temps de clacul sans aucune mesure avec celui de HPR 83 (à cause des choix explicités au paragraphe 4.1.2).

Le tableau 4.2 permet d'apprécier l'apport d'une phase de réduction avant toute mise en oeuvre d'un algorithme de résolution exacte. (Nous avons utilisé la méthode de séparation et d'évaluation pour les problèmes en variables 0-1 à plusieurs contraintes en inégalité, proposée par Wei-Shih en 1980, [61]). Les temps de calcul sont donnés en dixièmes de seconde.

Si le fait de réduire ne s'avère pas profitable pour les problèmes de très petite taille, l'utilisation du code de réduction FPR 83 commence à devenir bénéfique à partir du problème P5 de taille 10×28. Ceci ne fait

que corroborer le résultat bien connu que le temps de calcul d'une méthode de résolution par séparation et évaluation dépend crucialement du nombre n de variables du problème alors que l'on peut s'attendre pour le code FPR 83 à une complexité expérimentale linéaire en moyenne.

Problèmes	Taille initiale	Wei Shih	FPR 83	Taille réduite	Wei Shih	Temps total
P1	10 × 6	2	12	0 × 0	-	12
P2	10 × 10	5	23	1 × 3	0	23
P3	10 × 15	19	48	5 × 8	0	48
P4	10 × 20	18	57	2 × 10	1	58
P5	10 × 28	92	54	2 × 9	1	55
P6	5 × 39	> 249	75	4 × 28	175	250
P7	5 × 50	> 249	77	4 × 38	211	288

Tableau 4.2

Le tableau 4.3 indique en colonne, pour les 7 problèmes de Petersen et six problèmes de Weingartner, le nombre de variables et de contraintes éliminées respectivement par les tests v_i, i = 1, ..., 5, R1, R3 et les tests c_i, i = 1, 2, 3, R2.

Le tableau 4.4 fournit pour 10 problèmes réels les numéros des variables et des contraintes éliminées, ainsi que les temps de calcul en dixièmes de seconde.

Le tableau 4.5 donne pour 30 problèmes tirés au hasard la taille des problèmes réduits et les temps de calcul respectivement pour leur réduction par le code FPR 83 et pour leur résolution par la procédure énumérative de Wei Shih.

Problèmes	Taille initiale	Code	Minorant	Numéro des variables éliminées (<0, var.fixée à 0 ; >0 var.fixée à 1)	Numéro des contraintes éliminées	Taille réduite
P1	10 × 6	HPR76	3800	-1,2,3,-4,-5,6	1,2,3,4,5,6,7,8,9,10	0 × 0
		FPR83	3800	Idem	Idem	0 × 0
P2	10 × 10	HPR76	87061	-1,-3,4,-7,8,-9	1,3,4,5,6,7,8,9,10	1 × 4
		FPR83	87061	+ [10]	Idem	1 × 3
P3	10 × 15	HPR76	4015	2,6,10,-11,-13,14,15	1,2,3,4,5,6,9	3 × 8
		FPR83	4015	Idem	- [2,9]	5 × 8
P4	10 × 20	HPR76	6120	1,-4,-5,-6,-7,-8,-10,-11,-12,-13, 14,15,17,20	1,3,4,5,6,8,9,10	2 × 6
		FPR83	6120	- [-6,-10]	échange 7 et 10	2 × 8
P5	10 × 28	HPR76	12400	-4,-5,-7,-8,-11,-12,-13,14,15,17,20, 21,22,23,25,26,27,28	1,3,4,5,6,7,8,9	2 × 10
		FPR83	12400	+ [10]	Idem	2 × 9
P6	5 × 39	HPR76	10618	-3,4,-5,6,11,15,16,17,27,29,32	4	4 × 28
		FPR83	10552*	Idem	Idem	4 × 28
			10618	+ [8]	Idem	4 × 27
P7	5 × 50	HPR76	16537	-3,-5,6,11,15,16,17,32,40,41,42, 43,-46,47,48	4	4 × 35
		FPR83	16470**	- [-3,11,-46]	Idem	4 × 38
			16537	échange -2 et 11	Idem	4 × 35

Tableau 4.1

* la valeur optimale 10618 est atteinte au cours du processus de réduction
** le minorant 16470 reste inchangé au cours du processus de réduction

Problème Taille	Elimination variables									Elimination contraintes						
	V1/V2		V3		V4		V5	R1	R3	C1		C2		C3		R2
	MC	CC	MC	CC	MC	CC				MC	CC	MC	CC	MC	CC	
P1 6 × 10		2	1		1			2								10
P2 10 × 10		2		1		1		3		3		1				5
P3 10 × 15	2	4						1		4	1					
P4 10 × 20		12								3	3				1	1
P5 10 × 18		18					1			5	1					2
P6 5 × 39		11								1						
P7 5 × 50		12								1						
P6* 5 × 39		12								1						
P7* 5 × 50		14					1			1						
W1 2 × 28	1	16		1			1	4	1							1
W2 2 × 28		18						4	1			1				
W3 2 × 28		9														
W4 2 × 28		20						3								1
W5 2 × 28		21						4	3							2
W6 2 × 28		19					1	1								

Tableau 4.3

MC : utilisation de la "meilleure" contrainte (étape 2, 3 ou 7)

CC : utilisation d'une contrainte composite (étape 1 ou 4)

(*) : la réduction est menée en prenant comme minorant $\underline{v}(B)$ la valeur optimale $v(B)$.

notes

(i) Les performances des tests V1 et V2 sont données conjointement, car ces tests sont interchangeables et leurs performances sont directement fonction de leur ordre d'utilisation

(ii) Pour les tests V1/V2, V3/V4 la contrainte composite est utilisée avant la meilleure contrainte. Par contre c'est l'inverse pour les tests C1, C2, C3.

Problèmes	Taille initiale	Numéro des variables éliminées (<0, var.fixée à 0 ; >0, var.fixée à 1)	Numéro des contraintes éliminées	Taille réduite	Temps (dixième de sec.)
W1	2 × 28	-1,-2,3,-4,5—8,-9,10,-11,12,14,-16,19,-20,21,-22,23,24,-25 26,-27,-28	1	1 × 4	19
W2	2 × 28	-1,-2,3,-4,5,-6,7,8,-9,10,-12,14,-16,19,-20,21,-22,23,24, -25— -28	1	1 × 5	15
W3	2 × 28	-1,-4,-6,8,-9,14,21,-25,-28		2 × 19	31
W4	2 × 28	-1,3,-4,5,-6,7,8,-9,10,-11,12,-13,14,-16,19,-21,22—24, -25,26,27,-28	1	1 × 5	17
W5	2 × 28	Toutes les variables sont éliminées	1, 2	0 × 0	15
W6	2 × 28	-1,-2,3,-4,5,8,-9,10,-12,14,-15— -18,-20,21,-22,23,-25 -26,-28		2 × 7	20
W7	2 × 105	1—10,14—78,-79,81—83,85,86,-87,89—91,-98— -105		2 × 12	61
W8	2 × 105	1,2,-3,4,-7,-10— -12,14—17,19,21,22,24—26,-27,31,-33,34, 35,-39,-43,-47,-49,-51,-53— -59,-61— -63,-65— -92,-94 — -96, -98— -100,-102— -105		2 × 29	76
ST1	30 × 60	5,8,-12,14,16,25,-26,-32,37,-38,-40,46,47,-50,-51,53,-55, 56,-58,59.		30 × 40	493
ST2	30 × 60	5,8,14,16—19,-24,25,27,30,-32,37,-38,46,47,-50,-51,53,-55, 56,59,-60		30 × 37	484

Tableau 4.4

Problèmes Wei Shih	Taille initiale	Taille réduite	FPR 83	Algorithme Wei Shih	Problèmes Wei Shih	Taille initiale	Taille réduite	FPR 83	Algorithme Wei Shih
1	5 × 30	5 × 14	54	137	16	5 × 60	2 × 8	59	72
2	5 × 30	1 × 5	51	54	17	5 × 60	1 × 14	49	47
3	5 × 30	4 × 11	43	40	18	5 × 70	3 × 13	89	813
4	5 × 30	1 × 2	24	33	19	5 × 70	1 × 2	73	232
5	5 × 30	0 × 0	24	31	20	5 × 70	3 × 9	74	270
6	5 × 40	3 × 11	64	209	21	5 × 70	1 × 7	54	89
7	5 × 40	3 × 11	59	147	22**	5 × 80	3 × 29	100	978
8	5 × 40	2 × 10	58	51	23	5 × 80	2 × 13	93	896
9	5 × 40	1 × 3	32	32	24**	5 × 80	3 × 20	100	227
10**	5 × 50	3 × 18	61	288	25	5 × 80	3 × 10	80	153
11	5 × 50	2 × 12	64	163	26**	5 × 90	3 × 30	111	1654
12	5 × 50	1 × 5	76	110	27	5 × 90	2 × 7	86	262
13	5 × 50	3 × 15	65	100	28	5 × 90	2 × 8	85	236
14	5 × 60	3 × 11	84	261	29	5 × 90	2 × 9	97	104
15	5 × 60	3 × 6	67	173	30	5 × 90	0 × 0	73	49

Tableau 4.5 (*)

(*) Les pourcentages de variables et de contraintes éliminées sont respectivement 81 % et 55 %. Ces pourcentages très élevés de réduction permettent de penser que ces problèmes de Wei Shih sont très faciles.

(**) Pour ces 4 problèmes la valeur optimale n'est pas atteinte par les heuristiques au cours de l'algorithme de réduction. Les écarts $v(B) - \underline{v}(B)$ sont pour les problèmes n° 10, 22, 24, 26, respectivement 1, 39, 18 et 32. Ceci explique la moindre réduction constatée pour ces 4 problèmes.

Remarques

(i) Toutes les expériences numériques ont été menées avec le choix des paramètres suivants : au cours d'un processus de réduction,

- le nombre maximum d'appel du module de réduction lagrangienne (resp. composite) est fixé à 2 (resp. 1)

- le nombre d'itérations de l'algorithme de sous-gradients (resp. quasi-sous-gradients) est 80 (resp. 100)

- le paramètre μ du test de décision est suffisamment large pour que les tests V3/V4 et C2 soient le plus souvent utilisés : on a privilégié ainsi les performances en réduction par rapport au temps de calcul.

(ii) Il est essentiel de noter que ce n'est pas nécessairement le meilleur multiplicateur au sens de la valeur du dual qui sera le meilleur multiplicateur au sens de la réduction (plus grand nombre de variables et/ou de contraintes éliminées).

Par exemple, considérons le problème $(B) \equiv (P6)$ (voir tableau 4.1) tel que $v(B) = 10618$ et $|v(\bar{B})| = 10672$. Le multiplicateur lagrangien $w_1 = (6.00223, 2.72651, 1.20933, 0.00000, 5.40026)$, correspondant au majorant de $v(B)$: $v(\bar{B}^{\circ}(w_1)) = 10673.964844$, permet par l'intermédiaire du test V1 d'éliminer 11 variables. Par contre, le multiplicateur lagrangien $w_2 = (6.05055, 2.86449, 1.02625, 0.00000, 5.20942)$, correspondant à un majorant $v(\bar{B}^{\circ}(w_2)) = 10675.355469$ plus grand que le précédent permet l'élimination d'une variable supplémentaire (variable numéro 8)

(iii) grâce aux tableaux 4.1, 4.4 et 4.5 nous avons construit à partir des problèmes tests utilisés un certain nombre de problèmes "difficiles" (voir Annexe 2).

ENCADREMENT DE LA SOMME DES VARIABLES

CHAPITRE 5

INTRODUCTION

Le but de ce chapitre est de réduire le domaine des solutions réalisables du problème (B) suivant

$$(B)\quad \begin{cases} \max \ cx \\ \text{s.c. } Ax \leq b & (1) \\ \underline{v}(B) \leq cx \leq \bar{v}(B) & (2) \\ x \in V \end{cases}$$

(où $c \in \mathbb{R}_+^n$, $b \in \mathbb{R}_+^m$, $A \in R_+^{m \times n}$)

en adjoignant aux (m+1) contraintes de ce problème une contrainte supplémentaire d'encadrement de la somme des variables du type

$$\ell \leq c\,x = \sum_{j=1}^{n} x_j \leq u \qquad \text{avec } \ell, u \in \{1, 2, \ldots, n\}$$

l'encadrement de la fonction économique (2) étant obtenu par les méthodes proposées aux chapitres 2 et 3.

La génération de cette nouvelle contrainte peut aussi s'envisager pour le problème réduit issu du problème initial (B) après utilisation de l'algorithme de réduction décrit au chapitre 4.

Après le rappel dans une première partie (§ 5.1) des techniques utilisées par Glover (1965, [18]), nous proposons dans une deuxième partie (§ 5.2) un algorithme permettant un encadrement beaucoup plus serré de la somme des variables. Quelques expériences numériques sur les problèmes 6 et 7 de Petersen seront rapportées.

5.1 - Approche classique

Les (m+1) inégalités (1) et (2) permettent la détermination d'un majorant (inférieur à 2^n) du nombre de solutions réalisables de (B) à envisager :

En posant (voir Glover [18])

$$\ell = \min \{k \mid \sum_{j=1}^{k} c_j \geq \underline{v}(B),\ c_1 \geq c_2 \geq \ldots \geq c_n\}$$

$$u = \min \{u_o, u_1, \ldots, u_m\} \text{ avec}$$

$$u_o = \max \{k \mid \sum_{j=1}^{k} c_j \leq \bar{v}(B),\ c_1 \leq c_2 \leq \ldots \leq c_n\}$$

$$u_i = \max \{k \mid \sum_{j=1}^{k} a_{ij} \leq b_i,\ a_{i1} \leq a_{i2} \leq \ldots \leq a_{in}\} \quad i = 1, 2, \ldots, m$$

on déduit que $\forall\ x \in \{x \in V \mid Ax \leq b,\ \underline{v}(B) \leq cx \leq \bar{v}(B)\}$

$$\text{alors } \ell \leq \sum_{j=1}^{n} x_j \leq u$$

ce qui implique $|F(B)| \leq \sum_{k=}^{u} \binom{n}{k} < \sum_{k=1}^{n} \binom{n}{k} = 2^n$

Il est à remarquer que le calcul des (m+2) bornes ℓ, u_o, u_1, ..., u_m nécessite (m+2) appels d'algorithmes de type NKR, donc de complexité O(n) en moyenne.

Le tableau 5.1 répertorie les valeurs des bornes ℓ et u pour vingt problèmes concrets.

Problèmes	Taille	ℓ	u	Problèmes	Taille	ℓ	u
P1	10 × 6	2	4	W1	2 × 28	11	24
P2	10 × 10	3	8	W2	2 × 28	8	22
P3	10 × 15	7	12	W3	2 × 28	5	18
P4	10 × 20	7	15	W4	2 × 28	7	21
P5	10 × 28	12	24	W5	2 × 28	5	18
P6	5 × 39	10	36	W6	2 × 28	8	22
P7	5 × 50	13	46	W7	2 ×105	81	100
HP1	4 × 28	7	25	W8	2 ×105	17	56
HP2	4 × 35	5	29	ST1	30 × 60	17	42
F	10 × 20	9	12	ST2	30 × 60	30	48

Tableau 5.1

Remarque :

On peut éventuellement améliorer la borne inférieure ℓ en la déterminant par : $\ell = \max \{\ell_o, \ell_1, \ldots, \ell_m\}$

avec $\ell_o = \min \{k \mid \sum_{j=1}^{k} c_j \geq \underline{v}(B),\ c_1 \geq c_2 \geq \ldots \geq c_n\}$

$\ell_i = \min \{k \mid \sum_{j=1}^{k} a_{ij} \geq \underline{b}_i,\ a_{i1} \geq a_{i2} \geq \ldots \geq a_{in}\}, \quad i = 1,2,\ldots,m$

où $\underline{b}_i = \lceil v(K_i) \rceil$ est obtenu en résolvant

$$(K_i) \quad \begin{cases} \min\ A_i\, x \\ \text{s.c. } c\, x > \underline{v}(B) \\ \qquad x \in [V] \end{cases}$$

Cette mise en oeuvre nécessite 2m appels d'algorithmes de type NKR.

5.2 - Nouvelle approche

5.2.1 - Etudes des fonctions Ψ et $\bar{\Psi}$

Soit w^* un multiplicateur "presque optimal" (**) d'un des duaux lagrangien (D_L) ou composite $(\bar{D}_S)$ associés au problème (B)

$$(B) \quad \begin{cases} \max\ cx \\ \text{s.c. } Ax \leq b \\ \qquad x \in V \end{cases}$$

$$(D_L) \quad \begin{cases} \min\ v(BL(w)) \\ \text{s.c. } w \in \mathbb{R}_+^m \end{cases} \qquad \text{où } v(BL(w)) = \max_{x \in V} \{cx + w\,(b - Ax)\}$$

$$(\bar{D}_S) \quad \begin{cases} \min\ v(\overline{BS(w)}) \\ \text{s.c. } w \in B_2 \end{cases} \qquad \text{où } v(\overline{BS(w)}) = \max_{x \in \lceil V \rceil} \{cx \mid wAx \leq wb\}$$

$$\text{et } B_2 = \{w \in \mathbb{R}_+^m \mid ||w||_2 = 1\}$$

(*) ou $cx \geq \underline{v}(B) + 1$ si l'hypothèse $c \in \mathbb{N}^n$ est faite.

(**) "presque optimal" signifiant qu'en pratique w^* sera tel que :
$v(BS(w^*))$ (resp. $v(BL(w^*))$) = $v(\bar{B}) + \varepsilon$ ($\varepsilon > 0$ réel très petit).

Pour le multiplicateur w^* fixé et un entier p donné $\in \{1, 2, \ldots, n\}$, considérons les deux problèmes en variables bivalentes à 2 ocntraintes, $(R_\ell(w^*, p))$ et $(R^u(w^*, p))$ suivants :

$$(R_\ell(w^*, p)) \quad \begin{cases} \max \ cx \\ \text{s.c. } w^*Ax \leq w^*b \\ \qquad ex \leq p \\ \qquad x \in V \end{cases}$$

$$(R^u(w^*, p)) \quad \begin{cases} \max \ cx \\ \text{s.c. } w^*Ax \leq w^*b \\ \qquad ex \geq p \\ \qquad x \in V \end{cases}$$

Les fonctions duales composites en continu associées respectivement aux problèmes $(R_\ell(w^*, p))$ et $(R^u(w^*, p))$ sont définies par :

$\forall \lambda \in [0, 1]$

$$\bar{\phi}_\ell(p, \lambda) = \max_{x \in [V]} \{cx \mid (1-\lambda)\, w^*Ax + \lambda e\, x \leq (1-\lambda)\, w^*b + \lambda p\}$$

$$\bar{\phi}^u(p, \lambda) = \max_{x \in [V]} \{cx \mid (1-\lambda)\, w^*Ax - \lambda e\, x \leq (1-\lambda)\, w^*b - \lambda p\}$$

On définit de même les fonctions duales en entier ϕ_ℓ et ϕ^u en remplaçant dans les expressions de $\bar{\phi}_\ell$ et $\bar{\phi}^u$ le cube unité [V] de $\mathbb{R}^n$ par l'ensemble V de ses sommets.

Les valeurs des duaux composites associés respectivement aux problèmes $(R_\ell(w^*, p))$ et $(R^u(w^*, p))$ sont notées comme suit :

<u>cas réel</u>

$$\bar{\phi}_\ell(p) = \min_{0 \leq \lambda \leq 1} \bar{\phi}_\ell(p, \lambda)$$

$$\bar{\phi}^u(p) = \min_{0 \leq \lambda \leq 1} \bar{\phi}^u(p, \lambda)$$

cas entier $\quad \phi_\ell(p) = \min_{0\le\lambda\le 1} \phi_\ell(p, \lambda) \qquad \phi^u(p) = \min_{o\le\lambda\le 1} \phi^u(p, \lambda)$

D'après la proposition 1.6 les fonctions ϕ et $\bar{\phi}$ possèdent les propriétés suivantes :

Propriété 5.1

$\forall\, p \in \{1, 2, \ldots, n\}$, *les fonctions* $\bar{\phi}_\ell(p, .)$ *et* $\bar{\phi}^u(p, .)$ *sont quasi-convexes et continues sur* $[0, 1]$, *à valeurs dans* $\mathbb{R}_+$.

Propriété 5.2

$\forall\, p \in \{1, 2, \ldots, n\}$, *les fonctions* $\phi_\ell(p, .)$ *et* $\phi^u(p, .)$ *sont quasi-convexes et semi-continues supérieurement sur* $[0, 1]$, *à valeurs dans* $\mathbb{N}$.

Exemple 1

La figure 5.1 donne les représentations graphiques de la fonction $\bar{\phi}_\ell(p, .)$ pour le problème 6 de Petersen pour p = 20 et p = 21.

($w^* = (.6876, .2961, .1715, .0000, .6605)$, $v(B) = 10618$ et

$\bar{\phi}_\ell(p, 0) = \overline{v(BS(w^*))} = 10672.375$, $\forall\, p \in \{1, 2, \ldots, 39\}$

$\bar{\phi}_\ell(20, 1) = 13455$ et $\bar{\phi}_\ell(21, 1) = 13570$).

Exemple 2

La figure 5.2 donne la représentation graphique de la fonction $\phi_\ell(p, .)$ pour le problème 6 de Petersen pour p = 22.

($\phi_\ell(p, 0) = v(BS(w)) = 10662$, $\forall\, p \in \{1, 2, \ldots, 39\}$ $\phi_\ell(22, 1) = 13678$)

D'autre part, les fonctions Ψ et $\bar{\Psi}$ ont la propriété suivante.

Propriété 5.3

Ψ_ℓ *et* $\bar{\Psi}_\ell$ *(resp.* Ψ^u *et* $\bar{\Psi}^u$*) sont des fonctions non décroissantes (resp. non croissantes) de la variable* p.

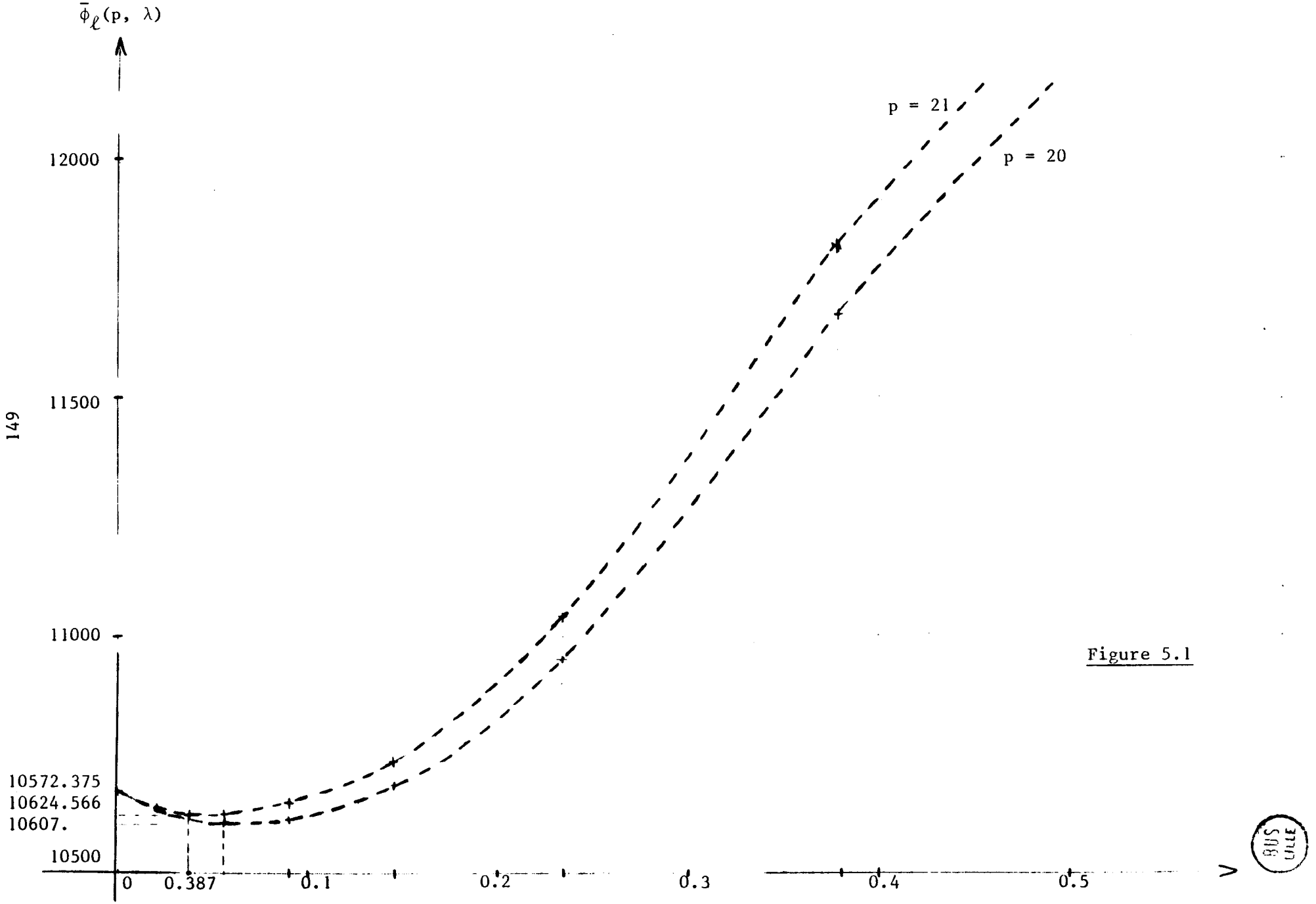


Figure 5.1

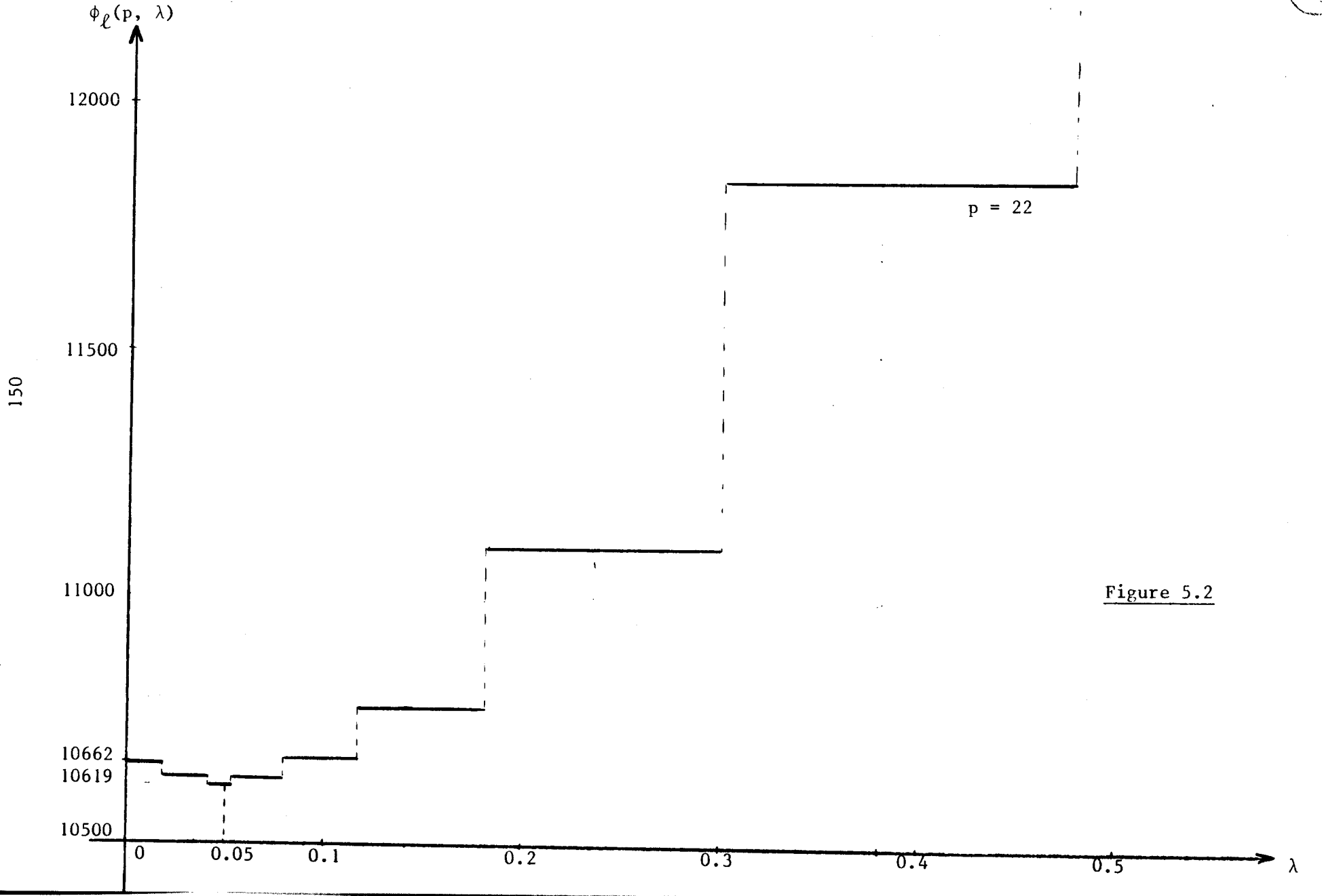

$\phi_\ell(p, \lambda)$
12000
11500
11000
10662
10619
10500
150
p = 22
0
0.05
0.1
0.2
0.3
0.4
0.5
λ

Figure 5.2

Démonstration :

$\forall \lambda \in [0, 1]$, pour tout couple $(p, p') \in \{1, 2, \ldots, n\}^2$ tel que $p < p'$, nous avons l'inclusion suivante :

$$\{x \in [V] \mid ((1-\lambda)w^* A+\lambda e)x \leq (1-\lambda)w^* b+\lambda p\} \subseteq \{x \in [V] \mid ((1-\lambda)w^* A+\lambda e)x \leq (1-\lambda)w^* b+\lambda p'\}$$

d'où $\bar{\phi}_\ell(p, \lambda) \leq \bar{\phi}_\ell(p', \lambda)$

et par conséquent $\bar{\Psi}_\ell(p) = \min_{0 \leq \lambda \leq 1} \bar{\phi}_\ell(p, \lambda) \leq \bar{\Psi}_\ell(p') = \min_{0 \leq \lambda \leq 1} \bar{\phi}_\ell(p', \lambda)$

En fait, l'inclusion est stricte pour $\lambda \in]0, 1]$ et nous aurons $\bar{\Psi}_\ell(p) < \bar{\Psi}_\ell(p')$ lorsque le minimum ne sera pas atteint pour $\lambda = 0$. Les démonstrations sont identiques pour Ψ_ℓ, Ψ^u et $\bar{\Psi}^u$.

□

Exemple 3 :

La figure 5.3 donne les représentations graphiques des fonctions $\bar{\Psi}_\ell(.)$ et $\bar{\Psi}^u(.)$ pour le problème 6 de Petersen.

5.2.2 - *Algorithme d'encadrement*

5.2.2.1 - Principe

L'algorithme d'encadrement repose essentiellement sur le théorème suivant :

Théorème 5.1

Soit x^* *une solution optimale du problème* (B) *et* $p \in \{1, 2, \ldots, n\}$

(i) si $\lfloor \bar{\Psi}_\ell(p) \rfloor < \underline{v}(B)$ *alors* $e\, x^* \geq p+1$

(ii) si $\Psi_\ell(p) < \underline{v}(B)$ *alors* $e\, x^* \geq p+1$

(iii) si $\lfloor \bar{\Psi}^u(p) \rfloor < \underline{v}(B)$ *alors* $e\, x^* \leq p-1$

(iv) si $\Psi^u(p) < \underline{v}(B)$ *alors* $e\, x^* \leq p-1$

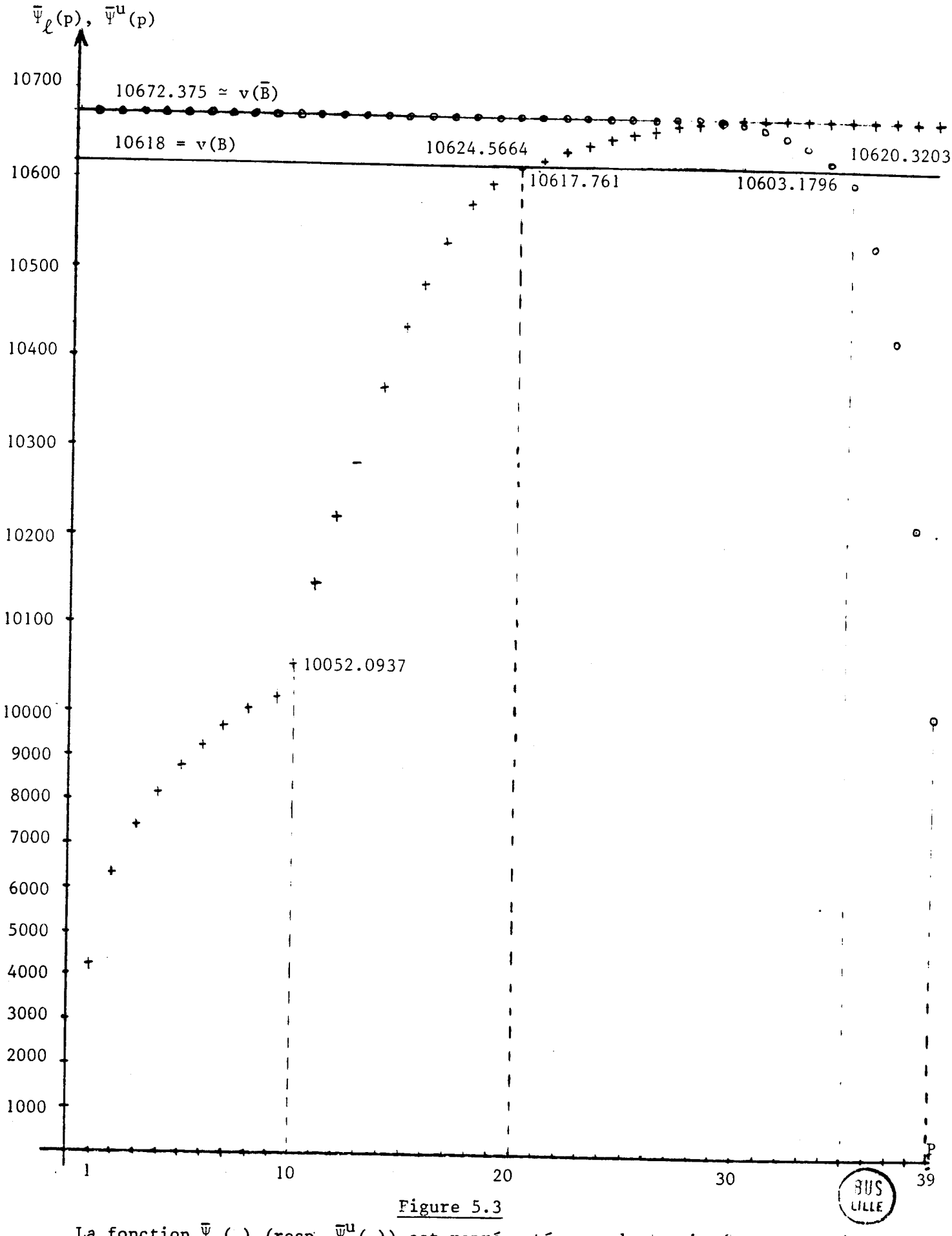


Figure 5.3

La fonction $\bar{\Psi}_\ell(.)$ (resp. $\bar{\Psi}^u(.)$) est représentée par des croix (resp. ronds)

Démonstration :

Vérifions que l'implication (ii) est vraie en montrant que :

$ex^* \leq p \Rightarrow \underline{v}(B) \leq \Psi_\ell(p)$

Sous l'hypothèse $ex^* \leq p$, le problème (B) a la même valeur optimale que (B_p) définie par :

$$(B_p) \quad \begin{cases} \max \ cx \\ \text{s.c. } Ax \leq b \\ \quad ex \leq p \\ \quad x \in V \end{cases}$$

D'autre part, $\Psi_\ell(p)$ n'est autre que la valeur du dual composite du problème $(R_\ell(w^*, p)$, lui-même relaxation du problème (B_p). Par conséquent $\underline{v}(B) \leq v(B) = v(B_p) \leq v(R\ (w^*, p)) \leq \Psi_\ell(p)$. Le (i) se déduit du (ii) grâce à l'inégalité $\Psi_\ell(p) \leq \lfloor \bar{\Psi}_\ell(p) \rfloor$. Les implications (iii) et (iv) se démontrent de la même manière par renversement de l'inégalité $ex^* \leq p$.

□

Ce théorème va nous permettre d'énoncer un processus d'augmentation et de diminution déterminant respectivement la borne inférieure ℓ et la borne supérieure u de la somme des variables.

5.2.2.2 - Algorithme détaillé

L'algorithme d'encadrement de la somme des variables peut s'énoncer comme suit :

Algorithme

1 p ← 1

1.1 *tant que* $\lfloor \bar{\Psi}(p) \rfloor < \underline{v}(B)$ *faire* p ← p+1

1.2 $\ell \leftarrow p$

2 p ← n

2.1 *tant que* $\lfloor \bar{\Psi}^u(p) \rfloor < \underline{v}(B)$ *faire* p ← p-1

2.2 u ← p

3 p ← ℓ

3.1 *tant que* $\Psi_\ell(p) < \underline{v}(B)$ *faire* p ← p+1

3.2 $\ell \leftarrow p$

4 p ← u

4.1 *tant que* $\Psi^u(p) < \underline{v}(B)$ *faire* p ← p-1

4.2 u ← p

5 *Stop*

La mise en oeuvre de cet algorithme implanté sur le calculateur CII HB IRIS 80 nécessite plusieurs remarques :

(i) Le calcul de $\bar{\Psi}_\ell(p)$ (resp. $\bar{\Psi}^u(p)$, $\Psi_\ell(p)$, $\Psi^u(p)$) consistant à minimiser la fonction quasi-convexe $\bar{\phi}_\ell(p, .)$ (resp. $\bar{\phi}^u(p, .)$, $\phi_\ell(p, .)$, $\phi^u(p, .)$) sur l'intervalle compact [0, 1], se réalise par l'intermédiaire d'une *recherche dichotomique* adaptée à la nature des fonctions ϕ (propriétés 5.1 et 5.2).

(ii) Pour $\lambda \in [0, 1]$ et $p \in \{1, 2, \ldots, n\}$ donnés,

- le calcul de $\bar{\phi}_\ell(p, \lambda)$ ou de $\bar{\phi}^u(p, \lambda)$, problèmes du type knapsack en continu, est effectué par l'algorithme NKR modifié pour tenir compte de la contrainte à coefficients de signe quelconque (algorithme de complexité O(n) en moyenne)

- le calcul de $\phi_\ell(p, \lambda)$ ou de $\phi^u(p, \lambda)$, problèmes du type knapsack en variables bivalentes, est effectué par une adaptation de l'algorithme FPK 79 (voir [10]) au cas d'une contrainte à coefficients de signe quelconque.

- le temps de calcul sensiblement plus important pour la procédure énumérative FPK 79 en comparaison de l'algorithme NKR (en moyenne deux fois plus) justifie la hiérarchie mise en place, les phases 1 et 2 avant les phases 3 et 4.

(iii) L'emploi de l'instruction "tant que" se justifie par la non décroissance des fonctions $\Psi_\ell(.)$ et $\bar{\Psi}_\ell(.)$ (resp. la non croissance des fonctions $\Psi^u(.)$ et $\bar{\Psi}^u(.)$) (propriété 5.3). En effet, dès qu'il existe $p \in \{1, 2, \ldots, n\}$ tel que :

$$\lfloor \bar{\Psi}_\ell(p) \rfloor \geq \underline{v}(B)$$

alors "propriété de non-décroissance de $\bar{\Psi}_\ell$" $\Rightarrow$ $\begin{cases} \forall\, p' \in \{p+1, \ldots, n\} \\ \lfloor \bar{\Psi}_\ell(p') \rfloor \geq \underline{v}(B) \end{cases}$

De même, dès qu'il existe $p \in \{1, 2, \ldots, n\}$ tel que :

$$\lfloor \bar{\Psi}^u(p) \rfloor \geq \underline{v}(B)$$

alors "propriété de non-croissance de $\bar{\Psi}^u$" $\Rightarrow$ $\begin{cases} \forall\, p' \in \{1, 2, \ldots, p-1\} \\ \lfloor \bar{\Psi}^u(p') \rfloor \geq v(B) \end{cases}$

(idem pour Ψ_ℓ et Ψ^u).

5.2.3 - *Résultats numériques*

Le tableau 5.2 compare sur quatre problèmes réels les performances de la procédure inspirée par Glover et de l'algorithme présenté précédemment. Deux cas sont envisagés ; phases 1 et 2 seulement (algorithme NKR) puis phases 1 à 4 (algorithmes NKR + FPK 79).

Problèmes et Taille	Glover			Phases 1 et 2			Phases 1 à 4		
	ℓ	u	temps	ℓ	u	temps	ℓ	u	temps
P6 : 5 39	10	36	1	21	34	23*	22	32	42*
P7 : 5 50	13	46	1	26	41	31*	27	38	67*
HP1 : 4 28	7	25	1	12	25	12*	15	25	11*
HP2 : 4 35	5	29	1	13	28	16*	14	28	14*

Tableau 5.2

Notes

(i) Les temps de calcul sont donnés en dixièmes de seconde. Toutefois les temps de calcul (*) ne sont qu'indicatifs car ils tiennent compte de très nombreuses impressions de textes.

(ii) Dans tous les cas la borne inférieure $\underline{v}(B)$ utilisée est la valeur optimale du problème.

□

Les résultats obtenus sont encourageants et l'adjonction de cette contrainte sur la somme des variables, à la réduction de la taille du problème, diminue de façon très sensible le nombre des solutions réalisables à explorer dans une procédure de résolution de type séparation et évaluation. Cette contrainte peut aussi servir à définir un schéma exploratoire original comme celui utilisé par Thesen dans [64].

CONCLUSION

Ce travail a abouti à la création d'un code de réduction, aussi bien en nombre de variables et de contraintes, qu'en nombre de solutions réalisables d'un problème de maximisation à plusieurs contraintes en variables bivalentes. Les performances de ce code, en réduction comme en temps calcul, s'avèrent très satisfaisantes sur des problèmes concrets ou tirés au hasard de petite et moyenne taille, sans structure particulière.

Cependant ce code expérimental mérite extension et amélioration. En effet, disposant d'un ensemble de problèmes tests de type investissement nous nous sommes intéressés en premier lieu aux problèmes à données entières non négatives. Le passage au cas de signe quelconque nécessite des modifications certes importantes mais uniquement d'ordre technique . De plus une mise en exploitation de ce code expérimental, tant en phase préliminaire qu'en cours de résolution exacte par une procédure de type séparation et évaluation progressive, demande une étude plus élaborée sur le plan informatique.

D'autre part, il serait intéressant d'analyser l'efficacité de cet algorithme de réduction en fonction de paramètres tels que : densité de la matrice des contraintes, dispersion des coefficients, coefficients de relâchement des contraintes. En effet les problèmes de grande taille sont généralement très structurés et cette étude permettrait de connaître a priori sur quel type de structure ce code de réduction donnerait des résultats satisfaisants.

D'autres prolongements peuvent être envisagés en incluant par exemple d'autres ingrédients dans le processus de réduction, tels que des relations de dominance et d'exclusion, des procédés de diminution des plus gros

coefficients, etc ...

En définitive nous espérons que ce travail sera un outil utile, à la fois sur le plan théorique et pratique, à tous ceux qui sont intéressés par la programmation mathématique et plus particulièrement la programmation en variables bivalentes.

GLOSSAIRE DES ELEMENTS THEORIQUES

ANNEXE 1

Il nous est apparu nécessaire pour faciliter la lecture de cet exposé de rassembler dans cet annexe toutes les propriétés connues et moins connues relatives aux notions de convexité et de quasi-convexité que l'on a pu utiliser dans ce travail à un moment ou à un autre.

Bibliographie : [55], [34], [25], [45], [57].

A - RAPPEL DES PROPRIETES ESSENTIELLES

Définition 1 :

Soit $C \subset \mathbb{R}^n$, convexe et $f : C \to \mathbb{R}$

f est convexe sur C si et seulement si

$\forall (x, x') \in C^2$, $\forall \lambda \in [0, 1]$, $f(\lambda x + (1-\lambda)x') \leq \lambda f(x) + (1-\lambda)f(x')$

Définition 2 :

Soit $C \subset \mathbb{R}^n$, convexe et $f : C \to \mathbb{R}$. L'épigraphe de f est l'ensemble

$$epi(f) = \{(x, z) \in C \times \mathbb{R} \mid f(x) \leq z\}$$

Proposition 1 :

Soit $C \subset \mathbb{R}^n$, convexe et $f : C \to \mathbb{R}$

f convexe <=> epi(f) convexe de $\mathbb{R}^n \times \mathbb{R}$

Définition 3 :

Soit $f : \mathbb{R}^n \to \mathbb{R}$

La conjuguée de f est la fonction f_* définie sur $\mathbb{R}^n$ par

$$f_*(y) = \sup_{x \in \mathbb{R}^n} \{x\, y - f(x)\}$$

Définition 4 :

Soit $C \subset \mathbb{R}^n$, convexe et $f : C \to \mathbb{R}$

f est quasi-convexe sur C si et seulement si

$\forall (x, x') \in C^2 : \forall z \in [x, x'], f(z) \leq \max\{f(x), f(x')\}$

(une fonction convexe est quasi-convexe, la réciproque est fausse)

Définition 5 :

Soit $C \subset \mathbb{R}^n$, convexe ; $f : C \to \mathbb{R}$; $z \in \mathbb{R}$. L'ensemble de niveau de f associé au réel z est l'ensemble.

$$L_z f = \{x \in C \mid f(x) \leq z\}$$

Proposition 2 :

Soit $C \subset \mathbb{R}^n$ convexe, $f : C \to \mathbb{R}$

f quasi-convexe $\Leftrightarrow \forall z \in \mathbb{R}$, $L_z f$ convexe de $\mathbb{R}^n$

Définition 6 :

Soit $f : \mathbb{R}^n \to \mathbb{R}$

La z quasi-conjuguée de f est la fonction quasi-convexe f_z^* définie par

$$f_z^*(y) = z - \inf_{x \in \mathbb{R}^n} \{f(x) \mid xy \geq z\}$$

Proposition 3 :

Soit $f : \mathbb{R}^n \to \mathbb{R}$

(i) $f_z^*(y) \leq f_*(y)$; $\forall (y, z) \in \mathbb{R}^{n+1}$

(ii) $f_*(y) = \sup_{z \in \mathbb{R}} \{f_z^*(y)\}$

Proposition 4 :

Soit $f : \mathbb{R}^n \to \bar{\mathbb{R}}$: les propositions suivantes sont équivalentes

(i) f s.c.i sur $\mathbb{R}^n$

(ii) $L_z f$ est fermé pour tout réel z

(iii) epi(f) est un fermé de $\mathbb{R}^{n+1}$

B - SOUS-DIFFERENTIABILITE

Définition 7 :

Soit $C \subset \mathbb{R}^n$ convexe, $f : C \to \mathbb{R}$ convexe, $x_o \in C^o$

g est un sous-gradient de f en x_o si

$$\forall\, x \in C,\ f(x) - f(x_o) \geq g(x - x_o)$$

(l'hyperplan affine $x \to f(x_o) + g(x - x_o)$ de vecteur normal

(1, g) est un hyperplan de support de epi(f) au point $(x_o, f(x_o))$

Définition 8 :

L'ensemble éventuellement vide des sous-gradients de f en x_o est appelé le sous-différentiel $\partial f(x_o)$ de f en x_o

Proposition 5 :

Soit $C \subset \mathbb{R}^n$ convexe, $f : C \to \mathbb{R}$ convexe, $x_o \in C^o$

$\partial f(x_o)$ est un convexe compact non vide

(f convexe en $x_o \in C^o$ => f continue en $x_o \in C^o$ si C est une partie pleine de $\mathbb{R}^n$, i.e. la dimension de la plus grande variété affine contenue dans C est n).

Proposition 6 :

Soit $C \subset \mathbb{R}^n$ convexe, $f : C \to \mathbb{R}$ convexe, $x_o \in C^o$, g un s/s-gradient de f en x_o

$$g \in \partial f(x_o) \iff f(x_o) + f_*(g) = x_o\, g$$

Proposition 7 :

Soit $C \subset \mathbb{R}^n$ convexe, $f : C \to \mathbb{R}$ convexe, $x_o \in C^o$

$$0 \in \partial f(x_o) \iff x_o \text{ est un minimum global de f sur C.}$$

Remarques

(i) Lorsque $\partial f(x_o)$ est un singleton, $\partial f(x_o) = \{\nabla f(x_o)\}$, c'est à dire que f est différentiable en x_o

(ii) Les fonctions convexes sont presque partout différentiables.

C - QUASI-SOUS-DIFFERENTIABILITE

Définition 9 :

Soit $C \subset \mathbb{R}^n$ convexe, $f : C \to \mathbb{R}$ quasi-convexe, $x_o \in C^o$.

y est un quasi-sous-gradient de f en x_o si

$$f(x_o) + f^*_{x_o y}(y) = x_o y$$

Définition 10 :

L'ensemble éventuellement vide des quasi-sous-gradients de f en x_o est appelé le quasi-sous-différentiel $\partial^* f(x_o)$ de f en x_o

Proposition 8 :

Soit $C \subset \mathbb{R}^n$ convexe, $f : C \to \mathbb{R}$ quasi-convexe, localement convexe en $x_o \in C^o$

(i) $\partial f(x_o) \subset \partial^* f(x_o)$

(ii) $\partial^* f(x_o)$ est convexe

(iii) si f est s.c.i. en $x_o \in fr(L_{f(x_o)f})$ alors $\partial^* f(x_o) \neq \emptyset$

(iv) $\partial^* f(x_o) = \{y \mid x'y \geq x\,y \Rightarrow f(x') \geq f(x)\}$

Proposition 9 :

Soit $C \subset \mathbb{R}^n$ convexe, $f : C \to \mathbb{R}$ quasi-convexe, $x_o \in C^o$

$0 \in \partial^* f(x_o) \Leftrightarrow x_o$ est un minimum global de f sur C

Proposition 10

Soit $C \subset \mathbb{R}^n$ convexe, $f : C \to \mathbb{R}$ quasi-convexe, $x_o \in C^o$, H l'hyperplan d'équation $x \to y\ (x-x_o)$

si $L_{f(x_o)}f \neq \emptyset$

(i) si $L_{f(x_o)}f = \overline{L^o_{f(x_o)}f}$ et si $y \in \partial^* f(x_o)$ alors H est un hyperplan de support de $L_{f(x_o)}f$ en x_o

(ii) si f est s.c.i en x_o et si H est un hyperplan de support de $L_{f(x_o)}f$ en x_o alors $y \in \partial^* f(x_o)$

Remarques

(i) une fonction quasi-convexe n'admet pas de sous-gradient en chaque point

(ii) comme les sous-gradients de f en x_o correspondent aux normales aux hyperplans de support de l'épigraphe épi(f), les quasi-sous-gradients de f en x_o correspondent aux normales aux hyperplans de support de l'ensemble de niveau $L_{f(x_o)}f$.

CATALOGUE DE PROBLEMES DIFFICILES

ANNEXE 2

A partir des 50 problèmes tests (dont les données figurent intégralement dans [13]) utilisés dans les expériences numériques, nous avons construit 7 problèmes "difficiles" grâce à l'algorithme de réduction décrit au chapitre 4.

Nous espérons que ces nouveaux problèmes tests seront d'une aide certaine pour les chercheurs intéressés par ce type de problème. Enfin il est possible de construire des problèmes difficiles au sens de notre réduction en appliquant le code de réduction FPR 83 à tout problème dont les données seraient tirées au hasard.

Problème 1 (tiré de P6)

nombre de contraintes : 4

nombre de variables : 27

Max {cx | Ax ≤ b ; x binaire} = 3090

c =

560	1125	68	47	122	196	41	25	115	82
22	631	132	420	86	42	103	81	26	49
316	72	71	49	108	116	90			

b =

207	185	168	160

A =

40	91	3	3	18	25	1	1	8	1
1	49	8	21	6	1	5	8	1	0
42	6	4	8	0	10	1			
16	92	4	6	0	8	2	1	6	2
1	70	9	22	4	1	5	6	0	4
8	4	3	0	10	0	6			
38	39	5	8	12	15	0	1	20	3
0	40	6	8	0	6	4	4	1	5
8	2	8	0	20	0	0			
38	52	7	0	3	4	1	2	4	6
1	18	15	38	10	4	8	0	3	0
6	1	3	0	3	5	4			

x^* =

1	1	0	1	0	0	1	0	1	1
1	0	0	1	0	1	0	1	0	1
0	1	1	1	1	1	1			

Problème 2 (tiré de P7)

nombre de contraintes : 4

nombre de variables : 34

Max {cx | Ax ≤ b ; x binaire} = 3186

C =

560	620	68	328	47	122	196	41	25	115
82	22	631	132	420	86	42	103	215	81
91	26	49	316	72	71	49	108	116	90
215	58	47	81						

b =

163	165	239	168

A =

40	30	3	12	3	18	25	1	1	8
1	1	49	8	21	6	1	5	10	8
2	1	0	42	6	4	8	0	10	1
8	3	2	4						

16	16	4	18	6	0	8	2	1	6
2	1	70	9	22	4	1	5	10	6
4	0	4	8	4	3	0	10	0	6
22	0	2	2						

38	71	5	40	8	12	15	0	1	20
3	0	40	6	8	0	6	4	22	4
6	1	5	8	2	8	0	20	0	0
13	6	1	2						

38	42	7	20	0	3	4	1	2	4
6	1	18	15	38	10	4	8	6	0
0	3	0	6	1	3	0	3	5	4
18	3	4	0						

x^* =

0	1	0	1	1	0	1	1	0	0
1	1	0	0	1	0	1	1	1	1
1	0	1	0	1	1	1	1	1	1
1	0	1	1						

Problème 3 (tiré de W3)

nombre de contraintes : 2

nombre de variables : 19

Max {cx | Ax ≤ b, x binaire} = 28642

c =

440	22507	14148	4650	4975	1160	4225	510	479	440
490	330	110	560	2885	11748	4550	3720	1950	

b =

135	157

A =

0	85	65	30	0	40	25	20	0	25
0	0	25	0	0	85	0	0	0	
20	125	80	35	15	15	40	5	10	12
10	9	0	20	40	50	36	40	19	

x^* =

0	1	0	0	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	

Problème 4 (tiré de W8)

nombre de contraintes : 2

nombre de variables : 29

Max {cx | Ax ≤ b ; x binaire} = 95168

c =

7074	5587	5500	3450	367	4235	9262	6155	32240	1600
4500	6570	7010	16020	8000	2568	2365	4350	4975	29400
7471	3840	1125	1790	2450	540	445	220		

b =

153	154

A =

25	17	20	22	15	10	50	10	150	0
0	0	40	60	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	2	5	6	40	2
6	10	13	30	15	5	5	10	15	91
24	15	15	5	10	15	10	10	10	

x^* =

1	1	1	0	1	1	1	1	0	1
1	1	0	0	1	1	0	1	0	1
0	0	0	0	0	0	0	0	0	

Problème 5 (Fleisher)

nombre de contraintes : 10

nombre de variables : 20

Max {cx | Ax ≤ b ; x binaire} = 2139

c =

245	177	291	237	114	237	194	211	231	211
97	168	174	134	308	271	131	211	97	282

b =

463	451	623	493	551	647	624	511	595	526

A =

77	12	57	7	21	20	85	52	72	90
62	71	93	8	79	68	68	44	13	52
21	85	87	82	26	61	36	1	14	19
74	26	85	24	56	63	0	12	1	39
72	58	72	93	3	93	17	54	92	79
12	20	24	30	99	79	15	11	8	89
76	14	86	37	12	48	54	58	32	15
6	54	32	32	71	61	29	92	20	73
11	62	93	56	80	52	93	38	89	33
17	32	35	76	51	31	40	73	42	78
28	85	3	31	93	44	93	73	35	28
11	37	99	73	91	67	7	95	15	97
32	42	54	90	59	98	94	88	59	86
10	0	59	15	76	97	55	87	70	1
77	18	87	61	64	37	41	39	3	18
20	55	50	30	51	92	88	55	60	86
53	83	68	50	86	20	19	58	48	99
57	67	21	36	41	13	34	94	50	50
41	68	47	16	5	9	35	62	88	73
64	96	9	62	79	27	16	25	30	60

x^* =

0	1	0	1	0	1	0	1	0	1
0	1	0	1	0	1	0	1	0	1

Problème 6 (tiré de ST1)

nombre de contraintes : 30

nombre de variables : 40

Max {cx | Ax ≤ b, x binaire} = 776

c =

82	77	110	67	61	3	6	85	2	22
238	33	56	63	69	47	63	75	6	83
47	47	7	6	53	76	200	29	91	6
27	52	6	51	55	72	13	6	65	95

b =

3317	1880	2740	4310	2681	3375	4704	3031	3115	2878
2609	3321	4142	4670	2130	3323	4766	2590	3573	2888
3578	3478	4189	1748	2039	2660	4645	3578	4418	2211

A =

524	774	818	56	26	22	42	20	47	37
662	21	42	57	97	75	53	49	76	96
834	785	42	39	979	46	608	25	23	41
667	123	805	116	19	26	760	81	699	27
16	67	792	99	21	794	53	81	73	87
837	234	64	65	908	16	758	22	27	91
13	435	86	29	377	73	504	5	86	61
67	255	118	332	97	370	32	23	165	75
33	42	644	96	23	49	57	931	36	5
596	978	82	59	37	22	71	788	56	96
55	283	348	66	873	90	33	94	42	864
464	931	33	350	15	22	9	82	358	6
51	114	716	668	23	43	55	973	47	3
350	56	77	97	28	75	61	95	26	33
430	950	590	32	297	140	714	828	97	27
571	449	602	789	15	85	27	11	778	19
638	720	809	31	12	82	688	868	61	17
775	82	81	860	85	49	72	72	7	13
740	830	72	30	221	2	9	36	150	18
586	996	519	719	320	13	654	408	70	66
978	53	983	3	93	37	704	581	698	97
712	42	45	91	56	32	41	664	290	86
22	234	57	784	503	2	59	37	75	36
105	47	760	15	190	3	17	328	210	294

504	37	619	93	218	23	47	303	71	56
593	744	214	67	53	22	96	16	978	53
884	796	36	3	724	31	468	3	81	5
509	845	293	700	75	59	9	58	67	734
16	27	206	57	7	87	21	624	33	92
770	15	32	13	1	2	644	4	13	100
604	854	96	910	372	39	12	990	140	39
876	813	559	952	640	59	290	644	232	76
53	36	650	210	218	520	33	12	16	86
324	100	894	818	39	7	27	318	538	43
46	206	92	65	944	16	7	57	174	65
320	143	382	566	878	678	284	39	755	90
16	51	310	21	7	43	25	860	45	82
354	42	72	76	73	37	99	79	59	85
52	798	45	26	560	36	45	12	62	62
790	920	390	44	62	83	27	43	574	86
75	51	23	26	99	426	92	99	11	51
77	66	85	554	19	604	96	77	97	55
7	85	760	104	580	95	927	62	51	476
716	17	37	868	267	91	2	234	444	45
95	89	5	81	270	491	6	72	410	12
644	79	910	595	85	167	430	21	414	89
6	948	904	67	994	13	295	65	408	211
11	49	844	11	417	570	43	39	16	430
7	63	3	63	338	402	664	21	553	128
88	458	508	222	804	720	75	45	35	55
750	1	96	31	846	37	496	29	210	930
653	51	35	43	393	2	834	1	418	36
59	6	218	36	92	321	22	138	423	53
828	16	93	622	13	189	36	75	58	968
66	89	440	91	62	67	628	23	83	943
129	950	170	67	613	608	27	554	730	710
748	654	46	69	3	769	82	624	967	948
684	15	22	586	700	343	83	57	46	65
21	66	51	19	882	13	254	788	760	542
339	34	13	230	327	65	87	72	23	
934	900	99	51	628	150	22	79	132	628
52	154	21	663	968	846	620	840	72	73
6	110	22	20	497	2	201	2	924	866
501	57	3	2	279	5	148	86	91	93
89	26	96	37	39	89	6	53	869	2
394	82	46	541	26	268	82	72	710	16
5	934	88	93	56	95	194	37	66	855
71	95	7	408	464	41	56	564	488	508

83	2	630	61	62	638	61	73	494	71
75	62	198	958	70	232	174	25	25	600
268	15	95	898	515	464	571	42	21	47
462	51	96	338	228	93	690	56	89	99
62	290	72	844	6	570	13	16	267	32
69	65	27	236	83	482	97	274	31	55
554	97	47	7	480	504	616	14	824	641
374	2	478	36	496	754	848	75	330	76
89	7	22	45	63	148	91	33	624	47
96	69	42	224	56	970	65	67	96	27
2	75	83	69	832	72	688	92	93	8
214	77	66	2	384	59	26	75	15	81
14	62	464	292	29	340	278	83	33	374
389	818	888	37	3	60	42	40	928	42
35	76	69	95	46	32	864	55	198	598
73	15	95	36	814	45	770	644	634	334
56	780	56	588	93	86	23	94	19	86
570	518	108	218	86	718	63	81	400	938
96	43	6	69	50	76	674	42	5	810
9	49	274	25	298	9	42	363	47	158
86	75	99	554	704	800	6	670	47	49
199	85	69	3	77	63	674	924	55	22
214	864	718	56	67	53	19	554	1	49
56	418	13	948	56	19	65	154	99	363
359	12	130	481	92	26	624	714	53	21
743	13	3	718	33	930	13	1	79	88
41	91	36	30	16	974	950	9	759	5
29	63	55	45	93	56	580	6	93	452
2	19	864	535	800	3	27	36	9	974
876	21	73	11	2	130	694	16	7	334
248	25	86	944	79	441	92	6	800	7
93	9	254	22	930	744	7	475	498	252
640	36	35	843	718	22	76	8	17	33
972	39	96	76	42	378	89	96	77	65
344	87	17	17	62	86	594	790	56	2
43	83	520	864	92	31	71	845	174	780
644	59	33	671	89	364	854	21	65	2
584	570	95	26	924	87	804	17	2	168
42	42	918	6	96	178	724	610	818	46
964	194	97	59	634	21	630	969	47	75
91	13	224	120	85	7	92	72	468	920
949	93	7	57	13	538	920	65	9	37
200	57	12	31	79	310	92	888	77	444
734	59	27	898	59	33	418	935	73	272

674	93	63	490	1	63	870	49	23	984
131	52	39	99	338	6	114	69	87	47
86	12	52	110	61	81	984	284	384	52
804	750	83	87	174	99	23	482	7	232
53	55	9	264	73	35	92	31	35	61
670	65	66	37	96	21	86	83	21	19
79	7	2	46	15	59	76	83	47	7
35	52	71	7	32	87	27	680	52	990

Problème 7 (tiré de ST2)

nombre de contraintes : 30

nombre de variables .: 37

Max {cx | Ax ≤ b, x binaire} = 1035

c =

47	77	110	67	65	3	6	39	33	63
6	21	56	29	69	61	85	6	47	76
47	51	7	53	6	2	52	82	91	63
27	13	6	72	55	72	31			

b =

5875	4351	5221	7099	5746	5560	6840	6069	6333	5229
5428	5842	7422	5461	5047	6064	7582	5220	6483	4929
5909	5057	6662	3514	4469	4153	7077	6163	6955	3373

A =

785	774	818	56	699	22	42	465	21	57
76	62	42	25	97	26	20	41	75	46
834	116	42	979	81	47	123	524	23	53
667	760	805	690	19	26	646			
435	67	792	99	165	794	53	268	234	65
27	97	64	5	908	21	81	61	18	73
13	332	86	377	23	73	255	16	86	758
67	32	118	266	97	370	834			
283	42	644	96	358	49	57	587	978	59
56	584	82	94	37	23	931	864	22	97
55	350	348	873	82	36	931	33	42	71
464	9	33	983	15	22	475			
950	114	716	668	778	43	55	936	56	97
26	7	77	828	28	23	973	27	75	140
430	789	590	297	11	47	449	51	97	61
571	27	602	264	15	85	80			
830	720	809	31	70	82	688	241	82	860
7	99	81	36	85	12	868	18	49	2
740	719	72	221	408	61	996	638	150	72
586	654	519	112	320	13	777			
234	53	983	3	210	37	704	678	42	91
290	11	45	37	56	93	581	36	32	2
22	15	57	503	328	698	47	978	75	41
105	17	760	719	190	3	891			

796	37	619	93	67	23	47	306	744	67
978	32	214	3	53	218	303	5	22	31
884	700	36	724	58	71	845	504	81	96
509	9	293	71	75	59	916			
854	27	206	57	232	87	21	513	15	13
13	420	32	990	1	7	624	39	2	39
604	952	96	372	644	33	813	16	140	644
876	290	559	460	640	59	8			
206	36	650	210	755	520	33	827	100	818
538	85	894	57	39	218	12	65	7	16
46	566	92	944	39	16	143	53	174	27
320	284	382	721	878	678	318			
798	51	310	21	574	43	25	150	42	76
59	45	72	12	73	7	860	62	37	36
52	44	45	560	43	45	920	16	62	99
790	27	390	968	62	83	368			
85	51	23	26	444	426	92	529	66	554
97	93	85	62	19	99	99	476	604	95
7	868	760	580	234	11	17	75	51	96
716	2	37	202	267	91	69			
948	89	5	81	16	491	6	939	79	595
414	288	910	65	85	270	72	211	167	13
6	11	904	994	39	410	49	95	408	430
11	43	844	703	417	570	134			
1	63	3	63	418	402	664	379	458	222
35	62	508	29	804	338	21	930	720	37
750	43	96	846	1	553	51	7	210	75
653	834	35	184	393	2	5			
89	6	218	36	730	321	22	910	16	58
93	23	13	92	138	943	189	67	66	67
440	62	554	423	950	59	83	36	129	27
170	790	613	608	540					
66	654	46	69	71	769	82	178	15	586
46	59	22	788	700	3	624	542	343	13
21	230	51	882	72	967	34	748	760	83
339	87	13	156	327	65	524			
110	900	99	51	91	150	22	132	154	663
72	75	21	2	968	628	79	866	846	2
6	2	22	497	86	132	57	964	924	620
501	148	3	10	279	5	52			
934	26	96	37	488	89	6	695	82	541
710	66	46	37	26	39	53	855	268	95
5	408	88	56	564	869	95	89	66	82
71	56	7	398	464	41	370			

15	2	630	61	89	638	61	287	62	958
25	86	198	42	70	62	73	47	232	464
268	338	95	515	56	494	51	83	21	174
462	690	96	585	228	93	48			
97	290	72	844	330	570	13	22	65	236
31	780	27	14	83	6	16	641	482	504
554	36	47	480	75	267	2	62	824	97
374	848	478	134	496	754	47			
75	7	22	45	15	148	91	840	69	224
96	45	42	92	56	63	33	8	970	72
2	2	83	832	75	624	77	89	93	65
214	26	66	660	384	59	63			
76	62	464	292	634	340	278	254	818	37
928	53	888	55	3	29	83	598	60	32
35	36	69	46	644	33	15	14	198	42
73	770	95	648	814	45	727			
43	780	56	588	47	86	23	175	518	218
400	1	108	42	86	93	94	810	718	76
96	25	6	50	363	19	49	56	5	63
9	42	274	534	298	9	751			
864	75	99	554	99	800	6	543	85	3
55	45	69	554	77	704	670	49	63	53
214	948	718	67	154	47	418	86	1	674
56	65	13	1	56	19	635			
91	12	130	481	93	26	624	171	13	718
79	608	3	9	33	92	714	5	930	974
41	45	36	16	6	53	63	359	759	13
29	580	55	418	93	56	629			
25	19	864	535	498	3	27	931	21	11
7	27	73	6	2	800	36	7	130	441
248	22	86	79	475	9	9	2	800	694
93	7	254	996	930	744	45			
87	36	35	843	174	22	76	569	39	76
77	850	96	790	42	718	8	2	378	86
344	864	17	62	845	17	83	640	56	89
43	71	520	629	92	31	365			
42	59	33	671	47	364	854	382	570	26
2	654	95	610	924	89	21	46	87	178
42	59	918	96	969	65	194	644	818	804
964	630	97	356	634	21	812			
57	13	224	120	73	7	92	79	93	57
9	22	7	888	13	85	72	444	538	310
200	898	12	79	935	468	59	91	77	920
734	418	27	185	59	33	591			

12	93	63	490	7	63	870	623	52	99
87	634	39	284	338	1	49	52	6	81
86	87	52	61	482	23	750	674	384	114
804	23	83	704	174	99	911			
7	55	9	264	52	35	92	310	65	37
21	52	66	83	96	73	31	7	21	59
79	7	2	15	680	35	52	53	47	86
35	27	71	608	32	87	844			

CODE FPR 83 EXPERIMENTAL

ANNEXE 3

```
      SUBROUTINE REDUC(CC,AA,BB,M,N,MN,NBREDL,NBREDC,PASL,PASC,MU1,MU2,
     SEPS,EPS1,EPS2,RSUPL,KV)
C                 ****************************
C                 * ALGORITHME DE REDUCTION *
C                 ****************************
      INTEGER CC(N),AA(MN),BB(M)
      INTEGER CH(60),AH(1800),BH(30)
      DIMENSION W(30),WMIN(30),NUMAX(30)
      DIMENSION NUV(60),NUC(30),MAXC(30),Y(60),Y1(60),Y2(60)
      INTEGER XMIN(60),SOMC(30),SOME(60),X(60),X1(60)
      INTEGER SOMVAFI,SOMINT,Z,ZSP,QSORT,GAP
      INTEGER PASL,PASC,RSUPL
      LOGICAL ECHEC,REDUCTION,ECHEC1,FIN
C                          //INITIALISATION//
      CALL UCTIME(0,INTER)
      WRITE(108,93)
   93 FORMAT(//,30X,'INITIALISATION',/)
      NR=N;MR=M;MNR=MN
      WRITE(108,49) M,N
   49 FORMAT(//,5X,'DONNEES :',I3,' CONTRAINTES',5X,I3,' VARIABLES',//)
      WRITE(108,1) (CC(J),J=1,N)
      WRITE(108,1) (BB(I),I=1,M)
      DO 48 I=1,M
   48 WRITE(108,1) (AA(I+(J-1)*M),J=1,N)
    1 FORMAT(5X,15A6)
      WRITE(108,2)NBREDL,NBREDC,PASL,PASC,MU1,MU2,EPS,EPS1,EPS2,RSUPL,KV
    2 FORMAT(//,5X,'NBREDL,NBREDC,PASL,PASC,MU1,MU2,EPS,EPS1,EPS2,RSUPL,
     SKV:',6I5,3(F6.4,2X),I4,I8)
      MIN=KV;SOMVAFI=0;SOMINT=0;QSORT=0;IREDL=0;IREDC=0
      FIN=.FALSE.
      DO 43 I=1,M
      W(I)=0.;WMIN(I)=0.
   43 CONTINUE
      DO 46 J=1,N
      NUV(J)=J;X(J)=0
   46 XMIN(J)=0
      DO 47 I=1,M
      NUC(I)=I;BH(I)=BB(I)
   47 CONTINUE
C                              //PREPARATION REDUCTION STRUCTURE//
      WRITE(108,91)
   91 FORMAT(//,30X,'PREPARATION REDUCTION STRUCTURE',/)
      DO 50 I=1,M
      SOMC(I)=0
      DO 50 J=1,N
      K=I+(J-1)*M
   50 SOMC(I)=SOMC(I)+AA(K)
      DO 51 J=1,N
      SOME(J)=0
      K=(J-1)*M
      DO 51 I=1,M
```

```
      IF(AA(I+K).NE.0) SOME(J)=SOME(J)+1
   51 CONTINUE
      DO 52 I=1,M
      MAXC(I)=AA(I);NUMAX(I)=1
      DO 52 J=2,N
      K=I+(J-1)*M
      IF(AA(K).GT.MAXC(I)) MAXC(I)=AA(K);NUMAX(I)=J
   52 CONTINUE
      WRITE(108,38) (SOMC(I),I=1,M)
      WRITE(108,39) (SOME(J),J=1,N)
      WRITE(108,58) (MAXC(I),I=1,M)
      WRITE(108,59) (NUMAX(I),I=1,M)
   38 FORMAT(//,5X,'SOMME ELEMENTS PAR CONTRAINTE ',15I8)
   39 FORMAT(//,5X,'NB ELEMENTS NON NULS PAR COLONNE ',15I4)
   58 FORMAT(//,5X,'PLUS GRAND ELEMENT PAR CONTRAINTE.',15I8)
   59 FORMAT(//,5X,'NUMERO DU PLUS GRAND ELEMENT PAR CONTRAINTE ',15I4)
      R=0.
      DO 53 J=1,N
   53 R=R+SOME(J)
      R1=MN;R=R/R1; WRITE(108,3)R
    3 FORMAT(//,5X,'DENSITE DE LA MATRICE :',F5.3)
      DO 54 I=1,M
      R=BB(I);R=SOMC(I)/R; WRITE(108,4)I,R
   54 CONTINUE
    4 FORMAT(//,5X,'COEFFICIENT DE RELACHEMENT DE LA CONTRAINTE ',I2,':',
     SF6.3)
C                          //PHASE PREPROCESSING//
      WRITE(108,80)
   80 FORMAT(//,30X,'PREPROCESSING')
      NR0=NR
      DO 82 K=1,MN
   82 AH(K)=AA(K)
      JR0=100;JR1=100
      CALL PREPROCES(AH,BH,MR,NR,MNR,JR0,JR1,NUV,NUC,W,SOMC,MAXC,SOME,
     SWMIN,NUMAX)
      JR0=0;JR1=0
      IF(NR.EQ.NR0) GOTO 94
      DO 89 J=NR+1,NR0
      J0=NUV(J);IF(J0.GT.0) SOMVAFI=SOMVAFI+CC(J0)
   89 CONTINUE
C                              //CALCUL DU MINORANT//
   94 WRITE(108,92)
   92 FORMAT(//,30X,'CALCUL PREMIER MINORANT PAR AGNES1',//)
      N1=0
      DO 62 J=1,NR
      J0=NUV(J)
   62 CH(J)=CC(J0)
      DO 60 J=1,NR
      J0=NUV(J);K=(J-1)*MR;K0=(J0-1)*M
      DO 60 I=1,MR
      I0=NUC(I);L=J+K;L0=I0+K0
```

```
      60 AH(L)=AA(LO)
         DO 70 KDA=1,41,20
         R=KDA;BETA=1.-(21.-R)/100.
         CALL AGNES1(CH,AH,BH,MR,NR,MNR,X,Z,BETA,W,N1,SOMC,SUME)
         Z=Z+SOMVAFI; IF(Z.LE.MIN) GOTO 70
         MIN=Z
         DO 72 J=1,NR
         JO=NUV(J)
      72 XMIN(JO)=X(J)
         DO 74 J=NR+1,N
         JO=ABS(NUV(J));IF(NUV(J).GT.0) XMIN(JO)=1;GOTO 74
         XMIN(JO)=0
      74 CONTINUE
      70 CONTINUE
         WRITE(108,8)IREDL,MIN
       8 FORMAT(//,20X,'REDUCTION ',I2,' VALEUR DU MINORANT ',I9)
         WRITE(108,9)((J,XMIN(J)),J=1,N)
       9 FORMAT(//,5X,'(NUM VARIABLE ,SOLUTION) ',10(2X,'(',I2,',',I1,')'))
C                //REDUCTION PAR METHODE DE S/S GRADIENTS//
         WRITE(108,5)
       5 FORMAT(//,30X,'REDUCTION PAR METHODE S/S GRADIENT')
C                //GENERATION MULTIPLICATEUR LAGRANGIEN//
         WRITE(108,6) NBREDL,PASL
       6 FORMAT(//,5X,'CALCUL DU MULTIPLICATEUR PENDANT ',I3,' FOIS',I4,' IT
      SERATIONS')
    1000 IREDL=IREDL+1
         IF(IREDL.GT.NBREDL) IREDL=0;GOTO 2100
         NITER=PASL;REDUCTION=.FALSE.;MNR=MR*NR
         ZSP=MIN-SOMVAFI
         DO 90 J=1,NR
         JO=NUV(J);X1(J)=XMIN(JO);CH(J)=CC(JO)
      90 CONTINUE
         DO 110 J=1,NK
         JO=NUV(J);K=(J-1)*MR;KO=(JO-1)*M
         DO 110I=1,MR
         IO=NUC(I);L=I+K;LO=IO+KO
     110 AH(L)=AA(LO)
         CALL REDULAG(CH,AH,BH,MR,NR,MNR,ZSP,X1,W,WMIN,NITER,SOMVAFI,EPS1)
         WRITE(108,10)((NUC(I),WMIN(I)),I=1,MR)
      10 FORMAT(//,5X,'(NUM CONT ,MEIL MULT) ',5(2X,'(',I2,',',F9.5,')'))
         Z=ZSP+SOMVAFI;IF(Z.LE.MIN) GOTO 910
         MIN=Z
         DO 165 J=1,NK
         JO=NUV(J);XMIN(JO)=X1(J)
     165 CONTINUE
         DO170 J=NR+1,N
         JO=ABS(NUV(J));IF(NUV(J).GT.0) XMIN(JO)=1; GOTO 170
         XMIN(JO)=0
     170 CONTINUE
         WRITE(108,11) IREDL,MIN; WRITE(108,9)((J,XMIN(J)),J=1,N)
      11 FORMAT(//,20X,'REDUCTION ',I2,'NOUVELLE VALEUR DU MINORANT',I9)
```

```
      C                //...IORATION EVENTUELLE DU MINORANT PAR ...//
  910 KFRAC=3;ECHEC1=.TRUE.
  940 ECHEC=.FALSE.
      CALL AGNES2(CH,AH,BH,MR,NR,MNR,X,Z,KFRAC,WMIN,SOMC,SOME,ECHEC)
      IF(.NOT.ECHEC) ECHEC1=ECHEC
      Z=Z+SOMVAFI;IF(Z.LE.MIN) GOTO 945
      MIN=Z
      DO 942 J=1,NR
      J0=NUV(J);XMIN(J0)=X(J)
  942 CONTINUE
      DO 944 J=NR+1,N
      J0=ABS(NUV(J));IF(NUV(J).GT.0) XMIN(J0)=1;GOTO 944
      XMIN(J0)=0
  944 CONTINUE
      WRITE(108,11) IREDL,MIN;WRITE(108,9) ((J,XMIN(J)),J=1,N)
  945 KFRAC=KFRAC+3;KAR=2*KFRAC
      IF(KFRAC.LE.9.AND.NR.GE.KAR) GOTO 940
      IF(.NOT.ECHEC1) ITERED=IREDL;NBRED=NBREDL;GOTO 200
      DO 946 KDA=1,41,20
      R=KDA;BETA=1.-(21.-R)/100.
      CALL AGNES1(CH,AH,BH,MR,NR,MNR,X,Z,BETA,WMIN,IREDL,SOMC,SOME)
      Z=Z+SOMVAFI;IF(Z.LE.MIN) GOTO 946
      MIN=Z
      DO 947 J=1,NR
      J0=NUV(J);XMIN(J0)=X(J)
  947 CONTINUE
      DO 948 J=NR+1,N
      J0=ABS(NUV(J));IF(NUV(J).GT.0) XMIN(J0)=1;GOTO 948
      XMIN(J0)=0
  948 CONTINUE
      WRITE(108,11) IREDL,MIN;WRITE(108,9) ((J,XMIN(J)),J=1,N)
  946 CONTINUE
      ITERED=IREDL;NBRED=NBREDL;GOTO 200
 1001 IF(REDUCTION) GOTO 1000
      IREDL=0
      C                //REDUCTION PAR METHODE QUASI-S/S-GRADIENTS//
 2100 WRITE(108,967)
  967 FORMAT(/,30X,'REDUCTION PAR METHODE QUASI-S/S-GRADIENTS')
      C                //GENERATION MULTIPLICATEUR COMPOSITE//
      WRITE(108,6) NBREDC,PASC
 2000 IREDC=IREDC+1
      IF(IREDC.GT.NBREDC) GOTO 5500
      NITER=PASC;REDUCTION=.FALSE.;MNR=MR*NR
      DO 952 J=1,NR
      J0=NUV(J);CH(J)=CC(J0)
  952 CONTINUE
      DO 954 J=1,NR
      J0=NUV(J);K=(J-1)*MR;K0=(J0-1)*M
      DO 954 I=1,MR
      I0=NUC(I);L=I+K;L0=I0+K0
  954 AH(L)=AA(L0)
```

```
      CALL REDUCOM(CH,AH,BH,MR,NR,MNR,W,WMIN,NITER,SOMVAFI,EPS2)
      WRITE(108,10) ((NUC(I),WMIN(I)),I=1,MR)
C                    //AMELIORATION EVENTUELLE DU MINORANT PAR AGNES2//
      KFRAC=3;ECHEC1=.TRUE.
  540 ECHEC=.FALSE.
      CALL AGNES2(CH,AH,BH,MR,NR,MNR,X,Z,KFRAC,WMIN,SOMC,SUME,ECHEC)
      IF(.NOT.ECHEC) ECHEC1=ECHEC
      Z=Z+SOMVAFI;IF(Z.LE.MIN) GOTO 545
      MIN=Z
      DO 542 J=1,NK
      J0=NUV(J);XMIN(J0)=X(J)
  542 CONTINUE
      DO 544 J=NR+1,N
      J0=ABS(NUV(J));IF(NUV(J).GT.0) XMIN(J0)=1;GOTO544
      XMIN(J0)=0
  544 CONTINUE
      WRITE(108,11) IREDC,MIN;WRITE(108,9) ((J,XMIN(J)),J=1,N)
  545 KFRAC=KFRAC+3;KAR=2*KFRAC
      IF(KFRAC.LE.9.AND.NR.GE.KAR) GOTO 540
      IF(.NOT.ECHEC1) ITERED=IREDC;NBRED=NBREDC;KCOMB=0;GOTO 300
      DO 546 KDA=1,41,20
      R=KDA;BETA=1.-(21.-R)/100.
      CALL AGNES1(CH,AH,BH,MR,NR,MNR,X,Z,WMIN,IREDC,SOMC,SUME)
      Z=Z+SOMVAFI;IF(Z.LE.MIN) GOTO 546
      MIN=Z
      DO 547 J=1,NK
      J0=NUV(J);XMIN(J0)=X(J)
  547 CONTINUE
      DO 548 J=NR+1,N
      J0=ABS(NUV(J));IF(NUV(J).GT.0) XMIN(J0)=1;GOTO 548
      XMIN(J0)=0
  548 CONTINUE
      WRITE(108,11) IREDC,MIN;WRITE(108,9) ((J,XMIN(J)),J=1,N)
  546 CONTINUE
      ITERED=IREDC;NBRED=NBREDC;KCOMB=0;GOTO 300
 2001 IF(REDUCTION) IREDC=IREDC-1;GOTO 2000
      GOTO 5000
C                  //PHASE REDUCTION//
C                 //ELIMINATION DE VARIABLES//
C                       //TEST V1//
  200 KCOMB=0
 1300 IF(KCOMB-1) 65,300,600
   65 CONTINUE
      WRITE(108,64)
   64 FORMAT(//,5X,'ELIMINATION S/S CONTRAINTE COMPOSITE')
      WRITE(108,95)
   95 FORMAT(///,30X,'TEST V1',//)
      SUP1=0.
      DO 201 I=1,MK
  201 SUP1=SUP1+BH(I)*WMIN(I)
      DO 205 J=1,NK
```

```
      R=0.;J0=NUV(J);K0=J0+K0
      DO 208 I=1,MR
      I0=NUC(I);K=I0+K0
  208 R=R+AA(K)*WMIN(I)
      Y(J)=CC(J0)-R;IF(Y(J).GT.0.) SUP1=SUP1+Y(J)
  205 CONTINUE
      WRITE(108,181) SUP1
      WRITE(108,182) (NUV(J),J=1,NR)
      WRITE(108,183) (Y(J),J=1,NR)
  181 FORMAT(5X,'VALEUR RELAXATION = ',F12.4)
  182 FORMAT(5X,'NUMERO ',15I6)
  183 FORMAT(5X,'COUTS REDUITS  ',10F10.3)
      JR=0;JR0=0;JR1=0;SOMINT=SOMVAFI
      GAP=INT(SUP1+SOMVAFI-MIN)
      WRITE(108,190) GAP
  190 FORMAT(5X,'GAP  ',I5)
      DO 210 J=1,NR
      KTEST=INT(SUP1-ABS(Y(J)))+SOMINT;IF(KTEST.GE.MIN) GOTO 210
      JR=JR+1
      IF(Y(J).GE.0.) X(JR)=J;J0=NUV(J);SOMVAFI=SOMVAFI+CC(J0);GOTO 210
      X(JR)=-J;JR0=JR0+1
  210 CONTINUE
      WRITE(108,120) SOMVAFI
  120 FORMAT(5X,'CHARGE FIXE =',I8)
      JR1=JR-JR0
      IF(JR.EQ.0) GOTO 300
      REDUCTION=.TRUE.
      DO 220 L=JR,1,-1
      K=ABS(X(L))
      IPV=NUV(K);NUV(K)=NUV(NR);NUV(NR)=IPV; SOME(K)=SOME(NR)
      IF(X(L).LT.0) NUV(NR)=-NUV(NR)
  220 NR=NR-1
      DO 230 K=NR+1,NR+JR
      J0=NUV(K)
      IF(J0.LT.0) J0=-J0;GOTO 235
      DO 232 I=1,MR
      I0=NUC(I);K0=I0+(J0-1)*M
  232 BH(I)=BH(I)-AA(K0)
  235 DO 234 I=1,MR
      I0=NUC(I);K0=I0+(J0-1)*M
  234 SOMC(I)=SOMC(I)-AA(K0)
  230 CONTINUE
      DO 240 I=1,MR
      I0=NUC(I)
      DO 242 J=NR+1,NR+JR
      J0=ABS(NUV(J));K0=I0+(J0-1)*M
      IF(AA(K0).EQ.MAXC(I)) GOTO 245
  242 CONTINUE
      GOTO 240
  245 J1=NUV(1);K1=I0+(J1-1)*M
      MAXC(I)=AA(K1);NUMAX(I)=J1
```

```
      DO 246 J=1,NR
      J0=NUV(J);K0=I0+(J0-1)*M
      IF(AA(K0).GT.MAXC(I)) MAXC(I)=AA(K0);NUMAX(I)=J0
  246 CONTINUE
  240 CONTINUE
      WRITE(108,13) ITERED,(NUV(J),J=NR+1,NR+JR)
   13 FORMAT(/,5X,'REDUCTION N°',I2,' TEST V1 : ',25I4)
      IF(NR.EQ.0) GOTO 6000
      DO 255 J=1,NR
      J0=NUV(J);K=(J-1)*MR;K0=(J0-1)*M
      DO 255 I=1,MR
      I0=NUC(I);L=I+K;L0=I0+K0
  255 AH(L)=AA(L0)
      NR0=NR
      CALL PREPROCES  (AH,BH,MR,NR,MNR,JR0,JR1,NUV,NUC,W,SOMC,MAXC,SOME,
     SWMIN,NUMAX)
      IF(NR.EQ.NR0) GOTO 270
      DO 260 J=NR+1,NR0
      J0=NUV(J);IF(J0.GT.0) SOMVAFI=SOMVAFI+CC(J0)
  260 CONTINUE
      WRITE(108,120) SOMVAFI
      IF(NR.EQ.0) GOTO 6000
  270 IF(MR.EQ.1) GOTO 5500
C                    //TEST V2//
  300 IF(KCOMB.EQ.0) GOTO 301
      MAXW=1,R=WMIN(1)
      DO 262 I=2,MR
      IF(WMIN(I).GT.R) R=WMIN(I);MAXW=I
  262 CONTINUE
      WRITE(108,63) NUC(MAXW)
   63 FORMAT(/,5X,'ELIMINATION S/S LA CONTRAINTE ',I3)
  301 CONTINUE
      WRITE(108,96)
   96 FORMAT(//,30X,'TEST V2',/)
      IF(KCOMB.EQ.1) GOTO 302
      DO 310 J=1,NR
      RS=0.;J0=NUV(J);K0=(J0-1)*M
      DO 315 I=1,MR
      I0=NUC(I);K=I0+K0
  315 RS=RS+AA(K)*WMIN(I)
  310 Y(J)=RS
      GOTO 303
  302 ICMAX=NUC(MAXW)
      DO 304 J=1,NR
      J0=NUV(J);K=ICMAX+(J0-1)*M
  304 Y(J)=AA(K)
  303 IF(KCOMB.EQ.1) RS=BH(MAXW);GOTO 306
      RS=0.
      DO 320 I=1,MR
  320 RS=RS+BH(I)*WMIN(I)
  306 DO 322 J=1,NR
```

```
      JU=NUV(J);CH(J)=CC(JU);X(J)=J
  322 CONTINUE
      N1=1
      CALL RESOL(CH,Y,RS,X,N1,NR,WOPT,QSORT)
      SUP2=0.
      DO 330 J=1,N1-1
  330 SUP2=SUP2+CH(J)
      SUP3=SUP2;SUP2=SUP2+WOPT*RS
      K=X(N1);JVBAS=NUV(K)
      IF(QSORT.EQ.1) GOTO 340
      N2=N1-1;Y3=CH(N2)/Y(N2)
      IF(N1.EQ.NR) VB1=0;GOTO 362
      N2=N1+1;Y4=CH(N2)/Y(N2);GOTO 360
  340 N2=N1-1; Y3=CH(N2)/Y(N2)
      DO 345 J=N2,1,-1
      R1=CH(J)/Y(J);IF(R1.LT.Y3) Y3=R1
  345 CONTINUE
      IF(N1.EQ.NR) VB1=0;GOTO 362
      N2=N1+1; Y4=CH(N2)/Y(N2)
      DO 350 J=N2,NR
      R1=CH(J)/Y(J);IF(R1.GT.Y4) Y4=R1
  350 CONTINUE
  360 VB1=SUP3+Y4*RS
  362 VB2=SUP3+CH(N1)+Y3*(RS-Y(N1))
      WRITE(108,181) SUP2
      WRITE(108,184) VB1,VB2
  184 FORMAT(5X,'MAJORANTS QUAND LA VARIABLE EN BASE EST FIXEE A 0 PUIS
     S 1',2F12.4)
      IF(VB1.LT.VB2) VB1=VB2
      DO 370 J=1,NR
  370 Y(J)=CH(J)-Y(J)*WOPT
      WRITE(108,182) (NUV(J),J=1,NR)
      WRITE(108,189) (X(J),J=1,NR)
      WRITE(108,183) (Y(J),J=1,NR)
  189 FORMAT(5X,'CHAINAGE ',10I6)
      JR=0;JR0=0;JR1=0;SOMINT=SOMVAFI
      GAP=INT(SUP2+SOMVAFI-MIN)
      WRITE(108,190) GAP
      DO 380 J=1,NR
      K=X(J);KTEST=INT(SUP2-ABS(Y(J)))+SOMINT;IF(KTEST.GE.MIN) GOTO 380
      JR=JR+1
      IF(Y(J).GE.0) X(JR)=K;JU=NUV(K);SOMVAFI=SOMVAFI+CC(JU); GOTO 380
      X(JR)=-K; JR0=JR0+1
  380 CONTINUE
      WRITE(108,120) SOMVAFI
      WRITE(108,189) (X(J),J=1,JR)
      JR1=JR-JR0
      IF(JR.EQ.0) GOTO 400
      REDUCTION=.TRUE.
      DO 335 L=JR,1,-1
      K=ABS(X(L))
```

```
      IPV=NUV(K);NUV(K)=NUV(NR); NUV(NR)=IPV; SOME(K)=SOME(NR)
      IF(X(L).LT.0) NUV(NR)=-NUV(NR)
      L1=L
      DO 356 L2=1,L1
      IF(ABS(X(L2)).EQ.NR) X(L2)=X(L)
  356 CONTINUE
  335 NR=NR-1
      WRITE(108,184) (NUV(J),J=1,NR+JR)
      DO 355 K=NR+1,NR+JR
      J0=NUV(K)
      IF(J0.LT.0) J0=-J0;GOTO 342
      DO 348 I=1,MR
      I0=NUC(I);K0=I0+(J0-1)*M
  348 BH(I)=BH(I)-AA(K0)
  342 DO 352 I=1,MR
      I0=NUC(I);K0=I0+(J0-1)*M
  352 SOMC(I)=SOMC(I)-AA(K0)
  355 CONTINUE
      DO 365 I=1,MR
      I0=NUC(I)
      DO 368 J=NR+1,NR+JR
      J0=ABS(NUV(J));K0=I0+(J0-1)*M
      IF(AA(K0).EQ.MAXC(I)) GOTO 372
  368 CONTINUE
      GOTO 365
  372 J1=NUV(1);K1=I0+(J1-1)*M
      MAXC(I)=AA(K1);NUMAX(I)=J1
      DO 374 J=1,NR
      J0=NUV(J);K0=I0+(J0-1)*M
      IF(AA(K0).GT.MAXC(I)) MAXC(I)=AA(K0);NUMAX(I)=J0
  374 CONTINUE
  365 CONTINUE
      WRITE(108,15) ITERED,(NUV(J),J=NR+1,NR+JR)
   15 FORMAT(/,5X,'REDUCTION N°',I2,' TEST V2 : ',25I4,/)
      IF(NR.EQ.0) GOTO 6000
      DO 390 J=1,NR
      J0=NUV(J); K=(J-1)*MR;K0=(J0-1)*M
      DO 390 I=1,MR
      I0=NUC(I);L=I+K;L0=I0+K0
  390 AH(L)=AA(L0)
      NR0=NR
      CALL PREPROCES  (AH,BH,MR,NR,MNR,JR0,JR1,NUV,NUC,W,SOMC,MAXC,SOME,
     SWMIN,NUMAX)
      IF(NR.EQ.NR0) GOTO 395
      DO 392 J=NR+1,NR0
      J0=NUV(J); IF(J0.GT.0) SOMVAFI=SOMVAFI+CC(J0)
  392 CONTINUE
      WRITE(108,120) SOMVAFI
      IF(NR.EQ.0) GOTO 6000
  395 IF(MR.EQ.1) GOTO 5500
C                    //TEST DECISION//
```

```
      400      NTO=MU1*(INT(SUP2-VB1)+1)
      461.        WRITE(108,97)
      462.     97 FORMAT(//,30X,'TEST DECISION',//)
      463.        WRITE(108,160) NTO,GAP
      464.    160 FORMAT(5X,'ON ENCLANCHE TESTS V3/V4 SI ',I6,' SUPERIEUR A ',I6)
      465.        IF(NTO.LT.GAP) GOTO 600
      466.  C                    //TEST V3,V4//
      467.        IF(KCOMB.EQ.0) GOTO 401
      468.        MAXW=1;R=WMIN(1)
      469.        DO 402 I=2,MR
      470.        IF(WMIN(I).GT.R) R=WMIN(I);MAXW=I
      471.    402 CONTINUE
      472.    401 CONTINUE
      473.        WRITE(108,98)
      474.     98 FORMAT(//,30X,'TEST V3,V4',//)
      475.        DO 405 J=1,NR
      476.        IF(JVBAS.NE.NUV(J)) GOTO 405
      477.        NUV(J)=NUV(NR); NUV(NR)=JVBAS
      478.        IPV=SOME(J);SOME(J)=SOME(NR);SOME(NR)=IPV; GOTO 410
      479.    405 CONTINUE
      480.        WRITE(108,16); STOP
      481.     16 FORMAT(/,5X,'VARIABLES EN BASE NON DETECTEE')
      482.    410 IF(KCOMB.EQ.1) GOTO 403
      483.        DO 412 J=1,NR-1
      484.        RS=0.;JO=NUV(J);KO=(JO-1)*N
      485.        DO 414 I=1,MR
      486.        IO=NUC(I);K=IO+KO
      487.    414 RS=RS+AA(K)*WMIN(I)
      488.        Y(J)=RS; Y1(J)=RS
      489.    412 CONTINUE
      490.        GOTO 404
      491.    403 ICMAX=NUC(MAXW)
      492.        DO 406 J=1,NR-1
      493.        JO=NUV(J);K=ICMAX+(JO-1)*M
      494.        Y(J)=AA(K)
      495.    406 Y1(J)=AA(K)
      496.    404 DO 416 J=1,NR-1
      497.        JO=NUV(J);CH(J)=CC(JO);AH(J)=CH(J);X(J)=J
      498.    416 CONTINUE
      499.        IF(KCOMB.EQ.1)RSO=BH(MAXW);K=ICMAX+(JVBAS-1)*M;RS1=RSO-AA(K);
      500.       SGOTO 407
      501.        RSO=RS1=0.
      502.        DO 418 I=1,MR
      503.    418 RSO=RSO+BH(I)*WMIN(I)
      504.        DO 420 I=1,MR
      505.        IO=NUC(I);K=IO+(JVBAS-1)*M
      506.    420 RS1=RS1+(BH(I)-AA(K))*WMIN(I)
      507.    407 NRO=NR-1;N1=1
      508.        CALL RESOL(CH,Y,RSO,X,N1,NRO,WOPT,QSORT)
      509.        SUPO=0.
      510.        DO 425 J=1,N1-1
```

```
      425 SUP0=SUP0+CH(J)
          SUP0=SUP0+WOPT*RS0
          DO 428 J=1,NR0
      428 Y(J)=CH(J)-Y(J)*WOPT
          DO 430 J=1,NR0
          K=X(J); Y2(K)=Y(J)
      430 CONTINUE
          DO 436 J=1,NR0
          Y(J)=Y2(J);X(J)=J
      436 CONTINUE
          N1=1
          CALL RESOL(AH,Y1,RS1,X,N1,NR0,WOPT,QSORT)
          SUP1=CC(JVBAS)
          IF(N1.EQ.1) GOTO 439
          DO 438 J=1,N1-1
      438 SUP1=SUP1+AH(J)
      439 SUP1=SUP1+WOPT*RS1
          DO 440 J=1,NR0
      440 Y1(J)=AH(J)-Y1(J)*WOPT
          DO 442 J=1,NR0
          K=X(J); Y2(K)=Y1(J)
      442 CONTINUE
          DO 444 J=1,NR0
      444 Y1(J)=Y2(J)
          WRITE(108,184) (NUV(J),J=1,NR)
          WRITE(108,186) SUP0,SUP1
          WRITE(108,188) (Y(J),J=1,NR-1)
          WRITE(108,187) (Y1(J),J=1,NR-1)
      186 FORMAT(5X,'VALEURS RELAXATIONS,VARIABLE EN BASE FIXEE A 0 PUIS 1'
         S,2F12.4)
      187 FORMAT(5X,'COUTS REDUITS(1) ',10F10.3)
      188 FORMAT(5X,'COUTS REDUITS(0) ',10F10.3)
          JR=JR0=JR1=0; IR0=IR1=0
          GAP=INT(SUP0+SOMVAFI-MIN)
          WRITE(108,197) GAP
          GAP=INT(SUP1+SOMVAFI-MIN)
          WRITE(108,198) GAP
      548 KTEST=INT(SUP0)+SOMVAFI; IF(KTEST.GE.MIN) GOTO 450
      549 REDUCTION=.TRUE.
      550 DO 446 I=1,MK
      551 I0=NUC(I);K=I0+(JVBAS-1)*M
      552 446 BH(I)=BH(I)-AA(K)
      553 SOMVAFI=SOMVAFI+CC(JVBAS); IR1=1; GOTO 460
      554 450 KTEST=INT(SUP1)+SOMVAFI; IF(KTEST.GE.MIN) GOTO 500
      555 IR0=1; NUV(NR)=-NUV(NR)
      556 460 NR=NR0
      557 DO 448 I=1,MK
      558 I0=NUC(I);K=I0+(JVBAS-1)*M
      559 448 SOMC(I)=SOMC(I)-AA(K)
      560 DO 452 I=1,MK
      561 I0=NUC(I);K=I0+(JVBAS-1)*M
```

```
      IF(AA(K).NE.MAXC(I)) GOTO 452
      J1=NUV(1);K0=I0+(J1-1)*M
      MAXC(I)=AA(K0);NUMAX(I)=J1
      DO 454 J=1,NR
      J0=NUV(J);L=I0+(J0-1)*M
      IF(AA(L).GT.MAXC(I)) MAXC(I)=AA(L);NUMAX(I)=J0
  454 CONTINUE
  452 CONTINUE
      WRITE(108,17)ITERED,NUV(NR+1)
   17 FORMAT(//,5X,'REDUCTION N',I2,' TEST V3 : ',I4,//)
      IF(IR1.EQ.0) GOTO 462
      DO 455 J=1,NR
  455 Y(J)=Y1(J)
      SUP0=SUP1
  462 SOMINT=SOMVAFI
      DO 465 J=1,NR
      KTEST=INT(SUP0-ABS(Y(J)))+SOMINT; IF(KTEST.GE.MIN) GOTO 465
      JR=JR+1
      IF(Y(J).GE.0) X(JR)=J; J0=NUV(J);SOMVAFI=SOMVAFI+CC(J0); GOTO 465
      X(JR)=-J; JR0=JR0+1
  465 CONTINUE
      WRITE(108,120) SOMVAFI
      JR1=JR-JR0; IF(JR.EQ.0) GOTO 580
      GOTO 550
  500 IPV=NUV(NR); NUV(NR)=NUV(1); NUV(1)=IPV
      IPV=SOME(NR);SOME(NR)=SOME(1);SOME(1)=IPV
      Y(NR)=Y(1); Y1(NR)=Y1(1);NR0=NR-1;SOMINT=SOMVAFI
  197 FORMAT(5X,'GAP:VARIABLE EN BASE A 0;',I6)
  198 FORMAT(5X,'GAP:VARIABLE EN BASE A 1;',I6)
      DO 470 J=2,NR
      KTEST=INT(SUP0-ABS(Y(J)))+SOMINT
      K=INT(SUP1-ABS(Y1(J)))+SOMINT; IF(K.GT.KTEST) KTEST=K; Y(J)=Y1(J)
      IF(KTEST.GE.MIN) GOTO 470
      JR=JR+1
      IF(Y(J).GE.0) X(JR)=J; J0=NUV(J); SOMVAFI=SOMVAFI+CC(J0); GOTO 470
      X(JR)=-J; JR0=JR0+1
  470 CONTINUE
      WRITE(108,120) SOMVAFI
      JR1=JR-JR0
      IF(JR.EQ.0) GOTO 599
      REDUCTION=.TRUE.
  550 DO 510 L=JR,1,-1
      K=ABS(X(L))
      IPV=NUV(K);NUV(K)=NUV(NR);NUV(NR)=IPV; SOME(K)=SOME(NR)
      IF(X(L).LT.0) NUV(NR)=-NUV(NR)
      L1=L
      DO 511 L2=1,L1
      IF(ABS(X(L2)).EQ.NR) X(L2)=X(L)
  511 CONTINUE
  510 NR=NR-1
      DO 520 L=NR+1,NR+JR
```

```
      JO=NUV(L)
      IF(JO.LT.0) JO=-JO;GOTO 525
      DO 522 I=1,MR
      IO=NUC(I);K=IO+(JO-1)*M
  522 BH(I)=BH(I)-AA(K)
  525 DO 524 I=1,MR
      IO=NUC(I);K=IO+(JO-1)*M
  524 SOMC(I)=SOMC(I)-AA(K)
  520 CONTINUE
      DO 530 I=1,MR
      IO=NUC(I)
      DO 532 J=NR+1,NR+JR
      JO=ABS(NUV(J));K=IO+(JO-1)*M
      IF(AA(K).EQ.MAXC(I)) GOTO 535
  532 CONTINUE
      GOTO 530
  535 J1=NUV(1);K0=IO+(J1-1)*M
      MAXC(I)=AA(K0);NUMAX(I)=J1
      DO 536 J=1,NR
      JO=NUV(J);L=IO+(JO-1)*M
      IF(AA(L).GT.MAXC(I)) MAXC(I)=AA(L);NUMAX(I)=JO
  536 CONTINUE
  530 CONTINUE
      WRITE(108,19) ITERED,(NUV(J),J=NR+1,NR+JR)
   19 FORMAT(//,5X,'REDUCTION N°',I2,' TEST V4 : ',25I4)
      IF(NR.EQ.0) GOTO 6000
  580 JR0=JR0+IR0; JR1=JR1+IR1
      DO 585 J=1,NR
      JO=NUV(J); K=(J-1)*MR;KU=(JO-1)*M
      DO 585 I=1,MR
      IO=NUC(I);L=I+K;L0=IO+KU
  585 AH(L)=AA(L0)
      NR0=NR
      CALL PREPROCES  (AH,BH,MR,NR,MNR,JR0,JR1,NUV,NUC,W,SOMC,MAXC,SOME,
     SWMIN,NUMAX)
      IF(NR.EQ.NR0) GOTO 590
      DO 582 J=NR+1,NR0
      JO=NUV(J);IF(JO.GT.0) SOMVAFI=SOMVAFI+CC(JO)
  582 CONTINUE
      WRITE(108,120) SOMVAFI
      IF (NR.EQ.0) GOTO 6000
  590 IF(MR.EQ.1) GOTO 5500
  599 KCOMB=KCOMB+1;GOTO 1300
C           //ELIMINATION DE CONTRAINTES//
C                 //TEST DE DECISION//
  600 ICR=0
      WRITE(108,99)
   99 FORMAT(//,30X,'TEST DECISION ELIMINATION CONTRAINTE',/)
      MAXW=1;R=WMIN(1)
      DO 615 I=1,MR
      IF(WMIN(I).GT.R) R=WMIN(I);MAXW=I
```

194

```
      615 CONTINUE
          DO 618 I=1,MK
          IF(WMIN(I).LT.R) ICR=ICR+1;X1(ICR)=I
      618 CONTINUE
          IF(.NOT.REDUCTION) GOTO 605
          IF(ITERED.LT.NBRED) GOTO 612
      605 CONTINUE
          WRITE(108,972)
      972 FORMAT(//,5X,'ESSAI ELIMINATION CONTRAINTES PAR TEST C2/C4/C6')
          GOTO 1800
      612 IF(MK.LE.10) GOTO 1700
          ICR=0
          DO 610 I=1,MK
          IF(WMIN(I).LE.EPS) ICR=ICR+1;X1(ICR)=I
      610 CONTINUE
          IF(ICR.EQ.0) GOTO 1900
C                //TEST C1,C3,C5//
C                    //TEST C1//
     1700 DO 122 I=1,ICR
          K=X1(I);K0=NUC(K)
      122 X(I)=K0
          WRITE(108,123) (X(I),I=1,ICR)
      123 FORMAT(5X,'CONTRAINTES SUSCEPTIBLES D ETRE ELIMINEES :',10I6)
          ICMAX=NUC(MAXW)
          DO 620 J=1,NK
          J0=NUV(J);K0=ICMAX+(J0-1)*M
      620 Y2(J)=AA(K0)
          IR=0
          DO 700 IC1=1,ICR
          WRITE(108,101)
      101 FORMAT(//,30X,'TEST C1 ',/)
          RS=BH(MAXW); N1=1
          I1=X1(IC1); ICMIN=NUC(I1)
          DO 625 J=1,NK
          J0=NUV(J);K0=ICMIN+(J0-1)*M
          X(J)=J
          Y(J)=Y2(J)
      625 CH(J)=AA(K0)
          CALL RESOL(CH,Y,RS,X,N1,NR,WOPT,QSORT)
          SUP1=0.
          IF(N1.EQ.1) GOTO 631
          DO 630 J=1,N1-1
      630 SUP1=SUP1+CH(J)
      631 SUP2=SUP1;SUP1=SUP1+WOPT*RS
          WRITE(108,194) NUC(I1),SUP1,BH(I1)
      194 FORMAT(5X,'CONTRAINTE ',I3,' VALEUR RELAXATION=',F12.3,' SECOND
         SMENBRE ',I5)
          IF(INT(SUP1)-BH(I1)) 635,635,640
      635 IR=IR+1
          WRITE(108,20) ITERED ,NUC(I1)
       20 FORMAT(/,5X,'REDUCTION N ',I2,' TEST C1 : ',I2)
```

```
      X1(IC1)=-I1; GOTO 700
  640 IF(QSORT.EQ.2) GOTO 700
      K=X(N1); JVBAS=NUV(K)
      IF(QSORT.EQ.1) GOTO 645
      IF(N1.EQ.1) VB2=0.;GOTO 632
      N2=N1-1; Y3=CH(N2)/Y(N2);VB2=SUP2+CH(N1)+Y3*(RS-Y(N1))
  632 IF(N1.EQ.NR) VB1=0.;GOTO 650
      N2=N1+1;Y4=CH(N2)/Y(N2);VB1=SUP2+Y4*RS;GOTO 650
  645 IF(N1.EQ.1) VB2=0;GOTO 633
      N2=N1-1; Y3=CH(N2)/Y(N2)
      DO 648 J=N2,1,-1
      R1=CH(J)/Y(J); IF(R1.LT.Y3) Y3=R1
  648 CONTINUE
      VB2=SUP2+CH(N1)+Y3*(RS-Y(N1))
  633 IF(N1.EQ.NR) VB1=0.;GOTO 650
      N2=N1+1; Y4=CH(N2)/Y(N2)
      DO 652 J=N2,NR
      R1=CH(J)/Y(J); IF(R1.GT.Y4) Y4=R1
  652 CONTINUE
      VB1=SUP2+Y4*RS
  650 CONTINUE
      WRITE(108,184) VB1,VB2
      IF(VB1.LT.VB2) VB1=VB2
C                //TEST DECISION//
      NTO=MU2*(INT(SUP1-VB1)+1);GAP=INT(SUP1-BH(I1))
      WRITE(108,162) NTO,GAP
  162 FORMAT(5X,'ON ENCLANCHE TEST C3 SI ',I5,' SUPERIEUR A ',I5)
      IF(NTO.LT.GAP) GOTO 700
C                //TEST C3//
C     WRITE(108,102)
  102 FORMAT(//,30X,'TEST C3',/)
      NOC5=0,N1C5=0
      DO 655 J=1,NR
      IF(JVBAS.NE.NUV(J)) GOTO 655
      NUV(J)=NUV(NR); NUV(NR)=JVBAS
      IPV=SOME(J);SOME(J)=SOME(NR); SOME(NR)=IPV
      VAL=Y2(J);Y2(J)=Y2(NR);Y2(NR)=VAL
      GOTO 660
  655 CONTINUE
      WRITE(108,21); STOP
   21 FORMAT(//,5X,'VARIABLE EN BASE NON DETECTEE')
  660 DO 662 J=1,NR-1
      J0=NUV(J);K=ICHIN+(J0-1)*M
      CH(J)=AA(K);X(J)=J
      Y(J)=Y2(J);Y1(J)=Y2(J)
  662 AH(J)=CH(J)
      N1=1;NR0=NR-1
      RS0=BH(MAXW)
      RS1=BH(MAXW)-Y2(NR)
      CALL RESOL(CH,Y,RS0,X,N1,NR0,WOPT,QSORT)
      SUP0=0.
```

```
      IF(N1.EQ.1) GOTO 656
      DO 664 J=1,N1-1
  664 SUP0=SUP0+CH(J)
  656 SUP0=SUP0+WOPT*RS0
      IF(QSORT.NE.2) K=X(N1);JVB0=NUV(K)
      IF(QSORT.EQ.2) N0C5=1
      DO 663 J=1,NR-1
  663 X(J)=J
      N1=1
      CALL RESUL(AH,Y1,RS1,X,N1,NR0,WOPT,QSORT)
      K=ICMIN+(JVBAS-1)*M;SUP1=AA(K)
      IF(N1.EQ.1) GOTO 657
      DO 666 J=1,N1-1
  666 SUP1=SUP1+AH(J)
  657 SUP1=SUP1+WOPT*RS1
      IF(QSORT.NE.2) K=X(N1);JVB1=NUV(K)
      IF(QSORT.EQ.2) N1C5=1
      WRITE(108,136) NUC(I1),BH(I1)
      WRITE(108,186) SUP0,SUP1
  136 FORMAT(5X,'CONTRAINTE ',I4,' SECOND MEMBRE ',I6)
      KT0=INT(SUP0);KT1=INT(SUP1)
      KTEST=KT1
      IF(KT0.GT.KT1) KTEST=KT0
      IF(KTEST.GT.BH(I1)) GOTO 790
      IR=IR+1; X1(IC1)=-I1
      WRITE(108,22) ITERED,NUC(I1)
   22 FORMAT(/,5X,'REDUCTION N ',I2,' TEST C3 : ',I2)
      GOTO 700
C               //TEST DECISION//
  790 IF(KT0.GT.BH(I1)) GOTO 795
      IF(N1C5.EQ.1) GOTO 700
      JVB=JVB1
      K=ICMIN+(JVBAS-1)*M;SUP=AA(K)
      SECMS=BH(MAXW)-Y2(NR);GOTO 799
  795 IF(KT1.GT.BH(I1)) GOTO 700
      IF(N0C5.EQ.1) GOTO 700
      JVB=JVB0;SUP=0;SECMS=BH(MAXW)
C               //TEST C5//
  799 NR0=NR-1
      WRITE(108,106)
  106 FORMAT(//,30X,'TEST C5 ',/)
      DO 1010 J=1,NR0
      IF(JVB.NE.NUV(J)) GOTO 1010
      NUV(J)=NUV(NR0);NUV(NR0)=JVB
      IPV=SOME(J);SOME(J)=SOME(NR0);SOME(NR0)=IPV
      VAL=Y2(J);Y2(J)=Y2(NR0);Y2(NR0)=VAL
      GOTO 1020
 1010 CONTINUE
      WRITE(108,16);STOP
 1020 DO 1022 J=1,NR0-1
      J0=NUV(J);K=ICMIN+(J0-1)*M
```

```
817.            CH(J)=AA(K)
818.            Y(J)=Y2(J);Y1(J)=Y2(J)
819.      1022  AH(J)=CH(J)
820.            N1=1;NR1=NR0-1
821.            RS0=SECMS
822.            RS1=SECMS-Y2(NR0)
823.            CALL RESOL(CH,Y,RS0,X,N1,NR1,WOPT,QSORT)
824.            SUP0=SUP
825.            IF(N1.EQ.1) GOTO 1023
826.            DO 1024 J=1,N1-1
827.      1024  SUP0=SUP0+CH(J)
828.      1023  SUP0=SUP0+WOPT*RS0
829.            N1=1
830.            CALL RESOL(AH,Y1,RS1,X,N1,NR1,WOPT,QSORT)
831.            K=ICMIN+(JVB-1)*M;SUP1=SUP+AA(K)
832.            IF(N1.EQ.1) GOTO 1025
833.            DO 1026 J=1,N1-1
834.      1026  SUP1=SUP1+AH(J)
835.      1025  SUP1=SUP1+WOPT*RS1
836.            WRITE(108,130) NUC(I1),BH(I1)
837.            WRITE(108,180) SUP0,SUP1
838.            KT0=INT(SUP0);KT1=INT(SUP1)
839.            KTEST=KT1
840.            IF(KT0.GT.KT1) KTEST=KT0
841.            IF(KTEST.GT.BH(I1)) GOTO 700
842.            IR=IR+1;X1(IR1)=-I1
843.            WRITE(108,1028) ITERED,NUC(I1)
844.      1028  FORMAT(/,5X,' REDUCTION N ',I2,' TEST C5 ',I2)
845.       700  CONTINUE
846.            IF(IR.EQ.0) GOTO 750
847.            REDUCTION=.TRUE.
848.            DO 665 I=1,IR
849.            IF(X1(I).GT.0) GOTO 665
850.            I1=-X1(I);NUC(I1)=-NUC(I1)
851.       665  CONTINUE
852.            M1=1
853.       670  IF(M1.GT.MR) GOTO 680
854.            IF(NUC(M1).GT.0) M1=M1+1; GOTO 670
855.            I0=-NUC(M1)
856.            DO 675 J=1,NK
857.            J0=NUV(J);K0=I0+(J0-1)*M
858.            IF(AA(K0).NE.0) SOME(J)=SOME(J)-1
859.       675  CONTINUE
860.            NUC(M1)=NUC(MR); NUC(MR)=I0; W(M1)=W(MR); W(MR)=0.
861.            WMIN(M1)=WMIN(MR);WMIN(MR)=0.
862.            BH(M1)=BH(MR);MAXC(M1)=MAXC(MR); SOMC(M1)=SOMC(MR)
863.            NUMAX(M1)=NUMAX(MR)
864.            MR=MR-1; GOTO 670
865.       680  IF(MR.EQ.0) NR=0; GOTO 6000
866.            JF1=0; N1=1
867.       685  IF(N1.GT.NR) GOTO 690
```

```
      IF(SOME(N1).NE.0) N1=N1+1; GOTO 685
      IPV=NUV(N1);NUV(N1)=NUV(NR); NUV(NR)=IPV; SOME(N1)=SOME(NR)
      NR=NR-1; JF1=JF1+1; GOTO 685
  690 IF(JF1.EQ.0) GOTO 695
      WRITE(108,23) (NUV(J),J=NR+1,NR+JF1)
   23 FORMAT(//,5X,'VARIABLES FIXEES A 1 :',25I4)
      DO 692 J=NR+1,NR+JF1
      J0=NUV(J); SOMVAFI=SOMVAFI+CC(J0)
  692 CONTINUE
      WRITE(108,120) SOMVAFI
  695 IF(MR.EQ.1) GOTO 5500
C                 //TEST C2,C4,C6//
  750 IF(FIN) GOTO 5500
      ICR=0
      DO 710 I=1,MR
      IF(WMIN(I).LE.EPS) ICR=ICR+1; X1(ICR)=I
  710 CONTINUE
      IF(ICR.EQ.0) GOTO 1900
C                 //TEST C2//
 1800 DO 132 I=1,ICR
      K=X1(I);K0=NUC(K)
  132 X(I)=K0
      WRITE(108,125) (X(I),I=1,ICR)
      IR=0
      DO 720 J=1,NR
      RS=0.;J0=NUV(J);K0=(J0-1)*M
      DO 725 I=1,MR
      I0=NUC(I);K=I0+K0
  725 RS=RS+WMIN(I)*AA(K)
  720 Y2(J)=RS
      SECM=0.
      DO 730 I=1,MR
  730 SECM=SECM+BH(I)*WMIN(I)
      DO 800 IC1=1,ICR
      WRITE(108,103)
  103 FORMAT(///,30X,'TEST C2',/)
      DO 740 J=1,NR
  740 Y(J)=Y2(J)
      I1=X1(IC1); ICMIN=NUC(I1)
      DO 745 J=1,NR
      J0=NUV(J);K=ICMIN+(J0-1)*M
      X(J)=J
  745 CH(J)=AA(K)
      RS=SECM; N1=1
      CALL RESOL(CH,Y,RS,X,N1,NR,WOPT,QSORT)
      SUP1=0.
      IF(N1.EQ.1) GOTO 749
      DO 748 J=1,N1-1
  748 SUP1=SUP1+CH(J)
  749 SUP2=SUP1; SUP1=SUP1+WOPT*RS
      WRITE(108,194) NUC(I1),SUP1,BH(I1)
```

```
      IF(INT(SUP1)-BH(I1)) 755,755,760
  755 IR=IR+1
      WRITE(108,24) ITERED,NUC(I1)
   24 FORMAT(/,5X,'REDUCTION N ',I2,' TEST C2 : ',I2)
      X1(IC1)=-I1; GOTO 800
  760 IF(QSORT.EQ.2) GOTO 800
      K=X(N1); JVBAS=NUV(K)
      IF(QSORT.EQ.1) GOTO 765
      IF(N1.EQ.1) VB2=0.;GOTO 763
      N2=N1-1; Y3=CH(N2)/Y(N2);VB2=SUP2+CH(N1)+Y3*(RS-Y(N1))
  763 IF(N1.EQ.NR) VB1=0;GOTO 770
      N2=N1+1;Y4=CH(N2)/Y(N2);VB1=SUP2+Y4*RS;GOTO 770
  765 IF(N1.EQ.1) VB2=0;GOTO 766
      N2=N1-1;Y3=CH(N1)/Y(N2)
      DO 762 J=N2,1,-1
      R1=CH(J)/Y(J); IF(R1.LT.Y3) Y3=R1
  762 CONTINUE
      VB2=SUP2+CH(N1)+Y3*(RS-Y(N1))
  766 IF(N1.EQ.NR) VB1=0;GOTO 770
      N2=N1+1; Y4=CH(N2)/Y(N2)
      DO 764 J=N2,NR
      R1=CH(J)/Y(J); IF(R1.GT.Y4) Y4=R1
  764 CONTINUE
      VB1=SUP2+Y4*RS
  770 CONTINUE
      WRITE(108,184) VB1,VB2
      IF(VB1.LT.VB2) VB1=VB2
C                // TEST DECISION//
      NTO=MU2*(INT(SUP1-VB1)+1);GAP=INT(SUP1-BH(I1))
      WRITE(108,164) NTO,GAP
  164 FORMAT(5X,'ON ENCLANCHE TEST C4 SI ',I6,' SUPERIEUR A ',I5)
      IF(NTO.LT.GAP) GOTO 800
C                //TEST C4//
      WRITE(108,104)
  104 FORMAT(//,30X,'TEST C4',/)
      NOC6=0;N1C6=0
      DO 768 J=1,NK
  768 Y(J)=Y2(J)
      DO 772 J=1,NK
      IF(JVBAS.NE.NUV(J)) GOTO 772
      NUV(J)=NUV(NK);NUV(NR)=JVBAS
      VAL=Y2(J);Y2(J)=Y2(NR);Y2(NR)=VAL
      IPV=SOME(J);SOME(J)=SOME(NR);SOME(NR)=IPV; GOTO 775
  772 CONTINUE
      WRITE(108,25); STOP
   25 FORMAT(/,5X,'VARIABLE EN BASE NON DETECTEE')
  775 DO 774 J=1,NR-1
      J0=NUV(J);K=ICMIN+(J0-1)*M
      Y(J)=Y2(J);Y1(J)=Y2(J);CH(J)=AA(K);X(J)=J
  774 AH(J)=CH(J)
      N1=1;NR0=NR-1
```

```
      970.          RS0=SECM;RS1=SECM-Y2(J)
      971.          CALL RESOL(CH,Y,RS0,X,N1,NR0,WOPT,QSORT)
      972.          SUP0=0.
      973.          IF(N1.EQ.1) GOTO 781
      974.          DO 782 J=1,N1-1
      975.      782 SUP0=SUP0+CH(J)
      976.      781 SUP0=SUP0+WOPT*RS0
      977.          IF(QSORT.NE.2) K=X(N1);JVB0=NUV(K)
      978.          IF(QSORT.EQ.2) NOC6=1
      979.          DO 783 J=1,NK-1
      980.      783 X(J)=J
      981.          N1=1
      982.          CALL RESOL(AH,Y1,RS1,X,N1,NR0,WOPT,QSORT)
      983.          K=ICMIN+(JVBAS-1)*M;SUP1=AA(K)
      984.          IF(N1.EQ.1) GOTO 785
      985.          DO 784 J=1,N1-1
      986.      784 SUP1=SUP1+AH(J)
      987.      785 SUP1=SUP1+WOPT*RS1
      988.          IF(QSORT.NE.2) K=X(N1);JVB1=NUV(K)
      989.          IF(QSORT.EQ.2) N1C6=1
      990.          WRITE(108,130) NUC(I1),BH(I1)
      991.          WRITE(108,180) SUP0,SUP1
      992.          KT0=INT(SUP0);KT1=INT(SUP1)
      993.          KTEST=KT1
      994.          IF(KT0.GT.KT1) KTEST=KT0
      995.          IF(KTEST.GT.BH(I1)) GOTO 890
      996.          IR=IR+1; X1(IC1)=-I1
      997.          WRITE(108,26)ITERED,NUC(I1)
      998.       26 FORMAT(/,5X,'REDUCTION N',I2,' TEST C4 : ',I2)
      999.          GOTO 800
     1000.    C                  //TEST DECISION//
     1001.      890 IF(KT0.GT.BH(I1)) GOTO 895
     1002.          IF(N1C6.EQ.1) GOTO 800
     1003.          JVB=JVB1
     1004.          K=ICMIN+(JVBAS-1)*M;SUP=AA(K)
     1005.          SECMS=SECM-Y2(NR);GOTO 899
     1006.      895 IF(KT1.GT.BH(I1)) GOTO 800
     1007.          IF(NOC6.EQ.1) GOTO 800
     1008.          JVB=JVB0;SUP=0;SECMS=SECM
     1009.    C                  //TEST C6//
     1010.      899 NR0=NR-1
     1011.          WRITE(108,107)
     1012.      107 FORMAT(//,30X,'TEST C6',/)
     1013.          DO 1110 J=1,NR0
     1014.          IF(JVB.NE.NUV(J)) GOTO 1110
     1015.          NUV(J)=NUV(NR0);NUV(NR0)=JVB
     1016.          IPV=SOME(J);SOME(J)=SOME(NR0);SOME(NR0)=IPV
     1017.          VAL=Y2(J);Y2(J)=Y2(NR0);Y2(NR0)=VAL
     1018.          GOTO 1120
     1019.     1110 CONTINUE
     1020.          WRITE(108,767 ;STOP
```

```
1021.      1120 DO 1122 J=1,NRO-1
1022.           JO=NUV(J);K=ICMIN+(JO-1)*M
1023.           CH(J)=AA(K)
1024.      1122 AH(J)=CH(J)
1025.           DO 1123 J=1,NRO-1
1026.           Y(J)=Y2(J)
1027.      1123 Y1(J)=Y2(J)
1028.           RSO=SECMS
1029.           RS1=SECMS-Y2(NRO)
1030.           N1=1;NR1=NRO-1
1031.           CALL RESOL(CH,Y,RSO,X,N1,NR1,WOPT,QSORT)
1032.           SUPO=SUP
1033.           IF(N1.EQ.1) GOTO 1130
1034.           DO 1127 J=1,N1-1
1035.      1127 SUPO=SUPO+CH(J)
1036.      1130 SUPO=SUPO+WOPT*RSO
1037.           N1=1
1038.           CALL RESOL(AH,Y1,RS1,X,N1,NR1,WOPT,QSORT)
1039.           K=ICMIN+(JVB-1)*M;SUP1=SUP+AA(K)
1040.           IF(N1.EQ.1) GOTO 1131
1041.           DO 1128 J=1,N1-1
1042.      1128 SUP1=SUP1+AH(J)
1043.      1131 SUP1=SUP1+WOPT*RS1
1044.           WRITE(108,136) NUC(I1),BH(I1)
1045.           WRITE(108,180) SUPO,SUP1
1046.           KTO=INT(SUPO);KT1=INT(SUP1)
1047.           KTEST=KT1
1048.           IF(KTO.GT.KT1) KTEST=KTO
1049.           IF(KTEST.GT.BH(I1)) GOTO 800
1050.           IR=IR+1;X1(IC1)=-I1
1051.           WRITE(108,1149) ITERED,NUC(I1)
1052.      1129 FORMAT(//,5X,'REDUCTION N',I2,' TEST C6 :',I2)
1053.       800 CONTINUE
1054.           IF(IR.EQ.0) GOTO 1900
1055.           REDUCTION=.TRUE.
1056.           DO 805 I=1,ICR
1057.           IF(X1(I).GT.0) GOTO 805
1058.           I1=-X1(I);NUC(I1)=-NUC(I1)
1059.       805 CONTINUE
1060.           M1=1
1061.       810 IF(M1.GT.MR) GOTO 820
1062.           IF(NUC(M1).GT.0) M1=M1+1; GOTO 810
1063.           IO=-NUC(M1)
1064.           DO 815 J=1,NK
1065.           JO=NUV(J);K=IO+(JO-1)*M
1066.           IF(AA(K).NE.0) SOME(J)=SOME(J)-1
1067.       815 CONTINUE
1068.           NUC(M1)=NUC(MR);NUC(MR)=IO; W(M1)=W(MR); W(MR)=0.
1069.           WMIN(M1)=WMIN(MR);WMIN(MR)=0.
1070.           BH(M1)=BH(MR); MAXC(M1)=MAXC(MR); SOMC(M1)=SOMC(MR)
1071.           NUMAX(M1)=NUMAX(MR)
```

202

```
1072.          MR=MR-1; GOTO 810
1073.      820 IF(MR.EQ.0) NR=0; GOTO 9000
1074.          JF1=0; N1=1
1075.      830 IF(N1.GT.NR) GOTO 840
1076.          IF(SOME(N1).NE.0) N1=N1+1; GOTO 830
1077.          IPV=NUV(N1); NUV(N1)=NUV(NR); NUV(NR)=IPV;SOME(N1)=SOME(NR)
1078.          NR=NR-1; JF1=JF1+1; GOTO 830
1079.      840 IF(JF1.EQ.0) GOTO 850
1080.          WRITE(108,23)(NUV(J),J=NR+1,NR+JF1)
1081.          DO 845 J=NR+1,NR+JF1
1082.          J0=NUV(J); SOMVAFI=SOMVAFI+CC(J0)
1083.      845 CONTINUE
1084.          WRITE(108,120) SOMVAFI
1085.      850 IF(MR.EQ.1) GOTO 5500
1086.     1900 IF(IREDL-IREDC) 2001,1501,1001
1087.     1501 WRITE(108,1502) ;STOP
1088.     1502 FORMAT(//,5X,'ERREUR BRANCHEMENT POST REDUCTION')
1089.     C
1090.     5000 IF(RSUPL.EQ.0) GOTO 5500
1091.     C                    //TEST V5//
1092.          WRITE(108,1100)
1093.     1100 FORMAT(///,30X,'TEST V5 ',/)
1094.          REDUCTION=.FALSE.
1095.          IRV=1
1096.     1200 IF(IRV.GT.MR) GOTO 4000
1097.          WRITE(108,1201) NUMAX(IRV),NUC(IRV)
1098.     1201 FORMAT(5X,'SEPARATION SUR VARIABLE ',I3,' DE LA CONTRAINTE ',I3)
1099.          IR=0;JR0=0;JR1=0
1100.          J1=NUMAX(IRV)
1101.          DO 1210 J=1,NR
1102.          IF(J1.NE.NUV(J)) GOTO 1210
1103.          NUV(J)=NUV(NR);NUV(NR)=J1
1104.          IPV=SOME(J);SOME(J)=SOME(NR);SOME(NR)=IPV;GOTO 1220
1105.     1210 CONTINUE
1106.          WRITE(108,1211) ;STOP
1107.     1211 FORMAT(//,5X,'VARIABLE SEPARABLE NON DETECTEE')
1108.     1220 GAP=MIN-SOMVAFI
1109.          WRITE(108,190) GAP
1110.          I0=NUC(IRV)
1111.          DO 1230 J=1,NR-1
1112.          J0=NUV(J);K=I0+(J0-1)*M
1113.          Y(J)=AA(K);Y1(J)=Y(J);CH(J)=CC(J0)
1114.     1230 AH(J)=CH(J)
1115.          N1=1;NR0=NR-1
1116.          RS0=BH(IRV)
1117.          K=I0+(J1-1)*M;RS1=RS0-AA(K)
1118.          CALL RESOL(CH,Y,RS0,X,N1,NR0,WOPT,QSORT)
1119.          SUP0=0
1120.          DO 1232 J=1,N1-1
1121.     1232 SUP0=SUP0+CH(J)
1122.          SUP0=SUP0+WOPT*RS0
```

```
      WRITE(108,1255) SUP0
 1253 FORMAT(5X,'VALEUR RELAXATION(0):',F12.4)
      K0=INT(SUP0)*IF(K0.GT.GAP) GOTO 1250
      JR1=1;SOMVAFI=SOMVAFI+CC(J1)
      DO 1234 I=1,MR
      I1=NUC(I);K=I1+(J1-1)*M
 1234 BH(I)=BH(I)-AA(K)
      GOTO 1260
 1250 N1=1
      CALL RESOL(AM,Y1,RS1,X,N1,NR0,WOPT,QSORT)
      SUP1=CC(J1)
      DO 1252 J=1,N1-1
 1252 SUP1=SUP1+AH(J)
      SUP1=SUP1+WOPT*RS1
      WRITE(108,1253) SUP1
 1253 FORMAT(5X,'VALEUR RELAXATION(1):',F12.4)
      K1=INT(SUP1)
      IF(K1.GT.GAP) IRV=IRV+1; GOTO 1200
      NUV(NR)=-NUV(NR);JR0=1
 1260 REDUCTION=.TRUE.
      WRITE(108,1261) NUV(NR)
 1261 FORMAT(5X,'VARIABLE ',I3,' ELIMINEE PAR VS')
      NR=NR0
      DO 1262 I=1,MR
      I1=NUC(I);K=I1+(J1-1)*M
 1262 SOMC(I)=SOMC(I)-AA(K)
      DO 1264 I=1,MR
      I1=NUC(I);J2=NUV(1);K0=I1+(J2-1)*M
      MAXC(I)=AA(K0);NUMAX(I)=J2
      DO 1264 J=1,NR
      J0=NUV(J);K=I1+(J0-1)*M
      IF(AA(K).GT.MAXC(I)) MAXC(I)=AA(K);NUMAX(I)=J0
 1264 CONTINUE
      DO 1266 J=1,NR
      J0=NUV(J);K=(J-1)*MR;K0=(J0-1)*M
      DO 1266 I=1,MR
      I1=NUC(I);L=I+K;L0=I1+K0
 1266 AH(L)=AA(L0)
      MNR=MR*NR
      NR0=NR
      CALL PREPROCESS(AH,BH,MR,NR,MNR,JR0,JR1,NUV,NUC,W,SOMC,MAXC,SOME,
     SWMIN,NUMAX)
      IF(NR.EQ.NR0) GOTO 1270
      DO 1268 J=NR+1,NR0
      J0=NUV(J);IF(J0.GT.0) SOMVAFI=SOMVAFI+CC(J0)
 1268 CONTINUE
      WRITE(108,120) SOMVAFI
      IF(NR.EQ.0) GOTO 6000
 1270 IF(MR.EQ.1) GOTO 5500
      GOTO 1200
C
```

204

```
      4000 IF(.NOT.REDUCTION) GOTO 5500
           MAXW=1;R=WMIN(1);ICR=0
           DO 4010 I=2,MR
           IF(WMIN(I).GT.R) R=WMIN(I);MAXW=I
      4010 CONTINUE
           DO 4015 I=1,MR
           IF(WMIN(I).LT.R) ICR=ICR+1;X1(ICR)=I
      4015 CONTINUE
           FIN=.TRUE.
           WRITE(108,4020)
      4020 FORMAT(//,20X,'ESSAI ELIMINATION CONTRAINTE PAR C1/C3/C5')
           GOTO 1700
C                    //FIN REDUCTION//
      5500 CALL UCTIME(1,INTER)
           WRITE(108,560) INTER
       560 FORMAT(//,20X,'TEMPS TOTAL REDUCTION :',I5)
           WRITE(108,28) MR,NR
        28 FORMAT(//,5X,'PROBLEME DIFFICILE DE TAILLE ',I3,1X,'X',I3,///)
           IF(MR.EQ.M) WRITE(108,29); GOTO 950
        29 FORMAT(//,5X,'AUCUNE CONTRAINTE ELIMINEE')
           WRITE(108,30) (NUC(I),I=MR+1,M)
        30 FORMAT(//,5X,'CONTRAINTE ELIMINEE : ',10I3)
       950 IF(NR.EQ.N) WRITE(108,37); GOTO 5600
        37 FORMAT(//,5X,'AUCUNE VARIABLE ELIMINEE')
           WRITE(108,31)(NUV(J),J=NR+1,N)
        31 FORMAT(//,5X,'VARIABLE ELIMINEE : ',25I4)
      5600 DO 280 J=1,NR
           J0=NUV(J);CH(J)=CC(J0)
       280 CONTINUE
           DO 282 J=1,NR
           J0=NUV(J);K=(J-1)*MR;K0=(J0-1)*M
           DO 282 I=1,MR
           I0=NUC(I);L=I+K;L0=I0+K0
       282 AH(L)=AA(L0)
           IF(MR.EQ.M.AND.NR.EQ.N) GOTO 5550
           WRITE(108,1) (NUV(J),J=1,NR)
           WRITE(108,1) (NUC(I),I=1,MR)
           WRITE(108,1) (CH(J),J=1,NR)
           WRITE(108,1) (BH(I),I=1,MR)
           DO 286 I=1,MR
       286 WRITE(108,1) (AH(I+(J-1)*MR),J=1,NR)
      5550 IF(MR-1) 5610,5610,5620
      5610 WRITE(108,32)
        32 FORMAT(//,5X,'PROBLEME REDUIT A 1 CONTRAINTE RESOLU PAR FPK79')
           ISORT=0;N1=0
           CALL FPK79(CH,AH,BH,NR,X,N1,ISORT)
           GOTO 7000
      5620 WRITE(108,33)
        33 FORMAT(//,5X,'PROBLEME RESOLU PAR ALGORITHME DE WEI SHIH')
           GOTO 8000
      6000 CALL UCTIME(1,INTER)
```

```
      WRITE(108,560) INTER
      WRITE(108,34)
   34 FORMAT(/,5X,'PROBLEME TOTALEMENT REDUIT')
      Z=0
      DO 960 J=1,N
      J0=ABS(NUV(J)); IF(NUV(J).LT.0) X(J0)=0; GOTO 960
      X(J0)=1; Z=Z+CC(J0)
  960 CONTINUE
      WRITE(108,35) Z; WRITE(108,36) ((J,X(J)),J=1,N);GOTO 8000
   35 FORMAT(/,5X,'VALEUR OPTIMALE : ',I9)
   36 FORMAT(/,5X,'(NUM VARIABLE,SOL OPT)',10(2X,'(',I2,',',I1,')'))
 7000 NOPT=SOMVAFI+N1
      WRITE(108,35) NOPT
      DO 290 J=1,NN
      J0=NUV(J);X1(J0)=X(J)
  290 CONTINUE
      IF(NR.EQ.N) GOTO 296
      DO 294 J=NR+1,N
      J0=ABS(NUV(J));IF(NUV(J).LT.0) X1(J0)=0;GOTO 294
      X1(J0)=1
  294 CONTINUE
  296 WRITE(108,36) ((J,X1(J)),J=1,N)
 8000 RETURN
      END
```

```
      SUBROUTINE PREPROCES(A,B,MR,NR,MNR,JR0,JR1,IP,JP,P,XS,XM,XNUL,PM,
     SXNU)
C              //RETOUR A UN PROBLEME BIEN POSE //
      INTEGER A(MNR),B(MR),XS(MR),XM(MR),XNUL(NR),XNU(MR)
      DIMENSION IP(NR),JP(MR),PM(MR),P(MR)
      M=MR;N=NR
      IF(JR0+JR1.EQ.0) GOTO 300
      IF(JR1.EQ.0) GOTO 200
C              //FIXATION DE VARIABLES A 0//
      DO 10 I=1,M
      IF(B(I).LT.XM(I)) GOTO 100
   10 CONTINUE
      GOTO 200
  100 JF0=0;N1=1;M1=1
   20 IF(N1.GT.N) GOTO 25
   15 IF(M1.GT.M) N1=N1+1;M1=1; GOTO 20
      K=M1+(N1-1)*M;IF(A(K).LE.B(M1)) M1=M1+1;GOTO 15
      IF(N1.EQ.N) IP(N)=-IP(N); GOTO 24
      IPV=-IP(N1);JP(N1)=IP(N);IP(N)=IPV;XNUL(N1)=XNUL(N)
      DO 22 I=1,M
      L1=I+(N1-1)*M;L2=I+(N-1)*M;IPV=A(L1);A(L1)=A(L2);A(L2)=IPV
   22 CONTINUE
   24 DO 23 I=1,M
      L=I+(N-1)*M; XS(I)=XS(I)-A(L)
   23 CONTINUE
      N=N-1;JF0=JF0+1;M1=1; GOTO 20
   25 WRITE(108,1)(IP(J),J=N+1,N+JF0)
    1 FORMAT(/,5X,'VARIABLES FIXEES A 0 PAR PREPROCES :',2>I4)
      IF(N.EQ.0) WRITE(108,2); GOTO 400
    2 FORMAT(/,5X,'TOUTES LES VARIABLES FIXEES EN PHASE 1:STOP')
      DO 40 I=1,M
      DO 50 J=N+1,N+JF0
      K=I+(J-1)*M;IF(A(K).EQ.XM(I)) GOTO 45
   50 CONTINUE
      GOTO 40
   45 XM(I)=A(I)
      XNU(I)=IP(1)
      DO 52 J=1,N
      K=I+(J-1)*M;IF(A(K).GT.XM(I)) XM(I)=A(K);XNU(I)=IP(J)
   52 CONTINUE
   40 CONTINUE
      IF(JR0+JF0.EQ.0) GOTO 400
C              // ELIMINATION DE CONTRAINTES REDONDANTES//
  200 KRE=0;M1=1
   60 IF(M1.GT.M) GOTO 70
      IF(XS(M1).GT.B(M1)) M1=M1+1; GOTO 60
      DO 65 J=1,N
      K=M1+(J-1)*MR;IF(A(K).NE.0) XNUL(J)=XNUL(J)-1
   65 CONTINUE
      IF(M1.EQ.M) GOTO 62
      DO 63 J=1,N
```

```
      L1=M1+(J-1)*MR;L2=M+(J-1)*MR;A(L1)=A(L2)
   63 CONTINUE
      IPV=JP(M1);JP(M1)=JP(M);JP(M)=IPV
      B(M1)=B(M);XS(M1)=XS(M);XM(M1)=XM(M);XNU(M1)=XNU(M)
      P(M1)=P(M);P(M)=0.;PM(M1)=PM(M);PM(M)=0.
   62 M=M-1;KRE=KRE+1; GOTO 60
   70 IF(KRE.EQ.0) GOTO 400
      IF(M.EQ.0) WRITE(108,3);N=0; GOTO 400
    3 FORMAT(/,5X,'TOUTES LES CONTRAINTES SONT REDONDANTES:STOP')
      WRITE(108,4)(JP(I),I=M+1,M+KRE)
    4 FORC  FORMAT(/,5X,'CONTRAINTES ELIMINEES PAR PREPROCES :',10I4)
C                //FIXATION DE VARIABLES A 1//
  300 JF1=0;N1=1
   80 IF(N1.GT.N) GOTO 90
      IF(XNUL(N1).NE.0) N1=N1+1; GOTO 80
      IF(N1.EQ.N) GOTO 95
      IPV=IP(N1);IP(N1)=IP(N);IP(N)=IPV;XNUL(N1)=XNUL(N)
   95 N=N-1;JF1=JF1+1; GOTO 80
   90 IF(JF1.EQ.0) GOTO 400
      WRITE(108,5)(IP(J),J=N+1,N+JF1)
    5 FORMAT(/,5X,'VARIABLES FIXEES A 1 PAR PREPROCES :',20I4)
      IF(N.EQ.0) WRITE(108,6)
    6 FORMAT(/,5X,'TOUTES LES VARIABLES FIXEES EN PHASE 3:STOP')
  400 NR=N;MR=M;MNR=M*N
      RETURN
      END
```

```
      SUBROUTINE AGNES1(C,H,B,M,N,MN,XHEU,ZHEU,ALPHA,PST,ITERED,XS,XNUL)
C          //HEURISTIQUE UTILISANT LA SOLUTION OPTIMALE //
      INTEGER C(N),H(MN),B(M),XHEU(N),XS(M),XNUL(N)
      DIMENSION PST(M)
      INTEGER X(60),YS(60),YNUL(60),PEH(300),PEB(30),PEC(10),L(30)
      DIMENSION PI(30),Q(30),IP(60),JP(30),A(60),P(30)
      INTEGER ZHEU ,Z,QSORT
      REAL NORME
C                 // INITIALISATION //
      JSEUIL=10;NPAS=5;IPAS=0;ZHEU=0;QSORT=0
      IF(ITERED.GT.0) GOTO 20
      DO 16 I=1,M
      KS=0
      DO 17 J=1,N
      K=I+(J-1)*M;KS=KS+H(K)
   17 CONTINUE
      R=KS;R=(R-B(I))/R
   16 P(I)=R
      GOTO 25
   20 DO 21 I=1,M
   21 P(I)=PST(I)
   25 R=NORME(P,M)
      DO 22 I=1,M
      P(I)=P(I)/R;Q(I)=1.
   22 CONTINUE
C                    //NOUVELLE ITERATION//
 2000 IPAS=IPAS+1
      JD=1;JF=N;MF=M;Z=0;ITER=0
      DO 26 I=1,M
      JP(I)=I;YS(I)=XS(I);L(I)=B(I)
   26 CONTINUE
      DO 27 J=1,N
      IP(J)=J;YNUL(J)=XNUL(J)
   27 CONTINUE
      DO 29 I=1,M
   29 PI(I)=P(I)*Q(I)
      R=NORME(PI,M)
      DO 30 I=1,M
   30 PI(I)=PI(I)/R
C          // NOUVELLE ETAPE //
 1000 ITER=ITER+1
      JD1=JD;JF1=JF;JX2=JF1-JD1+1
      IF(JF1-JD1) 1001,1002,1003
 1001 WRITE(108,1) IPAS,ITER;STOP
    1 FORMAT(/,5X,'ERREUR DANS HEURISTIQUE 1 A ITERATION ',I2,' ETAPE ',
     $I2)
 1002 DO 31 I=1,MF
      I1=JP(I);J1=IP(JD1);K=I1+(J1-1)*M
      IF(H(K).GT.L(I1)) JX=JD-1; GOTO 900
   31 CONTINUE
      JX=JD; GOTO 900
```

```
      1003 IF(JX2.LE.JSEUIL) GOTO 800
C          //CONTRACTION //
  100 IF(MF.EQ.1) GOTO 180
      IF(ITER.EQ.1) GOTO 120
      DO 101 I=1,MF
      I1=JP(I);KS=0
      DO 102 J=JD1,JF1
      J1=IP(J);K=I1+(J1-1)*M;KS=KS+H(K)
  102 CONTINUE
      R=KS;PI(I)=Q(I1)*(R-L(I1))/R
  101 CONTINUE
  120 DO 121 J=JD1,JF1
      R=0;J1=IP(J)
      DO 122 I=1,MF
      I1=JP(I);K=I1+(J1-1)*M;R=R+H(K)*PI(I)
  122 CONTINUE
  121 A(J)=R
      R=0
      DO 123 I=1,MF
      I1=JP(I);R=R+L(I1)*PI(I)
  123 CONTINUE
      B1=R; GOTO 150
  180 I1=JP(MF);B1=L(I1)
      DO 181 J=JD1,JF1
      J1=IP(J);K=I1+(J1-1)*M;A(J)=H(K)
  181 CONTINUE
  150 IF(ITER.EQ.1) B1=B1*ALPHA
C          // RESOLUTION EN CONTINU DU PROBLEME CONTRACTE//
      DO 201 J=JD1,JF1
      J1=IP(J);X(J)=C(J1)
  201 CONTINUE
      N1=JD1;WOPT=0.
      CALL RESOL(X(N1),A(N1),B1,IP(N1),N1,JX2,WOPT,QSORT)
      JD=N1;NVF1=JD-JD1
      IF(MF-1) 1001,250,300
  250 IF(B1.NE.0.) GOTO 700
      JX=JD-1; GOTO 900
C          // RETOUR ARRIERE SELON SENJU-TOYODA //
  300 IF(JD-JD1-1)1001,310,320
  310 J1=IP(JD1)
      DO 311 I=1,MF
      I1=JP(I);K=I1+(J1-1)*M;L(I1)=L(I1)-H(K);YS(I1)=YS(I1)-H(K)
  311 CONTINUE
      IF(JD.EQ.JF1) JD1=JD; GOTO 1002
      GOTO 400
  320 NCV=0
      DO 321 I=1,MF
      KS=0; I1=JP(I)
      DO 322 J=JD1,JD-1
      J1=IP(J);K=I1+(J1-1)*M;KS=KS+H(K)
  322 CONTINUE
```

210

```
      NCR=0;M1=1
  550 IF(M1.GT.MF) GOTO 510
      M2=JP(M1)
      IF(YS(M2).GT.L(M2)) M1=M1+1; GOTO 550
      DO 511 J=JD,JF
      J1=IP(J);K=M2+(J1-1)*M;IF(H(K).NE.0) YNUL(J1)=YNUL(J1)-1
  511 CONTINUE
      IF(M1.EQ.MF) GOTO 520
      IPV=JP(M1);JP(M1)=JP(MF); JP(MF)=IPV
  520 MF=MF-1;NCR=NCR+1; GOTO 550
  510 IF(NCR.EQ.0) GOTO 1000
      IF(MF.EQ.0) JX=JF; GOTO 900
C                 // FIXATION DE VARIABLE A 1 //
      NVF=0;N1=JD
  600 IF(N1.GT.JF) GOTO 1000
      J1=IP(N1)
      IF(YNUL(J1).NE.0) N1=N1+1; GOTO 600
      IF(N1.EQ.JD) GOTO 610
      IPV=IP(JD); IP(JD)=IP(N1); IP(N1)=IPV
  610 JD=JD+1; NVF=NVF+1;N1=N1+1; GOTO 600
C                 // REINSERTION CAS A 1 CONTRAINTE //
  700 WRITE(108,70);I1=JP(1); I(I1)=B1
   70 FORMAT(5X,'REINSERTION CAS A 1 CONTRAINTE')
      K0=0
      DO 710 J=JD,JF
      J1=JP(J);K=I1+(J1-1)*M
      IF(H(K).LE.L(I1)) K0=K0+1;X(K0)=IP(J)
  710 CONTINUE
      IF(K0.EQ.0) JX=JD-1; GOTO 900
      DO 721 J=1,K0
      J1=X(J);K=I1+(J1-1)*M;A(J)=C(J1)/H(K)
  721 CONTINUE
      K1=1
      DO 722 J=JD,JF
      IF(IP(J).NE.X(K1)) GOTO 722
      IF(J.EQ.JD+K1-1) GOTO 723
      IPV=IP(J);IP(J)=IP(JD+K1-1);IP(JD+K1-1)=IPV
  723 K1=K1+1;IF(K1.GT.K0) GOTO 730
  722 CONTINUE
  730 L1=2
  740 IF(L1.GT.K0) GOTO 750
      L2=L1-1
  735 IF(L2.LE.0) GOTO 736
      L3=L2+1;IF(A(L2).GE.A(L3)) GOTO 736
      IPV=IP(JD+L2-1);IP(JD+L2-1)=IP(JD+L2);IP(JD+L2)=IPV
      VAL=A(L2);A(L2)=A(L3);A(L3)=VAL
      L2=L2-1; GOTO 735
  736 L1=L1+1; GOTO 740
  750 J=1;K1=0
  755 IF(J.GT.K0) GOTO 770
      J1=IP(JD+J-1);K=I1+(J1-1)*M;L(I1)=L(I1)-H(K)
```

```
      L(I1)=L(I1)-KS;YS(I1)=YS(I1)-KS
      IF(L(I1).LT.0) NCV=NCV+1
  321 CONTINUE
      IF(NCV) 330,350,340
  330 IF(JD.EQ.JF1) JD1=JD; GOTO 1002
      GOTO 400
  340 DO 341 J=JD1,JD-1
      J1=IP(J);KS=0
      DO 342 I=1,MF
      I1=JP(I);IF(L(I1).LT.0) K=I1+(J1-1)*M;KS=KS-H(K)*L(I1)
  342 CONTINUE
      R=KS;A(J)=C(J1)/R
  341 CONTINUE
  350 L0=JD-1;JMIN=L0
      VAL =A(L0)
      DO 351 J=JD1,L0
      IF(A(J).LT.VAL) VAL=A(J);JMIN=J
  351 CONTINUE
      IF(JMIN.GE.L0) GOTO 360
      IPV=IP(L0);IP(L0)=IP(JMIN);IP(JMIN)=IPV
      VAL=A(L0);A(L0)=A(JMIN);A(JMIN)=VAL
  360 J1=IP(L0);NCV=0
      DO 361 I=1,MF
      I1=JP(I);K=I1+(J1-1)*M;L(I1)=L(I1)+H(K);YS(I1)=YS(I1)+H(K)
      IF(L(I1).LT.0) NCV=NCV+1
  361 CONTINUE
      JD=JD-1; NVF1=NVF1-1
      IF(NCV.GT.0) GOTO 350
C               // RETOUR A UN PROBLEME BIEN POSE //
C               // FIXATION DE VARIABLE A 0 //
  400 K1=JD
  410 IF(K1.GT.JF) GOTO 450
      J1=IP(K1)
      DO 411 I=1,MF
      I1=JP(I);K=I1+(J1-1)*M;IF(H(K).GT.L(I1)) GOTO420
  411 CONTINUE
      K1=K1+1; GOTO 410
  420 IF(K1.EQ.JF) JF=JF-1; GOTO 450
      IPV=IP(K1);IP(K1)=IP(JF); IP(JF)=IPV
      JF=JF-1; GOTO 410
  450 NVF0=JF1-JF
      IF(NVF0.EQ.0) GOTO 1000
      IF(JF-JD) 460,1000,500
  460 JX=JF; GOTO 900
C               // ELIMINATION CONTRAINTE REDONDANTE//
  500 DO 501 I=1,MF
      I1=JP(I);KS=0
      DO 502 J=JF+1,JF1
      J1=IP(J);K=I1+(J1-1)*M;KS=KS+H(K)
  502 CONTINUE
  501 YS(I1)=YS(I1)-KS
```

```
      IF(L(I1)) 759,761,761
  759 L(I1)=L(I1)+H(K);K1=K1-1;X(J)=0
  761 J=J+1; GOTO 755
  770 L1=1;L2=K0
  772 IF(L1.GT.L2) GOTO 780
      IF(X(L1).EQ.1) L1=L1+1; GOTO 772
  774 IF(L1.EQ.L2) GOTO 780
      IF(X(L2).EQ.0) L2=L2-1; GOTO 774
      IPV=IP(JD+L1-1);IP(JD+L1-1)=IP(JD+L2-1);IP(JD+L2-1)=IPV
      IPV=X(L1);X(L1)=X(L2);X(L2)=IPV
      L1=L1+1; GOTO 772
  780 JX=JD+K1-1; GOTO 900
C                 // PROCEDURE ENUMERATIVE //
  800 DO 801 J=JD1,JF1
      J1=IP(J);K=J-JD1+1;PEC(K)=C(J1)
  801 CONTINUE
      DO 802 I=1,MF
      I1=JP(I);PEB(I)=L(I1)
  802 CONTINUE
      DO 803 I=1,MF
      K1=JX2*(I-1);I1=JP(I)
      DO 803 J=JD1,JF1
      K2=J-JD1+1;J1=IP(J);K0=K1+K2;K=I1+(J1-1)*M;PEH(K0)=H(K)
  803 CONTINUE
      N0=JX2*MF;N1=JD1
      CALL LIFO(PEC,PEH,PEB,IP(N1),MF,JX2,N1,N0)
      JX=N1-1
C                 // FIN D'UNE ITERATION //
  900 Z=0
      DO 901 J=1,JX
      J1=IP(J);Z=Z+C(J1);X(J1)=1
  901 CONTINUE
      DO 902 J=JX+1,N
      J1=IP(J);X(J1)=0
  902 CONTINUE
      DO 903 I=1,M
      I1=JP(I);KS=0
      DO 904 J=1,JX
      J1=IP(J);K=I1+(J1-1)*M;KS=KS+H(K)
  904 CONTINUE
      L(I1)=B(I1)-KS;IF(L(I1).LT.0) GOTO 999
  903 CONTINUE
      IF(Z.LE.ZHEU) GOTO 920
      DO 906 I=1,M
      R=B(I);Q(I)=(R-L(I))/R
  906 CONTINUE
      DO 905 J=1,N
  905 XHEU(J)=X(J)
      ZHEU=Z;NX=0; GOTO 950
  920 IF(NX.GE.3) GOTO 3000
      DO 921 I=1,M
```

```
      921 Q(I)=Q(I)**2
          NX=NX+1
      950 IF(IPAS.LT.NPAS) GOTO 2000
          GOTO 3000
      999 WRITE(108,4)
        4 FORMAT(/,5X,'ERREUR DANS AGNES1:SOLUTION NON REALISABLE')
          ZHEU=0
          DO 104 J=1,N
      104 XHEU(J)=0
     3000 WRITE(108,2) ALPHA,ZHEU
          WRITE(108,3)(XHEU(J),J=1,N)
        2 FORMAT(/,5X,'AGNES1 : ALPHA = ',F3.1,3X,'MINORANT = ',I9)
        3 FORMAT(/,5X,'SOLUTION HEURISTIQUE :',50I2)
          RETURN
          END
```

```
      SUBROUTINE AGNES2(C,H,B,M,N,MN,X,Z,KFRAC,PST,XS,XNUL,ECHEC)
C          //HEURISTIQUE UTILISANT LES COUTS REDUITS//
      INTEGER C(N),H(MN),B(M),X(N),XS(M),XNUL(N)
      DIMENSION PST(M)
      INTEGER YS(30),YNUL(60),PEH(300),PEB(30),PEC(10),L(30)
      DIMENSION PI(30),IP(60),JP(30),A(60)
      INTEGER Z,QSORT
      LOGICAL ECHEC
      REAL NORME
C                 //INITIALISATION//
      JSEUIL=10;QSORT=0
      DO 21 I=1,M
   21 PI(I)=PST(I)
      R=NORME(PI,M)
      DO 22 I=1,M
   22 PI(I)=PI(I)/R
      JD=1;JF=N;MF=M;ITER=0
      DO 26 I=1,M
      JP(I)=I;YS(I)=XS(I);L(I)=B(I)
   26 CONTINUE
      DO 27 J=1,N
      IP(J)=J;YNUL(J)=XNUL(J)
   27 CONTINUE
C                 //NOUVELLE ETAPE//
 1000 ITER=ITER+1
      JD1=JD;JF1=JF;JX2=JF1-JD1+1
      IF(JF1-JD1) 1001,1002,1003
 1001 WRITE(108,1) ITER;STOP
    1 FORMAT(//,5X,'ERREUR DANS HEURISTIQUE 2 A ETAPE ',I2)
 1002 DO 31 I=1,MF
      I1=JP(I);J1=IP(JD1);K=I1+(J1-1)*M
      IF(H(K).GT.L(I1)) JX=JD-1;GOTO 900
   31 CONTINUE
      JX=JD; GOTO 900
 1003 IF(JX2.LE.JSEUIL)  GOTO 800
C                 //CONTRACTION//
  100 IF(MF.EQ.1) GOTO 180
      IF(ITER.EQ.1) GOTO 120
      DO 101 I=1,MF
      I1=JP(I);KS=0
      DO 102 J=JD1,JF1
      J1=IP(J);K=I1+(J1-1)*M;KS=KS+H(K)
  102 CONTINUE
      R=KS;PI(I)=(R-L(I1))/R
  101 CONTINUE
  120 DO 121 J=JD1,JF1
      R=0; J1=IP(J)
      DO 122 I=1,MF
      I1=JP(I);K=I1+(J1-1)*M;R=R+H(K)*PI(I)
  122 CONTINUE
  121 A(J)=R
```

```
      R=0
      DO 123 I=1,MF
      I1=JP(I); R=R+L(I1)*PI(I)
  123 CONTINUE
      B1=R;GOTO 150
  180 I1=JP(MF);B1=L(I1)
      DO 181 J=JD1,JF1
      J1=IP(J);K=I1+(J1-1)*M;A(J)=H(K)
  181 CONTINUE
C                 //RESOLUTION EN CONTINU DU PROBLEME CONTRACTE//
  150 DO 201 J=JD1,JF1
      J1=IP(J);X(J)=C(J1)
  201 CONTINUE
      N1=JD1
      CALL RESOL(X(N1),A(N1),B1,IP(N1),N1,JX2,WOPT,QSORT)
      N1=JD1
C           //SELECTION DES PLUS GROS COUTS REDUITS EN VALEUR ABSOLUE//
      DO 210 J=JD1,JF1
      A(J)=X(J)-WOPT*A(J);IF(A(J).LT.0.) IP(J)=-IP(J)
  210 A(J)=ABS(A(J))
      R=KFRAC;NVS=INT(JX2/R)
      CALL SELPPGN(A(N1),IP(N1),N1,JX2,NVS)
C           //FIXATION DES VARIABLES SELECTIONNEES//
      JF0=JD1+NVS-1;NVF1=0;NVF0=0;JD0=JD1
  220 IF(JD0.GT.JF0) GOTO 230
      IF(IP(JD0).GT.0) GOTO 240
      IPV=IP(JF);IP(JF)=-IP(JD);IP(JD)=IP(JF0);IP(JF0)=IPV
      J1=IP(JF)
      DO 221 I=1,MF
      I1=JP(I);K=I1+(J1-1)*M;YS(I1)=YS(I1)-H(K)
  221 CONTINUE
      JF=JF-1;JF0=JF0-1;NVF0=NVF0+1; GOTO 220
  240 J1=IP(JD0)
      DO 241 I=1,MF
      I1=JP(I);K=I1+(J1-1)*M;L(I1)=L(I1)-H(K);YS(I1)=YS(I1)-H(K)
  241 CONTINUE
      JD=JD+1;JD0=JD0+1; NVF1=NVF1+1; GOTO 220
  230 DO 243 J=JD1,JF1
  243 IF(IP(J).LT.0) IP(J)=-IP(J)
      DO 231 I=1,MF
      I1=JP(I);IF(L(I1).LT.0) GOTO 250
  231 CONTINUE
      GOTO 300
  250 DO 251 J=JD1,JD-1
      J1=IP(J)
      DO 251 I=1,MF
      I1=JP(I);K=I1+(J1-1)*M;L(I1)=L(I1)+H(K)
  251 CONTINUE
      M3=MF;GOTO 600
C           //RETOUR A UN PROBLEME BIEN POSE//
C           //FIXATION DE VARIABLE A 0//
```

```
      300 K0=0;IF(NVF1.EQ.0) GOTO 400
          K1=JD
      310 IF(K1.GT.JF) GOTO 350
          J1=IP(K1)
          DO 311 I=1,MF
          I1=JP(I);K=I1+(J1-1)*M;IF(H(K).GT.L(I1))K0=K0+1;GOTO320
      311 CONTINUE
          K1=K1+1; GOTO 310
      320 IF(K1.EQ.JF) JF=JF-1; GOTO 350
          IPV=IP(K1);IP(K1)=IP(JF);IP(JF)=IPV
          JF=JF-1; GOTO 310
      350 NVF0=JF1-JF
          IF(NVF0.EQ.0) GOTO 1000
          IF(JF-JD) 360,1000,400
      360 JX=JF; GOTO 900
C                     //ELIMINATION CONTRAINTE REDONDANTE//
      400 IF(K0+NVF0.EQ.0) GOTO 1000
          IF(K0.EQ.0) GOTO 415
          DO 401 I=1,MF
          I1=JP(I); KS=0
          DO 402 J=JF+1,JF+K0
          J1=IP(J);K=I1+(J1-1)*M;KS=KS+H(K)
      402 CONTINUE
      401 YS(I1)=YS(I1)-KS
      415 NCR=0;M1=1;M2=MF
      450 IF(M1.GT.MF) GOTO 410
          M2=JP(M1)
          IF(YS(M2).GT.L(M2)) M1=M1+1; GOTO 450
          DO 411 J=JD,JF
          J1=IP(J);K=M2+(J1-1)*M;IF(H(K).NE.0)YNUL(J1)=YNUL(J1)-1
      411 CONTINUE
          IF(M1.EQ.MF) GOTO 420
          IPV=JP(M1);JP(M1)=JP(MF);JP(MF)=IPV
      420 MF=MF-1;NCR=NCR+1; GOTO 450
      410 IF(NCR.EQ.0) GOTO 1000
          IF(MF.NE.0) GOTO 500
          DO 430 J=JD1,JD-1
          J1=IP(J)
          DO 430 I=1,M2
          I1=JP(I);K=I1+(J1-1)*M;L(I1)=L(I1)+H(K)
      430 CONTINUE
          GOTO 600
C                     //FIXATION DE VARIABLE A 1//
      500 NVF=0;N1=JD
      510 IF(N1.GT.JF) GOTO 1000
          J1=IP(N1)
          IF(YNUL(J1).NE.0) N1=N1+1; GOTO 510
          IF(N1.EQ.JD) GOTO 520
          IPV=IP(JD);IP(JD)=IP(N1);IP(N1)=IPV
      520 JD=JD+1;NVF=NVF+1; N1=N1+1; GOTO 510
C           //HEURISTIQUE SELON LES C(J) DECROISSANTS//
```

```
      600 ECHEC=.TRUE.
          DO 601 J=JD1,JF1
          J1=IP(J);X(J)=C(J1)
      601 CONTINUE
          K1=JD1+1
      620 IF(K1.GT.JF1) GOTO 625
          K2=K1-1
      615 IF(K2.LT.JD1) GOTO 630
          K3=K2+1
          IF(X(K2).GE.X(K3)) GOTO 630
          IPV=IP(K2);IP(K2)=IP(K3);IP(K3)=IPV
          IPV=X(K2);X(K2)=X(K3);X(K3)=IPV
          K2=K2-1; GOTO 615
      630 K1=K1+1; GOTO 620
      625 J=JD1;K0=0
      640 J1=IP(J);IF(J.GT.JF1) GOTO 650
          DO 642 I=1,MS
          I1=JP(I);K=I1+(J1-1)*M;L(I1)=L(I1)-H(K);I0=I
          IF(L(I1)) 644,642,642
      644 DO 643 K1=1,10
          L1=JP(K1);K=L1+(J1-1)*M;L(L1)=L(L1)+H(K)
      643 CONTINUE
          X(J)=0;J=J+1; GOTO 640
      642 CONTINUE
          X(J)=1;J=J+1; K0=K0+1; GOTO 640
      650 L1=JD1; L2=JF1
      660 IF(L1.GT.L2) GOTO 690
          IF(X(L1).EQ.1) L1=L1+1; GOTO 660
      670 IF(L1.EQ.L2) GOTO 690
          IF(X(L2).EQ.0) L2=L2-1; GOTO 670
          IPV=IP(L1);IP(L1)=IP(L2);IP(L2)=IPV
          IPV=X(L1);X(L1)=X(L2);X(L2)=IPV
          L1=L1+1; GOTO 660
      690 JX=JD1+K0-1
          WRITE(108,4)
          GOTO 900
        4 FORMAT(//,5X'AGNES2 :PROCEDURE ECHEC')
      C                  //PROCEDURE ENUMERATIVE//
      800 DO 801 J=JD1,JF1
          J1=IP(J);K=J-JD1+1;PEC(K)=C(J1)
      801 CONTINUE
          DO 802 I=1,MF
          I1=JP(I);PEB(I)=L(I1)
      802 CONTINUE
          DO 803 I=1,MF
          K1=JX2*(I-1); I1=JP(I)
          DO 803 J=JD1,JF1
          K2=J-JD1+1; J1=IP(J);K0=K1+K2;K=I1+(J1-1)*M;PEH(K0)=H(K)
      803 CONTINUE
          N0=JX2*MF;N1=JD1
          CALL LIFO(PEC,PEH,PEB,IP(N1),MF,JX2,N1,N0)
```

218

```
      JX=N1-1
C                  //FIN DE L'ALGORITHME//
  900 Z=0
      DO 901 J=1,JX
      J1=IP(J);Z=Z+C(J1);X(J1)=1
  901 CONTINUE
      DO 902 J=JX+1,N
      J1=IP(J);X(J1)=0
  902 CONTINUE
      DO 903 I=1,M
      KS=0
      DO 904 J=1,JX
      J1=IP(J);K=I+(J1-1)*M;KS=KS+H(K)
  904 CONTINUE
      K=B(I)-KS;IF(K.LT.0) GOTO 999
  903 CONTINUE
      GOTO 2000
  999 WRITE(108,5)
    5 FORMAT(//,5X,'ERREUR DANS AGNES2: SOLUTION NON REALISABLE')
      Z=0
      DO 104 J=1,N
  104 X(J)=0
 2000 WRITE(108,2) KFRAC,Z,ITER
      WRITE(108,3)(X(J),J=1,N)
    2 FORMAT(//,5X,'AGNES2 : KFRAC = ',I2,3X,'MINORANT = ',I9,3X,'ITER= '
     S,I2)
    3 FORMAT(/,5X,'SOLUTION HEURISTIQUE:',50I2)
      RETURN
      END
```

```
      SUBROUTINE RESOL(F,D,B,IP,N1,N,WOPT,QSORT)
C          //RESOLUTION EN CONTINU D'UN KNAPSACK//
      DIMENSION D(N),IP(N),DA(60)
      INTEGER F(N),QSORT
C           //CONTROLE//
      S=0.
      DO 48 J=1,N
   48 S=S+D(J)
      IF(S.LE.B) N1=N+1;WOPT=0.;B=B-S;QSORT=2;GOTO 500
C           //INITIALISATION//
      S=0.;ISEUIL=10;R=0.;QSORT=1
      DO 10 J=1,N
   10 DA(J)=F(J)/D(J)
      IB1=1;IC1=N
C           // TRI //
  100 IB=IB1;IH=IC1
      IF(IH-IB.GE.ISEUIL) GOTO 200
C           // TRI PAR INSERTION //
      J1=IB+1
   20 IF(J1.GT.IH) GOTO 500
      J2=J1-1
   30 IF(J2.LT.IB) GOTO 31
      IF(DA(J2).GE.DA(J2+1)) GOTO 31
      J3=J2+1
      IPV=IP(J2);IP(J2)=IP(J3);IP(J3)=IPV
      IPV=F(J2);F(J2)=F(J3);F(J3)=IPV
      VAL=DA(J2);DA(J2)=DA(J3);DA(J3)=VAL
      VAL=D(J2);D(J2)=D(J3);D(J3)=VAL
      J2=J2-1; GOTO 30
   31 J1=J1+1; GOTO 20
C           // TRI RAPIDE //
  200 J1=IB;J4=(IH+IB)/2;J2=J4
      IF(DA(J1).LE.DA(J2)) GOTO 35
      J1=J4; J2=IB
   35 J3=IH
      IF(DA(J3).GE.DA(J2)) GOTO 36
      J2=IH
      IF(DA(J1).GE.DA(J2)) J2=J1
   36 IPV=IP(J2);VAL=DA(J2);IPR=F(J2);VAR=D(J2)
      IP(J2)=IP(IH);DA(J2)=DA(IH);F(J2)=F(IH);D(J2)=D(IH)
      IF(IB.GE.IH) GOTO 8
    6 IF(DA(IB).LT.VAL) GOTO 7
      R=R+D(IB);IB=IB+1
      IF(IB.GE.IH) GOTO 8
      GOTO 6
    7 IP(IH)=IP(IB);DA(IH)=DA(IB);F(IH)=F(IB);D(IH)=D(IB);
    4 IH=IH-1
      IF(IB.GE.IH) GOTO 8
      IF(DA(IH).LE.VAL) GOTO 4
      IP(IB)=IP(IH);DA(IB)=DA(IH);F(IB)=F(IH);D(IB)=D(IH)
      R=R+D(IH);IB=IB+1
```

220

```
      IF(IB.LT.IH) GOTO 6
    8 IP(IB)=IPV;DA(IB)=VAL;F(IB)=IPR;D(IB)=VAR
      R=R+VAR
      IF(R.LT.B) GOTO 110
      IF(R.EQ.B) GOTO 400
      R=R-D(IB)
      IF(R.LT.B) GOTO 400
      IB=IB-1
      IF(R.EQ.B) IB=IB+1;GOTO 400
      R=S;IC1=IB;GOTO 100
  110 IB1=IB+1;S=R; GOTO 100
C           //FIN TRI PAR INSERTION//
  300 QSORT=0
      DO 302 J=IB,IH
      K=J;R=R+D(J)
  302 IF(R.GE.B) GOTO 310
  310 IF(R.EQ.B) IB=K+1;GOTO 400
      IB=K;R=R-D(K)
C           //INDICE DE BASE,MULTIPLICATEUR OPTIMAL//
  400 WOPT=DA(IB);N1=N1+IB-1;B=B-R
  500 RETURN
      END
```

```
      SUBROUTINE SELPPGN(E,KP,N1,N,P)
C        //SELECTIONNER LES P PLUS GRANDS ELEMENTS PARMI N//
      DIMENSION E(N),KP(N)
      INTEGER P
      ISEUIL=10;IB1=1;IC1=N
   10 IB=IB1;IH=IC1
      IF(IH-IB.GE.ISEUIL) GOTO 50
      J1=IB+1
   20 IF(J1.GT.IH) GOTO 100
      J2=J1-1
   30 IF(J2.LT.IB) GOTO 35
      J3=J2+1;IF(E(J2).GE.E(J3)) GOTO 35
      IPV=KP(J2);KP(J2)=KP(J3);KP(J3)=IPV
      VAL=E(J2);E(J2)=E(J3);E(J3)=VAL
      J2=J2-1; GOTO 30
   35 J1=J1+1; GOTO 20
   50 J1=IB;J4=(IH+IB)/2;J2=J4
      IF(E(J1).LE.E(J2)) GOTO 55
      J1=J4;J2=IB
   55 J3=IH
      IF(E(J3).GE.E(J2)) GOTO 56
      J2=IB
      IF(E(J1).GE.E(J2)) J2=J1
   56 VAL=E(J2);IPV=KP(J2)
      E(J2)=E(IH);KP(J2)=KP(IH)
      IF(IB.GE.IH) GOTO 80
   60 IF(E(IB).LT.VAL) GOTO 70
      IB=IB+1
      IF(IB.GE.IH) GOTO 80
      GOTO 60
   70 E(IH)=E(IB);KP(IH)=KP(IB)
   40 IH=IH-1
      IF(IB.GE.IH) GOTO 80
      IF(E(IH).LE.VAL) GOTO 40
      E(IB)=E(IH);KP(IB)=KP(IH);IB=IB+1
      IF(IB.LT.IH) GOTO 60
   80 E(IB)=VAL;KP(IB)=IPV
      K0=IB-P;IF (K0) 90,100,110
   90 IB1=IB+1; GOTO 10
  110 IC1=IB; GOTO 10
  100 RETURN
      END
```

222

```
      FUNCTION NORME(E,N)
C               //CALCUL DE LA NORME L2 D'UN VECTEUR //
      DIMENSION E(N)
      REAL NORME
      R=0.
      DO 1 J=1,N
    1 R=R+E(J)**2
      IF(R.LE.0.) WRITE(108,2); STOP
      NORME=SQRT(R)
    2 FORMAT(/,5X,'CALCUL DE LA NORME VECTEUR NUL;ABANDON')
      RETURN
      END
```

```
      SUBROUTINE REDUCOM(C,H,B,M,N,MN,PI,PM,NITER,VALFIN,EPS2)
C          //RESOLUTION DU DUAL COMPOSITE PAR UN
C            ALGORITHME DE TYPE QUASI SOUS-GRADIENT//
      INTEGER C(N),H(MN),B(M)
      DIMENSION PI(M),PM(M)
      DIMENSION A(60),X(60),IP(60),G(30),P(30),Q(30)
      INTEGER  VALFIN,QSORT
      REAL NORME
      LOGICAL DIMPAS
C                 //INITIALISATION//
      WRITE(108,74) (PI(I),I=1,M)
   74 FORMAT(5X,'MULTIPLICATEURS INITIAUX',10F10.4)
      INIT=0;NSW=INT(NITER/2)
 2000 PAR1=1.;ITER=1;QSORT=0
      R=NORME(PI,M)
      DO 10 I=1,M
      PI(I)=PI(I)/R;P(I)=PI(I)
   10 PM(I)=PI(I)
C                 //CALCUL D'UN QUASI-SOUS-GRADIENT//
 1000 DIMPAS=.TRUE.
      DO 12 J=1,N
      S=0.
      DO 13 I=1,M
      K=I+(J-1)*M;S=S+PI(I)*H(K)
   13 CONTINUE
   12 A(J)=S
      R=0.
      DO 14 I=1,M
   14 R=R+PI(I)*B(I)
      DO 16 J=1,N
      IP(J)=J;X(J)=0.
   16 CONTINUE
      N1=1;CALL RESOL(C,A,R,IP,N1,N,WOPT,QSORT)
      SUP=0.
      DO 18 J=1,N1-1
      SUP=SUP+C(J)*X(K);K=IP(J);X(K)=1.
   18 CONTINUE
      SUP=SUP+WOPT*R;K=IP(N1);X(K)=R/A(N1)
      DO 17 J=1,N
      K=IP(J);A(K)=C(J)
   17 CONTINUE
      DO 19 J=1,N
   19 C(J)=A(J)
      IF(ITER.EQ.1) VSUP=SUP;VSUP0=SUP; GOTO 45
      IF(SUP.GE.VSUP) GOTO 45
      DIMPAS=.FALSE.;VSUP=SUP
      DO 42 I=1,M
   42 PM(I)=PI(I)
   45 DO 46 I=1,M
      S=0.
      DO 47 J=1,N
```

224

```
      K=I+(J-1)*M;S=S+H(K)*X(J)
   47 CONTINUE
   46 G(I)=S-B(I)
      IF(WOPT.NE.0) GOTO 60
      DO 50 I=1,M
      IF(G(I)) 50,50,60
   50 CONTINUE
      WRITE(108,1)VSUP; WRITE(108,2)(X(J),J=1,N);STOP
    1 FORMAT(//,5X,' UNE SOLUTION OPTIMALE DU PROBLEME INITIAL
     S EST DETERMINEE; VALEUR OPTIMALE :',F11.1)
    2 FORMAT(//,5X,20I2)
C                 //CALCUL D'UN NOUVEAU MULTIPLICATEUR//
   60 DO 61 K=1,M
      S=0.
      DO 62 I=1,M
   62 S=S+G(I)*PI(I)
   61 G(K)=G(K)-S*PI(K)
      R=NORME(G,M)
      DO 63 I=1,M
   63 G(I)=G(I)/R
      IF(DIMPAS) PAR1=PAR1*.9
      DO 64 I=1,M
      PI(I)=PI(I)+PAR1*G(I)
      IF(PI(I)) 65,64,64
   65 PI(I)=-PI(I)
   64 CONTINUE
      R=NORME(PI,M)
      DO 66 I=1,M
   66 PI(I)=PI(I)/R
C                 //TEST  D'ARRET//
      IF(ITER.NE.10.OR.ITER.NE.11.OR.INIT.EQ.1) GOTO 120
      DO 110 I=1,M
  110 Q(I)=PI(I)-P(I)
      R=NORME(Q,M)
      WRITE(108,78) R
   78 FORMAT(5X,'NORME(P-PI) APRES 10 ET 11 ITERATIONS',F14.6)
      IF(R.LE.EPS2) GOTO 120
      WRITE(108,130)
  130 FORMAT(//,5X,'REINITIALISATION')
      DO 112 I=1,M
  112 PI(I)=1.
      INIT=1;GOTO 2000
  120 IF(ITER.GE.NITER) GOTO 140
      IF(ITER.EQ.NSW.AND.VSUP.GE.VSUP0) WRITE(108,76)NSW;GOTO140
   76 FORMAT(5X,'VALEUR NON AMELIOREE PENDANT ',I4,' ITERATIONS:ABANDON')
      ITER=ITER+1;GOTO 1000
  140 R=VSUP+VALFIN
      WRITE(108,3) VSUP,R
    3 FORMAT(//,5X,'MEILLEURE VALEUR DE LA RELAXATION DU PROBLEME COURANT
     S: ',F14.6,' DU PROBLEME INITIAL: ',F14.6)
      RETURN
```

```
      SUBROUTINE REDULAG(C,H,B,M,N,MN,VPINF,XH,PI,PM,NITER,VALFIN,EPS1)
C                  //RESOLUTION DU DUAL LAGRANGIEN PAR UN
C                   ALGORITHME DE TYPE SOUS-GRADIENT//
      INTEGER C(N),H(MN),B(M),XH(N)
      DIMENSION PI(M),PM(M)
      DIMENSION X(90),G(30),DES(30),P(30),Q(30)
      INTEGER VPINF,VALFIN
      LOGICAL DIMPAS
      REAL NORME
C                  //INITIALISATION//
      WRITE(108,81) (PI(I),I=1,M)
   81 FORMAT(5X,'MULTIPLICATEURS INITIAUX  ',10F10.4)
   82 FORMAT(5X,'PARAMETRES',I8,I6,I8,F6.4)
      NSW=INT(NITER/2)
 2000 PAR1=2.;PAR3=0;NCAM=5; ITER=1;INIT=1
      DO 110 I=1,M
      PM(I)=PI(I);P(I)=PI(I);IF(P(I).GT.0.) INIT=0
  110 CONTINUE
C                  //CALCUL D'UN SOUS-GRADIENT//
 1000 SUP=0.;DIMPAS=.TRUE.
      DO 10 K=1,M
   10 SUP=SUP+B(K)*PI(K)
      DO 12 J=1,N
      S=0.
      DO 14 I=1,M
      K=I+(J-1)*M;S=S+PI(I)*H(K)
   14 CONTINUE
      X(J)=C(J)-S
      IF(X(J).LE.0) X(J)=0.; GOTO 12
      SUP=SUP+X(J); X(J)=1.
   12 CONTINUE
      IF(ITER.EQ.1) VSUP=SUP;VSUP0=SUP; GOTO 20
      IF(SUP.GE.VSUP) GOTO 20
      DIMPAS=.FALSE.;VSUP=SUP
      DO 112 I=1,M
  112 PM(I)=PI(I)
   20 DO 25 I=1,M
      KS=0
      DO 26 J=1,N
      K=I+(J-1)*M;KS=KS+H(K)*X(J)
   26 CONTINUE
   25 G(I)=B(I)-KS
      DO 30 I=1,M
      IF(G(I)) 35,30,35
   30 CONTINUE
      WRITE(108,1) VSUP;WRITE(108,2)(X(J),J=1,N);STOP
    1 FORMAT(/,5X,' UNE SOLUTION OPTIMALE DU PROBLEME COURANT
     S EST DETERMINEE; VALEUR OPTIMALE :',F11.1)
    2 FORMAT(/,5X,20I2)
   35 DO 40 I=1,M
      IF(G(I)) 50,40,40
```

```
      40 CONTINUE
         KS=0
         DO 42 J=1,N
      42 KS=KS+C(J)*X(J)
         IF(KS.LE.VPINF) GOTO 50
         DO 44 J=1,N
      44 XH(J)=X(J)
C                //CALCUL D'UN NOUVEAU MULTIPLICATEUR//
      50 IF(ITER.GT.1) GOTO 60
         DO 52 I=1,M
      52 DES(I)=G(I)
         PAR2=0.;GOTO 100
      60 S=0.;R=0.
         IF(ITER.GT.NBAM) PAR3=1.5
         DO 62 I=1,M
      62 S=S+DES(I)*G(I)
         IF(S.GE.0.) PAR2=0.; GOTO 90
         DO 64 I=1,M
      64 R=R+DES(I)**2
         PAR2=-(PAR3*S)/R
      90 DO 66 I=1,M
         G(I)=G(I)+PAR2*DES(I);DES(I)=G(I)
      66 CONTINUE
     100 S=0.
         DO 70 I=1,M
      70 S=S+G(I)**2
         R=SUP-VPINF
         DO 72 I=1,M
         PI(I)=PI(I)-(PAR1*R*G(I))/S
         IF(PI(I)) 74,72,72
      74 PI(I)=0.
      72 CONTINUE
C                //TEST D'ARRET//
         IF(ITER.NE.10.OR.ITER.NE.11.OR.INIT.EQ.1) GOTO 120
         DO 114 I=1,M
     114 Q(I)=P(I)-PI(I)
         R=NORME(Q,M);R1=NORME(P,M);R2=NORME(PI,M)
         R=R/R1/R2
         WRITE(108,76) R
      76 FORMAT(5X,'NORME (P-PI) APRES 10 ET 11 ITERATIONS =',F12,6)
         IF(R.LE.EPS1) GOTO 120
         WRITE(108,130)
     130 FORMAT(/,5X,'REINITIALISATION')
         DO 116 I=1,M
     116 PI(I)=0.
         GOTO 2000
     120 IF(ITER.GE.NITER) GOTO 200
         IF(ITER.EQ.NSW.AND.VSUP.GE.VSUP0) WRITE(108,78) NSW;GOTO 200
      78 FORMAT(5X,'VALEUR NON AMELIOREE PENDANT ',I4,' ITERATIONS')
         IF(DIMPAS) PAR1=PAR1*.87
         ITER=ITER+1; GOTO 1000
```

228

```
      200 R=VSUP+VALFIN
          WRITE(108,4)VSUP,R
        4 FORMAT(/,5X,'MEILLEURE VALEUR DE LA RELAXATION DU PROBLEME COURANT
     S: ',F14.6,' DU PROBLEME INITIAL: ',F14.6)
          RETURN
          END
```

```
      SUBROUTINE LIFO(LC,LH,LB,KP,M,N,NX,MN)
C          //PROCEDURE ENUMERATIVE IMPLICITE UTILISANT LE
C          PRINCIPE ENUMERATIF,LAST IN FIRST OUT //
      DIMENSION LC(N),LB(M),LH(MN),KP(N)
      DIMENSION KS(30)
      INTEGER PF(10),XC(10),X(10)
      INTEGER ZINF,ZP
C            //ORDONNEMENT SUIVANT LES C(J) DECROISSANTS //
      K1=2
   20 IF(K1.GT.N) GOTO 25
      K2=K1-1
   15 IF(K2.LE.0) GOTO 30
      K3=K2+1
      IF(LC(K2).GE.LC(K3)) GOTO 30
      IPV=KP(K2);KP(K2)=KP(K3);KP(K3)=IPV
      IPV=LC(K2);LC(K2)=LC(K3);LC(K3)=IPV
      DO 16 I=1,M
      K=(I-1)*N;K4=K+K2;K5=K+K3
      IPV=LH(K4);LH(K4)=LH(K5);LH(K5)=IPV
   16 CONTINUE
      K2=K2-1; GOTO 15
   30 K1=K1+1; GOTO 20
C          // CALCUL D'UN MINORANT //
   25 ZINF=0
      DO 21 I=1,M
   21 KS(I)=0
      DO 22 J=1,N
      DO 23 I=1,M
      IO=I;KO=(I-1)*N;KS(I)=KS(I)+LH(KO+J)
      IF(LB(I)-KS(I)) 24,23,23
   24 DO 26 LO=1,IO
      KO=(LO-1)*N;KS(LO)=KS(LO)-LH(KO+J)
   26 CONTINUE
      X(J)=0; GOTO 22
   23 CONTINUE
      ZINF=ZINF+LC(J);X(J)=1
   22 CONTINUE
C            //INITIALISATION DE L'ENUMERATION IMPLICITE //
      ZP=0;INDEX=0
      DO 10 J=1,N
      PF(J)=0;XC(J)=-1
   10 CONTINUE
C          //DESCENTE DANS L'ARBORESCENCE //
  100 DO 110 J=1,N
      IF(XC(J).LT.0) JS=J; GOTO 150
  110 CONTINUE
      GOTO 800
  150 NSUP=0
      DO 155 J=1,N
  155 IF(XC(J).NE.0) NSUP=NSUP+LC(J)
  200 IF(NSUP.LE.ZINF) GOTO 800
```

230

```
      ISAT=0
      DO 210 I=1,M
      K0=(I-1)*N;IFIX=LB(I)-LH(K0+JS)
      IF(IFIX) 230,220,210
  220 ISAT=1
  210 CONTINUE
      INDEX=INDEX+1;PF(INDEX)=JS;XC(JS)=1;ZP=ZP+LC(JS)
      DO 211 I=1,M
      K0=(I-1)*N;LB(I)=LB(I)-LH(K0+JS)
  211 CONTINUE
      GOTO 300
  230 INDEX=INDEX+1;XC(J)=0;PF(INDEX)=-JS;NSUP=NSUP-LC(JS);JS=JS+1
      IF(N-JS) 800,200,200
C          //REDUCTION//
  300 IF(ISAT.EQ.0) GOTO 400
      DO 310 J=1,N
      IF(XC(J).LT.0) XC(J)=0;INDEX=INDEX+1;PF(INDEX)=-J
  310 CONTINUE
      GOTO 700
  400 DO 410 J=1,N
      IF(XC(J).GE.0) GOTO 410
      DO 420 I=1,M
      K0=(I-1)*N
      IF(LH(K0+J).GT.LB(I)) XC(J)=0;INDEX=INDEX+1;PF(INDEX)=-J;GOTO410
  420 CONTINUE
  410 CONTINUE
      IF(INDEX.EQ.N) GOTO 700
  600 DO 601 I=1,M
  601 KS(I)=0
      DO 610 J=1,N
      IF(XC(J)) 605,610,610
  605 DO 602 I=1,M
      K0=(I-1)*N;KS(I)=KS(I)+LH(K0+J)
  602 CONTINUE
  610 CONTINUE
      DO 620 I=1,M
      IF(KS(I).GT.LB(I)) GOTO 100
  620 CONTINUE
      DO 630 J=1,N
      IF(XC(J)) 631,630,630
  631 ZP=ZP+LC(J);XC(J)=1;INDEX=INDEX+1;PF(INDEX)=-J
      DO 632 I=1,M
      K0=(I-1)*N;LB(I)=LB(I)-LH(K0+J)
  632 CONTINUE
  630 CONTINUE
C          //REMONTEE DANS L'ARBORESCENCE//
  700 IF(ZP.LE.ZINF) GOTO 800
      ZINF=ZP
      DO 701 J=1,N
  701 X(J)=XC(J)
  800 DO 810 J=N,1,-1
```

```
      IF(PF(J)) 820,810,830
  820 IF(J.NE.INDEX) GOTO 890
      J1=-PF(J);INDEX=INDEX-1;PF(J)=0
      IF(XC(J1)) 890,840,850
  850 ZP=ZP-LC(J1)
      DO 851 I=1,M
      K0=(I-1)*N;LB(I)=LB(I)+LH(K0+J1)
  851 CONTINUE
  840 XC(J1)=-1
      IF(INDEX) 900,900,810
  810 CONTINUE
  830 IF(J.NE.INDEX) GOTO 890
      J2=PF(J);PF(J)=-J2;XC(J2)=0;ZP=ZP-LC(J2)
      DO 831 I=1,M
      K0=(I-1)*N;LB(I)=LB(I)+LH(K0+J2)
  831 CONTINUE
      GOTO 100
  890 WRITE(108,1); STOP
    1 FORMAT(//,5X,'ERREUR DANS LE SOUS-PROGRAMME LIFO')
C          //RETOUR AU PROGRAMME APPELANT //
  900 L1=1;L2=N
  910 IF(L1.GT.L2) GOTO 950
      IF(X(L1).EQ.1) L1=L1+1; GOTO 910
  920 IF(L1.EQ.L2) GOTO 950
      IF(X(L2).EQ.0) L2=L2-1; GOTO 920
      IPV=KP(L1);KP(L1)=KP(L2);KP(L2)=IPV
      IPV=X(L1);X(L1)=X(L2);X(L2)=IPV
      L1=L1+1; GOTO 910
  950 K0=0
      DO 951 J=1,N
      IF(X(J).EQ.0) GOTO 960
  951 K0=K0+1
  960 NX=NX+K0
      RETURN
      END
```

232

BIBLIOGRAPHIE

[1] AGMON S., *"The relaxation method for linear inequalities"*, Canadian Journal of mathematics 6 (1954) pp 382-392.

[2] BALAS E., MARTIN C.H., *"Pivot and complement - a heuristic 0-1 programming"*, Management Sciences research report n° 414 (1978).

[3] CAMERINI P.M., FRATTA L., MAFFIOLI F., *"On improving relaxation methods by modified gradient techniques"*, Mathematical programming study 3 (1975) pp 26-34.

[4] CLARKE F.H., *"Generalized gradients and applications"*, Trans. A.M.S. 205 (1975) pp 247-262.

[5] CROWDER H., JOHNSON E.L., PADBERG M.W., *"Solving large-scale zero-one linear programming problems"*, IBM research report RC 8888 Yorktown Heights (february 1982).

[6] DYER M.E. *"Calculating surrogate constraints"*, Mathematical programming Vol 19 (1980) pp 255-278.

[7] ETCHEBERRY J., CONCA C., STACCHETTI E., *"An implicit enumeration approach for integer programming using subgradient optimization"* Rapport N° 78/04/C. Université de Santiago-Chili (1978).

[8] FAYARD D., PLATEAU G., *Techniques de résolution du problème du knapsack en variables bivalentes"*. Publication 91 du laboratoire de calcul de l'Université des Sciences et Techniques de Lille (1977).

[9] FAYARD D., PLATEAU G., *"Contribution à la résolution des programmes mathématiques en nombres entiers"*, Thèse d'état (1979).

[10] FAYARD D., PLATEAU G.,*"An algorithm for the solution of the 0-1 knapsack problem"*, Computing 28 (1982) pp 269-287.

[11] FISHER M.L., NORTHUP W.D., SHAPIRO J.F., *"Using duality to solve discrete optimization problems : theory and computational experience"*, Mathematical programming study 3 (1975) p.56-94.

[12] FLEISHER J., Sigmap Neusletter n° 20 (February 1976).

[13] FREVILLE A., PLATEAU G., *"Méthodes heuristiques performantes pour les problèmes en variables 0-1 à plusieurs contraintes en inégalité"*. Publication ANO-91 (décembre 1982).

[14] FREVILLE A., PLATEAU G., *"Heuristics and reduction methods for 0-1 multiple constraints linear programming problems"*, Publication ANO-111 (juillet 1983).

[15] GARFINKEL R.S., NEMHAUSER G.L., *"Integer-programming"*, Wiley Interscience Publication (1972).

[16] GEOFFRION A.M., *"Duality in nonlinear programming"*, SIAM Review 13 (1971) pp 1-37.

[17] GEOFFRION A.M., *"Lagrangean relaxation for integer programming"* Mathematical programming study 2 (1974) pp 82-114.

[18] GLOVER F., *"A multiphase-dual algorithm for the 0-1 integer programming problem"*, Operations research 13 (1965) pp 879-919.

[19] GLOVER F., *"Surrogate constraints"*, Operations research 16 (1968) pp 741-749.

[20] GLOVER F., *"Surrogate constraints duality in mathematical programming"*, Operations research 23 (1975) pp 434-451.

[21] GLOVER F., *"Heuristics for integer programming using surrogate constraints"*, Decision sciences, vol 8 (1977) pp 156-166.

[22] GOFFIN J.L. *"On convergence rates of subgradient optimization methods"* Mathematical programming 13 (1977) pp 329-347.

[23] GOFFIN J.L. *"The relaxation methods for solving systems of linear inequalities"*, Mathematics of operations research Vol 5 n° 3 (august 1980)

[24] GONDRAN M., MINOUX M., *"Graphes et algorithmes"*, Collection de la Direction des études et recherches d'Electricité de France, Editions Eyrolles (1979).

[25] GUIGNARD M., *"Méthodes heuristiques de résolution d'un système d'inégalités linéaires en variables entières ou bivalentes"*, Rapport n° 32 - Laboratoire de Calcul de l'Université des Sciences et Techniques de Lille (janvier 1972).

[26] GREENBERG H., HEGERICH R.L., *"A branch search algorithm for the knapsack problem"*, Management science - Vol 16 n° 5 (1970) pp 327-332.

[27] GREENBERG H.J., PIERSKALLA W.P.,*"Surrogate mathematical programs"*, Opérations research 18 (1970) pp 924-939.

[28] GREENBERG H.J., PIERSKALLA W.P., *"A review of quasi-convex functions"*, Operations research 19 (1971) pp 1553-1570.

[29] GREENBERG H.J., PIERSKALLA W.P., *"Quasi-conjugate functions and surrogate duality"*, Cahiers du centre d'étude de recherches opérationnelle 15 (1973) pp 437-448.

[30] GRINOLD R.C., *"Lagrangian Subgradients"*, Management science - Vol 17 - n° 3 (November 1970) pp 185-188.

[31] HELD M., KARP R.M., *"The travelling-salesman problem and minimum spanning trees : Part II"*, Mathematical programming 1 (1971) pp 6-25.

[32] HELD M., WOLFE P., CROWDER H.P., *"Validation of subgradient optimization"*, Mathematical programming 6 (1974) pp 62-88.

[33] HILLIER F.S., *"Efficient heuristics procedures for integer linear programming with an interior path"*, Operations research 17 (1969) pp 600-636.

[34] HUARD P., *"La méthode simplex sans inverse explicite"*, Bulletin de la direction des études et recherches EDF, Série C n° 2 (1979) pp 79-98.

[35] HUARD P., *"Complements concernant la méthode des paramètres"*, Bulletin de la direction des études et recherches EDF, Série C n° 2 (1980) pp 63-68.

[36] HUARD P., *"Optimisation dans $\mathbb{R}^n$-éléments théoriques"*, Publication du Laboratoire de Calcul de l'Université des Sciences et Techniques de Lille (Janvier 1972).

[37] KARWAN M.H., RARDIN R.L., *"Surrogate duality in a branch and bound procedure"*, Naval Research logistics quaterly. 28(1) (1981).

[38] KARWAN M.H., RARDIN R.L., *"Some relationships between lagrangian and surrogate duality in integer linear programming"*, Mathematical programming 17 (1979) pp 320-324.

[39] KHACHYAN L., *"A polynomial algorithm in linear programming"*, Doklady Akademiia Nauk SSSR 244 (n° 5, february 1979) pp 1093-96.

[40] KIEHM J.L., VINCKE P., *"La méthode de Khachyan : description et justification mathematique"*, Cahiers du C.E.R.O. Vol n° 2 (1981).

[41] KIEHM J.L., VINCKE P., *"Méthode ellipsoidale et algorithme du simplexe : complexité théorique et pratique"*, Cahiers du C.E.R.O. vol 24, n° 2-3-4 (1982).

[42] KOCHENBERGER G.A., Mc KARL B.A., WYMAN F.P., *"A heuristic for general integer programming"*, Decision sciences - Vol 5 - n° 1 (january 1974) pp 36-44.

[43] LEGENDRE J.P., MINOUX M., *"Une application de la notion de dualité en programmation en nombres entiers : sélection et affectation optimales d'une flotte d'avions"*, RAIRO 11 n° 2, pp 201-222.

[44] LEMARECHAL C., *"Méthodes de sous-gradients"*, Bulletin direction et recherche EDF Série C n° 2 pp 5-14 (1974).

[45] LEMARECHAL C., *"An extension of Davidon methods to non differentiable problems"*, Mathematical programming study 3 (1975) pp 95-109.

[46] LOULOU R., MICHAELIDES E., *"New greedy-like heuristics for the multidimensional 0-1 knapsack problem"*, Operations research - Vol 27 - n° 6 (November 1979) pp 1101 -1114.

[47] LUENBERGER D.G. *"Quasi-convex programming"*, SIAM Journal of Applied Mathematics 16 (1968) pp 1090-1095.

[48] MARSTEN R.E., *"The use of the boxstep method in discrete optimization"* Mathematical programming Study 3 (1975) pp 127-144.

[49] MICHAELOPOULOS M., *"Méthodes de sous-gradient dans les problèmes d'optimisation avec contraintes : application à la programmation linéaire"*, Thèse de 3ème cycle - Université Scientifique et médicale de Grenoble (1982)

[50] MINOUX M., SEREAULT J.Y., *"Subgradients optimization and large scale programming : application to multicommodity network synthesis with security constraints"*, RAIRO Vol 15 (n° 2)(1980) pp 185-203.

[51] MINOUX M. *"Un algorithme de sous-gradients pour la recherche d'un point satisfaisant les conditions d'optimalité en optimisation non différentiable"*. Rapport interne. Centre National d'Etudes des Télécommunications (1982).

[52] MOTZKIN T., SCHOENBERG I.J., *"The relaxation method for linear inequalities"*, Canadian Journal of mathematics 6 (1954).

[53] PETERSEN C.C., *"Computational experience with variants of the Balas algorithm applied to the selection of R and D projects"*. Management science - Vol 13 - n° 9 (mai 1967) pp 736-750.

[54] PLATEAU G., *"Réduction de la taille des problèmes linéaires en variables 0-1"*, Publication 71 du Laboratoire de Calcul de l'Université des Sciences et Techniques de Lille 1 (1976).

[55] POLYAK B.T., *"A general method of solving extremum problems"*, Doklady Akademii Nauk SSSR 174 (1967) pp 33-36.

[56] POLYAK B.T., *"Minimization of unsmooth functionals"*, USSR computational Mathematics and Mathematics Physics 9 (1969) pp 509-521.

[57] RINNOOY KAN A.H.G., TELGEN J., *"The complexity of linear programming"* Statistica Neerlandica n° 2 (1981).

[58] ROCKAFELLAR R.T., *"Convex analysis"*, Princeton University Press, Princeton, N.J. (1970).

[59] SENJU S., TOYODA Y., *"An approach to linear programming with 0-1 variables"*, Management science Vol 15 n° 4 (1968) pp 196-207.

[60] SHAPIRO J.F., *"Mathematical programming : Structures and algorithms"* Wiley-Interscience Publication (1979).

[61] SHIH W., *"A branch and bound method for the multiconstraint zero-one knapsack problem"*. Journal of the operational research society - vol 30 - n° 4 (1979) pp 369-378.

[62] SHOR N.Z., *"Generalized Gradient methods of nondifferentiable optimization employing space dilatation operations"*, Mathematical programming. The state of the art - Bonn 1982 - pp 501-529.

[63] SPIELBERG K., *"Enumerative methods in integer programming"*, Annals of discrete mathematics 5 (1979) pp 139-183.

[64] THESEN ARNE, *"A recursive branch and bound algorithm for the multi-dimensional knapsack problem"*, Working paper Departement of Industrial Engineering, University of Wisconsin.

[65] TOYODA Y., *"A simplified algorithm for obtaining approximate solutions to 0-1 programming problems"*, Management science - Vol 21 - n° 12 (9175) pp 1417-1427.

[66] WEINGARTNER H.M., NESS D.N., *"Methods for the solution of the multi-dimensional 0-1 knapsack problem"*, Operations research Vol 15 n° 1 (1967) pp 83-103.

[67] WOLFE P., *"A method of conjugate subgradients for minimizing non-differentiable functions"*, Mathematical programming study 3 (1975) pp 143-173.