THÈSE

en vue de l'obtention du titre de

DOCTEUR DE L'UNIVESITÉ DE PAUL VERLAINE-METZ

(arrêté ministériel du 30 October 1993)

Spécialité Informatique

présentée par

NGUYEN Trong Phuc

Titre de la thèse :

TECHNIQUES D'OPTIMISATION EN TRAITEMENT D'IMAGE ET VISION PAR ORDINATEUR ET EN TRANSPORT LOGISTIQUE

Date de soutenance : 04 Juillet 2007

Composition du Jury :

Président	PHAM DINH Tao	*Professeur, INSA-Rouen*
Rapporteurs	Van-Dat CUNG	*Professeur, INP de Grenoble*
	Adnan YASSINE	*Professeur, Université du Havre*
Examinateurs	Abdelaziz BENSRHAIR	*Professeur, INSA-Rouen*
	Anass NAGIH	*Professeur, Université de Paul Verlaine-Metz*
	Amedeo NAPOLI	*Directeur de Recherche CNRS au LORIA*
Directeur de thèse	LE THI Hoai An	*Professeur, Université de Paul Verlaine-Metz*

Thèse préparée au sein de laboratoire
d'Informatique Théorique et Appliquée (LITA)
Université de Paul Verlaine-Metz

Remerciements

La préparation de cette thèse, sous la direction de Madame le Professeur LE THI Hoai An, a été faite au sein du laboratoire LITA de l'université Paul Verlaine-Metz.

Je voudrais exprimer ma dette à l'égard de Madame LE THI Hoai An, pour son aide inestimable, les conseils pertinents et formateurs, les encouragements qu'elle m'a donnés tout au long de ces 4 années. Je lui adresse toute ma gratitude pour la confiance dont elle m'a témoignée, pour la liberté qu'elle m'a donnée durant ce travail.

Je tiens à remercier plus particulièrement Monsieur le Professeur PHAM DINH Tao, directeur de l'équipe Modélisation et Optimisation Appliquée de l'INSA de Rouen pour ses conseils et son attention constante. Je lui exprime toute ma reconnaissance pour sa sympathie et les discussions très intéressantes qu'il a menées pour me suggérer les voies de recherche.

Je souhaite également exprimer ma gratitude à Monsieur le Professeur Van-Dat CUNG de l'Institut National Polytechnique de Grenoble et à Monsieur le Professeur Adnan YASSINE de l'université du Havre d'avoir accepté la charge de rapporteur de ma thèse, et d'avoir jugé mon travail.

Je tiens aussi à remercier Monsieur Amedeo NAPOLI, Directeur de Recherche CNRS au LORIA, Monsieur Anass NAGIH, Professeur à l'université de Paul Verlaine-Metz et enfin Monsieur Abdelaziz BENSRHAIR, Professeur à l'INSA de Rouen pour leur disponibilité et pour avoir accepté de faire parti de ce jury.

Je n'oublie pas de remercier toute l'équipe du personnel de l'école de Transport et de Communication d'Hanoï de m'avoir apporter de soutien.

Je me permets également de remercier l'ambassade de France au VIETNAM, la Région Haute-Normandie et l'Université de Paul Verlaine-Metz pour l'aide financière qu'elles m'ont attribuée.

Je témoigne toute mon affectation et reconnaissance à ma famille pour les sacrifices qu'elle a faite pour me soutenir lors des moments difficiles.

Je remercie tous mes collègues français et vietnamiens rencontrés à Rouen ainsi qu'à Metz pour les moments agréables lors de mon séjour en France. Je remercie particulièrement Minh, Nam, Vinh, Duy, Damien pour le partage dans le travail et dans la vie. Enfin je remercie tous ceux qui m'ont aidé de près ou de loin et tous ceux qui m'ont motivé même inconsciemment.

Liste des Publications et Conférences Internationales

NGUYEN Trong Phuc, 02/10/1976, HaiPhong, VietNam

Publications

- LE THI Hoai An, NGUYEN Trong Phuc and PHAM DINH Tao, *A continuous DC programming approach to the strategic supply chain design problem from qualified partner set,* European Journal of Operational Research, In Press, Available online 12 June 2006.
- LE THI Hoai An, OUANES Mohand and NGUYEN Trong Phuc, *A Brand and Bound method for Multivariate global optimization with box constraints,* Actes du Colloque International sur l'Optimisation et système d'Information, COSI'2006, pp 325-336, 2006.
- LE THI Hoai An, LE HOAI Minh, NGUYEN Trong Phuc and PHAM DINH Tao, *Noisy image segmentation by a robust Fuzzy C-Means clustering algorithm based DC programming and DCA,* Submitted to Optimization and Engineering.

Communications dans les Conférences Internationales et Nationales

- Les 12èmes journées de Mathématiques de l'Optimisation et de la Décision (Groupe MODE 2004), 25-27 Mars 2004, Le Havre.

  Une approche du problème de l'optimisation DC pour la conception d'une chaîne d'approvisionnement multi niveaux.
- The Fifth International Conference on Computer Sciences, Modelling, Computation and Optimization in Information Systems and Management Sciences(MCO-2004) July 1-3, 2004, Metz, France.

  A DC programming approach to the strategic supply chain design problem from qualified partner set.
- Les septièmes congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision 6-8 Février 2006 à Lille.

  Reconstruction d'images binaires : une nouvelle approche de l'optimisation DC.
- Les cinquièmes journées Francophones de Recherche Opérationnelle (FRANCORO) et le huitième congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision 20-23 Février 2007 à Grenoble.

  Segmentation des images IRM par la classification floue via DCA.
- Journées de Metz 2007, PDE and variational methods in image analysis, 3-5 May 2007, Metz.

  Noisy Image Segmentation by Fuzzy C-Means Clustering based DCA.

Table des matières

Table des figures

Liste des tableaux

Introduction générale

Les travaux présentés dans cette thèse concernent les nouvelles techniques d'optimisation pour la résolution de quatre problèmes importants issus de deux domaines - transport logistique et vision et traitement d'image :

1. *L'approvisionnement d'une chaîne logistique de multi-niveaux* : l'objectif est de concevoir la chaîne d'approvisionnements en choisissant un partenaire à chaque étape parmi les partenaires potentiels. Ce choix doit satisfaire la demande prévue sur l'horizon donné sans retard.
2. *La discrète tomographie appliquée à la construction d'image binaire* : la discrète tomographie se définit comme la reconstruction d'un sous ensemble de $\mathbb{Z}^n$ à partir de ses projections. Un de ses problèmes principaux est de reconstruire une matrice binaire $\mathbb{Z}^2$ à partir seulement de deux projections orthogonales, la projection horizontale H et la projection verticale V. La problématique est alors la suivante : Étant données les projections orthogonales H, V, existe-t-il une image binaire I ayant pour projections H et V et comment la reconstruire ?
3. *La segmentation d'image* : elle se définit comme un processus de découpage d'une image en régions connexes présentant une homogénéité selon un certain critère, comme par exemple les critères de texture et/ou de couleur. Dans ce domaine, de nombreuses méthodes basées sur différentes approches telles que contour, région, texture,..., ont été développées au cours de ces dernières années ([141]). Nous traitons ce problème par une approche de classification floue qui consiste en minimisation d'un critère non convexe.
4. L'estimation de la matrice fondamentale : la matrice fondamentale est un concept-clé pour toutes les questions touchant à l'emploi d'images non calibrées prises de points de vue multiples. Elle contient toute l'information géométrique disponible et permet d'obtenir la géométrie épipolaire à partir de deux vues perspectives non calibrées. Le but de ce problème est de reconstruire le modèle 3D numérique de la scène à partir de deux vues d'une même scène.

Il s'agit des problèmes d'optimisation non convexe de très grande dimension (sauf pour le dernier problème qui est de taille modeste) pour lesquels la recherche des bonnes méthodes de résolution est toujours d'actualité.

L'optimisation non convexe connaît une explosion spectaculaire depuis d'une quinzaine d'années car dans les milieux industriels, on a commencé à remplacer les modèles convexes par des modèles non convexes plus complexes mais plus fiables qui présentent mieux la nature des problèmes étudiés. L'analyse et l'optimisation convexes modernes se voient ainsi contrainte à une extension logique et naturelle à la non convexité et la non différentiabilité. La distinction entre mimimum local et minimum global et la non existence des caractérisations d'une solution optimale d'un problème d'optimisation non convexe provoquent des difficultés énormes dans le développement des méthodes pour sa résolution. Les méthodes numériques conventionnelles de l'optimisation convexe ne fournissent que des minima locaux bien souvent éloignés de l'optimum global. Ainsi, durant ces dernières années, la recherche en optimisation non convexe a largement bénéficié des efforts des chercheurs et s'est enrichie de nouvelles approches.

Dans le cadre d'optimisation déterministe on peut distinguer deux approches différentes mais complémentaires en programmation non convexe :

i. Approches globales combinatoires qui sont basées sur les techniques combinatoires de la Recherche Opérationnelle. Elles consistent à localiser les solutions optimales à l'aide des méthodes d'approximation, des techniques de coupe, des méthodes de décomposition, de séparation et évaluation. Elles ont connu de très nombreux développements importants au cours de ces dernières années à travers les travaux de H. Tuy (reconnu comme le pionnier) ([96]), R. Horst, P. Pardalos et N. V. Thoai ([76, 77]),... L'inconvénient majeur des méthodes globales est leur lourdeur (encombrement en places-mémoires) et leur coût trop important. Elles ne sont pas applicables aux problèmes d'optimisation non convexes réels qui sont souvent de très grande dimension.
ii. Approches locales et globales d'analyse convexe qui sont basées sur l'analyse et l'optimisation convexe. Ici la programmation DC (Différence de deux fonctions Convexes) et DCA (DC Algorithmes) jouent le rôle central car la plupart des problèmes d'optimisation non convexe sont formulés/reformulés sous la forme DC. Sur le plan algorithmique, l'essentiel repose sur les algorithmes de l'optimisation DC (DCA) introduits par Pham Dinh Tao en 1985 à l'état préliminaire et développés intensivement à travers de nombreux travaux communs de Le Thi Hoai An et Pham Dinh Tao depuis 1993 pour devenir maintenant classiques et de plus en plus utilisés par des chercheurs et praticiens de par le monde, dans différents domaines des sciences appliquées (voir [1]-[31] et [38]-[43]).

Notre travail s'appuie principalement sur la programmation DC et DCA. Cette démarche est motivée par la robustesse et la performance de la programmation DC et DCA comparées à des méthodes existantes, leur adaptation aux structures des problèmes traités et leur capacité de résoudre des problèmes industriels de grande dimension. A notre connaissance, DCA est actuellement parmi les rares algorithmes de la programmation non convexe étant capables de traiter des problèmes (différentiables ou non) de très grande dimension.

Un programme DC est de la forme

$$\alpha = \inf\{f(x) = g(x) - h(x) \ : \ x \in \mathbb{R}^n\} \quad (P_{dc})$$

où $g, h \in \Gamma_0(\mathbb{R}^n)$, le cône convexe de toutes les fonctions convexes semi-continues inférieurement et propres sur $\mathbb{R}^n$. Une telle fonction f est appelée fonction DC et g et h des composantes DC de f. DCA est une approche locale dont la construction est basée sur les composantes DC de f et non sur f elle-même. Puisqu'une fonction DC f admet une infinité de décompositions DC dont dépend DCA, il y a autant de DCA que des décompositions DC de f. Et les impacts de ces décompositions DC sur les qualités des DCA correspondants (rapidité, robustesse, globalité, ...) sont importants. La résolution d'un problème concret par DCA devrait répondre aux deux questions cruciales :

- La recherche d'une *bonne* décomposition DC : cette question est largement ouverte. En pratique on cherche des décompositions DC bien adaptées à la structure des problèmes traités. Les techniques de reformulation sont souvent utilisées et très efficaces pour l'obtention des décompositions DC intéressantes.
- La recherche d'un *bon* point initial : cette recherche est basée sur la combinaison de DCA avec les méthodes globales de type Séparation et Evaluation (SE) et/ou Approximation de l'Extérieur (AE), sur l'hybridation de DCA et les algorithmes heuristiques.

Dans le but de résolution de ces quatre problèmes d'applications importants notre travail est ainsi composé de

- La modélisation DC du problème traité par la formulation et la reformulation.
- La mise en oeuvre des schémas de DCA correspondants.
- La combinaison de DCA et d'autres approches pour chercher les bons points initials pour DCA et/ou pour prouver la globalité des solutions obtenues par DCA.
- La réalisation des logiciels à l'usage industriel immédiat.
- Les simulations numériques comparatives.

Du point de vue mathématique, les problèmes étudiés dans cette thèse sont classés en trois catégories :

1. *La programmation linéaire et/ou quadratique en variables mixtes 0-1* (l'approvisionnement d'une chaîne logistique et la reconstruction d'image binaire). Bien qu'il s'agit des problèmes d'optimisation combinatoire, nous les reformulons, grâce à la pénalité exacte, comme un problème d'optimisation continue qui est en fait une programmation DC. La transformation d'une programmation linéaire et/ou quadratique en variables mixtes binaires en une programmation quadratique non convexe (i.e. la minimisation d'une forme quadartique non convexe sur un polyèdre convexe borné) est bien connue dans la littérature ([9, 29]). Dans ce travail, *en utilisant une autre fonction de pénalité nous obtenons un problème DC polyédral pour lequel DCA jouit des propriétés de convergence intéressantes*. En plus de DCA, nous développons une méthode combinée de DCA et SE pour le problème d'approvisionnement d'une chaîne logistique.

Pour la reconstruction d'image binaire, nous considérons les trois formulations continues équivalentes parmi lesquelles la minimisation d'une forme quadratique sur un rectangle (seules les contraintes de bornes des variables sont présentées). Nous proposons *une nouvelle estimation inférieure d'une fonction quadratique non convexe sur un rectangle* qui pourrait être utiliser pour la recherche d'un bon point initial de DCA.

2. *La minimisation d'une fonction non convexe sur un simplexe* (la segmentation d'image par la classification floue) auquel les seuls algorithmes heuristiques sont disponibles à ce jour. Basée sur un schéma DCA extrêmement simple récemment développé dans [20] une procédure alternative combinée de DCA et l'algorithme standard de classification floue appelé FCM a été introduite dans le but de recherche d'un bon point initial de DCA.
3. *La minimisation d'une fonction non linéaire sur un ensemble non convexe* (l'estimation de la matrice fondamentale) pour laquelle nous développons une méthode de type de *région de confiance* qui est très efficace en optimisation non linéaire. Ce problème d'optimisation de petite taille (9 variables) est pourtant difficile car le Hessien de la fonction objectif est une matrice très mal conditionnée. Les méthodes classiques en vision par l'ordinateur pour la matrice fondamentale sont de type Levenbert-Marquard. La méthode de région de confiance (dont la supériorité par rapport à l'algorithme de Levenberd-Marquard est bien connue) consiste à la minimisation d'une forme quadratique non convexe (qui est l'approximation second ordre de la fonction objectif) sur une boulle de la norme Euclidienne. Nous utilisons diffétentes techniques (dont DCA) pour la résolution de ce sous problème.

Du point de vue informatique nos contributions propres portent sur l'introduction des informations spatiales aux modèles classiques pour les deux problèmes en traitement d'image (reconstruction d'image binaire et segmenation d'image). L'utilisation de l'information spatiale qui représente la relation entre le pixel et ses voisinages, est très importante en traitement d'image (car les voisinages possèdent souvent les valeurs semblables, et la probabilité qu'ils appartiennent à la même partition est très élevée). Il existe différentes manières de considération des informations spatiales qui se ramènent aux diférents modèles d'optimisation. Un choix adéquat de l'information spatiale devrait recueillir des bonnes informations sur l'image et conrrespondre à un modèle d'optimisation qui n'est pas trop compliqué. Les modèles avec l'information spatiale que nous utilisons sont toujours des programmes DC et DCA appliqué à ces modèles est aussi simple que celui aux modèles classiques sans l'information spatiale.

La thèse est composée de cinq chapitres. Le premier chapitre, servant des références aux autres, présente les outils de bases de la programmation DC et DCA.

Le chapitre deux porte sur le problème de conception d'une chaîne d'approvisionnement. Le chapitre trois concerne la tomographie discrète appliquée au problème de reconstruction d'image binaire. Dans le chapitre quatre, nous étudions le problème de segmentation d'image

via la classification floue par DCA. L'estimation non linéaire de la matrice fondamentale en vision par ordinateur par la méthode de région de confiance basée sur la méthode de gradient conjugué tronqué ou DCA est développée dans le dernier chapitre.

Chapitre 1

Introduction à la programmation DC et DCA

Le cadre des *programmes convexes* s'est avéré trop étroit et, à la notion de fonction convexe a succédé avec bonheur, celle plus générale, de fonction DC (différence de fonctions convexes). Les fonctions DC possèdent de nombreuses propriétés importantes qui ont été établies à partir des années 50 par Alexandroff (1949), Landis (1951) et Hartman (1959), une des principales propriétés est leur stabilité relative aux opérations fréquemment utilisées en optimisation. Cependant, il faut attendre le milieu des années 80 pour que la classe des fonctions DC soit introduite en optimisation, élargissant ainsi la classification des problèmes d'optimisation avec l'apparition de la programmation DC. On distingue deux grandes approches DC :

1. L'approche combinatoire (cette terminologie est due au fait que les nouveaux outils introduits ont été inspirés par les concepts de l'optimisation combinatoire) en optimisation globale continue, et
2. L'approche de l'analyse convexe en optimisation non convexe.

Les algorithmes de l'approche combinatoire utilisent les techniques de l'optimisation globale (méthode de séparation et d'évaluation, technique de coupe, méthodes d'approximation fonctionnelle et ensembliste) ; ces algorithmes relativement sophistiqués sont plutôt lourds à mettre en oeuvre, ils doivent donc être réservés à des problèmes de dimensions raisonnables possédant des structures bien adaptées aux méthodes lorsqu'il est important d'isoler l'optimum global.

Le pionnier de cette approche est H. Tuy dont le premier travail remonte à 1964. Ses travaux sont abondants, citons les livres de Horst-Tuy ([96, 97]) qui présentent la théorie, algorithmes et applications de l'optimisation globale. Viennent ensuite les principales contributions de l'Ecole Américaine (P. M. Pardalos, J. B. Rosen,...), Allemande (R. Horst, ...), Française (Le Thi Hoai An, Pham Dinh Tao,...) et l'Ecole Vietnamienne (Phan Thien Thach, Le Dung Muu, ...).

La seconde approche repose sur l'arsenal puissant d'analyse et l'optimisation convexes. Son premier travail dû à Pham Dinh Tao (1975) concerne le calcul des normes matricielles (problème fondamental en analyse numérique) qui est un problème de maximisation d'une fonction convexe sur un convexe. Le travail de Toland (1978) ([90]) sur la dualité et l'optimalité locale en optimisation DC généralise de manière élégante les résultats établis par Pham en maximisation convexe. La théorie de l'optimisation DC est ensuite développée notamment par Pham Dinh Tao, J. B. Hiriart Urruty, Jean - Paul Penot, Phan Thien Thach, Le Thi Hoai An. Sur le plan algorithmique dans le cadre de la seconde approche, on dispose actuellement que des DCA (DC Algorithms) introduits par Pham Dinh Tao (1986), qui sont basés sur les conditions d'optimalité et de dualité en optimisation DC. Mais il a fallu attendre les travaux communs de Le Thi Hoai An et Pham Dinh Tao (voir [1]-[31] et [38]-[43]) pour qu'il s'impose définitivement en optimisation non convexe comme étant des algorithmes les plus simples et performants, capables de traiter des problèmes de grande taille.

Nous reportons dans ce chapitre les principaux résultats relatifs à la programmation DC et DCA qui nous seront les plus utiles pour nos travaux. Ces résultats sont extraits de ceux présentés dans H. A. Le Thi 1994 ([1]), H. A. Le Thi 1997 ([2]). Pour une étude détaillée nous nous référons à ces deux références (voir également [1]-[31] et [38]-[43]).

1.1 Eléments de base de l'analyse DC

1.1.1 Notations et propriétés

Ce paragraphe est consacré à un rapide rappel d'analyse convexe pour faciliter la lecture de certains passages. Pour plus de détails, on pourra se référer aux ouvrages de P.J Laurent ([79]), de R.T Rockafellar ([85]) et d'A. Auslender ([66]). Dans toute la suite X désigne l'espace euclidien $\mathbb{R}^n$, muni du produit scalaire usuel noté $\langle .,. \rangle$ et de la norme euclidienne associée $\|x\| = \langle x, x \rangle^{\frac{1}{2}}$ et Y l'espace vectoriel dual de X relatif au produit scalaire, que l'on peut identifier à X. On note par $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$ muni d'une structure algébrique déduite de celle de $\mathbb{R}$ avec la convention que $-\infty - (+\infty) = +\infty$ ([85]). Etant donnée une fonction $f : S \longrightarrow \overline{\mathbb{R}}$ définie sur un ensemble S convexe de X, on appelle domaine effectif de f l'ensemble

$$dom(f) = \{x \in S : f(x) < +\infty\}$$

et épigraphe de f

$$epi(f) = \{(x, \alpha) \in S \times \mathbb{R} : f(x) < \alpha\}.$$

Si $dom(f) \neq \emptyset$ et $f(x) > -\infty$ pour tout $x \in S$ alors la fonction $f(x)$ est dite propre.

Une fonction $f : S \longrightarrow \overline{\mathbb{R}}$ est dite convexe si son épigraphe est un ensemble convexe de $\overline{\mathbb{R}} \times X$. Ce qui est équivalent de dire que S est un ensemble convexe et pour tout $\lambda \in [0, 1]$

on a

$$f((1-\lambda)x^1+\lambda x^2) \leq (1-\lambda)f(x^1)+\lambda f(x^2) : \forall x^1, x^2 \in S. \tag{1.1}$$

On note alors $Co(X)$ l'ensemble des fonctions convexes sur X.

Dans (1.1) si l'inégalité stricte est vérifiée pour tout $\lambda \in]0,1[$ et pour tout $x^1, x^2 \in S$ avec $x^1 \neq x^2$ alors f est dite strictement convexe.

On dit que $f(x)$ est fortement convexe sur un ensemble convexe C s'il existe un nombre $\rho > 0$ tel que

$$f((1-\lambda)x^1+\lambda x^2) \leq (1-\lambda)f(x^1)+\lambda f(x^2)-(1-\lambda)\lambda\frac{\rho}{2}\|x^1-x^2\|^2, \tag{1.2}$$

pour tout $x^1, x^2 \in C$, et pour tout $\lambda \in [0,1]$. Plus précisément f est fortement convexe sur C si

$$\rho(f,C) = Sup\{\rho \geq 0 : f-\frac{\rho}{2}\|.\|^2 \text{ est convexe sur } C\} > 0. \tag{1.3}$$

Il est clair que si $\rho(f,C) > 0$ alors (1.2) est vérifié pour tout $\lambda \in [0, \rho(f,C)[$. On dit que la borne supérieure est atteinte dans sa définition (1.3) si $f-\frac{\rho(f,C)}{2}\|.\|^2$ est convexe sur C. Si $C \equiv X$ on notera $\rho(f)$ au lieu de $\rho(f,X)$.

Remarque 1.1 *f fortement convexe $\Longrightarrow$ f strictement convexe $\Longrightarrow$ f convexe.*

Soit une fonction convexe propre f sur X, un élément $y^0 \in Y$ est dit un sous-gradient de f au point $x^0 \in dom(f)$ si

$$\langle y^0, x-x^0\rangle + f(x^0) \leq f(x) \quad \forall x \in X.$$

L'ensemble de tous les sous-gradients de f au point x^0 est dit sous-différentiel de f au point x^0 et est noté par $\partial f(x^0)$.

Etant donné un nombre positif $\epsilon > 0$, un élément $y^0 \in Y$ est dit ϵ-sous-gradient de f au point x^0 si

$$\langle y^0, x-x^0\rangle + f(x^0) \leq f(x)+\epsilon \quad \forall x \in X.$$

L'ensemble de tous les ϵ-sous-gradients de f au point x^0 est dit ϵ-sous-différentiel de f au point x^0 et est noté par $\partial_\epsilon f(x^0)$.

La fonction $f : S \longrightarrow \mathbb{R}$ est dite semi-continue inférieurement (s.c.i) en un point $x \in S$ si

$$\liminf_{y \to x} f(y) \geq f(x).$$

On note $\Gamma_0(X)$ l'ensemble des fonctions convexes s.c.i. et propre sur X.

Définition 1.1 *Soit une fonction quelconque $f : X \Longrightarrow \mathbb{R}$, la fonction conjuguée de f, notée $f^\star$, est définie sur Y par*

$$f^\star(y) = sup\{\langle x, y\rangle - f(x) : x \in X\}. \tag{1.4}$$

$f^\star$ est l'enveloppe supérieure des fonctions affines continues $y \mapsto \langle x, y\rangle - f(x)$ sur Y.

On résume dans la proposition suivante les principales propriétés dont on aura besoin pour la suite :

Proposition 1.1 *Si $f \in \Gamma_0(X)$ alors :*
- *$f \in \Gamma_0(X) \Longleftrightarrow f^\star \in \Gamma_0(Y)$. Dans ce cas on a $f = f^{\star\star}$,*
- *$y \in \partial f(x) \Longleftrightarrow f(x) + f^\star(y) = \langle x, y\rangle$ et $y \in \partial f(x) \Longleftrightarrow x \in \partial f^\star(y)$,*
- *$\partial f(x)$ est une partie convexe fermée,*
- *Si $\partial f(x) = \{y\}$ alors f est différentiable en x et $\nabla f(x) = y$,*
- *$f(x^0) = min\{f(x), x \in X\} \Longleftrightarrow 0 \in \partial f(x^0)$.*

1.1.2 Fonctions convexes polyédrales

Une partie convexe C est dite convexe polyédrale si

$$C = \bigcap_{i=1}^{m}\{x : \langle a_i, x\rangle - \alpha_i \leq 0\} \text{ où } a_i \in Y, \alpha_i \in \mathbb{R}, \quad \forall i = 1, ..., m.$$

Une fonction est dite convexe polyédrale si

$$f(x) = sup\{\langle a_i, x\rangle - \alpha_i : i = 1, ..., k\} + \chi_c(x).$$

où C est une partie convexe polyédrale et le symbole χ_c désigne la fonction indicatrice de C, i.e. $\chi_c(x) = 0$ si $x \in C$ et $+\infty$ sinon.

Proposition 1.2 *([85])*
- *Soit f une fonction convexe polyédrale. f est partout finie si et seulement si $C = X$,*
- *Si f est polyédrale alors $f^\star$ l'est aussi. De plus si f est partout finie alors*

$$f(x) = sup\{\langle a_i, x\rangle - a_i : i = 1, ..., k\},$$

$$dom(f^\star) = co\{a_i : i = 1, ..., k\},$$

$$f^\star(y) = min\{\Sigma_{i=1}^k \lambda_i \alpha_i : y = \Sigma_{i=1}^k \lambda_i a_i, \lambda_i \geq 0, \Sigma_{i=1}^k \lambda_i = 1\},$$

- *Si f est polyédrale alors $\partial f(x)$ est une partie convexe polyédrale non vide en tout point $x \in dom(f)$.*

1.1.3 Fonction DC

Une fonction $f : \Omega \mapsto \overline{\mathbb{R}}$ définie sur un ensemble convexe $\Omega \subset \mathbb{R}^n$ est dite DC sur Ω si elle peut s'écrire comme la différence de deux fonctions convexes sur Ω, i.e.

$$f(x) = g(x) - h(x),$$

où g et h sont des fonctions convexes sur Ω. On note par $DC(\Omega)$ l'ensemble des fonctions DC sur Ω, et par $DC_f(\Omega)$ le cas où les fonctions g et h sont convexes finies sur Ω.

Les fonctions DC possèdent de nombreuses propriétés importantes qui ont été établies à partir des années 50 par Alexandroff (1949), Landis (1951) et Hartman (1959) ; une des principales propriétés est leur stabilité relative aux opérations fréquemment utilisées en optimisation. Plus précisément

Proposition 1.3 *(i) Une combinaison linéaire de fonctions DC sur Ω est DC sur Ω,*
(ii) L'enveloppe supérieure d'un ensemble fini de fonctions DC à valeur finie sur Ω est DC sur Ω,
L'enveloppe inférieure d'un ensemble fini de fonctions DC à valeur finie sur Ω est DC sur Ω,
(iii) Soit $f \in DC_f(\Omega)$, alors $|f(x)|, f^+(x) = max\{0, f(x)\}$ et $f^-(x) = min\{0, f(x)\}$ sont DC sur Ω.

Ces résultats se généralisent aux cas des fonctions à valeur dans $\mathbb{R} \cup \{+\infty\}$ ([2]). Il en résulte que l'ensemble des fonctions DC sur Ω est un espace vectoriel ($DC(\Omega)$) : c'est le plus petit espace vectoriel contenant l'ensemble des fonctions convexes sur $\Omega(Co(\Omega))$.

Remarque 1.2 *Etant donnée une fonction DC f et sa représentation DC $f = g - h$, alors pour toute fonction convexe finie φ, $f = (g + \varphi) - (h + \varphi)$ donne une autre représentation DC de f. Ainsi, une fonction DC admet une infinité de décomposition DC.*

Désignons par $C^2(\mathbb{R}^n)$, la classe des fonctions deux fois continûment différentiables sur $\mathbb{R}^n$.

Proposition 1.4 *Toute fonction $f \in C^2(\mathbb{R}^n)$ est DC sur un ensemble convexe compact quelconque $\Omega \cup \mathbb{R}^n$.*

Puisque le sous-espace des polynômes sur Ω est dense dans l'espace $C(\Omega)$ des fonctions numériques continues sur Ω on en déduit :

Corollaire 1.1 *L'espace des fonctions DC sur un ensemble convexe compact $\Omega \cup \mathbb{R}^n$ est dense dans $C(\Omega)$, i.e.*

$$\forall e > 0, \exists F \in C(\Omega) : |f(x) - F(x)| \leq \epsilon \quad \forall x \in \Omega.$$

Soulignons que les fonctions DC interviennent très fréquemment en pratique, aussi bien en optimisation différentiable que non différentiable. Un résultat important établi par Hartman (1959) permet d'identifier les fonctions DC dans de nombreuses situations, en ayant recours simplement à une analyse locale de la convexité (localement convexe, localement concave, localement DC).

Une fonction $f : D \mapsto \mathbb{R}$ définie sur un ensemble convexe ouvert $D \in \mathbb{R}^n$ est dite localement DC si pour tout $x \in D$ il existe un voisinage convexe ouvert U de x et une paire de fonctions convexes g, h sur U telle que $f|_U = g|_U - h|_U$.

Proposition 1.5 *Une fonction localement DC sur un ensemble convexe D est DC sur D.*

1.2 Optimisation DC

De par la prépondérance et de la richesse des propriétés des fonctions DC, le passage du sous-espace $Co(\Omega)$ à l'espace vectoriel $DC(\Omega)$ permet d'élargir significativement les problèmes d'optimisation convexe à la non convexité tout en conservant une structure sous-jacente fondamentalement liée à la convexité. Le domaine des problèmes d'optimisation faisant intervenir des fonctions DC est ainsi relativement large et ouvert, couvrant la plupart des problèmes d'applications rencontrés.

Ainsi on ne peut d'emblée traiter tout problème d'optimisation non convexe et non différentiable. La classification suivante devenue maintenant classique :

$$(1) \quad sup\{f(x) : x \in C\}, \text{f et C sont convexes}$$

$$(2) \quad inf\{g(x) - h(x) : x \in X\}, \text{g et h sont convexes}$$

$$(3) \quad inf\{g(x) - h(x) : x \in C, f_1(x) - f_2(x) \leq 0\},$$

où g, h, f_1, f_2 et C sont convexes semble assez large pour contenir la quasi-totalité des problèmes non convexes rencontrés dans la vie courante. Le problème (1) est un cas spécial du problème (2) avec $g = \chi_C$, la fonction indicatrice de C, et $h = -f$. Le problème (2) peut être modélisé sous la forme équivalent de (1)

$$inf\{t - h(x) : g(x) - t \leq 0\}.$$

Quant au problème (3) il peut être transformé sous la forme (2) via la pénalité exacte relative à la contrainte DC $f_1(x) - f_2(x) \leq 0$. Sa résolution peut être aussi ramenée, sous certaines conditions techniques, à celle d'une suite de problèmes (1).

Problème (2) est communément appelé *la programmation DC*. Elle est d'un intérêt majeur aussi bien d'un point de vue pratique que théorique. Du point de vue théorique, on peut

souligner que, comme on a vu en haut, la classe des fonctions DC est remarquablement stable par rapport aux opérations fréquemment utilisées en optimisation. En outre, on dispose d'une élégante théorie de la dualité ([32, 33, 90, 99, 1, 2, 5]) qui, comme en optimisation convexe, a de profondes répercussions pratiques sur les méthodes numériques.

Sur le plan algorithmique, les algorithmes de l'optimisation DC (DCA) dus à Pham Dinh Tao ([37, 38]) constituent une nouvelle approche originale basée sur la théorie DC. Ces algorithmes représentent en fait une généralisation des algorithmes de sous-gradients étudiés par le même auteur sur la maximisation convexe ([32, 37]). Cependant, il a fallu attendre les travaux communs de Le Thi et Pham au cours de ces dix dernières années (voir [1]-[31] et [38]-[43]) pour que les DCA deviennes maintenant classiques et populaires.

1.2.1 Dualité DC

En analyse convexe, le concept de la dualité (fonctions conjuguées, problème dual, etc.) est une notion fondamentale très puissante. Pour les problèmes convexes et en particulier linéaires, une théorie de la dualité a été développée depuis déjà plusieurs décennies ([85]). Plus récemment, en analyse non convexe d'importants concepts de dualité ont été proposés et développés, tout d'abord, pour les problèmes de maximisation convexe, avant de parvenir aux problèmes DC. Ainsi la dualité DC introduite par Toland (1978) peut être considérée comme une généralisation logique des travaux de Pham Dinh Tao (1975) sur la maximisation convexe. On va présenter ci-dessous les principaux résultats (en optimisation DC) concernant les conditions d'optimalité (locale et globale) et la dualité DC. Pour plus de détails, le lecteur est renvoyé au document de Le Thi (1997) (voir également [5]).

Soit l'espace $X = \mathbb{R}^n$ muni du produit scalaire usuel $\langle ., . \rangle$ et de la norme euclidienne $\|.\|$. Désignons par Y l'espace dual de X que l'on peut identifier à X lui-même et par $\Gamma_0(X)$ l'ensemble de toutes les fonctions propres s.c.i. sur X.

Soient $g(x)$ et $h(x)$ deux fonctions convexes propres sur $X(\ g, h \in \Gamma_0(X))$, considérons le problème DC

$$inf\{g(x) - h(x) : x \in X\} \quad (P)$$

et le problème dual

$$inf\{h^\star(y) - g^\star(y) : y \in Y\} \quad (D)$$

où $g^\star(y)$ désigne la fonction conjuguée de g.

Ce résultat de dualité DC défini à l'aide des fonctions conjuguées donne une importante relation en optimisation DC ([90]).

Théorème 1.1 *Soient g et $h \in \Gamma_0(X)$, alors*

(i)

$$\inf_{x\in dom(g)}\{g(x)-h(x)\}=\inf_{y\in dom(h^\star)}\{h^\star(y)-g^\star(y)\} \tag{1.5}$$

(ii) Si y^0 est un minimum de $h^\star-g^\star$ sur Y alors chaque $x^0\in\partial g^\star(y^0)$ est un minimum de $g-h$ sur X.

Preuve :
(i)

$$\begin{aligned}\alpha &= \inf\{g(x)-h(x):x\in X\}\\ &= \inf\{g(x)-sup\{\langle x,y\rangle-h^\star(y):y\in Y\}:x\in X\}\\ &= \inf\{g(x)+inf\{h^\star(y)-\langle x,y\rangle:y\in Y\}:x\in X\}\\ &= \inf_x\inf_y\{h^\star(y)-\langle x,y\rangle-g(x)\}\\ &= \inf\{h^\star(y)-g^\star(y):y\in Y\}.\end{aligned}$$

(ii) cf. Toland ([90]).

□

Le théorème (1.1) montre que résoudre le problème primal (P) implique la résolution du problème dual (D) et vice-versa.

De par la parfaite symétrie entre le problème primal (P) et le problème dual (D), il apparaît clairement que les résultats établis pour l'un se transpose directement à l'autre. Cependant, nous choisissons ici de ne pas les présenter simultanément afin de simplifier la présentation.

1.2.2 Optimalité globale en optimisation DC

En optimisation convexe, x^0 minimise une fonction $f\in\Gamma_0(X)$ si et seulement si : $0\in\partial f(x^0)$. En optimisation DC, la condition d'optimalité globale suivante ([100]) est formulée à l'aide des ϵ-sous-différentiels de g et h. Sa démonstration (basée sur l'étude du comportement du ϵ-sous-différentiel d'une fonction convexe en fonction du paramètre ϵ) est compliquée. La démonstration dans [2] est plus simple et convient bien au cadre de l'optimisation DC : elle exprime tout simplement que cette condition d'optimalité globale est une traduction géométrique de l'égalité des valeurs optimales dans les programmes DC primal et dual.

Théorème 1.2 ***(Optimalité globale DC)*** *Soit $f=g-h$ où $g,h\in\Gamma_0(X)$ alors. x^0 est un minimum global de $g(x)-h(x)$ sur X si et seulement si,*

$$\partial_\epsilon h(x^0)\subset\partial_\epsilon g(x^0)\quad\forall\epsilon>0. \tag{1.6}$$

Remarque 1.3 –

(i) Si $f \in \Gamma_0(X)$, on peut écrire $f = g - h$ avec $f = g$ et $h = 0$. Dans ce cas l'optimalité globale dans (P) - qui est identique à l'optimalité locale car (P) est un problème convexe - est caractérisée par,

$$0 \in \partial f(x^0). \tag{1.7}$$

Du fait que $\partial_\epsilon h(x^0) = \partial h(x^0) = \{0\}, \quad \forall \epsilon > 0, \forall x \in X$, et la croissance du ϵ-sousdifférentiel en fonction de ϵ, la relation (1.7) est équivalente à (1.6).

(ii) D'une manière plus générale, considérons les décompositions DC de $f \in \Gamma_0(X)$ de la forme $f = g - h$ avec $g = f + h$ et $h \in \Gamma_0(X)$ finie partout sur X. Le problème DC correspondant est un "faux" problème DC car c'est un problème d'optimisation convexe. Dans ce cas, la relation (1.7) est équivalente à

$$\partial h(x^0) \subset \partial g(x^0).$$

(iii) On peut dire ainsi que (1.6) marque bien le passage de l'optimisation convexe à l'optimisation non convexe. Cette caractéristique de l'optimalité globale de (P) indique en même temps toute la complexité de son utilisation pratique car il fait appel à tous les ϵ-sous-différentiels en x^0.

1.2.3 Optimalité locale en optimisation DC

Nous avons vu que la relation $\partial h(x^0) \subset \partial g(x^0)$ (faisant appel au sous-différentiel "exact") est une condition nécessaire et suffisante d'optimalité globale pour un "faux" problème DC (problème d'optimisation convexe). Or dans un problème d'optimisation globale, la fonction à minimiser est localement convexe "autour" d'un minimum local, il est alors clair que cette relation d'inclusion sous-différentielle permettra de caractériser un minimum local d'un problème DC.

Définition 1.2 *Soient g et $h \in \Gamma_0(X)$. Un point $x^\bullet \in dom(g) \cap dom(h)$ est un minimum local de $g(x) - h(x)$ sur X si et seulement si*

$$g(x) - h(x) \geq g(x^\bullet) - h(x^\bullet), \quad \forall x \in V_{x^\bullet}, \tag{1.8}$$

où V_x désigne un voisinage de x.

Proposition 1.6 ***(Condition nécessaire d'optimalité locale)*** *Si $x^\bullet$ est un minimum local de $g - h$ alors*

$$\partial h(x^\bullet) \subset \partial g(x^\bullet), \tag{1.9}$$

Preuve : Si $x^\bullet$ est un minimum local de $g-h$, alors il existe un voisinage V_x de x tel que

$$g(x) - g(x^\bullet) \geq h(x) - h(x^\bullet), \quad \forall x \in V_{x^\bullet}. \tag{1.10}$$

Par suite si $y^\bullet \in \partial h(x^\bullet)$ alors

$$g(x) - g(x^\bullet) \geq \langle x - x^\bullet, y^\bullet \rangle, \quad \forall x \in V_{x^\bullet} \tag{1.11}$$

Ce qui est équivalent, en vertu de la convexité de g, à $y^\bullet \in \partial g(x^\bullet)$. □

Remarquons que pour un certain nombre de problème DC et en particulier pour h polyédrale, la condition nécessaire (1.9) est également suffisante, comme nous le verrons un peu plus loin. On dit que $x^\bullet$ est un point critique de $g-h$ si $\partial h(x^\bullet) \cup \partial g(x^\bullet)$ est non vide ([90]). C'est une forme affaiblie de l'inclusion sousdifférentielle. La recherche d'un tel point critique est à la base de DCA (forme simple) qui sera étudiée dans la section suivante. En général DCA converge vers une solution locale d'un problème d'optimisation DC. Cependant sur le plan théorique, il est important de formuler des conditions suffisantes pour l'optimalité locale.

Théorème 1.3 ***(Condition suffisante d'optimalité locale ([2, 5]))*** *Si $x^\star$ admet un voisinage V tel que*

$$\partial h(x) \cap \partial g(x^\star) \neq \emptyset, \quad \forall x \in V \cap dom(g), \tag{1.12}$$

alors $x^\star$ est un minimum local de $g-h$.

Corollaire 1.2 *Si $x \in int(dom(h))$ vérifie*

$$\partial h(x) \in int(\partial g(x)),$$

alors x est un minimum local de $g-h$.

Corollaire 1.3 *Si $h \in \Gamma_0(X)$ est convexe polyédrale alors $\partial h(x) \subset \partial g(x)$ est une condition nécessaire et suffisante pour que x soit un minimum local de $g-h$.*

Preuve : Ce résultat généralise le premier obtenu par C. Michelot dans le cas où $g, h \in \Gamma_0(X)$ sont finies partout et h convexe polyédrale (cf. ([2, 5])). □

Pour résoudre un problème d'optimisation DC, il est parfois plus facile de résoudre le problème dual (D) que le problème primal (P). Le théorème (1.1) assure le transport par dualité des minima globaux. On établit de même le transport par dualité des minima locaux.

Corollaire 1.4 ***(Transport par dualité DC des minima locaux ([2, 5]))*** *Supposons que $x^\bullet \in dom(\partial h)$ soit un minimum local de $g-h$, soient $y^\bullet \in \partial h(x^\bullet)$ et $V_{x^\bullet}$ un voisinage de $x^\bullet$ tel que $g(x) - h(x) \geq g(x^\bullet) - h(x^\bullet), \quad \forall x \in V_{x^\bullet} \cap dom(g)$. Si*

$$x^\bullet \in int(dom(g^\star)) \quad et \quad \partial g^\star(y^\bullet) \subset V_{x^\bullet}, \tag{1.13}$$

alors $y^\bullet$ est un minimum local de $h^\star - g^\star$.

Preuve : Immédiate d'après la proposition (1.1) en se restreignant à l'intervalle $V_{x^\bullet} \cap dom(g)$. □

Remarque 1.4 *Bien sûr, par dualité, tous les résultats de cette section se transposent au problème dual D. Par exemple :*
si y est un minimum local de $h^\star - g^\star$ alors $\partial g^\star(y) \subset \partial h^\star(y)$.

1.3 DCA

Il s'agit d'une nouvelle méthode de sous-gradient basée sur l'optimalité et la dualité en optimisation DC (non différentiable). Cette approche est complètement différente des méthodes classiques de sous-gradient en optimisation convexe. Dans les DCA, la construction algorithmique cherche à exploiter la structure DC du problème. Elle nécessite, en premier lieu, de disposer d'une représentation DC de la fonction à minimiser, i.e. $f = g - h$ (g, h convexe), car toutes les opérations s'effectueront uniquement sur les composantes convexes. Ainsi, la séquence des directions de descente est obtenue en calculant une suite de sous-gradient non directement à partir de la fonction f, mais des composantes convexes des problèmes primal et dual.

1.3.1 Principle de DCA

La construction des DCA, découverte par Pham Dinh Tao (1986) s'appuie sur la caractérisation des solutions locales en optimisation DC des problèmes primal (P) et dual (D)

$$\alpha = inf\{g(x) - h(x) : x \in X\} \quad (P),$$

$$\alpha = inf\{h^\star(y) - g^\star(y) : y \in Y\} \quad (D).$$

Les DCA consistent en la construction de deux suites $\{x^k\}$ et $\{y^k\}$. La première suite est candidate à être solution du problème primal et la seconde du problème dual. Ces deux suites sont liées par dualité et vérifient les propriétés suivantes :
– les suites $\{g(x^k) - h(x^k)\}$ et $\{h^\star(y^k) - g^\star(y^k)\}$ sont décroissantes,
– et si $(g-h)(x^{k+1}) = (g-h)(x^k)$ alors l'algorithme s'arrête à la $(k+1)^{ieme}$ itération et le point x^k (resp. y^k) est un point critique de $g-h$ (resp. $h^\star - g^\star$),
– sinon toute valeur d'adhérence $x^\bullet$ de $\{x^k\}$ (resp. $y^\bullet$ de $\{y^k\}$) est un point critique de $g-h$ (resp. $h^\star - g^\star$).
L'algorithme cherche en définitif un couple $(x^\bullet, y^\bullet) \in X \times Y$ tel que $x^\bullet \in \partial g^\star(y^\bullet)$ et $y^\bullet \in \partial h(x^\bullet)$.

Schéma de DCA simplifié

L'idée principale de la mise en oeuvre de l'algorithme (forme simple) est de construire une suite $\{x^k\}$, vérifiant à chaque itération $\partial g(x^k) \cap \partial h(x^{k-1}) \neq \emptyset$, convergente vers un point critique $x^\bullet(\partial h(x^\bullet) \cap \partial g(x^\bullet) \neq \emptyset)$ et symétriquement, de façon analogue par dualité, une suite $\{y^k\}$ telle que $\partial g^\star(y^{k-1}) \cap \partial h^\star(y^k) \neq \emptyset$ convergente vers un point critique.

On construit ainsi :

Algorithme 1. [DCA]

Etape 0. x^0 donné.
Etape 1. Pour chaque k, x^k étant connu, déterminer $y^k \in \partial h(x^k)$.
Etape 2. Trouver $x^{k+1} \in \partial g^\star(y^k)$.
Etape 3. Si test d'arrêt vérifié **STOP** ; Sinon $k \leftarrow k+1$.

Cette description, avec l'aide de schémas d'itération de points fixes des multi-applications ∂h et $\partial g^\star$, apparaît ainsi être d'une grande simplicité.

1.3.2 Existence des suites générées

L'algorithme DCA est bien défini si on peut effectivement construire les deux suites $\{x^k\}$ et $\{y^k\}$ comme ci-dessus à partir d'un point initial arbitraire x^0.

- Par construction, si $x^0 \in dom(\partial h)$, alors $y^0 \in \partial h(x^0)$ est bien défini.
- Pour $k \geq 1$, y^k est bien défini si et seulement si x^k est défini et contenu dans $dom(\partial h)$, par suite, x^k et y^k sont bien définis si et seulement si $\partial g^\star(y^{k+1}) \cap dom(\partial h)$ est non vide, ce qui entraîne que $y^{k+1} \in dom(\partial g^\star)$.

Lemme 1.1 *([5]) Les suites $\{x^k\}$, $\{y^k\}$ dans DCA sont bien définies si et seulement si*

$$dom(\partial g) \subset dom(\partial h), \quad et \quad dom(\partial h^\star) \subset dom(\partial g^\star).$$

La convergence de l'algorithme est assurée par les résultats suivants ([5]) :

Soient ρ_i et $\rho_i^\star, (i = 1, 2)$ des nombres réels positifs tels que $0 \leq \rho_i < \rho(f_i)$ (resp. $0 \leq \rho_i^\star < \rho_i^\star(f_i^\star)$) où $\rho_i = 0$ (resp $\rho_i^\star = 0$) si $\rho(f_i) = 0$ (resp $\rho(f_i^\star) = 0$) et ρ_i (resp $\rho_i^\star$) peut prendre la valeur $\rho(f_i)$ (resp $\rho(f_i^\star)$) si cette borne supérieure est atteinte. Nous poserons pour la suite $f_1 = g, f_2 = h$.

Théorème 1.4 *Si les suites $\{x^k\}$ et $\{y^k\}$ sont bien définies. Alors on a :*
(i)

$$(g-h)(x^{k+1}) \leq (h^\star - g^\star)(y^k) - \frac{\rho_h}{2}\|dx^k\|^2 \leq (g-h)(x^k) - \frac{\rho_1+\rho_2}{2}\|dx^k\|^2$$

(ii)

$$(h^\star - g^\star)(y^{k+1}) \leq (g-h)(x^{k+1}) - \frac{\rho_1^\star}{2}\|dy^k\|^2 \leq (h^\star - g^\star)(y^k) - \frac{\rho_1^\star + \rho_2^\star}{2}\|dy^k\|^2$$

où $dx^k = x^{k+1} - x^k$

Corollaire 1.5 *([5])**(Convergence)***
1.

$$\begin{aligned}(g-h)(x^{k+1}) &\leq (h^\star - g^\star)(y^k) - \tfrac{\rho_2}{2}\|dx^k\|^2 \\ &\leq (g-h)(x^k) - [\tfrac{\rho_2}{2}\|dx^{k-1}\|^2 + \tfrac{\rho_1^\star}{2}\|dy^k\|^2]\end{aligned}$$

2.

$$\begin{aligned}(g-h)(x^{k+1}) &\leq (h^\star - g^\star)(y^k) - \tfrac{\rho_2^\star}{2}\|dx^k\|^2 \\ &\leq (g-h)(x^k) - [\tfrac{\rho_2^\star}{2}\|dx^{k-1}\|^2 + \tfrac{\rho_1^\star}{2}\|dy^k\|^2]\end{aligned}$$

3.

$$\begin{aligned}(h^\star - g^\star)(y^{k+1}) &\leq (g-h)(x^{k+1}) - \tfrac{\rho_1^\star}{2}\|dy^k\|^2 \\ &\leq (h^\star - g^\star)(y^k) - [\tfrac{\rho_1^\star}{2}\|dy^k\|^2 + \tfrac{\rho_2^\star}{2}\|dx^k\|^2]\end{aligned}$$

4.

$$\begin{aligned}(h^\star - g^\star)(y^{k+1}) &\leq (g-h)(x^{k+1}) - \tfrac{\rho_1}{2}\|dy^{k+1}\|^2 \\ &\leq (h^\star - g^\star)(y^k) - [\tfrac{\rho_1}{2}\|dx^{k+1}\|^2 + \tfrac{\rho_2}{2}\|dx^k\|^2]\end{aligned}$$

Corollaire 1.6 *([5]) Si les égalités ont lieu, il vient :*
1. $(g-h)(x^{k+1}) = (h^\star - g^\star)(y^k) \Longleftrightarrow y^k \in \partial h(x^{k+1})$
2. $(g-h)(x^{k+1}) = (g-h)(x^k) \Longleftrightarrow x^k \in \partial g^\star(y^k), \quad y^k \in \partial h(x^{k+1})$
3. $(h^\star - g^\star)(y^k) = (g-h)(x^k) \Longleftrightarrow x^k \in \partial g^\star(y^k)$
4. $(h^\star - g^\star)(y^{k+1}) = (h^\star - g^\star)(y^k) \Longleftrightarrow y^k \in \partial h(x^{k+1}), \quad x^{k+1} \in \partial g^\star(y^{k+1})$

En général, les qualités (robustesse, stabilité, vitesse de convergence, bonnes solutions locales) de DCA dépendent des décompositions DC de la fonction objectif f = g - h. Le théorème 1.4 montre que la forte convexité des composantes convexes dans les problèmes primal et dual peut influencer sur DCA. Pour rendre les composantes convexe g et h fortement convexes, on peut usuellement appliquer l'opération suivante

$$f = g - h = \left(g + \frac{\lambda}{2}\|.\|^2\right) - \left(h + \frac{\lambda}{2}\|.\|^2\right).$$

Dans ce cas, les composantes convexes dans le problème dual seront continûment différentiable.

1.3.3 Calcul des sous-gradients

La description de DCA à l'aide de schémas d'itération de points fixes des multi-applications ∂h et $\partial g^\star$ (∂g et $\partial h^\star$) se présente schématiquement :

$$\begin{array}{lcl} x^k & \leftarrow & y^k \in \partial h(x^k) \\ x^{k+1} \in \partial g^\star(y^k) & \leftarrow & y^{k+1} \in \partial h(x^{k+1}) \\ (y^k \in \partial g(x^{k+1})) & & (x^{k+1} \in \partial h^\star(y^{k+1})) \end{array} \tag{1.14}$$

On voit ainsi une parfaite symétrie des suites $\{x^k\}$ et $\{y^k\}$ relative à la dualité de l'optimisation DC.

Le calcul du sous-gradient de la fonction h en un point x^k est en général aisé : dans de nombreux problèmes concrets on connaît l'expression explicite de ∂h. Par contre, le calcul d'un sous gradient de la conjuguée de la fonction convexe g en un point y^k, nécessite en général la résolution du programme convexe,

$$\partial g^\star(y^k) = argmin\{g(x) - \langle y^k, x\rangle : x \in X\}, \tag{1.15}$$

en effet, rappelons que l'expression explicite de la conjuguée d'une fonction donnée n'est en pratique pas connue.

D'après (1.15), remarquons que le calcul de x^{k+1} revient à minimiser une fonction convexe déduite de la fonction DC $f = g - h$, en approximant la composante concave $-h$ par une de ses minorantes affines au point x^k, i.e.

$$x^{k+1} \in \partial g^\star(y^k) : \quad x^{k+1} \in argmin\{g(x) - \big[\langle y^k, x - x^k\rangle + h(x^k)\big] : x \in X\}.$$

Et similairement, par dualité

$$y^{k+1} \in \partial h(x^{k+1}) : \quad y^{k+1} \in argmin\{h^\star(y) - \big[\langle x^{k+1}, y - y^k\rangle + g^\star(y^k)\big] : y \in Y\}.$$

1.3.4 Optimisation DC polyédrale

L'optimisation DC polyédrale survient lorsque l'une des composantes convexes g ou h est convexe polyédrale. A l'instar des problèmes d'optimisation convexe polyédrale, cette classe de problème d'optimisation DC se rencontre fréquemment en pratique et possèdent d'intéressantes propriétés. Nous allons voir que la description de DCA y est particulièrement simple ([1, 2, 5]).

Soit le programme DC

$$inf\{g(x) - h(x) : x \in X\} \quad (P),$$

lorsque la composante convexe h est polyédrale, i.e.

$$h(x) = \max_{x \in X}\{\langle a^i, x\rangle - b^i : i = 1, ..., m\},$$

alors le calcul des sous-gradients $y^k = \partial h(x^k)$ est immédiat. Il est clair qu'en limitant (naturellement) le choix des sous-gradients aux gradients des fonctions affines minorantes de h, i.e. $\{y^k\} \in \{a^i : i = 1, ..., m\}$, qui est un ensemble fini, la suite des itérés $\{y^k\}$ sera finie ($k \leq m$). En effet, la suite $\{(h^\star - g^\star)(y^k)\}$ est par construction de DCA décroissante et les choix possibles des itérés y^k sont finis. De même, par dualité les suites $\{x^k\}$ et $\{(g - h)(x^k)\}$ sont décroissantes.

Théorème 1.5 ***(Convergence finie)***
- *les suites* $\{g(x^k) - h(x^k)\}$ *et* $\{h^\star(y^k) - g^\star(y+)\}$ *sont décroissantes,*
- *lorsque* $(g-h)(x^{k+1}) = (g-h)(x^k)$ *alors l'algorithme s'arrête à la* $(k+1)^{ieme}$ *itération et le point* x^k *(resp.* y^k*) est un point critique de* $g-h$ *(resp.* $h^\star - g^\star$*).*

Remarquons que si c'est la composante g qui est polyédrale, de par la conservation du caractère polyédrale par la conjugaison fonctionnelle et de l'écriture du problème dual, on retrouve les mêmes résultats ci-dessus.

1.3.5 Interprétations de DCA

A chaque itération on remplace dans le programme DC primal la deuxième composante DC h par sa minorante affine $h_k(x) = h(x^k) + \langle x - x^k, y^k\rangle$ au voisinage de x^k pour obtenir le programme convexe suivant

$$\inf\{\overline{f_k} = g(x) - h_k(x) : x \in \mathbb{R}^n\} \tag{1.16}$$

dont l'ensemble des solutions optimales n'est autre que $\partial g^\star(y^k)$.

De manière analogue, la deuxième composante DC $g^\star$ du programme DC dual (1.5) est remplacée par sa minorante affine $(g^\star)_k(y) = g^\star(y^k) + \langle y - y^k, x^{k+1}\rangle$ au voisinage de y^k pour donner naissance au programme convexe

$$\inf\{h^\star(y) - (g^\star)_k(y) : y \in \mathbb{R}^n\} \tag{1.17}$$

dont $\partial h(x^{k+1})$ est l'ensemble des solutions optimales. DCA opère ainsi une double linéarization à l'aide des sous-gradients de h et $g^\star$. Il est à noter que DCA travaille avec les composantes DC g et h et non pas avec la fonction f elle-même. Chaque décomposition DC de f donne naissance à un DCA.

Comme $\overline{f_k}$ est une fonction convexe, le minimum x^{k+1} est défini par $0 \in \partial\overline{f_k}(x^{k+1})$ et la majoration de f par $\overline{f_k}$ assure la décroissance de la suite $\{f(x^k)\}$. En effet, comme h_k est une fonction affine minorante de h en x^k, $\overline{f_k}$ est bien une fonction convexe majorante de f,

$$f(x) \leq \overline{f_k}(x), \quad \forall x \in X,$$

qui coïncide en x^k avec f,

$$f(x^k) = \overline{f_k}(x^k),$$

donc en déterminant l'itéré x^{k+1} comme le minimum du programme convexe (1.16), la décroissance de la suite des itérés est assurée,

$$f(x^{k+1}) \leq f(x^k).$$

Si à l'itération $k+1$, $f(x^{k+1}) = f(x^k)$ alors x^{k+1} est un point critique de $f(0 \in \partial\overline{f_k}(x^{k+1}) \Longrightarrow 0 \in \partial f(x^{k+1}))$.

Remarque 1.5 *Si $\overline{f_k}$ est strictement convexe alors il existe un unique minimum x^{k+1}.*

Commentaire : il est important de remarquer que l'on remplace, non localement au voisinage de x^k, mais globalement sur tout le domaine, la fonction f par la fonction :

$$\overline{f_k}(x) = g(x) - (\langle y^k, x - x^k\rangle + h(x^k)) \quad avec \quad y^k \in \partial h(x^k), \forall x \in X$$

qui, considérée localement au voisinage de x^k, est une approximation du premier ordre de f et globalement sur $\mathbb{R}^n$. Il faut souligner que $\overline{f_k}$ n'est pas définie restrictivement à partir d'information locale de f au voisinage de x^k (i.e. $f(x^k), \partial f(x^k)$, ...) mais incorpore toute la première composante convexe de f dans sa définition, i.e. $\overline{f_k} = g - h_k = f - (h + h_k)$. En d'autre terme, $\overline{f_k}$ n'est pas simplement une approximation locale de f au voisinage de x^k, mais doit être plutôt qualifiée de "convexification majorante" de f globalement liée à la fonction DC par la première composante convexe définie sur $\mathbb{R}^n$ tout entier. Par conséquent, les pas de déplacement de x^k à x^{k+1} sont déterminés à partir de f définie globalement pour tout $x \in \mathbb{R}^n$. DCA ne peut donc être simplement considéré, comme une méthode d'approximation locale ou de descente locale, telle que l'on connaît classiquement, de par le caractère globale de la "convexification majorante". Ainsi, à la différence des approches locales conventionnelles (déterministes ou heuristiques), DCA exploite simultanément des propriétés locales et globales de la fonction à minimiser au cours du processus itératif et converge en pratique vers une bonne solution locale, voire parfois globale.

Pour une étude complète de la programmation DC et DCA, se reporter aux [1]-[31] et [38]-[43] et références incluses. Le traitement d'un programme non convexe par une approche DC et DCA devrait comporter donc deux tâches : la recherche d'une décomposition DC adéquate et celle d'un bon point initial. Pour un programme DC donné, la question de décomposition DC optimale reste ouverte, en pratique on cherche des décompositions DC bien adaptées à la structure spécifiques du programme DC étudié pour lesquelles les suites $\{x^k\}$ et $\{y^k\}$ sont faciles à calculer, si possible explicites pour que les DCA correspondants soient moins coûteux en temps et par conséquent capables de supporter de très grandes dimensions.

Chapitre 2

La conception d'une chaîne d'approvisionnement multi-niveaux

Résumé *Ce chapitre concerne une nouvelle approche continue de la programmation DC et DCA pour la résolution du problème de la conception d'une chaîne d'approvisionnement multi-niveaux. Ce problème se définit lorsque la production d'une occasion de marché est lancée parmi un ensemble de partenaires potentiels. L'occasion de marché est caractérisée par une prévision déterministe dans tout l'horizon donné. On suppose que le produit soit traité dans un ordre fixe de quelques étapes distinctes, et que chaque étape puisse avoir un certain nombre de partenaires potentiels. L'objectif de ce problème stratégique est de concevoir la chaîne d'approvisionnements en choisissant un partenaire pour chaque étape. Ce choix doit satisfaire la demande prévue sur l'horizon donné sans retard. La plupart des méthodes existantes pour résoudre ce problème sont basées sur l'heuristique. Dans ce travail, nous traitons ce problème par DCA via la pénalité exacte en basant sur les décompositions DC appropriées et proposons une technique de recherche de bon point initial. Les simulations numériques sur plusieurs jeux d'essais empiriques montrent l'efficacité de notre approche par rapport aux méthodes standards.*

2.1 Introduction

De nos jours, le fait de conserver son avantage concurrentiel sur le marché est un défi de plus en plus complexe. Les clients s'attendent toujours à obtenir de meilleurs services, tout en les payant moins cher. La gestion de la chaîne d'approvisionnement est une fonction qui consiste essentiellement à satisfaire les besoins des clients de façon efficiente. Dans ce contexte, de plus en plus d'entreprises se rendent compte qu'une gestion efficace de leur chaîne d'approvisionnement est un facteur clé de la réussite : prévoir la demande est une tâche remplie d'incertitude ; la gestion des stocks par le fournisseur figure parmi les attentes courantes de votre clientèle ; les acquisitions, les nouveaux canaux et les désinvestissements deviennent monnaie courante.

La gestion d'une chaîne d'approvisionnement est un ensemble d'approches utilisées pour intégrer efficacement les fournisseurs, les manufacturiers, les entrepôts, les distributeurs, les détaillants et les clients de manière à produire et à distribuer les bonnes quantités de produits, aux bons endroits et au bon moment pour réduire les coûts inhérents à l'ensemble du système, tout en rencontrant les niveaux de services désirés par les clients. A haut niveau, une chaîne d'approvisionnement est composée de deux processus intégrés : le processus de planification de la production et le processus de distribution et de logistique. Ils sont illustrés dans le schéma 2.1 ci-dessous.

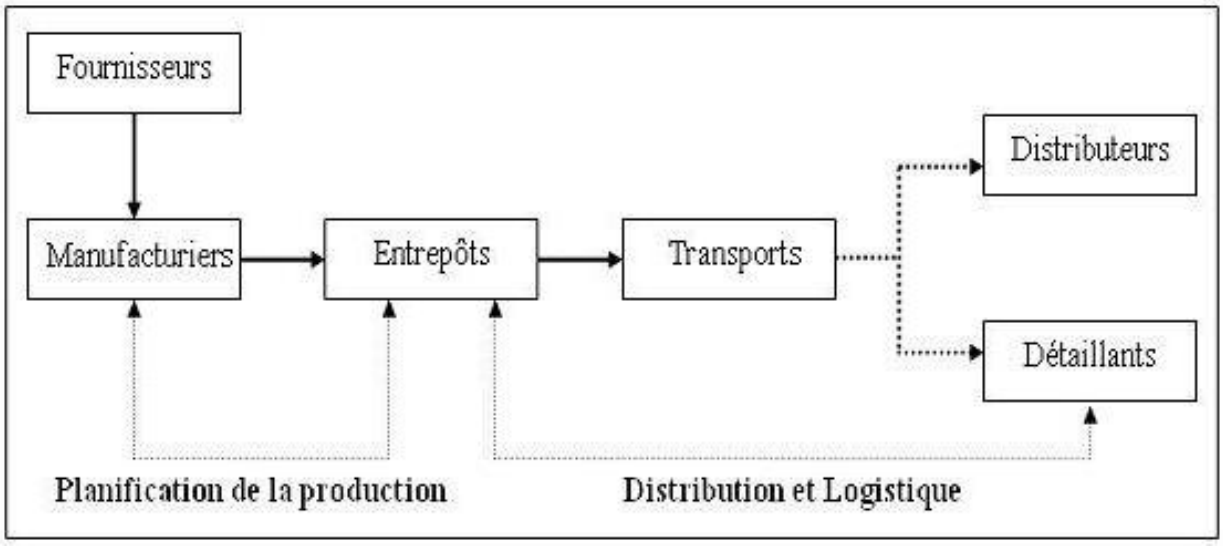

FIG. 2.1 – Le processus d'une chaîne d'approvisionnement

Le processus de planification de la production contient des sous processus de fabrication et de stockage. Plus spécifiquement, la planification de la production décrit la conception et la gestion du processus de fabrication entier et des politiques, des procédures de stockage.

Le processus de distribution et de logistique détermine comment des produits sont récupérés et transportés de l'entrepôt aux détaillants. Ces produits peuvent être transportés aux détaillants directement, ou peuvent d'abord être déplacés aux distributions qui, à leur tour, transportent des produits aux détaillants. Ce processus inclut la gestion de la récupération des produits, du transport, et de la livraison finale de produits.

Il y a principalement trois tâches les plus importantes de la gestion d'une chaîne d'approvisionnement :

1. La conception et l'optimisation d'une chaîne d'approvisionnement : déterminer quelle est la façon la plus optimale de fabriquer les produits, de les stocker et de les livrer.
2. L'évaluation et la sélection des technologies de gestion d'une chaîne d'approvisionnement : déterminer rapidement et précisément les exigences auxquelles devront répondre tous les

nouveaux systèmes pour une chaîne d'approvisionnement.
3. La mise en œuvre des technologies de gestion d'une chaîne d'approvisionnement : mise en œuvre et intégration d'applications, et gestion des projets des partenaires visant à mettre en place des technologies de gestion de chaîne d'approvisionnement.

Dans le cadre de ces travaux, nous nous contentons d'étudier la première tâche. C'est une mission assez complexe pour des raisons suivantes :
- La chaîne d'approvisionnement est un réseau très complexe d'entreprises, d'équipements industriels et d'organisations ayant des objectifs conflictuels.
- L'équilibre entre l'offre et la demande est très difficile à atteindre.
- L'ensemble des systèmes varient dans le temps (planification de la production, stratégies de prix, disponibilités, coûts d'approvisionnement,...).
- Les problèmes de conception et de gestion sont nouveaux, mal compris par les acteurs et aucune solution générique n'est disponible.

Trois niveaux de conception d'une chaîne d'approvisionnement sont stratégique, tactique et opérationnel qui peuvent être distingués selon le temps horizontal ([104]). Le niveau stratégique considère des décisions à long terme. Il demande des données approximatives et agrégées. Le niveau opérationnel comporte des décisions à court terme et demande des données transactionnelles. Le niveau tactique dépend de deux éléments : le temps horizontal et la quantité et l'exactitude des données demandées.

Les études relatives à ces trois niveaux sont vastes. Dans ce travail, nous abordons seulement le niveau stratégique, explicitement, c'est le problème d'affectation essentiellement liés au choix des participants à une chaîne d'approvisionnement ainsi que le problème de structuration des chaînes d'approvisionnement. Plusieurs modèles de conception à ce niveau ont été étudiés et la plupart de ces modèles sont formulés sous forme de programmation linéaire en variables mixtes 0-1.

Dans un des articles avancés dans le domaine de la conception de système de distribution, Geoffrion & Graves ([112]) présentent une formulation de programmation linéaire en variables mixtes 0-1 de conception d'un système de distribution multi-produits. Ces auteurs proposent une décomposion de Bender pour la résoudre. Ce modèle représente un système de production-distribution avec plusieurs usines en connaissant les capacités, les centres de distribution et le certain nombre de zones du client. L'objective du problème est de minimiser le coût intégré de la distribution, de la production et du transport. Cependant, ce modèle ne considère pas le coût de stockage de chaque centre de distribution.

Plus récemment, Cohen & Lee ([110]) présentent une programmation déterministe non linéaire en variables mixtes 0-1 qui est une version simplifiée du modèle présenté par Cohen & al. 1989 ([111]) en déterminant le prix de décision stratégique de déploiement de ressource dans un réseau global de fabrication et de distribution. Dans ce modèle, les auteurs décrivent

différentes stratégies de déploiement des ressources pour chacune des étapes primaires de la chaîne d'approvisionnement. La fonction objective utilisée dans leur modèle est la maximisation de tout le bénéfice après impôts pour les fabrications et les distributions. Les coûts considérés sont le coût de la fourniture, de la production, de la distribution et du transport aussi bien que les droits de douane et le prix de transfert.

Une conception d'un système de distribution et production multi-produits est représentée par Brown et al. ([106]) sous la formulation linéaire en variables mixtes 0-1. Les auteurs proposent une méthode de décomposition basée sur l'attribution de l'équipement des usines, la production de chaque usine. Les méthodes heuristiques sont étudiées pour résoudre ce problème et les auteurs soulignent que cette sorte de décomposition fournisse une amélioration significative par rapport aux méthodes traditionnelles.

Tout récemment, dans l'industrie, il existe plusieurs modèles stratégiques extensifs de la conception d'une chaîne d'approvisionnement (voir [105, 112, 113, 114] pour le détail).

- Van Roy ([116]) développe un modèle d'optimisation du réseau de production et de distribution pour une compagnie pétrochimique, avec plusieurs niveaux de distribution.
- Arntzen et al. ([102]) décrivent le développement d'une large programmation entier mixte pour modeler les décisions d'une chaîne d'approvisionnement au "Digital Equipment Corporation".
- Ashayeri et al. ([103]) décrivent le développement et l'utilisation d'une large programmation entier mixte au "Volvo Car BV".

Le modèle que nous considérons est une conception stratégique d'une chaîne d'approvisionnement. Ce problème [107, 108] est modélisé comme une programmation linéaire en variables mixtes 0-1, et de nombreuses difficultés ont surgit quand le nombre de variables binaires augmente. Il y a plusieurs algorithmes developpés pour résoudre le problème linéaire en variables mixtes 0-1 tels que l'algorithme Séparation et Évaluation, la méthode de coupes ou la programmation dynamique ([112]). Quelques méthodes sont basées sur la conjonction avec d'autres. L'une d'entre elles est la méthode de Séparation et Coupe. Parmi des approches basées sur les propriétés principales des problèmes étant considérés, les méthodes heuristiques sont proposées pour trouver une bonne solution optimale globale. Dans notre travail, en utilisant la technique de la pénalité exacte, nous traitons ce problème comme une programmation DC dans le contexte de l'optimisation continue. De plus, nous combinons le DCA avec l'algorithme de Séparation et Évaluation pour trouver la solution optimale globale.

Le reste du chapitre est organisé de la façon suivante. Dans la deuxième section, nous présentons la description et la formulation du problème ([107]) comme une programmation linéaire en variables mixtes 0-1. La troisième section décrit comment reformuler le problème sous la formule continue via la technique de la pénalité exacte. Cette section est consacrée à la programmation DC et DCA pour résoudre ce problème continu. Un algorithme combiné de DCA et Séparation et Évaluation est présenté dans la quatrième section tandis que les

résultats numériques sont rapportés dans la cinquième section. Nous présentons finalement quelques commentaires de conclusion dans la dernière section.

2.2 Description et formulation

2.2.1 Description

Le modèle que nous considérons (voir [107, 108]) est composé de φ opérations (notés E_1, E_2,..., E_φ) pour fabriquer le produit. Une opération est exécutée à chaque étape en choisissant un partenaire potentiel. Chaque partenaire potentiel a un coût d'activité unitaire et un coût de stockage. Il y a un coût de transport entre deux partenaires potentiels successifs situés dans deux étapes successives et un coût fixe pour faire une colaboration entre eux. Ces coûts peuvent changer dans chaque étape, dans chaque période et par chaque partenaire. Le problème est représenté par un réseau G=(N, A), où N comme ensemble de sommets et A comme ensemble d'arrêts. Chaque sommet représente une location du partenaire potentiel et chaque arrêt représente une connexion entre deux étapes successives. L'objectif du problème est de rechercher la "meilleure" chaîne reliée d'un sommet dans la première étape à un sommet dans la dernière étape. L'objectif est de sélectionner un partenaire à chaque étape de la chaîne d'approvisionnement pour satisfaire les demandes sur toutes les périodes, bien entendu, le coût est le plus bas.

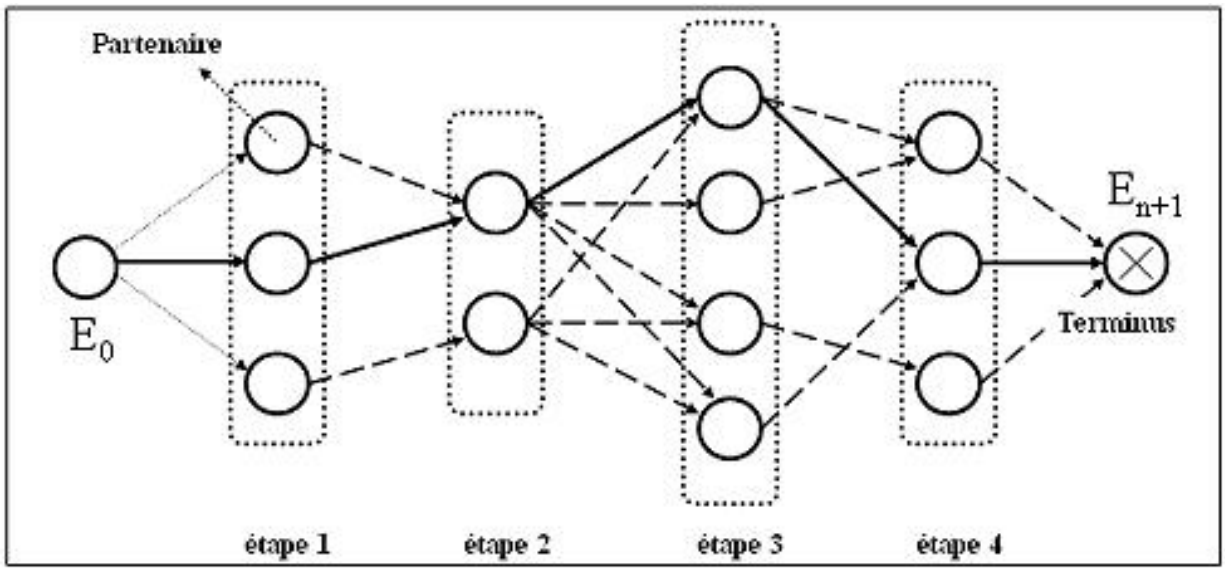

FIG. 2.2 – Le modèle de la chaîne d'approvisionnement du problème

La formulation du problème est considérée dans [107, 108] avec les principales caractéristiques du système :

- Il s'agit de satisfaire une opportunité du marché.
- La demande est connue sur l'horizon, mais peut varier d'une période à la suivante.
- La rupture de stock est interdite à la dernière étape. En d'autre termes, le client final doit être satisfait tout au long de l'horizon donné.
- La capacité disponible de chaque sommet est connue sur l'horizon de l'étude, mais peut varier d'une période élémentaire à l'autre.
- La somme du temps opératoire à une étape et du temps de transport de cette étape à la suivante est égale à une période élémentaire.
- Il n'y a pas de limite de capacité sur le transport d'une étape à la suivante.

2.2.2 Formulation mathématique ([107, 108])

Les paramètres utilisées :
- $P =< 1, ..., \varphi >$: l'ensemble des étapes requises pour réaliser le projet.
- T : le nombre de périodes sur lesquelles se fait l'optimisation.
- d_t avec $t \in T$: la demande de la période t.
- M : une valeur supérieure à la somme des demandes.
- N_α avec $\alpha \in P$: le nombre de partenaires potentiels à l'étape α.
- $F_{i,\alpha,j,\alpha+1}$ avec $\alpha = 1, 2, ..., \varphi - 1, i \in \{1, 2, ..., N_\alpha\}$ et $j \in \{1, 2, ..., N_{\alpha+1}\}$: le coût fixe d'établissement d'une connection entre le nœud i de l'étape α et le nœud j de l'étape $\alpha + 1$.
- $C_{i,\alpha,j,\alpha+1,t}$ avec $\alpha = 1, 2, ..., \varphi - 1, i \in \{1, 2, ..., N_\alpha\}, j \in \{1, 2, ..., N_{\alpha+1}\}$ et $t \in T$: le coût de transport unitaire du nœud i de l'étape α au nœud j de l'étape $\alpha + 1$ dans la période t.
- $H^f_{i,\alpha,t}$ avec $\alpha \in P, i \in \{1, 2, ..., N_\alpha\}$ et $t \in T$: le coût de stockage à la sortie du nœud i à l'étape α dans la période t.
- $H^r_{i,\alpha,t}$ avec $\alpha \in P, i \in \{1, 2, ..., N_\alpha\}$ et $t \in T$: le coût de stockage à l'entrée du nœud i à l'étape α dans la période t.
- $U_{i,\alpha,t}$ avec $\alpha \in P, i \in \{1, 2, ..., N_\alpha\}$ et $t \in T$: le coût unitaire de réalisation de l'operation par le nœud i de l'étape α durant la période t.
- $\Phi_{i,\alpha,t}$ avec $\alpha \in P, i \in \{1, 2, ..., N_\alpha\}$ et $t \in T$: la capacité disponible chez le nœud i de l'étape α durant la période t.

Variables :
- $h^f_{i,\alpha,t}$ avec $\alpha \in P, i \in \{1, 2, ..., N_\alpha\}$ et $t \in T$: le nombre d'unités en attente à la sortie du nœud i à l'étape α durant la période t.
- $h^r_{i,\alpha,t}$ avec $\alpha \in P, i \in \{1, 2, ..., N_\alpha\}$ et $t \in T$: le nombre d'unités en attente à l'entrée du nœud i à l'étape α durant la période t.
- $z_{i,\alpha,t}$ avec $\alpha \in P, i \in \{1, 2, ..., N_\alpha\}$ et $t \in T$: le quantité produite par le nœud i dans l'étape α durant la période t.
- $x_{i,\alpha,j,\alpha+1,t}$ avec $\alpha = 1, 2, ..., \varphi - 1, i \in \{1, 2, ..., N_\alpha\}, j \in \{1, 2, ..., N_{\alpha+1}\}$ et $t \in T$: la quantité du stock de sortie du nœud i de l'étape α vers le stock d'entrée du nœud j de l'étape $\alpha + 1$ durant la période t.

- $W_{i,\alpha}$: $\begin{cases} 1 & \text{si le nœud } i \text{ dans l'étape } \alpha \text{ est inclus dans la chaîne} \\ 0 & \text{sinon.} \end{cases}$
  Ici, $\alpha \in P$ et $i \in \{1, 2, ..., N_\alpha\}$.
- $Y_{i,\alpha,j,\alpha+1}$: $\begin{cases} 1 & \text{si tous les deux nœuds sont inclus dans la chaîne} \\ 0 & \text{sinon.} \end{cases}$
  Ici, $\alpha = 1, 2, ..., \varphi - 1$, $i \in \{1, 2, ..., N_\alpha\}$ et $j \in \{1, 2, ..., N_{\alpha+1}\}$.

Formulation de programmation linéaire en variables mixtes 0-1 :

$$(MIP) \quad \min \quad \sum_{\alpha=1}^{\varphi-1} \sum_{i=1}^{N_\alpha} \sum_{j=1}^{N_{\alpha+1}} F_{i,\alpha,j,\alpha+1} Y_{i,\alpha,j,\alpha+1}$$

$$+ \sum_{\alpha=1}^{\varphi-1} \sum_{t=1}^{T} \sum_{i=1}^{N_\alpha} \sum_{j=1}^{N_{\alpha+1}} C_{i,\alpha,j,\alpha+1,t} x_{i,\alpha,j,\alpha+1,t}$$

$$+ \sum_{\alpha=1}^{\varphi} \sum_{t=1}^{T} \sum_{i=1}^{N_\alpha} (H^f_{i,\alpha,t} h^f_{i,\alpha,t} + H^r_{i,\alpha,t} h^r_{i,\alpha,t} + U_{i,\alpha,t} z_{i,\alpha,t})$$

Tels que :

$$\sum_{i=1}^{N_\alpha} W_{i,\alpha} = 1 \quad \forall \alpha \in P \tag{2.1}$$

$$Y_{i,\alpha,j,\alpha+1} \geq W_{i,\alpha} + W_{j,\alpha+1} - 1 \tag{2.2}$$

$$\alpha = 1, 2, ..., \varphi - 1 \quad i = 1, ..., N_\alpha \quad j = 1, ..., N_{\alpha+1}$$

$$z_{i,\alpha,t} \leq \Phi_{i,\alpha,t} W_{i,\alpha} \quad \forall \alpha \in P \quad i = 1, ..., N_\alpha \quad t = 1, ..., T \tag{2.3}$$

$$\sum_{t=1}^{T} \sum_{j=1}^{N_{\alpha+1}} x_{i,\alpha,j,\alpha+1,t} \leq W_{i,\alpha} M \quad \alpha = 1, 2, ..., \varphi - 1 \quad i = 1, 2, ..., N_\alpha \tag{2.4}$$

$$\sum_{t=1}^{T} \sum_{i=1}^{N_\alpha} x_{i,\alpha,j,\alpha+1,t} \leq W_{j,\alpha+1} M \quad \alpha = 1, 2, ..., \varphi - 1 \quad j = 1, 2, ..., N_{\alpha+1} \tag{2.5}$$

$$h^f_{i,\alpha,t} = h^f_{i,\alpha,t-1} + z_{i,\alpha,t} - \sum_{j=1}^{N_{\alpha+1}} x_{i,\alpha,j,\alpha+1,t} \tag{2.6}$$

$$\alpha = 1, 2, ..., \varphi - 1 \quad i = 1, 2, ..., N_\alpha \quad t = 1, 2, ..., T$$

$$h^f_{i,\varphi,t} = h^f_{i,\varphi,t-1} + z_{i,\varphi,t} - d_t W_{i,\varphi} \quad i = 1,2,...,N_\varphi \quad t = 1,2,...,T \tag{2.7}$$

$$h^r_{j,\alpha+1,t} = h^r_{j,\alpha+1,t-1} - z_{j,\alpha+1,t} + \sum_{i=1}^{N_\alpha} x_{i,\alpha,j,\alpha+1,t} \tag{2.8}$$

$$\alpha = 1,2,...,\varphi-1 \quad j = 1,2,...,N_{\alpha+1} \quad t = 1,2,...,T$$

$$h^f_{i,\alpha,0} = 0 \quad i = 1,2,...,N_\alpha \quad \forall \alpha \in P \tag{2.9}$$

$$h^r_{i,\alpha,0} = 0 \quad i = 1,2,...,N_\alpha \quad \forall \alpha \in P \tag{2.10}$$

$$h^f_{i,\alpha,t}, h^f_{i,\alpha,t}, z_{i,\alpha,t} \geq 0 \quad i = 1,2,...,N_\alpha \quad \forall \alpha \in P \quad t = 1,2,...,T \tag{2.11}$$

$$x_{i,\alpha,j,\alpha+1,t}, Y_{i,\alpha,j,\alpha+1} \geq 0 \tag{2.12}$$

$$i = 1,2,...,N_\alpha \quad j = 1,2,...,N_{\alpha+1} \quad \alpha = 1,2,...,\varphi-1 \quad t = 1,2,...,T$$

$$W_{i,\alpha} \in \{0,1\} \quad i = 1,2,...,N_\alpha \quad \alpha = 1,2,...,\varphi \tag{2.13}$$

L'objectif du problème est de minimiser le minimum de la somme intégrée du coût d'établissement des connexions, du coût de transport, des coûts de stokage à l'entrée et à la sortie et du coût de production. Les contraintes (2.1) et (2.2) assurent qu'à chaque étape, un partenaire et un seul est choisi dans la chaîne d'appovisionnement. La contrainte (2.3) assure une capacité de production de chaque partenaire. Pour obtenir la collaboration entre deux partenaires à deux étapes successives, nous déduisons les contraintes (2.4) et (2.5). Pour assurer une relation entre le transport, la production et le stockage, les contraintes de (2.6) à (2.8) sont nécessaires. Les contraintes (2.9) et (2.10) spécifient les conditions initiales des partenaires au lancement de la chaîne d'appovisionnement. Les contraintes (2.11) et (2.12) indiquent que les quantités de production, de transport, et de stockage sont non négatives. La dernière contrainte (2.13) signifie que les variables W_i sont binaires. Dans ce modèle, nous considérons que toutes les variables en dehors de la production sont nulles.

Ce problème est une programmation linéaire en variables mixtes 0-1 et ainsi une programmation non convexe. La difficulté de ce problème dépend du nombre d'étapes, du nombre de partenaires(i.e. les nœuds) dans chaque étape et du nombre de périodes. Par exemple, si nous considérons un projet qui contient de 10 étapes avec 10 nœuds à chaque étape et 6 périodes, donc il y a 9400 variables, 100 variables binaires et 3030 contraintes. Le nombre de variables, le nombre de variables binaires et le nombre de contraintes sont calculés par des formules suivantes.

Le nombre de variables :

$$(T+1)*\sum_{\alpha=1}^{\varphi-1} N_\alpha N_{\alpha+1} + (3T+4)*\sum_{\alpha=1}^{\varphi} N_\alpha.$$

Le nombre de variables binaires : $\sum_{\alpha=1}^{\varphi} N_\alpha$.

Le nombre de contraintes :

$$(T+2)*\sum_{\alpha=1}^{\varphi-1} N_\alpha N_{\alpha+1} + (6T+5)*\sum_{\alpha=1}^{\varphi} N_\alpha + \varphi - N_\varphi - (T+1)N_1.$$

2.3 Programmation DC et DCA pour la résolution du problème

2.3.1 Reformulation du type minimisation concave

Dans cette section, utilisant les résultats bien connus de la technique de pénalité exacte, nous formulerons (**MIP**) sous la forme d'un problème d'optimisation concave. Considérez maintenant le problème linéaire en variables mixtes 0-1 sous la forme générale :

$$\textbf{(GMIP)} \quad \alpha = \min\left\{c^T x : (x,y) \in \mathcal{D}, y \in \{0,1\}^m\right\}$$

où $\mathcal{D}$ est un polyèdre convexe borné, non vide sur $\mathbb{R}^n \times \mathbb{R}^m$ défini par un nombre fini de contraintes linéaires.

Dans [9], (**GMIP**) est reformulé comme un programme quadratique concave via la fonction de pénalité exacte $\theta(y) = \sum_{i=1}^{m}(y_i, 1-y_i)$. Dans notre travail, nous utilisons une autre fonction de pénalité qui donne une programmation DC plus appropriée.

Considérons la fonction p définie par

$$p(x,y) = \theta(y) = \sum_{i=1}^{m} min(y_i, 1-y_i).$$

Soit $\mathcal{K} = \{(x,y) \in \mathcal{D} : y \in [0,1]^m\}$. Clairement la fonction p est concave et finie sur $\mathcal{K}$, $p(x,y) \geq 0$ pour tout $(x,y) \in \mathcal{K}$, et

$$\{(x,y) \in \mathcal{D}, \quad y \in \{0,1\}^m\} = \{(x,y) \in \mathcal{K}, \quad p(x,y) \leq 0\}.$$

Par conséquent (**GMIP**) peut être récrit par

$$\alpha = \min : \{c^T x : (x,y) \in \mathcal{K}, \quad p(x,y) \leq 0\}. \tag{2.14}$$

A partir du Théorème 2.1 ci-dessous nous obtenons, pour un nombre suffisamment grand t ($t \geq t_0$), le problème de minimisation concave équivalent à (**GMIP**) :

$$\min : \{c^T x + tp(x,y) : (x,y) \in \mathcal{K}\}. \tag{2.15}$$

Théorème 2.1 *([29]) Soient $\mathcal{K}$ un polyèdre convex borné non vide, f une fonction finie concave sur $\mathcal{K}$ et p une fonction finie concave non négative sur $\mathcal{K}$. Il existe $t_0 \geq 0$ tel que pour tout $t \geq t_0$, deux problèmes ci-dessous sont équivalents :*

$$(P_t) \quad \alpha(t) = \inf\{f(x) + tp(x) : x \in \mathcal{K}\}$$

$$(P) \quad \alpha = \inf\{f(x) : x \in \mathcal{K}, p(x) \leq 0\}.$$

Précisément, si l'ensemble de sommet de $\mathcal{K}$, dénoté par $V(\mathcal{K})$, est contenu dans $\{x \in \mathcal{K}, p(x) \leq 0\}$, alors $t_0 = 0$, sinon $t_0 = \min\left\{\dfrac{f(x) - \alpha(0)}{S} : x \in \mathcal{K}, p(x) \leq 0\right\}$, où $S = \min\{p(x) : x \in V(\mathcal{K}), p(x) > 0\} > 0$.

Soit $x \in \mathbb{R}^n$ les variables continues et $y \in \mathbb{R}^m$ les variables binaires du (**MIP**). Il est clair que l'ensemble $\mathcal{D}$ de points réalisables (x,y) déterminé par le système des contraintes $\{(2.1), ..., (2.12)\}$ est non vide, borné, polyèdre convex dans $\mathbb{R}^n \times \mathbb{R}^m$. (**MIP**) peut être exprimé sous une formule de (**GMIP**) et le résultat ci-dessus est satisfait du (**MIP**).

2.3.2 Résolution de (2.15) par DCA

Nous montrons d'abord que (2.15) est une programmation DC et puis nous présentons le DCA appliqué à ce problème.

Soit $\chi_{\mathcal{K}}$ la fonction indicatrice sur $\mathcal{K}$, $\chi_{\mathcal{K}}(x,y) = 0$ si $(x,y) \in \mathcal{K}$, $+\infty$ sinon. Soit g et h les fonctions définies par

$$g(x,y) = \chi_{\mathcal{K}}(x,y) \quad \text{et} \quad h(x,y) = -c^T x - t\sum_{i=1}^{m} min(y_i, 1 - y_i). \tag{2.16}$$

Par conséquent, g et h sont des fonctions convexes, donc le problème (2.15) est une programmation DC sous la forme

$$\min\{g(x,y) - h(x,y) \ : \ (x,y) \in \mathbb{R}^n \times \mathbb{R}^m\}. \tag{2.17}$$

Selon la description de DCA, la résolution de (2.15) via la formulation (2.17) par DCA consiste en la détermination de deux suites

$$(u^k, v^k) \in \partial h\ (x^k, y^k) \quad et$$

$$(x^{k+1}, y^{k+1}) \in \partial g^*(u^k, v^k).$$

La fonction h est différentiable et son gradient au point (x^k, y^k) est calculé de la manière suivante :

$$(u, v) \in \partial h(x, y) \Leftarrow u = -c, \quad v = (v_i)_i, \text{où} \quad v_i = \begin{cases} -t & si \quad y_i \leq 0.5 \\ t & sinon. \end{cases} \tag{2.18}$$

Le calcul de $(x^{k+1}, y^{k+1}) \in \partial g^*(u^k, v^k)$ se ramène à la résolution du problème suivant :

$$\min \left\{ -\langle (u^k, v^k), (x, y) : (x, y) \in \mathcal{K} \right\}. \tag{2.19}$$

Algorithme 2.1 *Schéma de DCA*

Initialisation :
- *Choisir* $(x^1, y^1) \in \mathbb{R}^n \times \mathbb{R}^m$ *et* $k = 1$.
- *Choisir les tolérances* ϵ_1 *et* ϵ_2 *positives suffisamment petites.*

Répeter
- *Calculer* $(u^k, v^k) \in \partial h(x^k, y^k)$ *via (2.18).*
- *Calculer* $(x^{k+1}, y^{k+1}) \in \partial g^*(u^k, v^k)$ *en résoudrant le programme linéaire (2.19) ;*
- $k + 1 \leftarrow k$

Jusqu'à $\left\|(x^{k+1}, y^{k+1}) - (x^k, y^k)\right\| \leq \epsilon_1$ *ou*

$$\left| \langle c, x^{k+1} - x^k \rangle + t \sum_{i=1}^{m} \left(min(y_i^{k+1}, 1 - y_i^{k+1}) - min(y_i^k, 1 - y_i^k) \right) \right| < \epsilon_2.$$

La convergence de l'algorithme 2.1 peut être récapitulée dans le prochain théorème dont la preuve est essentiellement basée sur le théorème de convergence d'un programme DC polyédral ([13, 16, 42]).

Théorème 2.2 *(Propriétés de la convergence de l'algorithme 2.1)*

(i) L'algorithme 2.1 génère une suite $\{(x^k, y^k)\}$ *dans* $V(\mathcal{K})$ *tel que la suite* $\{c^T x^k + t\theta(y^k)\}$ *est décroissante.*

(ii) Il existe un nombre non négatif t *tel que pour chaque* $t \geq t_1$ *la suite* $\{\theta(y^k\}$ *est décroissante. En particulier, si* (x^r, y^r) *est une solution réalisable de (**GMIP**) alors* (x^k, y^k)*, pour tout* $k \geq r$*, est réalisable également.*

(iii) La suite $\{(x^k, y^k)\}$ *converge à* $(x^*, y^*) \in V(\mathcal{K})$ *après un nombre fini d'itérations. Le point* (x^*, y^*) *est un point critique du problème (2.15). En plus, si* $y_i^* \neq \frac{1}{2}$ *pour* $i = 1, ..., m$*, alors* (x^*, y^*) *est une solution locale du problème (2.15).*

Preuve :

(i) est la conséquence du théorème de la convergence de DCA pour une programmation DC générale (voir [9, 16, 42, 43]).

(ii) Comme mentionné précédemment, si $V(\mathcal{K})$ est contenu dans l'ensemble de la solution réalisable de (**GMIP**) alors l'affirmation est triviale avec $t_1 = 0$. Sinon, posons

$$\xi = min\{\theta(y') - \theta(y) : ((x,y),(x',y')) \in V(\mathcal{K}) \times V(\mathcal{K}), \theta(y') > \theta(y)\},$$

$$\eta = max\{c^T x' - c^T x : ((x,y),(x',y')) \in V(\mathcal{K}) \times V(\mathcal{K})\},$$

donc $0 < \xi < +\infty$ et $0 \leq \eta < +\infty$ car l'ensemble $V(\mathcal{K})$ est finie. Considérons maintenant le nombre non négatif t_1 défini par $t_1 = \frac{\xi}{\eta}$ et $t > t_1$. Soit $\{(x^k, y^k)\}$ un ensemble généré par **Algorithme 2.1** appliqué au (2.15) à partir de cette valeur t. Supposons qu'il existe $r \geq 1$ tel que $\theta(y^{r+1}) > \theta(y^r)$. Alors

$$t[\theta(y^{r+1}) - \theta(y^r)] > t_1[\theta(y^{r+1}) - \theta(y^r)] = \frac{\xi}{\eta}[\theta(y^{r+1}) - \theta(y^r)] \geq \xi.$$

Par conséquent,

$$t[\theta(y^{r+1}) - \theta(y^r)] \geq c^T x^r - c^T x^{r+1},$$

i.e.,

$$c^T x^{r+1} + t\theta(y^{r+1}) \geq c^T x^r + t\theta(y^r),$$

ce qui contredit la décroissance de la suite $c^T x^k + t\theta(y^k)$.

(iii) Le problème (2.15) avec la décomposition DC (2.16) est une programmation polyédrale DC puisque les deux composants h et g sont des fonctions convexes polyédrales. Dans le chapitre 1, nous avons montré la convergence de DCA. Selon ce théorème de convergence, pour n'importe quelle programmation DC, la solution déterminée par DCA est un point critique de $g - h$, i.e.,

$$\partial g(x^*, y^*) \cap \partial h(x^*, y^*) \neq \emptyset. \tag{2.20}$$

Si $y_i^* \neq \frac{1}{2}, \forall i = 1, ..., m$, alors h est différentiable en (x^*, y^*) et la condition (2.20) devient $\partial h(x^*, y^*) \subset \partial g(x^*, y^*)$. Cet inclusion sous-differentielle, qui est une condition nécessaire de l'optimalité locale dans la programmation DC, est également suffisante dans le cas d'une programmation polyédrale DC dont le deuxième composant h de la décomposition DC est une fonction convexe polyédrale ([13, 16, 42]). La preuve est alors complète.

□

Remarque 2.1 *Selon le théorème 2.2, laissez-nous résumer les dispositifs principaux du DCA appliqués à (2.15) :*

(i) Toutes les deux suites $\{c^T x^k + t\theta(y^k)\}$ et $\theta(y^k)$ sont décroissantes.

*(ii) Si (x^r, y^r) est réalisable pour (**GMIP**) alors (x^k, y^k), pour tout $k \geq r$, est réaliable. Dans ce cas, la suite $\{(x^k, y^k)\}$ se déplace dans l'ensemble des solutions réalisables de (**GMIP**), $((x^k, y^k) \in V(\mathcal{K}), y^k \in \{0,1\}^m)$, et la fonction objective est diminuée.*

2.4 Relancer DCA : Un algorithme combiné de DCA et B&B

Pour trouver la solution optimale globale de ce problème, nous combinons le DCA avec l'algorithme de Séparation et Évaluation. La relaxation linéaire est utilisée pour calculer les bornes inférieures tandis que les bornes supérieures sont déterminées en appliquant le DCA à (2.15). Notre algorithme combiné peut être résumé comme suit : en commençant par le rectangle $R_0 = [0,1]^m$, nous considérons, à chaque itération $k \geq 0$, le rectangle R_k correspondant à la plus petite borne inférieure β_k. Le rectangle choisi R_k est divisé en deux sous-rectangles $R_{k,i=0,1}$ et la borne inférieure est améliorée en résolvant les programmations linéaires correspondantes. La détermination de la borne supérieure γ_k est effectuée grâce à l'application du DCA à (2.15). La procédure s'arrête quand $\gamma_k - \beta_k \leq \epsilon$. Dans ce cas, elle fournit une ϵ-solution optimale de (**MIP**).

Le schéma de l'algorithme combiné :

1. Choisir $R_0 = [0,1]^m$, $\gamma_0 = +\infty$, $\beta_0 = -\infty$, $restart = true$, $\mathcal{R} = \{R_0\}$ et k=0. Choisir une tolérance positive ϵ.
2. Sélectionner le rectangle R_k tel que $\beta_k = \beta(R_k) = \min\{\beta(R) : R \in \mathcal{R}\}$. Diviser R_k entre deux sous-rectangles R_{k_0} et R_{k_1} via l'index $j*$

$$R_{k_i} = \{y \in R_k : y_{j*} = i, \quad i = 0,1\}.$$

3. Calculer la borne inférieure $\beta_{k_i}(i = 0,1)$ en résolvant les programmations linéaires relaxées correspondantes à l'ensemble R_{k_i}.
4. Si ($restart = true$), alors mettre à jour γ_k, la meilleure borne supérieure de la valeur optimale de (**MIP**) en appliquant le DCA à (2.15) à partir d'un point initial trouvé dans l'étape 3.
5. Si $\mathcal{R} = \emptyset$ (i.e., $\gamma - \beta \leq \epsilon$), alors arrêter. Dans ce cas, (x^k, y^k) est la solution optimale. Sinon, mettre à jour $\mathcal{R} \leftarrow \mathcal{R} \cup \{R_{k_i} : \beta(R_{k_i}) < \gamma_k - \epsilon, i = 0,1\} \setminus R_k$ et retouner à l'étape 2.

La différence entre l'algorithme combiné et l'algorithme standard de Séparation et Évaluation est la détermination de la borne supérieur dans l'étape 3 en utilisant DCA. D'où, la variable *restart* est une variable booléenne qui nous permet de décider de relancer le DCA ou pas. La question *Quand le DCA est relancé ?* est intéressante selon le point de vue numérique. Elle sera étudiée dans la section 2.4.2. Comme dans plusieurs programmations DC, le DCA fournit une solution globale à (2.15) (et ainsi à (**MIP**)) à partir d'un bon point initial. Ce bon point peut être trouvé efficacement en calculant les bornes inférieures (voir la section 2.4.1 pour les détails). Selon les expériences numériques, nous remarquons que le DCA fournit une solution globale après une ou deux premières itérations de l'algorithme de Séparation et

Évaluation. Néanmoins, nous devons continuer le processus de l'algorithme de Séparation et Évaluation pour améliorer la borne inférieure jusqu'à ce que celle-ci devient très proche de la meilleure borne supérieure. En fait, l'algorithme de Séparation et Évaluation est présenté pour trouver un bon point initial de DCA et pour vérifier la globalité de DCA.

2.4.1 Recherche du bon point initial du DCA

A partir du théorème 2.2, nous constatons que, en commençant par une solution réalisable de (**MIP**), DCA fournit une meilleure solution réalisable, bien qu'il travaille sur un ensemble réalisable continu de (2.15). Il est important de trouver un bon point réalisable de (**MIP**) pour relancer le DCA. Pendant le processus de l'algorithme de Séparation et Évaluation, nous pouvons relancer le DCA à partir de la meilleure solution réalisable de (**MIP**) qui est découverte en calculant les bornes inférieures. L'algorithme combiné de DCA et de l'algorithme de Séparation et Évaluation pour la programmation quadratique non convexe ([9]) est une voie de recherche efficace. D'autre part, pour obtenir rapidement un bon point réalisable de (**MIP**), nous proposons également une nouvelle procédure pour calculer une solution optimale du programme quadratique concave ([9]).

$$0 = \min\Big\{ \sum_{i=1}^{m} y_i(1 - y_i) : (x, y) \in \mathcal{K} \Big\}. \tag{2.21}$$

Dans notre algorithme, un bon point initial est calculé en choisissant une des deux procédures qui dépend de la situation actuelle.

2.4.2 Quand DCA est relancé ?

D'une part, pendant le processus de l'algorithme de Séparation et Évaluation, nous relançons le DCA lorsqu'une solution réalisable à (**MIP**) qui améliore la meilleure borne supérieure courante obtenue.

D'autre part, le DCA est également relancé quand le nombre de composantes 0-1 de variables binaires (dénoté $N_{y^{R_k}}$) de la solution (x^{R_k}, y^{R_k}) au problème linéaire correspondant de relaxation est suffisamment grand, par exemple $N_{y^{R_k>}} \geq m/2$. Dans ce cas, le point initial de DCA est la solution du problème (2.21).

Nous décrivons maintenant l'algorithme combiné de DCA et l'algorithme de Séparation et Évaluation pour obtenir une solution optimale globale du problème (**MIP**).

2.4.3 Algorithme DCA-B&B

Algorithme 2.2 *Algorithme DCA-B&B*

Initialisation :
- *Choisir* $R_0 = [0,1]^m$.
- *Choisir une tolérance* $\epsilon > 0$.

Étape 0 :
- *Résoudre le problème linéaire relaxé de (**MIP**) pour obtenir une solution optimale* (x^{R_0}, y^{R_0}) *et la première borne inférieure* $\beta_0 = \beta(R_0)$.
- *Résoudre (2.15) en utilsant DCA à partir du point initial* (x^{R_0}, y^{R_0}) *pour obtenir* $(x_t^{R_0}, y_t^{R_0})$.
- ***Si*** $(x_t^{R_0}, y_t^{R_0})$ *est réalisable à (**MIP**)* ***Alors***
  Poser $\gamma_0 = c^T x_t^{R_0}$ *et* $(x^0, y^0) = (x_t^{R_0}, y_t^{R_0})$.
  Sinon $\gamma_0 = +\infty$.
- ***Si*** $(\gamma_0 - \beta_0) \leq \epsilon|\gamma_0|$ ***Alors***
  (x^0, y^0) *est une* ϵ*-solution optimale de (**MIP**).*
  Sinon
  $\mathcal{R} \leftarrow \{R_0\}, k \leftarrow 0$.

Tant que Stop $=$ *false* ***Faire***
- *Sélectionner le rectangle* R_k *tel que* $\beta_k = \beta(R_k) = \min\{\beta(R) : R \in \mathcal{R}\}$.
- *Diviser* R_k *entre deux sous-rectangles* R_{k_0} *et* R_{k_1} *via l'index* $j*$

$$R_{k_i} = \{y \in R_k : y_{j*} = i, \quad i = 0, 1\}.$$

- *Résoudre le sous-problème* (P_{k_i}) *pour obtenir* $\beta(R_{k_i})$ *et* $(x^{R_{k_i}}, y^{R_{k_i}})$:

$$(P_{k_i}) \quad \beta(R_{k_i}) = \min\left\{c^T x : (x, y) \in K, \quad y \in R_{k_i}\right\} \quad (i = 0, 1).$$

- ***Si*** $(x^{R_{k_i}}, y^{R_{k_i}})$ *est la meilleure solution réalisable de (**MIP**)* ***Alors***
  Mettre à jour γ_k *et la meilleure solution réalisable* (x^k, y^k) *à l'aide de DCA au (2.15) à partir du point* $(x^{R_{k_i}}, y^{R_{k_i}})$.
  Sinon Si $(N_{y^{R_{k_i}}} \geq m/2)$ ***Alors***
  Résoudre (2.21) en appliquant DCA pour obtenir $(x_t^{R_{k_i}}, y_t^{R_{k_i}})$.
  Appliquer DCA à (2.15) à partir du point $(x_t^{R_{k_i}}, y_t^{R_{k_i}})$.
  Mettre à jour γ_k *et la meilleure solution réalisable* (x^k, y^k).
  FinSi
  FinSi
- *Mettre à jour* $\mathcal{R} \leftarrow \mathcal{R} \cup \{R_{k_i} : \beta(R_{k_i}) < \gamma_k - \epsilon, i = 0, 1\} \setminus R_k$.
- ***Si*** $\mathcal{R} = \emptyset$ ***Alors***
  (x^k, y^k) *est la* ϵ*-solution optimale,*
  ARRÊTER.
  Sinon
  $k \leftarrow k + 1$.

FinTantque

2.5 Expériences numériques

L'algorithme a été exécuté sur un ordinateur DELL de 1GHz, 512Mb RAM en C avec la double précision. Pour résoudre la programmation linéaire, nous avons utilisé le logiciel CPLEX en version 9.1. Pour vérifier l'efficacité de notre méthode proposée, nous avons généré aléatoirement les données où le nombre de variables binaires est augmenté en changeant le nombre d'étapes et le nombre de nœuds à chaque étape. Tous les exemples générés ont eu au moins une solution réalisable. Dans cette expérience numérique, nous comparons l'efficacité de notre **Algorithme DCA-B&B** par rapport à l'algorithme de Séparation et Évaluation classique. Pour tous les jeux d'essais, nous avons toujours obtenu la ϵ-solution optimale, avec $\epsilon \leq 5.10^{-2}$. Les résultats de chaque jeu d'essai sont récapitulés dans les tableaux suivants avec 10 jeux d'essais.

Nous utilisons les notations suivantes :

- N^oit : le nombre d'itérations de chaque algorithme.
- UB, LB : la dernière borne supérieure et la dernière borne inférieure de chaque algorithme.
- CPU : le temps de calcul en seconde de chaque algorithme.
- R : le nombre de relances de DCA.
- N^oF : le nombre de relances de DCA dans l'**Algorithme DCA-B&B** quand le DCA fournit la première solution réalisable à (**MIP**).
- $Gap = \dfrac{UB - LB}{UB}$
- $\% = \dfrac{UB_0 - LB}{LB}$ où UB_0 est la première borne supérieure finie.

Le tableau **2.1** contient les résultats des jeux d'essais avec 5 étapes 6 noeuds à chaque étape et 8 périodes. Il y a 2280 variables avec 30 variables binaires et 929 contraintes. Les tableaux **2.2, 2.3, 2.4** contiennent les résultats des jeux d'essais avec 10 étapes, 10 noeuds à chaque étape et 2, 4, 6 périodes correspondantes. Il y a 100 variables binaires. La dernière ligne de chaque tableau indique le résultat moyen sur 10 jeux d'essais.

Commentaires. A partir des résultats dans les tableaux ci-dessous, nous constatons que :

- DCA lancé une seule fois a trouvé une ϵ-solution optimale à (**MIP**) dans plusieurs cas (13/40 jeux d'essais). Dans ce cas, l'**Algorithme DCA-B&B** a besoin plus d'itérations pour améliorer uniquement les bornes inférieures.
- La relance de DCA fournit très rapidement une première solution réalisable de (**MIP**). C'est à dire, le nombre moyen de relances de DCA est 1.6 sur 40 jeux d'essais.
- DCA fonctionne bien avec des problèmes où le nombre de variables binaires est grand. D'ailleurs, la supériorité de l'**Algorithme DCA-B&B** à l'algorithme de Séparation et Évaluation augmente quand le nombre de variables binaires augmente. L'**Algorithme DCA-B&B** est rapide pour des problèmes à grande taille tandis que l'algorithme de Séparation et Évaluation est très lent ou il ne peut pas résoudre certains problèmes avec

le temps raisonnable.

Conclusion. Nous avons présenté une approche efficace pour le problème de conception de chaîne d'approvisionnement multi-niveaux. L'objectif du problème est de sélectionner un partenaire à chaque niveau de la chaîne d'approvisionnement. Notre méthode est basée sur le DCA. Les résultats montrent que l'**Algorithme DCA-B&B** est très intéressant. Le DCA est original parce qu'il peut donner une solution entière tandis qu'il travaille sur un domaine continu.

N^o	Séparation et Évaluation					DCA et B&B							
	N^{o}it	UB	LB	CPU	Gap	N^{o}it	R	UB	LB	CPU	Gap	N^{o}F	%
01	23	12751540	12749999	17.250	0.01	6	3	13009398	12527906	27.703	3.84	2	3.84
02	6	13096550	12936516	4.656	1.24	1	1	13096550	12886344	3.875	1.63	1	1.63
03	26	12311228	12274897	20.797	0.30	10	3	12311228	12076966	31.500	1.94	1	5.92
04	7	11041377	10994134	8.687	0.43	1	1	11041377	10912780	5.906	1.18	1	1.18
05	12	12254463	12246380	16.016	0.07	8	2	12529261	12192327	35.969	2.76	2	2.76
06	27	12173214	12160131	32.062	0.11	10	2	12173214	11881686	36.141	2.45	2	2.45
07	143	13354261	13353492	93.000	0.01	39	7	13694028	13155286	115.00	4.10	2	4.42
08	6	11399543	11280127	7.454	1.06	1	1	11399543	11245948	6.813	1.37	1	1.37
09	19	12024911	11974646	16.422	0.42	11	2	12239793	11920423	23.735	2.68	2	2.68
10	51	12943749	12829760	48.469	0.89	17	3	13140308	12626941	57.391	4.07	3	4.07
T	**32.0**	/	/	**26.481**	**0.454**	**10.4**	**2.5**	/	/	**34.403**	**2.60**	**1.7**	**3.03**

TAB. 2.1 – La performance de l'algorithme
Il y a 5 étapes avec 6 nœuds à chaque étape. L'horizon choisi est 8. Dans ce cas, ce problème contient de 2280 variables avec 30 variables binaires et 929 contraintes.

N^o	Séparation et Évaluation					DCA et B&B							
	N^{o}it	UB	LB	CPU	Gap	N^{o}it	R	UB	LB	CPU	Gap	N^{o}F	%
01	28	8104834	8104460	32.05	0.01	10	4	8396799	8006590	35.28	4.87	2	10.59
02	24	8027372	7989584	22.16	0.47	9	3	8196882	7951863	34.53	3.08	3	3.08
03	11	5833312	5816744	12.61	0.28	1	1	5851223	5784518	4.20	1.15	1	1.15
04	171	6428269	6425561	128.49	0.04	31	8	6609466	6462683	67.39	2.27	2	6.25
05	20	6439370	6437419	16.94	0.03	1	1	6495153	6416677	3.99	1.22	1	1.22
06	250	7160614	7052829	225.95	1.53	19	7	7268901	6923530	65.34	4.99	2	9.17
07	65	8480361	8386595	53.99	1.12	44	4	8480361	8310454	65.25	2.04	3	7.69
08	145	8937962	8800644	118.84	1.56	20	7	8973170	8611216	73.00	4.20	7	4.20
09	28	8281668	8095728	27.11	2.30	10	4	8227518	8030711	39.84	2.45	2	13.67
10	113	8494666	8366524	110.06	1.53	54	3	8729058	8334181	79.70	4.74	3	4.74
T	**85.5**	/	/	**74.82**	**0.89**	**19.9**	**4.2**	/	/	**46.85**	**3.10**	**2.6**	**6.18**

TAB. 2.2 – La performance de l'algorithme
Il y a 10 étapes avec 10 nœuds à chaque étape. L'horizon choisi est 2. Dans ce cas, ce problème contient de 4600 variables avec 100 variables binaires et 1870 contraintes.

N^o	Séparation et Évaluation					DCA et B&B							
	N^{o}it	UB	LB	CPU	Gap	N^{o}it	R	UB	LB	CPU	Gap	N^{o}F	%
01	342	13775283	13668217	1024.03	0.78	16	1	13931102	13280286	126.64	4.90	1	4.90
02	174	12587549	12517932	428.41	0.56	45	8	12694792	12421078	306.31	2.20	1	5.58
03	77	15176654	15008953	211.75	1.12	1	1	15291922	14593939	20.52	4.78	1	4.78
04	78	14457193	14301062	317.97	1.09	1	1	14554540	13904302	17.38	4.68	1	4.68
05	1319	16762655	16555092	3444.72	1.25	189	4	17085218	16301870	1012.21	4.81	1	4.81
06	67	14435312	14271973	165.99	1.14	17	1	14846049	14184674	94.41	4.66	1	4.66
07	33	15021565	14911221	101.05	0.74	20	4	15467196	14852319	150.13	4.14	1	6.91
08	17	13164280	13120270	63.45	0.34	1	1	13164280	13018489	16.72	1.12	1	1.12
09	23	13776803	13404263	70.47	2.78	1	1	13776803	13267787	17.94	3.84	1	3.84
10	620	14959440	14696923	1342.77	1.79	118	22	14934627	14457342	747.49	3.30	1	11.57
T	**275.0**	/	/	**717.06**	**1.16**	**40.9**	**4.4**	/	/	**250.96**	**3.84**	**1**	**5.29**

TAB. 2.3 – La performance de l'algorithme
Il y a 10 étapes avec 10 nœuds à chaque étape. L'horizon choisi est 4. Dans ce cas, ce problème contient de 7000 variables avec 100 variables binaires et 2450 contraintes.

N^o	Séparation et Évaluation					DCA et B&B							
	N^{o}it	UB	LB	CPU	Gap	N^{o}it	R	UB	LB	CPU	Gap	N^{o}F	%
01	482	20333800	19896506	3672.70	2.20	136	20	20326982	19632766	2321.84	3.54	1	11.38
02*	144	20750750	20635325	1506.38	0.56	5	1	20859678	19883544	196.88	4.91	1	4.91
03	622	21501251	21199952	4163.52	1.42	186	29	21855651	20965148	2936.67	4.25	1	8.17
04*	52	20490694	20281185	804.86	1.03	28	3	21233824	20303946	661.95	4.58	1	4.58
05*	99	21419045	21203578	1080.16	1.02	4	1	21727324	20700557	170.92	4.96	1	4.96
06	301	22394747	22024054	2241.97	1.68	102	14	22881899	21825177	1633.05	4.84	1	4.84
07*	352	21659415	21539517	3868.56	0.56	138	14	21667224	21346899	2715.89	1.50	1	8.51
08	333	21616715	21367102	3312.48	1.17	76	26	22120621	21152122	1801.58	4.58	1	7.08
09*	259	21999296	21737766	2085.94	1.20	101	13	22655609	21734990	1869.78	4.24	1	4.24
10	425	20551240	20281236	2928.49	1.33	156	26	21043910	20084409	2474.47	4.78	1	10.15
T	**306.9**	/	/	**2566.51**	**1.22**	**93.2**	**14.7**	/	/	**1678.30**	**4.22**	**1**	**6.89**

TAB. 2.4 – La performance de l'algorithme
Il y a 10 étapes avec 10 nœuds à chaque étape. L'horizon choisi est 6. Dans ce cas, ce problème contient de 9400 variables avec 100 variables binaires et 3030 contraintes.

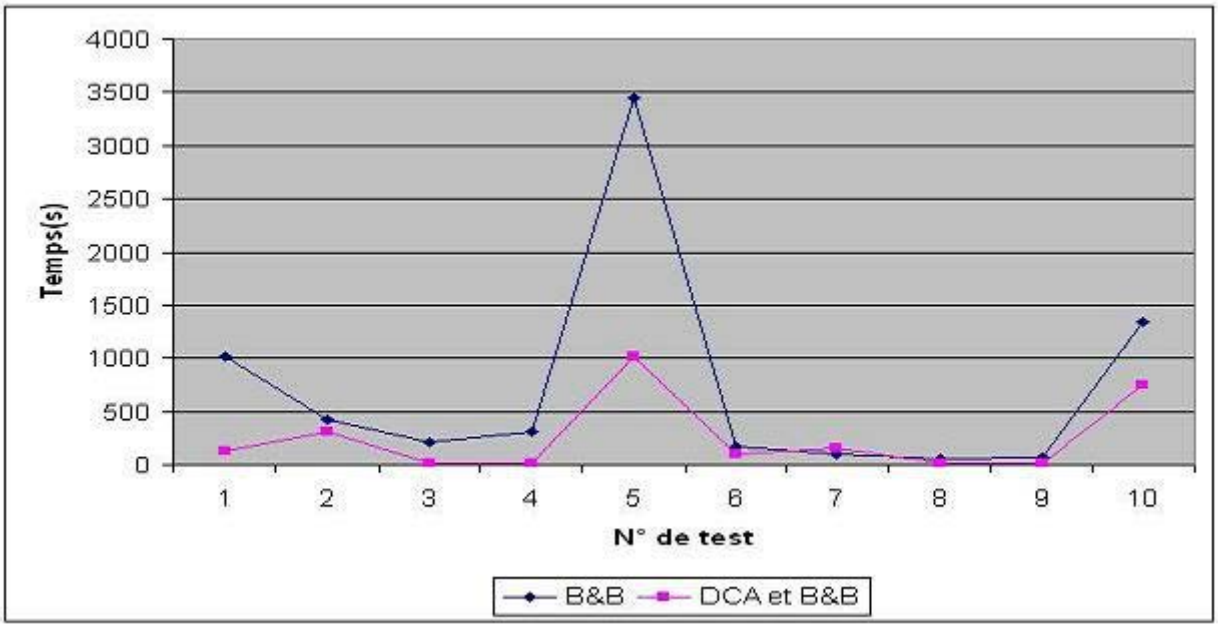

FIG. 2.3 – La comparaison sur le temps de calcul.
Avec 10 étapes, 10 nœuds à chaque étape et 4 périodes

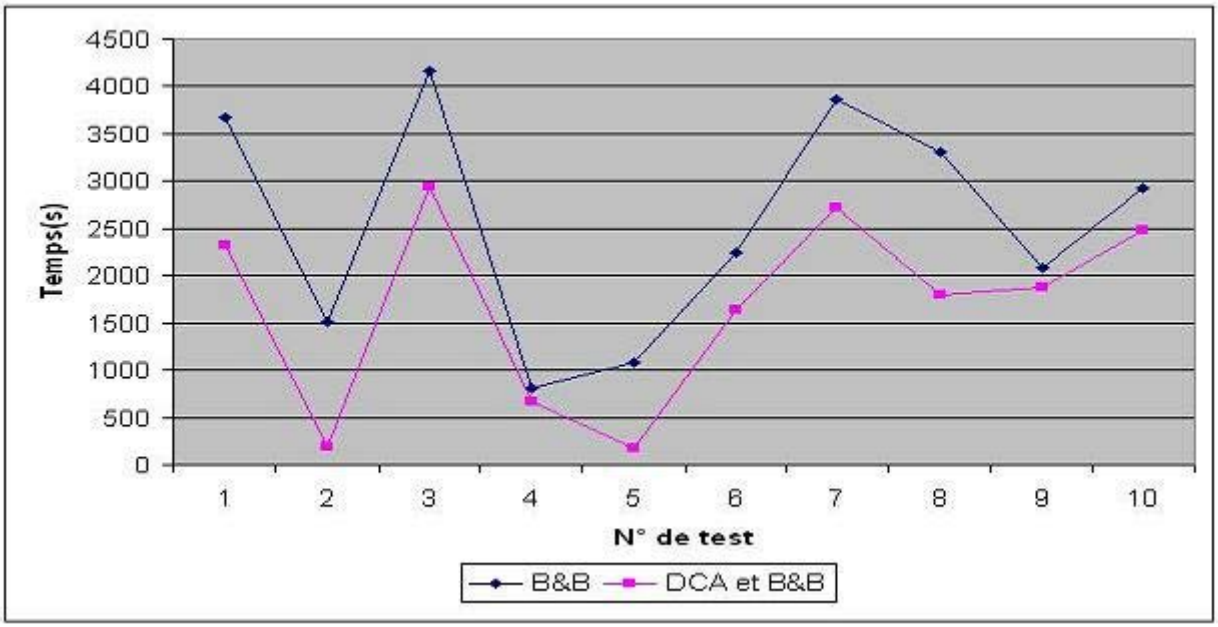

FIG. 2.4 – La comparaison sur le temps de calcul.
Avec 10 étapes, 10 nœuds à chaque étape et 6 périodes

Chapitre 3

Reconstruction d'images binaires

Résumé *La discrète tomographie se définit comme la reconstruction d'un sous ensemble de $\mathbb{Z}^n$ à partir de ses projections. Un de ses problèmes principaux est de reconstruire une matrice binaire $\mathbb{Z}^2$ à partir seulement de deux projections orthogonales, la projection horizontale H et la projection verticale V. La problématique est alors la suivante : Étant données les projections orthogonales H, V, existe-t-il une image binaire I ayant pour projections H et V et comment la reconstruire ? En général, le problème de la reconstruction à partir d'un nombre fini de projections est toujours sous-déterminé, de plus, pour plus de trois directions, il est NP-difficile. Il est nécessaire d'imposer les contraintes supplémentaires sur la partie de reconstruction. Une stratégie est basée sur l'augmentation du nombre de directions de la projection et une autre stratégie est d'ajouter les propriétés géométriques des objets qui peuvent être connues à priori (i.e. la convexité, la connexité, la périodicité ...).*

Dans ce chapitre, nous traitons le problème de reconstruction d'une image binaire par les trois approches différentes basées sur DCA et nous comparons l'efficacité de nos approches par rapport aux méthodes standards.

3.1 Introduction

La discrète tomographie (Discrete Tomography [DT] en Englais) qui a été nommée par Larry Shepp en 1994, est un domaine de la tomographie traitant les structures physiques discrètes. La discrète tomographie a sa propre théorie mathématique et la plupart des méthodes de résolution sont basées sur les mathématiques discrètes, ce qui a permis de présenter la discrète tomographie comme les problèmes combinatoires dans les années 1960. Ses applications sont nombreuses, notamment en médecine, en traitement d'images, en sécurité de données et en théorie des graphes ([127]).

Ce problème consiste à reconstruire un sous ensemble de $\mathbb{Z}^n$ à partir d'un ensemble de projections. Une de ses applications principales est la reconstruction d'une matrice binaire à partir seulement de deux projections orthogonales, la projection horizontale et la projection

verticale (voir la figure 3.1). La projection horizontale est la somme des éléments de chaque ligne et la projection verticale est la somme des éléments de chaque colonne de la matrice. En 1957, H. Ryser ([130]) et D. Gale ([121]) ont indépendamment tiré les conditions nécessaires et suffisantes pour l'existence d'une solution. Ryser a également fourni un algorithme en temps polynomial pour trouver une solution.

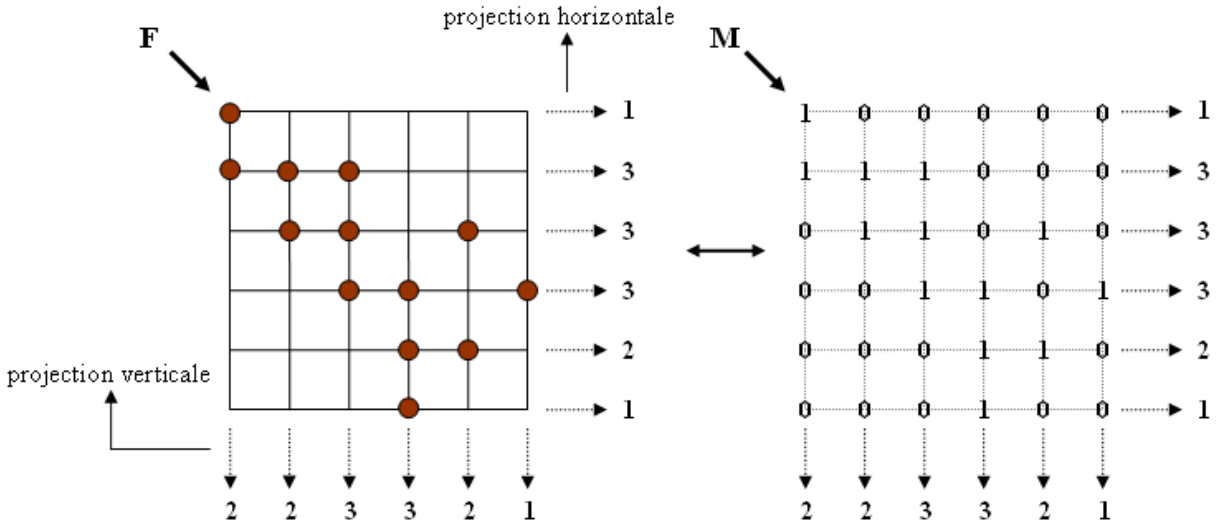

FIG. 3.1 – Un sous ensemble F de $\mathbb{Z}^2$ et la matrice binaire M équivalente

Cependant, le problème de la reconstruction d'un sous ensemble à partir d'un nombre fini de projections est toujours sous-déterminé et un grand nombre de solutions peuvent exister. La figure 3.2 illustre les solutions possibles d'un ensemble si nous ne considèrons que les projections orthogonales. Il est nécessaire d'imposer les contraintes supplémentaires sur la partie de reconstruction. Il y a deux approches pour résoudre ce problème :

i. La première stratégie est basée sur l'augmentation du nombre de directions de la projection. Malheureusement, dans ce cas, le problème de reconstruction devient insurmontable quand leur nombre de directions est supérieur à deux, de plus, si ce nombre est supérieur ou égal à trois directions, il est NP-difficile ([122]). Plusieurs méthodes en temps polynomial ont été développées pour trouver les solutions approchées (i.e. [120, 131, 132]). Dans ce contexte, une solution approchée est proche de la solution optimale si ses projections de l'ensemble de directions sont proches de celles de l'ensemble original. En 2000, P. Gritzmann et al. ([124]) ont présenté un modèle linéaire en variables $0-1$ et quelques algorithmes simples pour ce problème. Cependant si avec les mêmes données entrées nous n'obtenons pas une solution unique, les projections de cette solution peuvent être très différentes de celles de l'ensemble original. Dans [118], les auteurs montrent que les petits changements de données peuvent entrainer les solutions complètement différentes. En conséquence, l'approche basée seulement sur les mathématiques n'est pas suffisante pour obtenir une solution correcte.

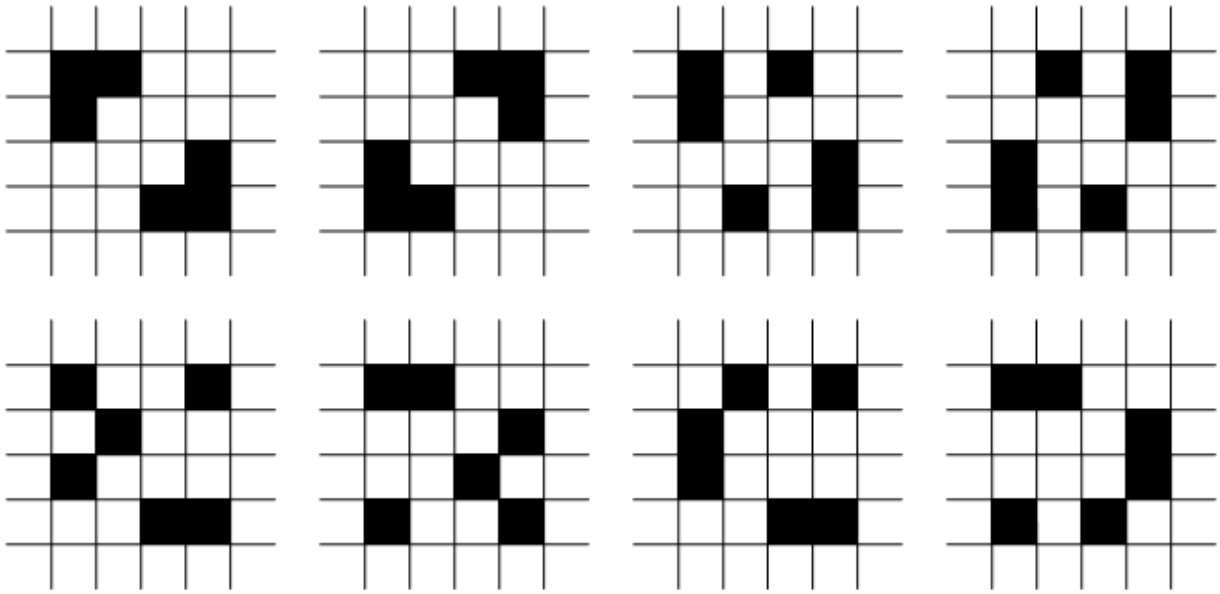

FIG. 3.2 – Les sous ensembles F de $\mathbb{Z}^2$ avec les mêmes projections orthogonales

ii. La deuxième stratégie se base sur l'ajout des propriétés géométriques qui peuvent être connues a priori par exemple la distribution, la convexité, la connexité, la périodicité ... Ces informations supplémentaires permettent à l'algorithme de recontruction de supprimer les solutions approchées incorrectes. Les premières approches sont présentées dans ([126]). S. Matej et al. ([129]) décrivent un algorithme modifié de Métropolis basé sur la distribution de Gibbs avec les trois projections données. T. Frese et al. ([119]) proposent une modélisation statistique dans un contexte bayesien et la solution est déterminée par la méthode multi-échelle. Les deuxièmes approches considèrent les informations à priori en terme de sous-ensembles auxquels la solution doit appartenir. Plusieurs travaux étudient le problème sur des classes des matrices binaires ayant des propriétés de convexité, de connectivité ou de périodicité. Par exemple, A.D Lungo et al. ([128]) présentent un algorithme basé sur la propriété de la connexité à partir des projections orthogonales. La convexité dans la recontruction d'une matrice binaire est décrite par R. Gardner et P. Gritzmann dans ([123]). En 2005, S. Weber et al. ([58]-[64]) proposent plusieus modèles équivalents qui sont basés sur le modèle de P. Gritzmann en considérant la distribution de Gibbs. Ils ont utilisé DCA pour la résolution de ces problèmes.

Notre travail traite le problème de reconstruction d'une image binaire en combinant tous les deux stratégies. Nous considérons ce problème sous les trois formes équivalentes en y ajoutant les informations spatiales (la relation entre le pixel et ses voisinages) :

– La première formulation est la minimisation d'une forme quadratique convexe en variables binaires. En utilisant la pénalité exacte nous le transformons en une programmation quadratique non convexe sur un rectangle pour laquelle nous utilisons le schéma de DCA

développé dans ([28]) qui est explicite et très efficace. En plus, nous proposons une nouvelle procédure d'estimation d'une borne inférieure d'une fonction quadratique dont les calculs sont assez simples. Cette procédure est utilisée pour la recherche d'un bon point initial pour DCA.
– La deuxième correspond à une programmation quadratique convexe en variables binaires. En utilisant un nouveau résultat concernant la pénalité exacte en programmation DC ([26]) nous reformulons le problème comme une programmation DC polyédrale et ainsi appliquons DCA.
– La troisième est basée sur le modèle linéaire en variables $0-1$ de P. Gritzmann. Grâce à la pénalité exacte, avec la même fonction de pénalité proposée dans le chapitre 2, nous obtenons une programmation DC polyédrale de même forme que celui du chapitre 2. Nous pouvons ainsi utilisé le même schéma DCA développé dans le chapitre précédent.

La question de recherche des bons paramètres de pénalité est également étudiée dans ce chapitre.

Le chapitre est organisé de la manière suivante. La deuxième section concerne les notations préliminaires et la formulation mathématique du problème. Dans la troisième section nous présentons les trois formulations équivalentes et nos algorithmes correspondants. La recherche des bons paramètres de pénalité pour améliorer la qualité de l'image reconstruite est présentée dans la quatrième section. Les résultats numériques sur certaines images binaires sélectionées sont présentés dans la dernière section.

3.2 Préliminaire et formulation

3.2.1 Préliminaire

Dans cette section, nous utilisons les notations employées par J.T Herman et al. ([127]). Considérons l'espace Euclidien d-dimensions, nous donnons quelques notations et définitions suivantes :
◦ $\mathbb{Z}$: un ensemble de nombres entiers.
◦ $\mathbb{N}_0$: un ensemble de nombres entiers non négatif.
◦ F : un ensemble de treillis (lattice sets en Anglais) fini et discret $F \subset \mathbb{Z}^d$.
◦ x : un élément de l'ensemble F.
◦ La direction de treillis est représentée par un vecteur non nul de $\mathbb{Z}^d$.
◦ Un ensemble fini des directions distinctes de trellis sera dénoté par D, par conséquent,

$$D = (v^1, v^2, ..., v^q), \quad q \geq 2.$$

◦ Une droite de treillis l dans l'espace Euclidien d-dimensions est une droite parallèle à un vecteur $v^k \in D$ et en outre $l \cap \mathbb{Z}^d$ est non vide.

- L'ensemble de toutes les droites de treillis qui sont parallèles au $v^k \in D$ est dénoté par L^k. La collection de l'ensemble de droites de treillis déterminées par D est réprésentée ensuite :

$$L = (L^1, L^2, ..., L^q), \quad q \geq 2.$$

- Soit Ω la classe des ensembles finis dans $\mathbb{Z}^d$.

La figure 3.3 illustre les définitions du problème de discrète tomographie.

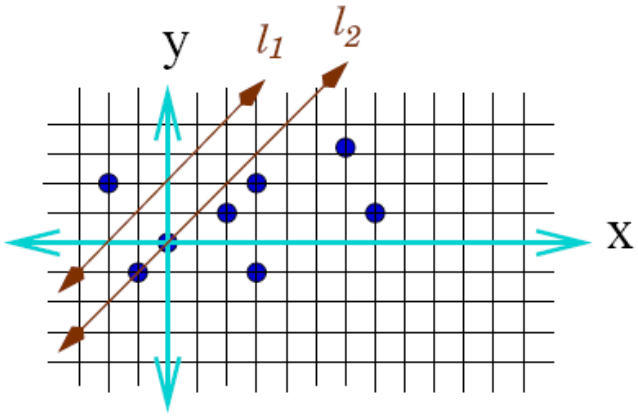

FIG. 3.3 – Illustration des définitions du problème de discrète tomographie.

Définition 3.1 *Soit $F \subset \mathbb{Z}^d$ un ensemble de treillis. Sa projection sur la direction v^k est définie par une fonction*

$$\mathcal{P}_F^k : L^k \to \mathbb{N}_0$$
$$\mathcal{P}_F^k(l) = \|F \cap l\| = \sum_{x \in l} f(x),$$

où f dénote la fonction caractéristique de l'ensemble discret F.

Définition 3.2 *Soient $F \subset \mathbb{Z}^d$ et $F^* \subset \mathbb{Z}^d$ les ensembles de treillis. F et F^* sont équivalents selon les directions $D = (v^1, v^2, ..., v^q) \quad q \geq 2$ si*

$$\mathcal{P}_F^k = \mathcal{P}_{F^*}^k, \quad \forall k = 1, 2, ..., q.$$

Définition 3.3 *Soit Ω être la classe des ensembles finis dans $\mathbb{Z}^d$. Soit $F \in \Omega$ un ensemble de treillis. F est déterminé selon les directions $D = (v^1, v^2, ..., v^q)$ dans la classe Ω s'il existe aucun F^* tel que F et F^* sont équivalents selon les directions D.*

Les problématiques sont alors les suivantes :

1. **Existence**(Ω, L)
   Données : Étant données les fonctions $p^k : L^k \to \mathbb{N}_0, \quad \forall k = 1, 2, ..., q.$
   Question : Existe-t-il un ensemble $F \in \Omega$ tel que $\mathcal{P}_F^k = p^k, \quad \forall k = 1, 2, ..., q$?

2. **Unicité**(Ω, L)
   Données : Étant donné l'ensemble $F \in \Omega$.
   Question : Existe-t-il un ensemble $F^* \in \Omega$ tel que F et F^* sont équivalents selon les directions $D = (v^1, v^2, ..., v^q)$?

3. **Reconstruction**(Ω, L)
   Données : Étant donné les fonctions $p^k : L^k \to \mathbb{N}_0$, $\forall k = 1, 2, ..., q$.
   Question : Construire un ensemble $F \in \Omega$ tel que $\mathcal{P}_F^k = p^k, \quad \forall k = 1, 2, ..., q.$

Supposont que nous avons les fonctions $p^k : L^k \to \mathbb{N}_0, \quad \forall k = 1, 2, ..., q$ tel que chaque fonction p^k consiste m_k droites de treillis. Soit $M = \sum_{i=1}^{q} m_i$ et $|F| = N$. Le problème de discrète tomographie est représenté sous un système d'équations linéaires suivant :

$$Px = b \quad \text{tel que} \quad x \in \{0, 1\}^N, \tag{3.1}$$

où $P \in \{0, 1\}^{M \times N}, b \in \mathbb{N}_0^M$. Ici, P est une matrice qui représente la relation entre un point de treillis de F et une droite de treillis l. Le vecteur x représente l'ensemble F et le vecteur b contient les valeurs des fonctions $p^k, \quad \forall k = 1, 2, ..., q$. Chaque projection correspond à une ligne de la matrice P et sa valeur de projection est la composante correspondante du vecteur b. Chaque élément $x_i \in \{0, 1\}$ indique si le pixel appartient à la projection. La figure 3.4 affiche l'ensemble $F \in \mathbb{Z}^2$ et le système d'équations linéaires équivalent.

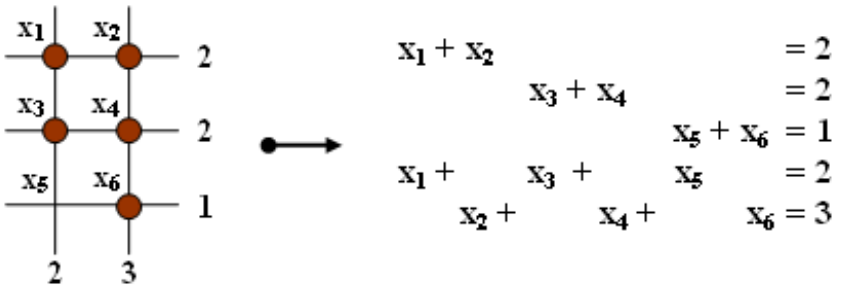

FIG. 3.4 – Le sous ensemble $F \in \mathbb{Z}^2$ et le système d'équations linéaires équivalent

Exemple : Considérer le sous ensemble F de $\mathbb{Z}^2$ dans la figure 3.4. C'est une 3×2-rectangle donc $M = 5$ et $N = 6$. Nous avons un système d'équations linéaires équivalent $Px = b$

suivant :

$$\begin{array}{cccccccccccc} x_1 & + & x_2 & & & & & & & & = & 2 \\ & & & x_3 & + & x_4 & & & & & = & 2 \\ & & & & & & x_5 & + & x_6 & = & 1 \\ x_1 & + & & x_3 & + & & x_5 & & & = & 2 \\ & & x_2 & + & & x_4 & + & & x_6 & = & 3. \end{array}$$

Dans ce cas, nous avons

$$P = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix}, \quad b = \begin{pmatrix} 2 \\ 2 \\ 1 \\ 2 \\ 3 \end{pmatrix}.$$

En général, le problème 3.1 peut transformer en problème de la programmation quadratique convexe en variables $0-1$

$$0 = \min \left\{ \|Px - b\|^2 \quad t.q \quad x \in \{0,1\}^N \right\}, \tag{3.2}$$

ou en problème de la programmation linéaire en variables $0-1$

$$\min \left\{ -\langle e, x \rangle \quad t.q \quad Px = b, \quad x \in \{0,1\}^N \right\}, \tag{3.3}$$

et le problème relaxé

$$\min \left\{ -\langle e, x \rangle \quad t.q \quad Px = b, \quad x \in [0,1]^N \right\}. \tag{3.4}$$

En 2000, P. Gritzmann et al. ([124]) ont proposé une approche approximative pour résoudre le problème 3.2

$$\min \left\{ -\langle e, x \rangle \quad t.q \quad Px \leq b, \quad x \in \{0,1\}^N \right\}. \tag{3.5}$$

C'est un modèle linéaire en variables mixtes $0-1$ et ils ont développé quelques algorithmes simples comme la méthode gloutonne pour résoudre les problèmes **BIF** (Best Inner Fit) et **BOF** (Best Outer Fit). Dans leur méthode gloutonne, on construit tout simplement une solution incrémentalement en rajoutant à chaque pas un élément selon un critère glouton.

$$\textbf{BIF} \quad \min \left\{ -\langle e, x \rangle \quad t.q \quad Px \leq b \quad x \in \{0,1\}^N \right\}. \tag{3.6}$$

$$\textbf{BOF} \quad \max \left\{ \langle e, x \rangle \quad t.q \quad Px \geq b \quad x \in \{0,1\}^N \right\}. \tag{3.7}$$

3.2.2 Formulations mathématiques

Le problème de reconstruction d'une image binaire est équivalent au problème de reconstruction d'une matrice binaire si nous considérons un pixel de l'image comme un élément de la matrice. La figure 3.6 illustre l'image binaire et la matrice binaire équivalente. Donc, le problème de la reconstruction d'une image binaire est représenté par un système d'équations linéaires $Px = b$ où P est une $M \times N$-matrice et b est un vecteur dans $\mathbb{R}^M$. Si nous considérons une image binaire de la taille $n_1 \times n_2$, alors $M = n_1 + n_2$ et $N = n_1 \times n_2$.

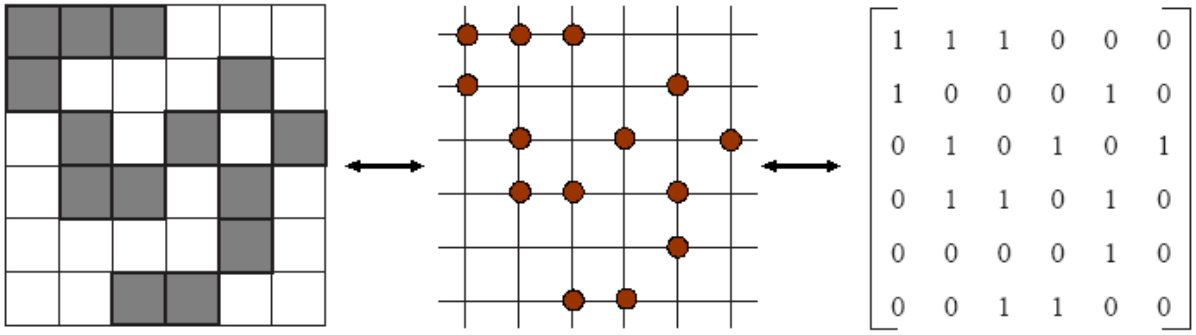

FIG. 3.5 – L'image binaire et la matrice binaire équivalente

Notre modèle dans ce travail est basé sur le modèle linéaire en variables $0-1$ de P. Gritzmann en y ajoutant la relation entre le pixel et ses voisinages dans le but de supprimer les solutions approchées incorrectes. En fait, cette relation (appelé l'information spatiale) est une des caractéristiques importantes d'une image car les voisinages possèdent souvent les valeurs semblables, et la probabilité qu'ils appartiennent à la même partition est très élevée. Il y a différentes manières de considérer cette information. Dans notre cadre, l'information spatiale est la somme de distance entre le pixel et ses voisinages et nous incorporons cette somme dans la fonction objective avec un coefficient associé. La figure 3.6 affiche le pixel et ses voisinages dans le cas où nous considérons les projections orthogonales.

3.2.2.1 Premier modèle

Dans [58], les auteurs ont considéré la minimisation d'une forme quadratique en variables 0-1 suivante :

$$\textbf{(QPB)} \qquad \begin{cases} \min \frac{1}{2}\|Px - b\|^2 + \alpha \sum_{i=1}^{N} \sum_{j \in V_i} (x_i - x_j)^2 \\ t.q \qquad x \in \{0,1\}^N, \end{cases} \tag{3.8}$$

où V_i est un ensemble de l'index des voisinages du pixel i.

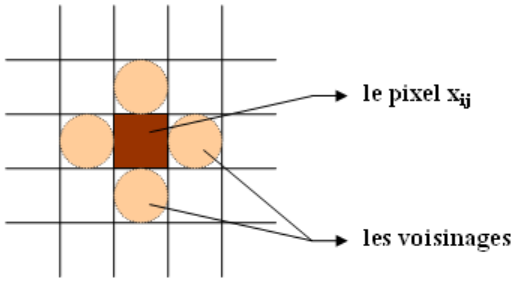

FIG. 3.6 – Le pixel et ses 4 voisinages

3.2.2.2 Deuxième modèle

A partir du modèle (3.3), nous considérons la minimisation d'une forme quadratique sur un polyèdre. Le modèle mathématique du problème s'écrit ainsi :

$$\textbf{(IQP)} \quad \begin{cases} \min -\langle e, x\rangle + \alpha \sum_{i=1}^{N} \sum_{j \in V_i} (x_i - x_j)^2 \\ t.q \qquad Px \leq b \\ \qquad\quad x \in \{0,1\}^N, \end{cases} \tag{3.9}$$

où e designe le vecteur dont tous les composantes sont égales à 1.

3.2.2.3 Troisième modèle

En utilisant les variables auxiliares $\{z_{i,j}\}$ où $z_{i,j} = (x_i - x_j)^2$, ce problème peut être exprimé sous la forme d'une programmation linéaire en variable mixtes 0-1 suivante :

$$\textbf{(MIP)} \quad \begin{cases} \min -\langle e, x\rangle + \alpha \sum_{i=1}^{N} \sum_{j \in V_i} z_{i,j} \\ t.q \qquad Px \leq b \\ \qquad\quad z_{i,j} \geq x_i - x_j \quad \forall i = 1, ..., N \quad j \in V_i \\ \qquad\quad z_{i,j} \geq x_j - x_i \quad \forall i = 1, ..., N \quad j \in V_i \\ \qquad\quad x \in \{0,1\}^N. \end{cases} \tag{3.10}$$

Les problèmes (**QPB**) et (**IQP**) sont les problèmes quadratiques convexes en variables $0-1$ tandis que le problème (**MIP**) est linéaire en variables mixtes $0-1$. La complexité dépend de la forme et du nombre de variables $0-1$ ainsi que du nombre de contraintes. Dans toutes les trois problèmes ci-dessus, le nombre de variables et le nombre de contraints sont très grands. Par exemple, considérons une image de la taille $n_1 \times n_2$ avec les projections orthogonales, il y a $n_1 n_2$ variables $0-1$ dans le premier modèle, $n_1 n_2$ variables $0-1$ et $n_1 + n_2$ contraintes

dans le deuxième et n_1n_2 variables $0-1$ et $4n_1n_2$ variables continues et plus de $4n_1n_2$ contraintes dans le troisième. Nous voyons bien que le nombre de variables et de contraintes a considérablement augmenté d'un problème à l'autre. Ainsi, l'approche par relaxation de variables $0-1$ comme l'algorithme BIF ne permet pas d'obtenir le meilleur résultat en raison de la valeur arrondie des variables $0-1$. Pour obtenir l'image avec de bonne qualité, nous proposons l'algorithme DCA pour tous les trois modèles en question en les transformant sous la forme du problème d'optimisation continue grâce à la technique de pénalité exacte. Il y a plusieurs approches de DCA pour ce type de problème en choisissant la fonction de pénalité différente ou la manière de la décomposition DC. Dans la prochaine section, vous allez trouver les détails sur la fonction de pénalité ou la manière de la décomposition DC.

3.3 Algorithmes

3.3.1 Résolution du problème (QPB) par DCA

Considérons le problème (**QPB**) ci-dessus. Clairement, le terme $\sum\limits_{i=1}^{N}\sum\limits_{j\in V_i}(x_i-x_j)^2$ est une fonction quadratique convexe où V_i désigne un ensemble de l'index des voisinages du pixel i. Dont, dans ce cas, le problème (**QPB**) peut s'écrire comme

$$\textbf{(GQPB)} \qquad \begin{cases} \min \frac{1}{2}\|Px-b\|^2+\frac{\alpha}{2}\langle x,Qx\rangle \\ t.q \qquad x\in\{0,1\}^N, \end{cases} \tag{3.11}$$

où

$$\sum_{i=1}^{N}\sum_{j\in V_i}(x_i-x_j)^2=\frac{1}{2}\langle x,Qx\rangle$$

avec Q est une matrice $N\times N$ semi-définie positive.

Considérons la fonction de pénalité p définie par

$$p(x)=\langle x,e-x\rangle.$$

Clairement la fonction p est concave et finie et

$$\{x\in\{0,1\}^N\}=\{x\in[0,1]^N,\quad p(x)\leq 0\}.$$

En utilisant un nouveau résultat concernant la pénalité exacte en programmation DC ([26]) sur le problème $\{\min f(x) : x\in\mathcal{K}\}$ où $\mathcal{K}$ est un polyèdre et $f(x)$ est une fonction DC, pour un nombre suffisamment grand t ($t\geq t_0$), nous reformulons le problème (**GQPB**) comme la programmation DC polyédrale suivante

$$\begin{cases} \min \frac{1}{2}\|Px-b\|^2+\frac{\alpha}{2}\langle x,Qx\rangle+tp(x) \\ t.q \qquad x\in[0,1]^N, \end{cases} \tag{3.12}$$

ou précisément,

$$\begin{cases} \min -\langle e, x\rangle + \frac{\alpha}{2}\langle x, Qx\rangle + t\langle x, e-x\rangle \\ t.q \qquad x \in [0,1]^N. \end{cases} \tag{3.13}$$

Le problème (**GQPB**) est simplifié par la forme suivante

$$\min\left\{\frac{1}{2}\langle x, Ax\rangle + \langle c, x\rangle : x \in [0,1]^N\right\}, \tag{3.14}$$

où

$$A = P^T P + \alpha Q - 2tI,$$

et

$$c = -P^T b + te.$$

Le problème (3.14) est une programmation quadratique non convexe et ainsi une programmation DC avec la décomposition DC suivante

$$g(x) = \frac{1}{2}\rho\|x\|^2 + c^T x + \chi_{\mathcal{B}}(x), \tag{3.15}$$

et

$$h(x) = \frac{1}{2}\langle x, (\rho I - A)x\rangle. \tag{3.16}$$

Ici, $\chi_{\mathcal{B}}$ est la fonction indicatrice sur le rectangle $\mathcal{B} = \{x \in \mathbb{R}^N : 0 \leq x_i \leq 1 \quad \forall i = 1, 2...N\}$, $\chi_{\mathcal{B}}(x) = 0$ si $x \in \mathcal{B}$, $+\infty$ sinon et ρ est un nombre positif pour que la matrice $(\rho I - A)$ soit semi-définie positive. Clairement, les fonctions g et h sont convexes sur $\mathcal{B}$.

Selon la description de DCA dans [28], la résolution de (3.14) par DCA consiste en la détermination de deux suites x^k et y^k

$$y = (\rho I - A)x^k, \tag{3.17}$$

$$x^{k+1} \in argmin\left\{\frac{\rho}{2}\|x\|^2 + \langle (c - y^k), x\rangle + \chi_{\mathcal{B}}(x) : x \in \mathbb{R}^N\right\}. \tag{3.18}$$

Il est clair que, (3.18) est équivalente à

$$\min_{x\in\mathcal{B}} \left\| x - \frac{y^k - c}{\rho} \right\|^2.$$

Dont, x^{k+1} est la projection de $\frac{y^k - c}{\rho}$ sur $\mathcal{B}$,i.e.,

$$x^{k+1} = P_{\mathcal{B}}\left(x^k - \frac{Ax^k + c}{\rho}\right).$$

Algorithme 3.1 *Algorithme* ***DCA-1***

Initialisation :
- *Choisir* $x^0 \in \mathbb{R}^N$ *et* $k = 0$.
- *Choisir les paramètres* α *et* t *positives.*
- *Choisir la tolérance* ϵ *positive suffisamment petite.*
- *Calculer les coefficients de la matrice* A*, du vecteur* c *et* ρ.

Répeter
- *Si* $0 \leq (\rho I - A)x^k - c \leq \rho$*, alors*

$$x^{k+1} = \frac{(\rho I - A)x^k - c}{\rho}.$$

- *Sinon,*
  $x^{k+1} = 0$ *si* $(\rho I - A)x^k - c < 0$,
  $x^{k+1} = 1$ *si* $(\rho I - A)x^k - c > \rho$.
- $k + 1 \leftarrow k$

Jusqu'à

$$\left\|x^{k+1} - x^k\right\| \leq \epsilon \quad ou \quad \left\|f(x^{k+1}) - f(x^k)\right\| \leq \epsilon_2.$$

Reconstruction d'une image binaire Soit x^* la solution calculée par DCA. Pour reconstruire l'image binaire, nous arrondissons les varibles x_i^* qui ne sont pas entiers. Dans ce cas, $x_i^* = 1$ si $x_i^* \geq 0.5$ et $x_i^* = 0$ sinon.

3.3.2 Le bon point initial du DCA pour le problème (QPB)

Le problème (3.14) est une programmation quadratique non convexe. Pour déterminer le bon point initial du DCA, nous devrons résoudre le problème relaxé du problème initial. Dans l'Annexe II, nous avons construit une fonction quadratique convexe sous-estimé de sa fonction objective. Dans le premier pas, nous devons résoudre ce nouveau problème quadratique convexe afin de trouver un bon point initial.

Considérons la fonction objective du problème (3.14) suivante

$$f(x) = \frac{1}{2}\langle x, Ax\rangle + \langle c, x\rangle. \tag{3.19}$$

La fonction quadratique convex sous-estimé de sa fonction est

$$\varphi(x) = \frac{1}{2}\langle x, (A - D + KI)x\rangle + \langle \beta, x\rangle + \beta_0, \tag{3.20}$$

où D est la matrice diagonale avec $D_{ii} = a_{ii}$, I est la matrice identité, K est un nombre positif tels que $K \geq ||H_f(x)||, \quad \forall x \in [0,1]^N$ et $H_f(x)$ est la matrice Hessienne de la fonction f.

Dans l'Annexe II, nous avons montré que la fonction $\varphi(x)$ est quadratique convexe et $\varphi(x) \leq f(x), \quad \forall x \in [0,1]^N$. Pour déterminer $\beta_i, \quad \forall i = 0, 1, ..., N$, nous considérons les $n+1$ sommets $v^0, ..., v^N$ de la boite $[0,1]^N$ et résolvons le système linéaire suivant

$$\begin{cases} \varphi(v^0) = f(v^0) \\ \varphi(v^1) = f(v^1) \\ \quad ... \\ \varphi(v^N) = f(v^N) \end{cases}. \tag{3.21}$$

Choisissons les sommets $v^0, ..., v^N$ de la manière suivante : $v^0 = \{0, 0, ..., 0\}$ est le vecteur 0, $v^i = e^i = \{0, 0, ..., 1, 0, ...0\}$ est le vecteur dont toutes les composantes sont nulles sauf la i^{ieme} qui est égale à 1. En conséquence,

$$\begin{cases} \beta_0 = 0 \\ \beta_1 = c_1 - \frac{1}{2}(K - a_{11}) \\ \quad ... \\ \beta_N = c_N - \frac{1}{2}(K - a_{NN}) \end{cases}. \tag{3.22}$$

Le problème quadratique convexe sous-estimé est représenté par

$$\min \left\{ \frac{1}{2}\langle x, (A - D + KI)x \rangle + \sum_{i=1}^{N} (c_i - \frac{1}{2}(K - a_{ii}))x_i : x \in [0,1]^N \right\}. \tag{3.23}$$

3.3.3 Résolution du problème (IQP) par DCA

Considérons le problème (**IQP**) sous la forme simplifiée suivante

$$\textbf{(GIQP)} \qquad \begin{cases} \min -\langle e, x \rangle + \frac{\alpha}{2}\langle x, Qx \rangle \\ t.q \qquad Px \leq b \\ \qquad\quad x \in \{0,1\}^N, \end{cases} \tag{3.24}$$

où

$$\sum_{i=1}^{N} \sum_{j \in V_i} (x_i - x_j)^2 = \frac{1}{2}\langle x, Qx \rangle$$

avec Q est une matrice $N \times N$ semi-définie positive.

Considérons la fonction de pénalité p définie par

$$p(x) = \sum_{i=1}^{N} min(x_i, 1 - x_i).$$

Soit

$$\mathcal{D}_1 = \{x : \quad Px \leq b\},$$

et $\mathcal{K}_1 = \{x \in \mathcal{D}_1 : x \in [0,1]^N\}$ où $\mathcal{D}_1$ est un polyèdre convexe borné, non vide sur $\mathbb{R}^N$ défini par un nombre fini de contraintes linéaires. Clairement la fonction p est concave et finie sur $\mathcal{K}_1$, $p(x) \geq 0$ pour tout $x \in \mathcal{K}_1$, et

$$\{x \in \mathcal{D}_1, \quad x \in \{0,1\}^N\} = \{x \in \mathcal{K}_1, \quad p(x) \leq 0\}.$$

Grâce à la pénalité exacte en programmation DC ([26]), nous reformulons le problème (**GIQP**) comme la programmation DC polyédrale suivante

$$\begin{cases} \min -\langle e, x\rangle + \frac{\alpha}{2}\langle x, Qx\rangle + tp(x) \\ t.q \qquad Px \leq b \\ \qquad\quad x \in [0,1]^N, \end{cases} \tag{3.25}$$

ou précisément,

$$\begin{cases} \min -\langle e, x\rangle + \frac{\alpha}{2}\langle x, Qx\rangle + t\sum_{i=1}^{N} min(x_i, 1-x_i) \\ t.q \qquad Px \leq b \\ \qquad\quad x \in [0,1]^N. \end{cases} \tag{3.26}$$

Le problème (3.26) est une programmation quadratique non convexe et ainsi une programmation DC avec la décomposition DC suivante

$$g(x) = \frac{\alpha}{2}\langle x, Qx\rangle - \langle e, x\rangle, \tag{3.27}$$

et

$$h(x) = -t\sum_{i=1}^{N} min(x_i, 1-x_i). \tag{3.28}$$

Selon la description de DCA dans la chapitre 1, la résolution de (3.26) par DCA consiste en la détermination de deux suites

$$y^k \in \partial h(x^k) \quad et$$
$$x^{k+1} \in \partial g^*(y^k).$$

La fonction h est différentiable et son gradient au point x^k est calculé de la manière suivante :

$$y \in \partial h(x)$$
$$y_i = \begin{cases} -t & x_i \leq 0.5 \\ t & sinon. \end{cases} \tag{3.29}$$

Le calcul de $x^{k+1} \in \partial g^*(y^k)$ à chaque itération se ramène à la résolution du problème quadratique convexe suivant :

$$\min\left\{\frac{\alpha}{2}\langle x, Qx\rangle - \langle (e + y^k), x\rangle : x \in \mathcal{K}_1\right\} \tag{3.30}$$

Algorithme 3.2 *Algorithme* ***DCA-2***

Initialisation :
– *Choisir* $x^0 \in \mathbb{R}^N$ *et* $k = 0$.
– *Choisir les paramètres* α *et* t.
– *Choisir les tolérances* ϵ_1 *et* ϵ_2 *positives suffisamment petites.*
Répeter
– *Calculer* $y^k \in \partial h(x^k)$ *via (3.29).*
– *Calculer* $x^{k+1} \in \partial g^*(y^k)$ *en résolvant le programme quadratique convexe (3.30) ;*
– $k + 1 \leftarrow k$
Jusqu'à

$$\left\|x^{k+1} - x^k\right\| \leq \epsilon_1 \quad ou \quad \left\|f(x^{k+1}) - f(x^k)\right\| \leq \epsilon_2.$$

La convergence de l'algorithme 3.2 peut être récapitulée dans le chapitre 1 dont la preuve est essentiellement basée sur le théorème de convergence d'un programme DC polyédral ([13, 16, 42]).

3.3.4 Résolution du problème (MIP) par DCA

Dans cette section, utilisant les résultats bien connus de la technique de pénalité exacte, nous formulerons (**MIP**) sous une forme du problème d'optimisation concave. Grâce à la pénalité exacte, avec la même fonction de pénalité proposée dans le chapitre 2, nous obtenons une programmation DC polyédrale équivalente et nous utilisons le même schéma DCA dans ce chapitre pour le résoudre.

Considérons la même fonction de pénalité p définie par

$$p(x, z) = \theta(x) = \sum_{i=1}^{N} min(x_i, 1 - x_i).$$

Soit

$$\mathcal{D}_2 = \{(x, z) : \quad Px \leq b, \quad z_{i,j} \geq x_i - x_j, \quad z_{i,j} \geq x_j - x_i\},$$

et $\mathcal{K}_2 = \{(x, z) \in \mathcal{D}_2 : x \in [0, 1]^N\}$ où $\mathcal{D}_2$ est un polyèdre convexe borné, non vide sur $\mathbb{R}^N \times \mathbb{R}^M$ défini par un nombre fini de contraintes linéaires. Clairement la fonction p est concave et finie sur $\mathcal{K}_2$, $p(x, z) \geq 0$ pour tout $(x, z) \in \mathcal{K}_2$, et

$$\{(x, z) \in \mathcal{D}_2, \quad x \in \{0, 1\}^N\} = \{(x, z) \in \mathcal{K}_2, \quad p(x, z) \leq 0\}.$$

Par conséquent (**MIP**) peut être récrit par

$$\begin{cases} \min -\langle e, x\rangle + \alpha \sum_{i=1}^{N} \sum_{j\in V_i} z_{i,j} \\ t.q \quad Px \leq b \\ \qquad z_{i,j} \geq x_i - x_j \quad \forall i = 1, ..., N \quad j \in V_i \\ \qquad z_{i,j} \geq x_j - x_i \quad \forall i = 1, ..., N \quad j \in V_i \\ \qquad p(x, z) \leq 0. \end{cases} \tag{3.31}$$

A partir du théorème 2.1 dans le chapitre 2, nous obtenons, pour un nombre suffisamment grand t ($t \geq t_0$), le problème de minimisation concave équivalent à (**MIP**) :

$$\begin{cases} \min -\langle e, x\rangle + \alpha \sum_{i=1}^{N} \sum_{j\in V_i} z_{i,j} + tp(x, z) \\ t.q \quad Px \leq b \\ \qquad z_{i,j} \geq x_i - x_j \quad \forall i = 1, ..., N \quad j \in V_i \\ \qquad z_{i,j} \geq x_j - x_i \quad \forall i = 1, ..., N \quad j \in V_i \\ \qquad x \in [0, 1]^N, \end{cases} \tag{3.32}$$

où précisément,

$$\begin{cases} \min -\langle e, x\rangle + \alpha \sum_{i=1}^{N} \sum_{j\in V_i} z_{i,j} + t \sum_{i=1}^{N} min(x_i, 1 - x_i) \\ t.q \quad Px \leq b \\ \qquad z_{i,j} \geq x_i - x_j \quad \forall i = 1, ..., N \quad j \in V_i \\ \qquad z_{i,j} \geq x_j - x_i \quad \forall i = 1, ..., N \quad j \in V_i \\ \qquad x \in [0, 1]^N, \end{cases} \tag{3.33}$$

Nous montrons d'abord que (3.33) est une programmation DC et puis nous présentons l'algorithme DCA pour résoudre ce problème. Soit

$$f(x, z) = -\langle e, x\rangle + \alpha \sum_{i=1}^{N} \sum_{j\in V_i} z_{i,j} + t \sum_{i=1}^{N} min(x_i, 1 - x_i).$$

Soit $\chi_{\mathcal{K}_2}$ la fonction indicatrice sur $\mathcal{K}_2$, $\chi_{\mathcal{K}_2}(x, z) = 0$ si $(x, z) \in \mathcal{K}_2$, $+\infty$ sinon. Soit g et h les fonctions définies par

$$g(x, z) = \chi_{\mathcal{K}_2}(x, z)$$

et

$$h(x, z) = \langle e, x\rangle - \alpha \sum_{i=1}^{N} \sum_{j\in V_i} z_{i,j} - t \sum_{i=1}^{N} min(x_i, 1 - x_i).$$

Par conséquent, g et h sont des fonctions convexes donc le problème (3.33) est une programmation DC sous la forme

$$\min\{g(x,z) - h(x,z) \ : \ (x,z) \in \mathbb{R}^N \times \mathbb{R}^M\} \tag{3.34}$$

Selon la description de DCA dans le chapitre 1, la résolution de (3.33) via la formulation (3.34) par DCA consiste en la détermination de deux suites

$$(u^k, v^k) \in \partial h\ (x^k, z^k) \quad et$$

$$(x^{k+1}, z^{k+1}) \in \partial g^*(u^k, v^k).$$

La fonction h est différentiable et son gradient au point (x^k, z^k) est calculé de la manière suivante :

$$(u,v) \in \partial h(x,z)$$

$$\begin{cases} u = (u_i)_i \leftarrow u_i = \begin{cases} 1-t & x_i \leq 0.5 \\ 1+t & sinon, \end{cases} \\ v = -\alpha e. \end{cases} \tag{3.35}$$

Le calcul de $(x^{k+1}, z^{k+1}) \in \partial g^*(u^k, v^k)$ se ramène à la résolution du problème suivant :

$$\min\left\{-\langle (u^k, v^k), (x,z) : (x,z) \in \mathcal{K}_2\right\} \tag{3.36}$$

Algorithme 3.3 *Algorithme* ***DCA-3***

Initialisation :
- *Choisir* $(x^0, z^0) \in \mathbb{R}^N \times \mathbb{R}^M$ *et* $k = 0$.
- *Choisir les paramètres* α *et* t.
- *Choisir les tolérances* ϵ_1 *et* ϵ_2 *positives suffisamment petites.*

Répeter
- *Calculer* $(u^k, v^k) \in \partial h(x^k, y^k)$ *via (3.35).*
- *Calculer* $(x^{k+1}, z^{k+1}) \in \partial g^*(u^k, v^k)$ *en résolvant le programme linéaire (3.36) ;*
- $k + 1 \leftarrow k$

Jusqu'à

$$\left\|(x^{k+1}, z^{k+1}) - (x^k, z^k)\right\| \leq \epsilon_1 \quad or \quad \left\|f(x^{k+1}, z^{k+1}) - f(x^k, z^k)\right\| \leq \epsilon_2.$$

La convergence de l'algorithme 3.3 peut être récapitulée dans le théorème 2.2 dans le chapitre 2.

3.4 Sélection du bon paramètre de pénalité

Nous avons utilisé la technique de pénalité exacte pour reformuler le problème sous la forme d'un problème d'optimisation concave. Généralement, le paramètre de pénalité joue en rôle

très important car il influence fortement sur la vitesse de la résolution et le résultat. Si le paramètre de pénalité est grand, l'algorithme DCA converge lentement, cependant si le paramètre de pénalité est petit, l'image obtenue n'est pas non plus, de bonne qualité. Il est donc très intéressant d'introduire une étude plus approfondie sur le choix d'un tel paramètre.

D'autre part, la sélection un bon point initial de l'algorithme DCA est très importante. A partir d'un bon point initial, DCA fournirait peut-être un résultat très semblable à l'image originale.

Ici, nous combinons la phase de recherche du paramètre de pénalité et la phase de recherche d'un bon point initial. L'idée de cette approche est simple : en commençant par une valeur de paramètre de pénalité $t = t_{min}$ suffisament grande, à chaque itération, nous résoudrons le problème (3.26) (resp. (3.33)) par DCA pour obtenir la solution locale x^* (resp.(x^*, z^*)). Si la condition d'arrêt $\sum_{1}^{N} x_i^*(1 - x_i^*) \leq \epsilon$ est satisfaite, nous terminons cet algorithme et obtenons le résultat en arrondissant les variables x_i^*. Sinon, y^* est l'arrondissement de x^*. En augmentant la valeur de paramètre de pénalité t, nous relançons DCA à partir du point y^* (resp. (y^*, z^*)).

Algorithme 3.4 *Algorithme* ***DCA-4***

Initialisation :

- *Choisir* $y^{*0} \in \mathbb{R}^N$ *(resp.* $(y^{*0}, z^{*0}) \in \mathbb{R}^N \times \mathbb{R}^M$*).*
- *Choisir une valeur initiale du parmètre de pénalité* $t = t_{min}$*.*
- *Choisir les paramètres* $\alpha > 0$*,* $\gamma > 0$ *et* t_{max}*.*
- *Choisir une tolérance* $\epsilon > 0$*. et* $k = 0$*.*

Répeter

- *Déterminer la solution* x^{*k+1} *(resp.* (x^{*k+1}, z^{*k+1})*) du problème (3.26) (resp. (3.33)) par l'algorithme 3.2 (resp. l'algprithme 3.3) à partir du point initial* y^{*k} *(resp.* (y^{*k}, z^{*k})*).*
- *Calculer* y^{*k+1} *à partir de* x^{*k+1}*.*
- $k + 1 \leftarrow k$*.*
- $t + \gamma \leftarrow t$*.*

Jusqu'à

$$t \geq t_{max} \quad ou \quad \sum_{i=1}^{N} x_i^{*k}(1 - x_i^{*k}) \leq \epsilon.$$

3.5 Expériences numériques

Nous avons codé les algorithmes en langage C avec la double précision et testé sur l'ordinateur DELL de 1GHz, 512Mb RAM. A chaque itération de DCA, nous avons utilisé le

logiciel CPLEX 9.1 pour trouver la solution des sous-problèmes de programmation linéaire ou quadratique convexe. Pour vérifier l'efficacité des algorithmes proposés, nous utilisons les images ayant des caractéristiques spéciales. Par exemple, la première image est elliptique mais la deuxième est elliptique et rectangulaire. Nous avons traité ici les images de taille moyenne. Le paramètre de pénalisation α de la relation entre le pixel et ses voisinages a été choisi dans l'intervalle $[0.1, 0.5]$.

L'algorithme de DCA est exécuté, dans le premier temps, avec les paramètres de pénalité différents pour vérifier s'il est efficace. Nous avons comparé ici le résultat de nos algorithmes avec celui de l'algorithme **BIF** dans les deux cas, avec ou sans l'information spatiale.

Nous avons, ensuite, appliqué l'algorithme **DCA-4** sur ces mêmes images. Le paramètre de pénalité t_{min} a été initialisé à 0.125 et t_{max} à 10. A chaque itération, nous avons incrémenté la valeur de t de 0.125. La figure 3.7 affiche le résultats de l'algorithme **DCA-1** avec ou sans recherche du bon point initial, la figure 3.8 et 3.9 illustrent le résultat des algorithmes **DCA-2**, **DCA-3** tandis que la figure 3.10 et 3.11 illustrent le résultat de l'algorithme **DCA-4**.

Commentaires. Ces résultats numériques montrent que :
- Les algorithmes de DCA donnent l'image de qualité supérieure par rapport à celle obtenue en appliquant l'algorithme **BIF** dans les deux cas, avec ou sans l'information spatiale.
- Les algorithmes de DCA dépendent fortement du choix du paramètre de pénalité. La figure 3.8 et 3.9 montrent les résultats différents selon les changements de ce paramètre. Par exemple, sur la figure 3.8, l'image reconstruite est identique à l'originale avec $t = 0.25$ et $\alpha = 0.125$.
- La suite de images obtenues en appliquant l'algorithme **DCA-1** présentée sur la figure 3.7 montre qu'à partir du bon point initial obtenu comme solution du problème (3.23), algorithme **DCA-1** nous donne le meilleur résultat par rapport à un choix aléatoire du point initial. Malheureusement, le temps nécessaire pour obtenir cette solution n'est pas négligeable. Par exemple, avec l'image 64×64, il est dans l'intervalle de 397 secondes.
- La suite de images obtenues en appliquant l'algorithme **DCA-4** présentée sur la figure 3.10 et 3.11 montre qu'après certaines itérations, nous pouvons obtenir l'image identique à l'originale en changeant le paramètre de pénalité et le point initial. Dans 80% des cas, nous obtenons le résultat après environ 3-7 lancements de DCA. Cependant, ce nombre de lancements peut changer si nous modifions le paramètre de pénalisation α.
- A partir du tableau de résultat 3.1, nous constatons que : le temps pour les calculs initiaux (les paramètres, les matrices, ...) de l'algorithme **DCA-2** est plus court que l'algorithme **DCA-3**, tandis que celui de l'algorithme **DCA-1** est multiplié approximativement par 60 fois par rapport au temps nécessaire pour l'algorithme **DCA-2**. Clairement, dans l'initialisation, il faut créer une matrice pleine de grande taille avec l'algorithme **DCA-1** tandis que l'algorithme **DCA-2** a besoin d'une matrice creuse de même taille.
- Le tableau 3.2 montre que le temps moyen de chaque itération de DCA de l'algorithme **DCA-1** est le plus court. Ici, à chaque itération, l'algorithme **DCA-1** utilise une projection

explicite tandis que l'algorithme **DCA-2** doit résoudre un problème quadratique convexe et l'algorithme **DCA-3** doit résoudre un problème linéaire. Cependant, le nombre d'itérations de chaque algorithme est différent et la taille du problème linéaire est 4 fois plus grand de la taille du problème quadratique convexe.

- Dans le tableau 3.3, le temps moyen pour obtenir le résultat dépend fortement de l'image en entrée, et pourtant, l'algorithme **DCA-3** demande le plus de temps pour donner le résultat. Le nombre moyen d'itérations de l'algorithme **DCA-1** est plus grand que celui de l'algorithme **DCA-2**.

Conclusion. Dans ce chapitre, nous avons présenté les trois versions de DCA pour trois modèles différents du problème de la reconstruction d'une image binaire. Les résultats montrent que les algorithmes DCA apportent l'image reconstruite très semblable à l'image originale. Cependant, nous constatons que le temps de calcul consacré à l'exécution de l'algorithme DCA est relativement important. Cet inconvénient est dû au fait de résolution des problèmes linéaires ou quadratiques de grande taille dans chaque itération de DCA.

TAB. 3.1 – Le temps pour les calculs initiaux de chaque algorithme.

	Temps(s)		
	Image 64×64	Image 128×128	Image 256×256
DCA-1	63.75	——	——
DCA-2	0.093	0.344	1.406
DCA-3	0.141	0.546	2.281

TAB. 3.2 – Le temps moyen de chaque itération de DCA

	Temps(s)		
	Image 64×64	Image 128×128	Image 256×256
DCA-1	0.60	—-	———
DCA-2	5.12	43.72	404.14
DCA-3	7.38	23.37	456.37

TAB. 3.3 – Le temps et le nombre d'itérations de DCA de chaque algorithme

	Image 64×64		**Image** 128×128		**Image** 256×256	
	N°Iter	Temps	N°Iter	Temps	N°Iter	Temps
DCA-1	115	69.08	–	———	–	———-
DCA-2	9	46.08	8	425.38	8	3896.47
DCA-3	11	85.75	12	249.01	11	4587.38

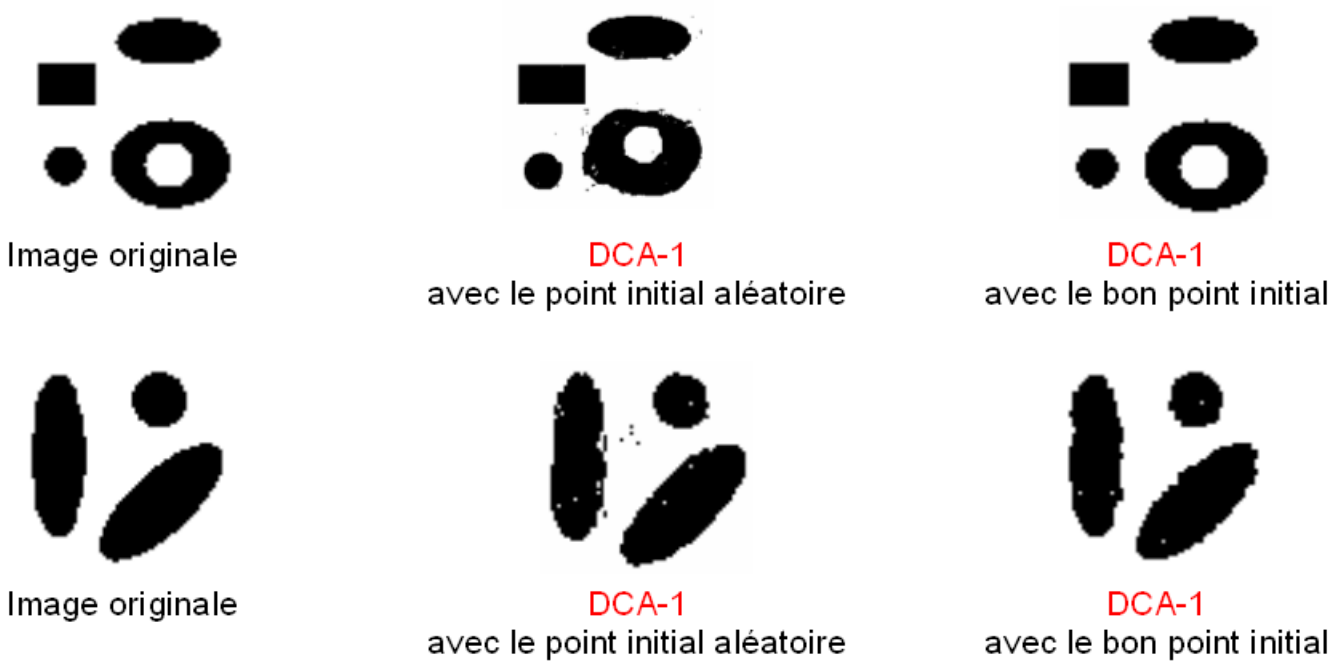

FIG. 3.7 – Les résultats de l'algorithme **DCA-1** avec ou sans le bon point initial.

FIG. 3.8 – Les résultats avec les choix des paramètres.

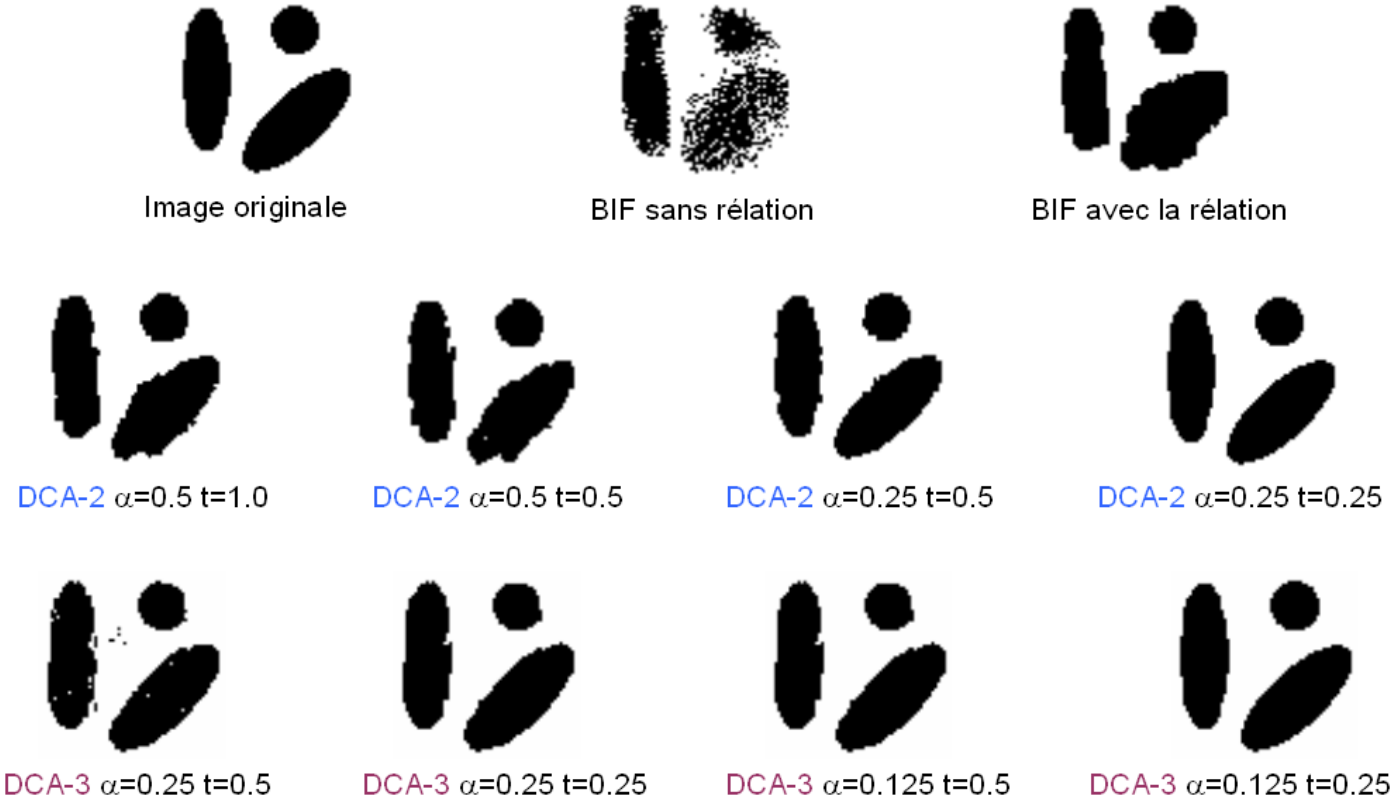

FIG. 3.9 – Les résultats avec les choix des paramètres.

FIG. 3.10 – Le résultat de l'algorithme **DCA-4** avec le paramètre t changé

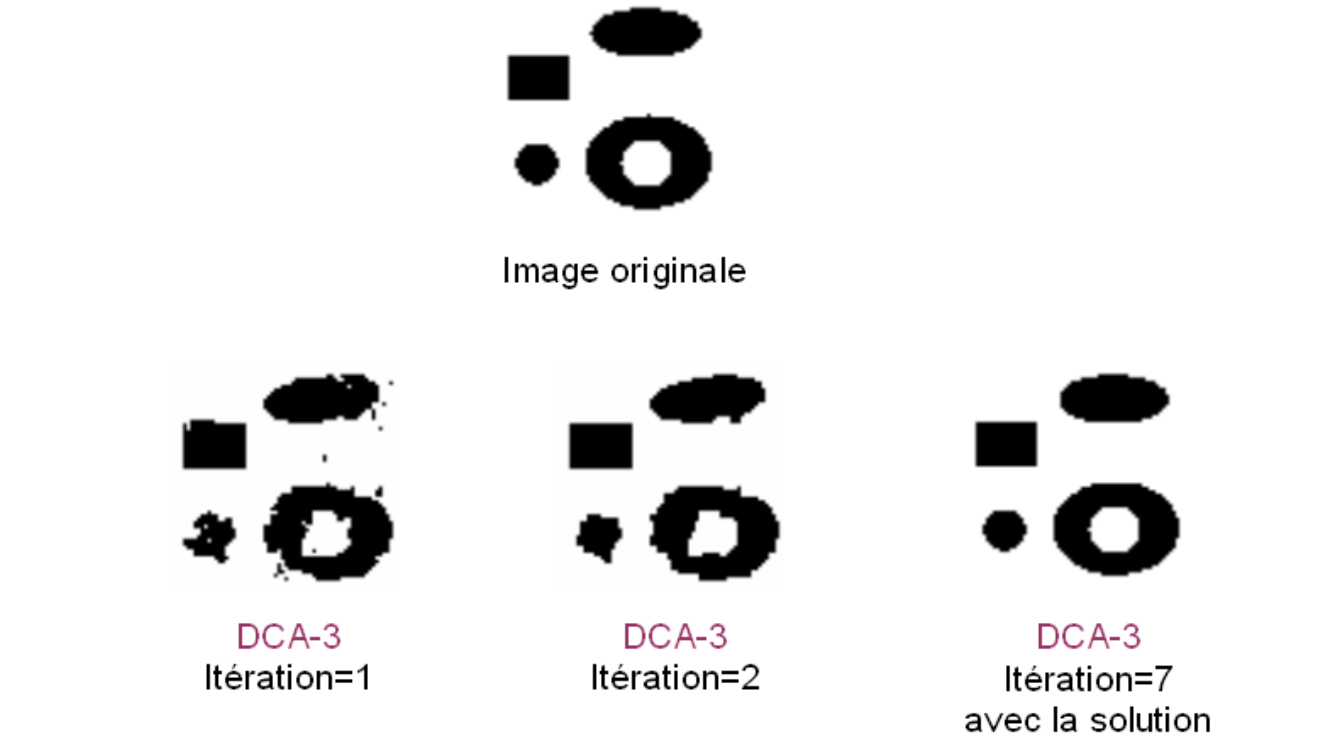

FIG. 3.11 – Le résultat de l'algorithme **DCA-4** avec le paramètre t changé

Chapitre 4

Segmentation d'image IRM par la classification floue via DCA

Résumé *Dans ce chapitre, nous proposons une nouvelle méthode de segmentation d'image via la classification floue, une des approches les plus utilisées en classification non supervisée de données, en utilisant un nouvel et robust algorithme basé sur la programmation DC et DCA. La programmation DC et DCA constituent un cadre théorique et algorithmique enrichissant pour la fouille de données ([18]-[20] et [49]-[62]), en particulier pour la classification floue. C'est pourquoi nous avons choisi volontairement la classification floue parmi les approches existantes pour la segmentation d'image. En exploitant un schéma de DCA extrêmement simple pour la classification floue ([20]) nous proposons un nouveau modèle par l'introduction des informations spatiales au modèle FCM standard. DCA appliqué au nouveau modèle est de même forme que DCA pour le modèle FCM standard, et avec les informations spatiales notre algorithme améliore nettement la segmentation des images brutées. Pour trouver un bon point initial de l'algorithme, nous proposons une procédure alternative de DCA et FCM. Les simulations numériques sur plusieurs imageries médicales, qui sont très haute résolution issue d'un signal, montrent l'efficacité de notre approche par rapport aux méthodes standards sur le temps de calcul et la qualité des solutions.*

4.1 Introduction

La segmentation d'image joue un rôle important dans une variété d'applications telle que la médecine, la géologie, la biométrie, la bureautique, etc... Le premier champs d'application est le traitement de l'imagerie médicale qui est fondé sur des images de l'intérieur du corps d'un patient (échographies, radiographies,...). En effet, on peut repérer sur ces images la présence d'anomalies ce qui permet de détecter certaines maladies. Par exemple, la détection de micro-calcifications dans une mammographie peut rélever la présence d'un cancer du sein. La géologie applique des cartes dont les couleurs peuvent représenter la densité de population, le climat. En plus, le champs d'applications de la segmentation d'image peut s'élargir dans bien d'autres domaines tels que : le suivi de forme dans des documents vidéo, la détection d'objets satellaires dans des images astronomiques, la détection de fronts dans les images

satellites pour l'assimilation de données en météologie,...

La segmentation d'image est une étape primaire dans la plupart des applications de la vision d'ordinateur à l'analyse d'image. C'est une des tâches les plus importantes de la phase de pré-traitement. L'identification d'objets réels, de pseudo-objets et d'ombres, ou la recherche de tout élément d'intérêt présent dans l'image, tous nécessitent une forme de segmentation. La segmentation se définit comme un processus qui consiste à découper une image en régions connexes présentant une homogénéité selon un certain critère, comme par exemple les critères de texture et/ou de couleur. Ce processus est connu par sa complexité en raison de la subjectivité de la définition de régions connexes, et de l'objectif qu'on veut atteindre de notre image segmentée. Par exemple, dans un mur de brique, doit-on considérer que chacune des briques forme une région autonome ?

Dans ce domaine, de nombreuses méthodes basées sur différentes approches telles que contour, région, texture,..., ont été développées au cours de ces dernières années ([141]). On peut confirmer qu'aucune méthode ne semble prévaloir sur les autres, chacune ayant son domaine de prédilection. En absence de méthode universelle, il est classique de retrouver les différentes approches classifiées en quatre thèmes : clustering, approches contours, approches régions et méthodes hybrides. Rajapakse([143]) a classifié les différentes méthodes en tant que quatre catégories :

- Les méthodes classiques telles que le seuillage, la croissance de région, la segmentation basée sur les contours. La première méthode consiste à déterminer le seuil à appliquer à l'image : le seuillage permet de sélectionner les parties de l'image qui intéressent l'opérateur. La deuxième méthode propose à faire croître chaque région autour d'un pixel de départ dont l'agglomération n'exploite aucune connaissance a priori de l'image ou du bruit et la décision d'intégrer à la régoin un pixel voisin repose seulement sur un critère d'homogénéité imposé à la zone en croissance. Et la troisième méthode s'intéresse aux contours des objets extraits de l'image. La plupart du temps, ces contours sont morcelés et peu précis, il faut donc utiliser des techniques de reconstruction de contours par interpolation ou connaître a priori la forme de l'objet recherché.
- Les méthodes statistiques telles que la segmentation bayesienne ou la segmentation au maximum de vraisemblance sont basées sur les développements des chaînes de Markov. Dans [136], on peut trouver un algorithme de segmentation non supervisé "Multiple Resolution Segmentation" (MRS), formulé dans le contexte bayesien. La méthode utilise un modèle AR-2D causal (Gaussian Autoregressive) : l'image observée est considérée comme le mélange de champs aléatoires statistiquement homogénes. Le champ des classes est modélisé par un champ aléatoire markovien. La segmentation optimale est définie au sens du Maximum a Posteriori (MAP) et estimée grâce à un algorithme de minimisation locale (Iterated Conditional Mode).
- Les méthodes de réseaux de neurones : Une des stratégies possibles pour la segmentation est celle de la classification de pixels. Quelques exemples de segmentation d'images colorées par réseaux de neurones ont été publiés récemment. Dans [137], les réseaux utilisés étaient

des réseaux de Hopfield, configurés à l'aide de l'histogramme des couleurs. Dans [140], les auteurs utilisent les réseaux auto-organisés du type Kohonen pour la segmentation.
– Les méthodes de classification floue : Le problème de classification automatique (clustering) est considéré comme une des problématiques majeures en extraction des connaissances à partir de données. Parmi les techniques de classification, la classification floue (fuzzy) via le modèle Fuzzy C-Means (FCM) est plus étudiée. FCM a été introduite par Jim Bezdek en 1981 ([134]) comme une amélioration des méthodes de clustering précédentes ([138]), et a été beaucoup développée dans les années 90. Cette approche a été appliquée avec succès dans plusieurs problèmes dont la segmentation d'image ([135, 142, 133, 144, 139]). FCM consiste à déterminer simultanément, via un modèle d'optimisation, les centres des classes et le degré d'appartenance des points aux classes. Dans la segmentation d'image, elle permet d'obtenir une partition floue de l'image en donnant à chaque pixel de l'image un degré d'appartenance à une région donnée.

Un inconvénient du modèle standard de FCM dans la segmentation d'image est de ne pas tenir compte de l'information spatiale qui une relation entre le pixel et ses voisinages. Pourtant, cette information rend l'algorithme très sensible au bruit et à d'autres objets façonnés dans l'image. En fait, cette relation est une des caractéristiques importantes d'une image car les voisinages possèdent souvent les valeurs semblables, et la probabilité qu'ils appartiennent à la même partition est très élevée. Par ailleurs, si nous considérons une image bruitée, le FCM n'est pas une méthode pour surmonter ce problème. Récemment, de nombreux chercheurs ont ajouté l'information spatiale à l'algorithme original de FCM pour améliorer l'efficacité de la segmentation d'image([142, 144, 139]).

Le but de notre travail est double. Le modèle de la classification dans la segmentation d'image est, en général, un problème de très grande dimension pour lequel, la recherche des méthodes efficaces est toujours nécessaire. Premièrement, nous proposons une nouvelle méthode de segmentation d'image via le modèle de FCM en utilisant un nouveau et robust algorithme basé sur la programmation DC et DCA. Deuxièmement, pour la segmentation d'image bruitée, nous considérons un modèle adaptatif de FCM (appelé FCM-Spatial) qui incorpore l'information spatiale à la fonction de clustering.

La programmation DC et DCA ont été appliqués avec succès à nombreux problèmes d'optimisation non convexe différentiable ou non de grande dimension dans différents domaines des sciences appliquées, en particulier aux problèmes de la fouille de données (voir par exemple [18]-[20] et [49]-[62]). Récemment un schéma de DCA extrêmement simple et robust a été introduit dans [20] pour la classification floue grâce à une formulation DC intéressante obtenue par des techniques de reformulations rigoureuses. Nous utilisons ce schéma pour la résolution du nouveau modèle FCM-Spatial qui est de même forme que FCM standard mais le nombre de variables a doublé. Pour calculer le bon point initial et accélérer la convergence de DCA, nous proposons une procédure alternative de FCM-DCA qui combine le DCA avec l'algorithme classique de FCM. Les résultats expérimentaux sur plusieurs images bruitées

ont illustré l'efficacité de l'algorithme proposé et sa supériorité par rapport à l'algorithme standard de FCM sur le temps de calculs et la qualité des solutions. D'ailleurs, avec le modèle spatial, notre algorithme réduit considérablement l'effet du bruit.

Ce chapitre est organisé de la façon suivante. Dans la deuxième section, nous présentons la formulation du modèle FCM et le modèle adaptatif de FCM-Spatial pour la segmentation d'image bruitée. La résolution de ce problème par la programmation DC et DCA est étudiée dans la troisième section, tandis que l'accélération du DCA par une procédure alternative de FCM-DCA est présentée dans la quatrième section. Finalement, les résultats numériques de nos algorithmes sont rapportés dans la dernière section.

4.2 Classification floue

4.2.1 Modèle de FCM

Soit $X = \{x_1, x_2..., x_n\}$ l'ensemble des points x_k dans l'espace $\mathbb{R}^d$. Dans la segmentation d'image IRM, on considère chaque pixel comme un point, i.e. $x_i \in \mathbb{R}$ où x_i est la valeur de l'intensité du pixel. Dans l'image couleur, un pixel peut être représenté sur trois valeurs pour chacune des couleurs : rouge, vert et bleu, i.e $x_i \in \mathbb{R}^3$.

Soit c le nombre de régions d'image. Les valeurs des degrés d'appartenance sont représentées dans une matrice $U = [u_{ik}]$ pour $1 \leq i \leq c, 1 \leq k \leq n$ où u_{ik} désigne le degré d'appartenance du point x_k à la région (classe) C_i dont le centre est v_i. Il est clair que

$$u_{i,k} \in [0,1] \text{ pour } i = 1, ..., c \text{ et } k = 1, ..., n \ ;$$

$$\sum_{i=1}^{c} u_{i,k} = 1, \text{ pour } k = 1, ..., n.$$

Si la matrice de partition U est déterminée, on en déduit la classification selon la règle suivante : le point x_k (pour $k = 1, \ldots, n$) est classé dans la classe C_i (pour $i = 1, \ldots, c$) si et seulement si

$$u_{i,k} = \max\{u_{j,k} : j \in \{1, \ldots, c\}\}.$$

Considérons la fonction J_m définie par :

$$J_m(U, V) = \sum_{k=1}^{n} \sum_{i=1}^{c} u_{i,k}^m ||x_k - v_i||^2,$$

où $\|.\|$ désigne la norme Euclidienne de l'espace correspondant, V est une $(c \times p)$ - matrice dont chaque ligne v_i correspond au centre de la classe C_i, et $m \geq 1$ un paramètre entier

qui définit le degré de flou du modèle. Chercher une classification revient ainsi chercher la matrice de partition U et les centres v_i. Le modèle mathématique de FCM s'écrit ainsi :

$$\begin{cases} \min J_m(U,V) = \sum_{k=1}^{n}\sum_{i=1}^{c} u_{i,k}^m ||x_k - v_i||^2 \\ \text{tel que} \quad u_{i,k} \in [0,1] \quad 1 \le i \le c, 1 \le k \le n \\ \qquad\qquad \sum_{i=1}^{c} u_{i,k} = 1 \quad \forall k \quad et \quad \sum_{k=1}^{n} u_{i,k} > 0 \quad \forall i. \end{cases} \tag{4.1}$$

Algorithme 4.1 *Schéma de FCM ([134])*

Initialisation :
- *Choisir le nombre de classes c.*
- *Initialise la matrice de partition U ansi que les centres v_i aléatoirement.*

Répeter
- *Calculer les centres V suivant l'équation :*

$$v_i = \sum_{k=1}^{n} u_{ik}^m x_k / \sum_{k=1}^{n} u_{ik}^m, \quad \forall i = 1, 2, ..., c. \tag{4.2}$$

- *Calculer la matrice de partition U :*

$$u_{ik} = \left[\sum_{k=1}^{c} \frac{\|x_k - v_i\|^{2/(m-1)}}{\|x_k - v_j\|^{2/(m-1)}} \right]^{-1}, \forall i = 1, 2, ..., c \quad et \quad \forall k = 1, 2, ..., n. \tag{4.3}$$

Jusqu'à *[les centres ont stabilisé (i.e. : le changement de U est plus petit qu'une valeur ϵ)].*

4.2.2 Modèle de FCM avec l'information spatiale

Dans la segmentation d'image par le modèle standard de FCM, chaque pixel $x_k \in \mathbb{R}^d$ représente les données multispectrales. Cependant, comme mentionné précédemment, une des caractéristiques importantes d'une image est que les voisinages de pixel possèdent les valeurs semblables, le rapport spatial est donc intéressant pour la segmentation d'image. L'information spatiale est la relation entre le pixel et ses voisinages. Il y a différentes manières d'incorporer l'information spatiale (voir la figure 4.1).

Dans notre cadre, nous considérons l'information spatiale de x_k comme une valeur moyenne de ses voisinages 3×3, et chaque point x_k dans (4.1) a deux groupes de valeurs : les valeurs du pixel et les valeurs moyennes de ses voisinages 3×3.

Soit N_k les voisinages 3×3 du pixel x_k. Les données entrées x_k dans notre modèle spatial de FCM sont $x_k = (x_{k1}, x_{k2})$ où x_{k1} représente les valeurs du pixel de k^{th} de l'image et

$x_{k2} = (x_{k1} + \sum_{i \in N_k} x_{i1})/9$. D'où, le nombre de variables U n'est pas changé et V devient une matrice de $c \times 2d$ dont la i^{eme} ligne est, $v_i \in \mathbb{R}^{2d}$, le centre de la région C_i.

Le modèle spatial de FCM dans notre approche n'est pas tellement différent par rapport à son modèle standard FCM (4.1) sauf le fait que chaque $x_k \in \mathbb{R}^d$ est remplacé par $x_k = (x_{k1}, x_{k2}) \in \mathbb{R}^{2d}$. Par conséquent, n'importe quel algorithme pour le modèle standard de FCM (4.1) peut être appliqué au modèle spatial de FCM. Au point de vue numérique, le problème spatial de FCM est plus difficile car le nombre de variables V est doublé.

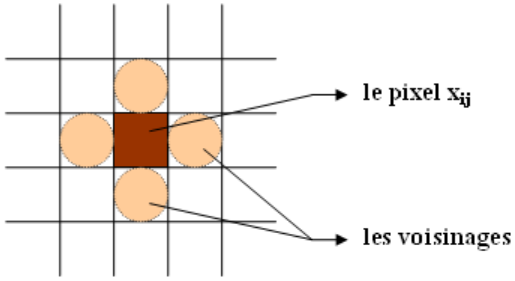

FIG. 4.1 – Le pixel et ses 4 voisinages

4.3 La programmation DC et DCA pour la résolution de FCM

Dans cette section nous présentons la formulation DC et le schéma DCA correspondant pour le modèle FCM développés dans [20].

4.3.1 La nouvelle formulation du modèle de FCM

Dans la modèle (4.1), seulement la variable U est a priori bornée. En fait on peut aussi restreindre la variable V à un domaine borné. En effet, la condition nécessaire d'optimalité du premier ordre en (U, V) implique

$$\nabla_V J_m(U, V) = 0,$$

i.e.,

$$\partial_{v_i} J_m(U, V) = \sum_{k=1}^{n} u_{i,k}^m 2(v_i - x_k), \quad \forall i = 1, ..., c, \ k = 1, ..., n$$

ou

$$v_i \sum_{k=1}^{n} u_{i,k}^m = \sum_{k=1}^{n} u_{i,k}^m x_k.$$

D'autre part, la non-vacuité des classes assure que $\sum_{k=1}^{n} u_{i,k}^m > 0$, $\forall i = 1, ..., c$. Par suite

$$\|v_i\|^2 \leq \frac{\left(\sum_{k=1}^{n} u_{i,k}^m \max_{k=1,...,n} \|x_k\|\right)^2}{(\sum_{k=1}^{n} u_{i,k}^m)^2} = \max^2{}_{k=1,...,n}\|x_k\| = r^2.$$

Considérons les nouvelles variables $t_{i,k}$ telles que $u_{i,k} = t_{i,k}^2$. La contrainte $\sum_{i=1}^{c} u_{i,k} = 1$ devient $\sum_{i=1}^{c} t_{i,k}^2 = 1$ ou $\|t_k\|^2 = 1$ avec $t_k \in \mathbb{R}^c$. Soient S_k la sphère de rayon 1 dans $\mathbb{R}^c$ et R_i la boule Euclidienne de rayon r dans $\mathbb{R}^d$, on peut reformuler le problème FCM comme :

$$\begin{cases} \min J_{2m}(T, V) = \sum_{k=1}^{n} \sum_{i=1}^{c} t_{i,k}^{2m} ||x_k - v_i||^2 \\ \text{tel que} \quad T \in \mathcal{S} = \Pi_{k=1}^{n}\ S_k,\ V \in \mathcal{C} = \Pi_{i=1}^{c} R_i \end{cases} . \tag{4.4}$$

Ce dernier est un problème d'optimisation non convexe dont la résolution sera décrite dans la suite.

Remarque 4.1 *Le changement de variables en $t_{i,k}$ nous amène à travailler sur $\mathcal{S}$, le domaine réalisable des variables $t_{i,k}$ qui est le produit des sphères et non le produit des simplexes comme dans le cas des variables initiales $u_{i,k}$. On pourrait penser alors que la non convexité de $\mathcal{S}$ rend le problème plus difficile qu'avec sa formulation initiale. Mais ce n'est pas le cas, comme on verra dans la suite, car le problème (4.4) sera reformulé sous une forme équivalente où $\mathcal{S}$ est remplacé par le produit des boules de rayon 1. Ce qui est intéressant, tant sur le plan algorithmique que numérique : la nouvelle formulation DC de (4.4) donne naissance à un schéma DCA extrêmement simple qui ne nécessite que des calculs explicites et donc non coûteux. Puisqu'il s'agit des calculs des projections d'un point sur une boule Euclidienne à chaque itération.*

4.3.2 Formulation DC de (4.9)

La fonction objectif de (4.4) peut s'écrire de la manière suivante :

$$J_{2m}(T, V) = \frac{\rho}{2}\|T\|^2 + \frac{\rho}{2}\|V\|^2 - \left[\frac{\rho}{2}\|(T, V)\|^2 - J_{2m}(T, V)\right].$$

Pour tout $(T, V) \in \mathcal{S} \times \mathcal{C}$ on a

$$J_{2m}(T,V) = \frac{\rho}{2}n + \frac{\rho}{2}\|V\|^2 - H(T,V).$$

avec

$$H(T,V) \quad = \tfrac{\rho}{2}\|(T,V)\|^2 - J_{2m}(T,V). \tag{4.5}$$

Dans le lemme suivant nous donnerons les conditions pour que la fonction H soit convexe.

Lemme : soit $\mathcal{B} = \Pi_{k=1}^{n} B_k$, où B_k est la boule de centre 0 et de rayon 1 dans $\mathbb{R}^c$. La fonction $H(U,V)$ est convexe sur $\mathcal{B} \times \mathcal{C}$ pour toute valeur de ρ telle que

$$\rho \geq \frac{m}{n}(2m-1)\alpha^2 + 1 + \sqrt{\left[\frac{m}{n}(2m-1)\alpha^2 + 1\right]^2 + \frac{16}{n}m^2\alpha^2},$$

où

$$\alpha = r + \max_{1\leq k\leq n} \|x_k\| .$$

Preuve : on remarque tout d'abord que $\rho > 0$ car $m \geq 1$.

Puisque

$$H(T,V) = \sum_{k=1}^{n}\sum_{i=1}^{c}\left[\frac{\rho}{2}t_{i,k}^2 + \frac{\rho}{2}\|v_i\|^2 - t_{i,k}^{2m}\|x_k - v_i\|^2\right],$$

H est convexe si toutes les fonctions

$$h_{i,k}(t_{i,k}, v_i) = \frac{\rho}{2}t_{i,k}^2 + \frac{\rho}{2}\|v_i\|^2 - t_{i,k}^{2m}\|x_k - v_i\|^2, \quad \forall i = 1,...c, k = 1,...n$$

sont convexes.

Considérons la fonction suivante :

$$f \quad : \quad \mathbb{R}\times\mathbb{R} \to \mathbb{R}$$

$$f(x,y) = \frac{\rho}{2}x^2 + \frac{\rho}{2}y^2 - x^{2m}y^2. \tag{4.6}$$

Le Hessien de la fonction f est donné par :

$$J(x,y) = \begin{pmatrix} \rho - 2m(2m-1)y^2x^{2m-2} & -4mx^{2m-1}y \\ -4mx^{2m-1}y & \frac{\rho}{n} - 2x^{2m} \end{pmatrix}. \tag{4.7}$$

Pour tout $(x,y) : 0 \leq x \leq 1; \|y\| \leq \alpha$, on a :

$$\begin{aligned} \mid J(x,y) \mid \; &= (\rho - 2m(2m-1)y^2x^{2m-2})\,(\tfrac{\rho}{n} - 2x^{2m}) - 16m^2x^{4m-2}y^2 \\ &\geq \tfrac{\rho}{n}^2 - \left[2\tfrac{m}{n}(2m-1)y^2x^{2m-2} + 2x^{2m}\right]\rho - 16m^2x^{4m-2}y^2 \\ &\geq \tfrac{1}{n}\rho^2 - 2\left(\tfrac{m}{n}(2m-1)\alpha^2 + 1\right)\rho - 16m^2\alpha^2. \end{aligned}$$

Par suite, si

$$\rho \geq \frac{m}{n}(2m-1)\alpha^2 + 1 + \sqrt{\left[\frac{m}{n}(2m-1)\alpha^2 + 1\right]^2 + \frac{16}{n}m^2\alpha^2}, \tag{4.8}$$

alors $\mid J(x,y) \mid \geq 0$, pour tout $(x,y) \in \mathbb{R}^2$ tels que $0 \leq x \leq 1, \mid y \mid \leq \alpha$.

Ainsi avec ρ défini par (4.8), la fonction f est convexe sur $[0,1] \times [-\alpha, \alpha]$. Par conséquent les fonctions

$$\theta_{i,k}(t_{i,k}, v_i) = \frac{\rho}{2}\ t_{i,k}^2 + \frac{\rho}{2}\|x_k - v_i\|^2\ - t_{i,k}^{2m}\|x_k - v_i\|^2$$

sont convexes sur $\{0 \leq t_{i,k} \leq 1, \|v_i\| \leq r\}$ avec ρ donné dans (4.8) et $\alpha = r + \max_{1\leq k\leq n} \|x_k\|$.

Il en est de même pour les fonction $h_{i,k}$, car

$$h_{i,k}(t_{i,k}, v_i) = \theta_{i,k}(t_{i,k}, v_i) + \rho\langle x_k, v_i\rangle - \frac{\rho}{2}\|x_k\|^2.$$

Ainsi, avec les valeurs données ci-dessus de ρ et α, la fonction $H(T,V)$ est convexe sur $\mathcal{B}\times\mathcal{C}$. Dans toute la suite nous travaillons avec ces valeurs de ρ et α.

Il est clair que, pour tout $T \in \mathcal{B}$ et un $V \in \mathcal{C}$ fixé, la fonction $J_{2m}(T,V)$ est concave en variable T (car $H(T,V)$ est convexe), par suite son minimum sur $\mathcal{B}$ est atteint sur la frontière $\mathcal{S}$ de $\mathcal{B}$,i.e.,

$$\begin{aligned}&\min\left\{\frac{\rho}{2}\|V\|^2 - H(T,V) : (T,V) \in \mathcal{B}\times\mathcal{C}\right\}\\ &= \min\left\{\frac{\rho}{2}\|V\|^2 - H(T,V) : (T,V) \in \mathcal{S}\times\mathcal{C}\right\}.\end{aligned}$$

Le problème (4.4) peut être alors reformulé comme

$$\min\left\{\frac{\rho}{2}\|V\|^2 - H(T,V) : (T,V) \in \mathcal{B}\times\mathcal{C}\right\},$$

ou encore

$$\min\left\{\ \chi_{\mathcal{B}\times\mathcal{C}}(T,V) + \frac{\rho}{2}\|V\|^2 - H(T,V) : (T,V) \in \mathbb{R}^{c\times n}\times\mathbb{R}^{c\times d}\right\} \tag{4.9}$$

qui est un programme DC avec la décomposition DC suivante :

$$\chi_{\mathcal{B}\times\mathcal{C}}(T,V) + \frac{\rho}{2}\|V\|^2 - H(T,V) = G(T,V) - H(T,V),$$

où

$$G(T,V)\ \ = \chi_{\mathcal{B}\times\mathcal{C}}(T,V) + \tfrac{\rho}{2}\|V\|^2 \tag{4.10}$$

est bien évidemment une fonction convexe grâce à la convexité de $\mathcal{B}$ et $\mathcal{C}$.

4.3.3 Résolution de (4.9) par DCA

Selon la description de DCA dans la chapitre 1, la résolution de FCM via la formulation (4.9) par DCA consiste en la détermination de deux suites

$$(Y^l, Z^l) \in \partial H\ (T^l, V^l) \quad et$$
$$(T^{l+1}, V^{l+1}) \in \partial G^*(Y^l, Z^l).$$

La fonction H est différentiable et son gradient au point (T^l, V^l) est calculé de la manière suivante :

$$\nabla H(T^l, V^l) \quad = \rho(T^l, V^l) - (2mt_{i,k}^{2m-1}\|x_k - v_i\|^2, 2\sum_{k=1}^{n}(v_i - x_k)t_{i,k}^{2m}). \tag{4.11}$$

Le calcul de $(T^{l+1}, V^{l+1}) \in \partial G^*(Y^l, Z^l)$ se ramène à la résolution du problème suivant :

$$\min\left\{\frac{\rho}{2}\|V\|^2 - \langle (T,V), (Y^l, Z^l)\rangle : (T,V) \in \mathcal{B} \times \mathcal{C}\right\}.$$

Il s'en suit que (Proj étant l'application de projection)

$$T^{l+1} = \mathrm{Pr\,oj}_{\mathcal{B}}(Y^l) \quad et$$
$$V^{l+1} = \mathrm{Pr\,oj}_{\mathcal{C}}(\frac{1}{\rho}Z^l).$$

Plus précisément :

$$V_{i,.}^{l+1} = \quad \begin{cases} \frac{(Z^l)_{i,.}}{\rho} \text{ si } \|(Z^l)_{i,.}\| \leq \rho r \\ \frac{(Z^l)_{i,.}r}{\|(Z^l)_{i,.}\|} \text{ sinon} \end{cases}, \quad \forall i = 1, ...c, \tag{4.12}$$

et

$$T_{.,k}^{l+1} = \quad \begin{cases} Y_{.,k}^l \text{ si } \|Y_{.,k}^l\| \leq 1 \\ \frac{(Y^l)_{.,k}}{\|(Y^l)_{.,k}\|} \text{ sinon} \end{cases}, \quad \forall k = 1, ...n. \tag{4.13}$$

Algorithme 4.2 *Schéma de DCA*

Initialisation :
- *Choisir* $T^0 \in \mathbb{R}^{c\times n}$ *et* $V^0 \in \mathbb{R}^{c\times p}$. *Soit* $l = 0$.
- *Choisir une tolérance* $\epsilon > 0$.

Répeter
- *Calculer* $(Y^l, Z^l) \in \nabla H(T^l, V^l)$ *à l'aide de (4.11) ;*
- *Calculer* $(T^{l+1}, V^{l+1}) \in \partial G^*(Y^l, Z^l)$ *à l'aide de (4.12) et (4.13) ;*
- $l + 1 \longleftarrow l$

Jusqu'à $\|(T^{l+1}, V^{l+1}) - (U^l, V^l)\| \leq \epsilon(\|(T^{l+1}, V^{l+1})\|)$.

Construction des classes Soient (T^*, V^*) la solution calculée par DCA et $u_{i,k} = t_{i,k}^{*2}$. Le point x_k appartient à la classe C_i si $u_{ik} = \max_{j=1..c} u_{j,k}$.

4.4 Accélération du DCA par une procédure alternative de FCM-DCA

La recherche d'un bon point initial joue un rôle crucial dans la solution d'une programmation D.C par DCA. Elle dépend de la structure du problème considéré et elle peut être effectuée par, par exemple, une méthode heuristique bien adaptée au problème. D'une manière générale, un bon point initial pour le DCA ne doit pas être un minimum local, parce qu'à partir d'un tel point, le DCA est stationnaire. En plus, nous observons qu'à partir de n'importe quel point non minimum local, la fonction objective diminue rapidement pendant quelques premières itérations de DCA. Nous avons la même remarque pour l'algorithme standard de FCM. C'est pourquoi, nous proposons une procédure alternative de FCM-DCA pour le problème (4.9).

4.4.1 Procédure alternative de FCM-DCA

Algorithme 4.3 *Algorithme combiné de FCM-DCA*

Initialisation :

- *Choisir le nombre de classes c.*
- *Initialise la matrice de partition U^o ansi que les centres V^o aléatoirement.*
- *Soit $l = 0$. Choisir une tolérance $\epsilon > 0$.*

Répeter

Une itération de FCM :
- *Calculer les centres V^l à partir de (4.2).*
- *Déterminer la matrice de partition U^l à l'aide de (4.3).*
- $t_{ik} = \sqrt{u_{ik}}$, $\forall i = 1, ..., c$ *and* $\forall k = 1, ..., n$.

Une itération de DCA :
- *Calculer $(Y^l, Z^l) \in \nabla H(T^l, V^l)$ à partir de (4.11) ;*
- *Calculer $(T^{l+1}, V^{l+1}) \in \partial G^*(Y^l, Z^l)$ à l'aide de (4.12) et (4.13) ;*

$l + 1 \longleftarrow l$

Jusqu'à $\|(T^{l+1}, V^{l+1}) - (U^l, V^l)\| \leq \epsilon(\|(T^{l+1}, V^{l+1})\|)$.

4.4.2 La recherche d'un bon point initial de DCA

Si nous utilisons l'algorithme combiné de FCM-DCA jusqu'à sa convergence, l'efficacité du DCA ne peut être bien exploitée. Pour remédier à cette situation, nous proposons un autre algorithme efficace basé sur le DCA. C'est un algorithme qui contient de deux phases. Dans la première phase, nous exécutons quelques itérations de l'algorithme combiné de FCM-DCA pour trouver un bon point initial. Et dans la deuxième, à partir du point trouvé, nous utilisons DCA jusqu'à sa convergence. L'algorithme est résumé comme ci-dessous.

Algorithme 4.4 *Algorithme de deux phases*

Initialisation :

- *Choisir le nombre de classes c.*
- *Initialise la matrice de partition U^o ansi que les centres V^o aléatoirement.*
- *Soit $l = 0$. Choisir une tolérance $\epsilon > 0$.*

La première phase :

- *Exécuter q itérations de l'algorithme combiné de FCM-DCA pour obtenir (T^{q+1}, V^{q+1}).*
- *Mise à jour $(T^o, V^o) \longleftarrow (T^{q+1}, V^{q+1})$.*

La deuxième phase :

- *Appliquer l'algorithme 4.2 à partir de bon point initial (T^o, V^o) jusqu'à sa convergence.*

4.5 Expériences numériques

Pour vérifier l'efficacité de notre méthode proposée, d'abord nous utilisons l'image originale où les contours et les régions sont parfaitement localisées. Ensuite, dans la même image, nous ajoutons le bruit gaussien avec le rapport différent de signal-bruit pour démontrer les efficacités de chaque méthode proposée. Troisièmement, nous comparons le temps de calcul entre l'algorithme accéléré de DCA et l'agorithme standard de FCM. Tous les tests ont été réalisés sur un ordinateur de Pentium[R] 4 CPU 3.00GHz 1.00Go RAM. A la fin, nous observons les résultats de ces méthodes sur l'image **Blume** qui contient plusieurs régions.

Pour comparer la performance de **Algorithme 4.4** selon le temps de calcul, nous appliquons **Algorithme de FCM** et **Algorithme 4.4** sur les mêmes images avec les mêmes paramètres initiaux. Le tableau **4.1** donne le temps de calcul de chaque méthode. Nous utilisons les notations suivantes :

- $Taille$: la taille de l'image.
- c : le nombre de régions d'image.
- $N°F$: le nombre d'itérations dans **Algorithme de FCM**.
- q : le nombre d'itérations de FCM-DCA dans la première phase dans **Algorithme 4.4**.
- $N°D$: le nombre d'itérations de DCA dans la deuxième phase dans **Algorithme 4.4**.
- $Temps$: le temps de calcul de chaque algorithme en second.

Commentaires. A partir des résultats expérimentaux, nous constatons que :

- Dans plusieurs images, notre algorithme donne une segmentation presque parfaite. En plus, sans information spatiale, nos **Algorithme 4.3** et **Algorithme 4.4** peuvent surmonter la segmentation d'image bruitée dans certains cas.
- Avec l'information spatiale, **Algorithme 4.4** fonctionne bien sur toutes les images bruitées.

Il peut supprimer les bruits de manière efficace. Nous pourrons confirmer, dans ce cas, que les deux algorithmes DCA apportent l'image de meilleure qualité par rapport à l'algorithme FCM.

– Dans la plupart de jeux d'essais, **Algorithme 4.3** est plus rapide que **Algorithme de FCM** parce que le calcul de projection des points sur les boules Euclidiennes est explicite. De plus, **Algorithme 4.4** permet d'avoir un gain de calcul important si le choix du paramètre q est bon. Le tableau **4.1** indique l'efficacité du choix du paramètre q en temps de calcul.

Conclusion. Dans ce chapitre, nous proposons une nouvelle méthode de segmentation d'image via le modèle de FCM en utilisant un nouveau et robust algorithme basé sur la programmation DC et DCA. D'une autre manière, le modèle de FCM a été reformulé comme une programmation DC pour lequel un schéma simple et rapide de DCA a été appliqué. La procédure alternative de FCM-DCA est efficace pour trouver un bon point initial de DCA et pour accélérer sa convergence. L'algorithme de deux phases de DCA peut alors être appliqué dans le problème de classification de grande dimension. D'autre part, l'utilisation d'information spatiale pour la segmentation d'image bruitée semble être efficace. Les simulations numériques préliminaires prouvent que les algorithmes proposés sont prometteurs pour la segmentation d'image bruitée.

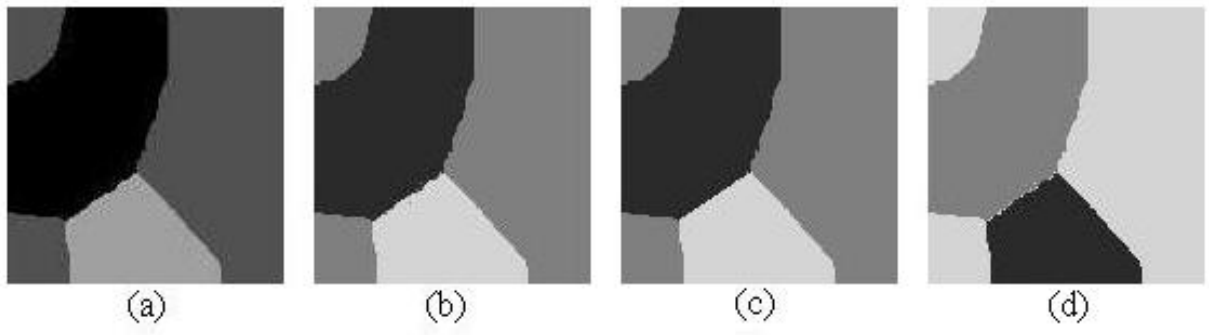

FIG. 4.2 – L'image originale et les résultats de segmentation (c=3).
(a) : L'image originale (b) : Le résultat de **Algorithme de FCM** (c) : Le résultat de **Algorithme 4.3** (d) : Le résultat de **Algorithme 4.4**.

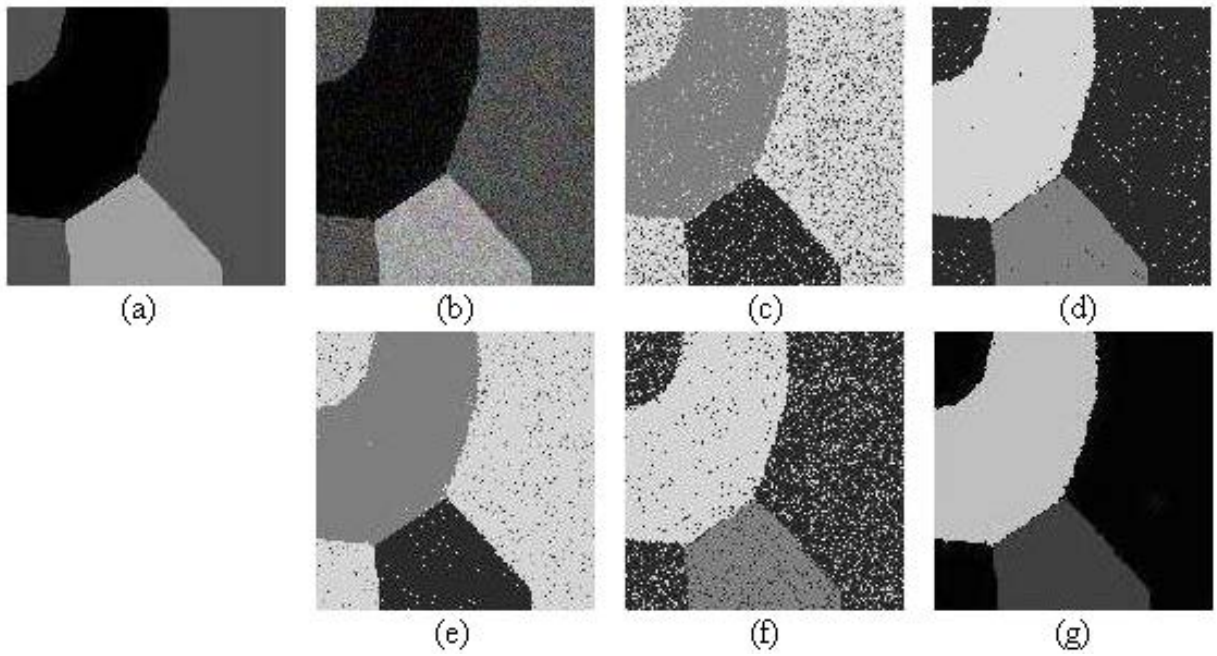

FIG. 4.3 – L'image originale avec le bruit et les résultats de segmentation (c=3).
(a) : L'image originale (b) : L'image originale avec le bruit Gaussien (c) : Le résultat de **Algorithme de FCM** (d) : Le résultat de **Algorithme de FCM** avec l'information spatiale. (e) : Le résultat de **Algorithme 4.3** (f) : Le résultat de **Algorithme 4.4** (g) : Le résultat de **Algorithme 4.4** avec l'information spatiale.

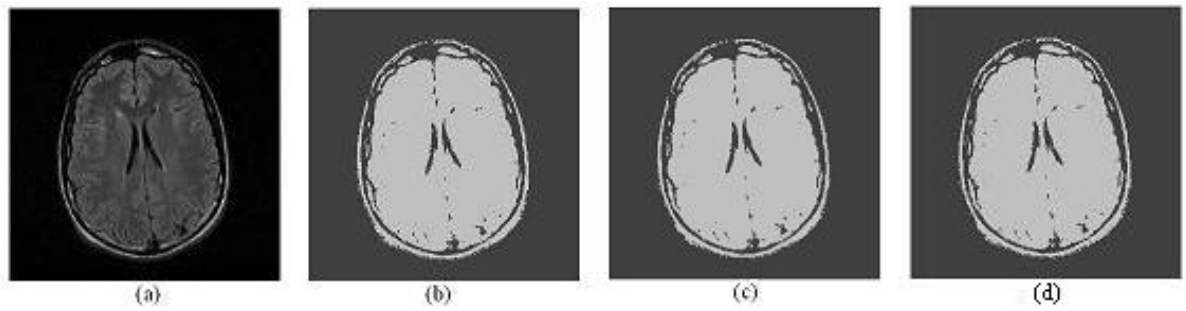

FIG. 4.4 – L'image médical originale et les résultats de segmentation (c=2).
(a) : L'image médicale (b) : Le résultat de **Algorithme de FCM** (c) : Le résultat de **Algorithme 4.3** (d) : Le résultat de **Algorithme 4.4**.

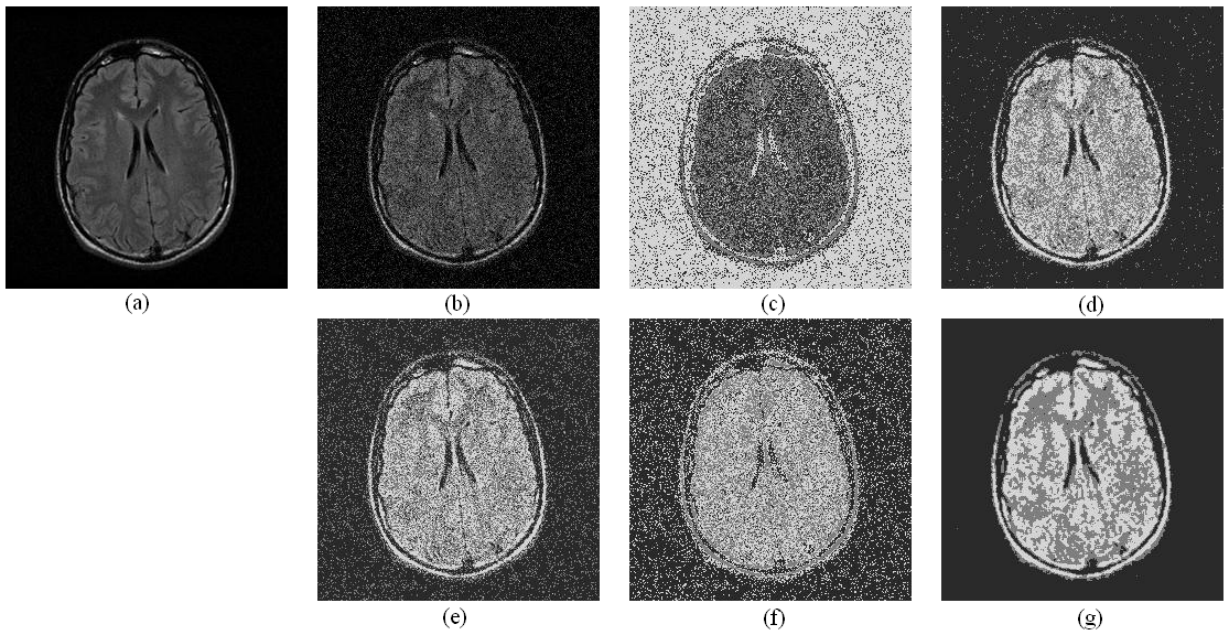

FIG. 4.5 – L'image médicale avec le bruit Gaussien et les résultats de segmentation (c=3).
(a) : L'image médicale originale (b) : L'image médicale avec le bruit Gaussien (c) : Le résultat de **Algorithme de FCM** (d) : Le résultat de **Algorithme de FCM** avec l'information spatiale. (e) : Le résultat de **Algorithme 4.3** (f) : Le résultat de **Algorithme 4.4** (g) : Le résultat de **Algorithme 4.4** avec l'information spatiale.

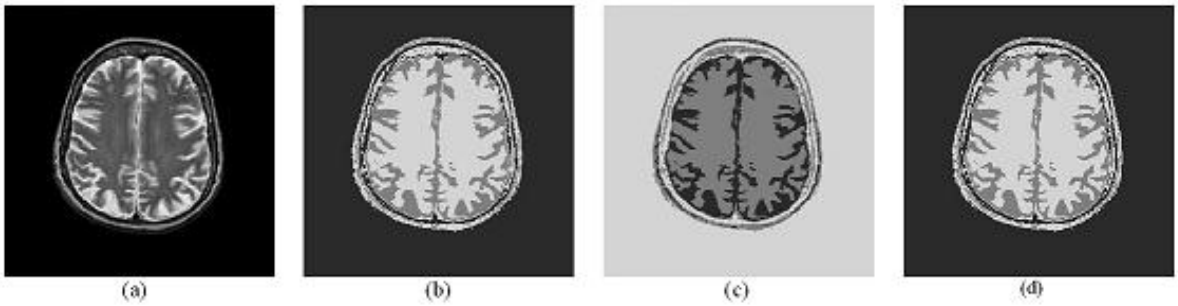

FIG. 4.6 – L'image médical originale et les résultats de segmentation (c=3).
(a) : L'image médicale (b) : Le résultat de **Algorithme de FCM** (c) : Le résultat de **Algorithme 4.3** (d) : Le résultat de **Algorithme 4.4**.

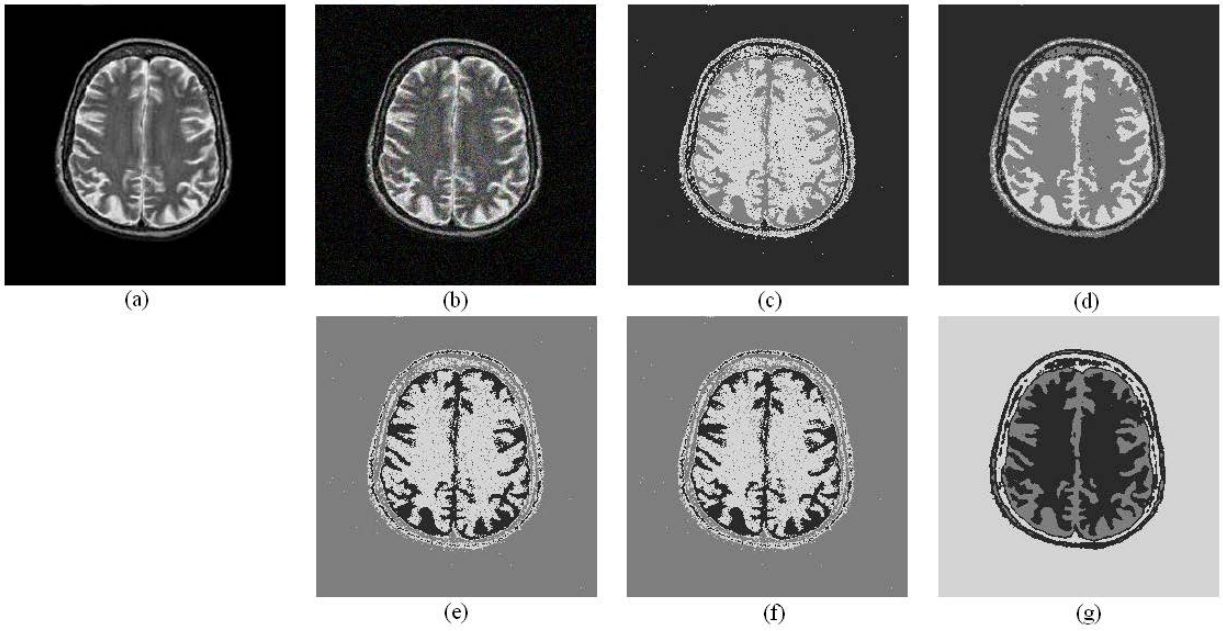

FIG. 4.7 – L'image médical avec le bruit Gaussien et les résultats de segmentation (c=3).
(a) : L'image médicale originale (b) : L'image médicale avec le bruit Gaussien (c) : Le résultat de **Algorithme de FCM** (d) : Le résultat de **Algorithme de FCM** avec l'information spatiale. (e) : Le résultat de **Algorithme 4.3** (f) : Le résultat de **Algorithme 4.4** (g) : Le résultat de **Algorithme 4.4** avec l'information spatiale.

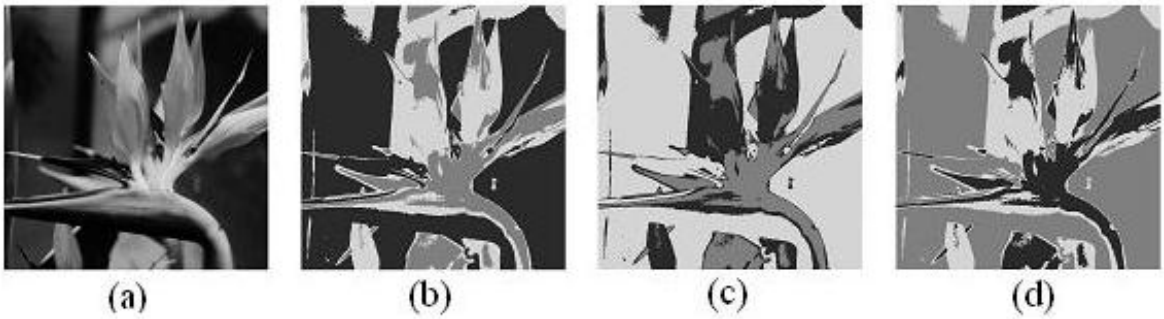

FIG. 4.8 – L'image **Blume** et les résultats de segmentation (c=3).
(a) : L'image **Blume** (b) : Le résultat de **Algorithme de FCM** (c) : Le résultat de **Algorithme 4.3** (d) : Le résultat de **Algorithme 4.4**.

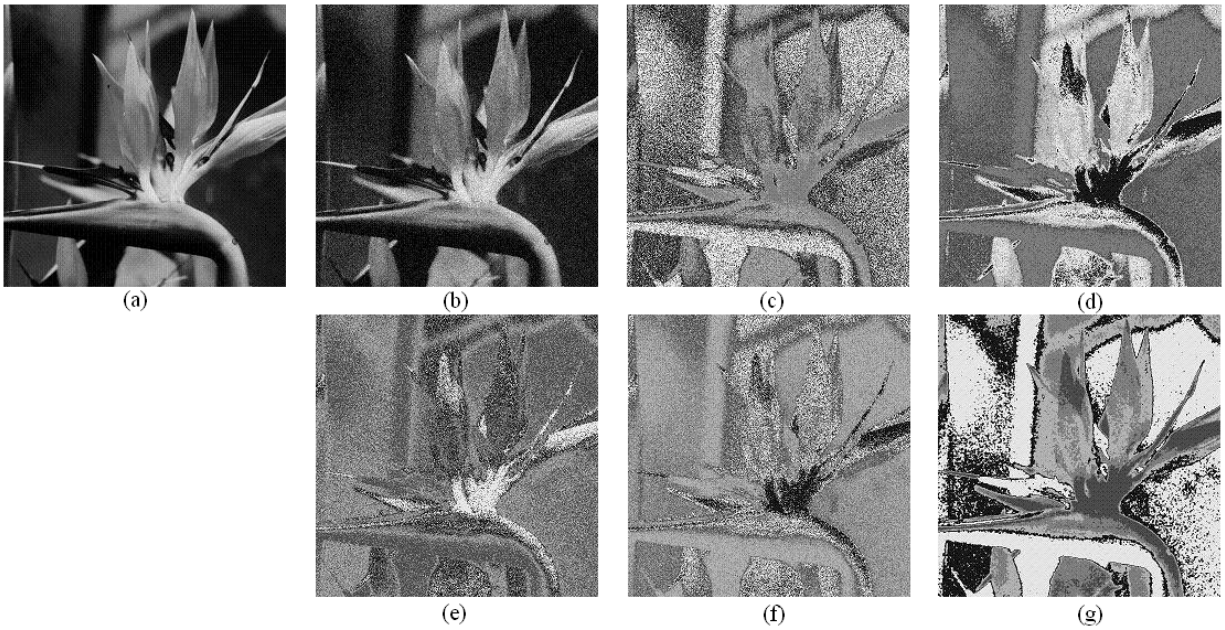

FIG. 4.9 – L'image médical avec le bruit Gaussien et les résultats de segmentation (c=5).
(a) : L'image **Blume** originale (b) : L'image **Blume** avec le bruit Gaussien (c) : Le résultat de **Algorithme de FCM** (d) : Le résultat de **Algorithme de FCM** avec l'information spatiale. (e) : Le résultat de **Algorithme 4.3** (f) : Le résultat de **Algorithme 4.4** (g) : Le résultat de **Algorithme 4.4** avec l'information spatiale.

TAB. 4.1 – Comparaison sur le temps de calcul de **Algorithme FCM** et **Algorithme 4.3, 4.4**.

Data			**FCM**		**Algorithm 4.3**		**Algorithm 4.4**		
$N°$	Size	c	N^oF	Time	N^oI	Time	q	N^oD	Time
1	128^2	2	24	1.453	16	1.312	12	10	1.219
2	128^2	2	17	1.003	12	0.985	10	2	0.765
3	256^2	3	36	15.340	24	13.297	20	2	10.176
4	256^2	3	75	31.281	57	30.843	30	12	26.915
5	256^2	3	39	15.750	27	14.687	20	14	13.125
6	256^2	5	91	84.969	75	86.969	40	78	61.500
7	256^2	3	73	31.094	62	34.286	15	21	24.188
8	256^2	3	78	34.512	52	32.162	20	13	29.182
9	512^2	3	49	92.076	41	102.589	30	46	74.586
10	512^2	5	246	915.095	196	897.043	120	86	691.854

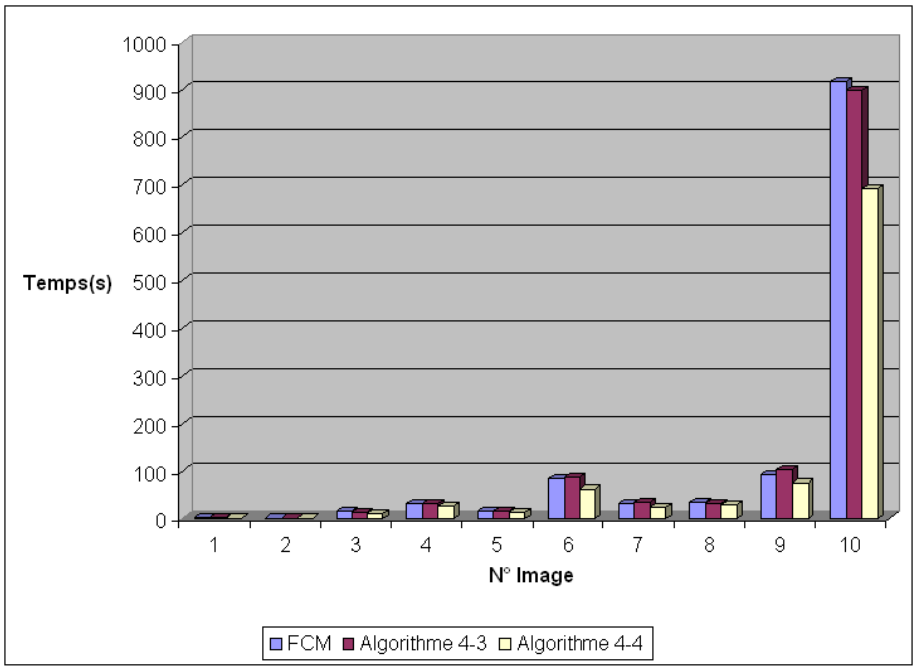

FIG. 4.10 – Comparaison sur le temps de calcul de **Algorithme FCM** et **Algorithme 4.3, 4.4**.

Chapitre 5

Une estimation non linéaire de la matrice fondamentale en vision par ordinateur

Résumé *Ce chapitre concerne une nouvelle approche d'une estimation non linéaire de la matrice fondamentale en vision par ordinateur. La matrice fondamentale est un concept-clé pour toutes les questions touchant à l'emploi d'images non calibrées prises de points de vue multiples. Elle contient toute l'information géométrique disponible et permet d'obtenir la géométrie épipolaire à partir de deux vues perspectives non calibrées. Le but de ce problème est de reconstruire le modèle 3D numérique de la scène à partir de deux vues d'une même scène.*

Des techniques linéaires d'estimation de la matrice fondamentale sont proposées mais ces techniques entraînent des erreurs considérables. Dans ce travail, nous approchons ce problème à partir d'un modèle non linéaire existant en appliquant la méthode région de confiance. A chaque itération, la méthode de région de confiance fournit le sous-problème et nous utilisons l'algorithme DCA pour résoudre ce sous problème. A la fin, nous comparons le résultat et la qualité de l'image obtenue à partir de cette combinaison avec celui obtenu à partir de la combinaison de la méthode de région de confiance et la méthode de gradient conjugué tronqué pour évaluer l'efficacité de chaque algorithme.

5.1 Introduction

La vision par ordinateur est la branche de l'informatique pour inférer les informations à partir d'images digitales, en général des photographies, des films vidéos qui peuvent être acquis par des capteurs comme les caméras vidéos, infrarouges, les radars ou les senseurs spécialisés. Parmi les nombreux exemples d'applications de la vision par ordinateur, nous pouvons citer : la surveillance, la création de modèles 3D numériques, la recherche d'information dans des bases de données d'images ou de séquences vidéos... Les outils fondamentaux de la vision par ordinateur, par exemple la détection, la segmentation, l'extraction d'indices visuels, la géométrie et la calibration des capteurs, la localisation et reconnaissance d'objets,

la reconstruction ou le traitement d'images, sont présentés dans un langage mathématique simple. Parallèlement aux avancées théoriques en vision par ordinateur, le développement des technologies est une des raisons de l'utilisation de la vision par ordinateur : l'expansion des systèmes informatiques a fourni la puissance de calcul nécessaire au traitement des données.

La vision 3D par ordinateur est un sous-ensemble de la vision par ordinateur, dont le but est d'inférer de l'information tridimensionnelle à partir d'images. Dans ce contexte, nous nous intéressons à la reconstruction de modèle 3D numérique de la scène à partir de deux vues d'une même scène. Ce modèle 3D numérique peut être utilisé dans les phases ensuites de la vision par ordinateur, par exemple, un véhicule autonome conduisant sur un terrain inconnu doit être capable de détecter les obstacles et d'estimer leur distance afin de pouvoir les éviter.

La détermination de la matrice fondamentale qui représente la relation entre deux vues de la même scène est une tâche très importante dans la vision 3D par ordinateur car elle nous permet d'avoir une reconstruction de modèle 3D numérique. La matrice fondamentale est une représentation algébrique de la géométrie épipolaire non calibrée qui contient toute l'information géométrique disponible et permet d'obtenir la géométrie épipolaire à partir de deux vues perspectives non calibrées. Plus précisement, elle est une corrélation transformant un point d'image sur la droite épipolaire correspondante dans l'autre image. La géométrie épipolaire est surtout considérée en vue de deux objectifs : l'appariement de primitives images et l'orientation relative de deux vues. Pour ce qui est de l'aspect orientation de deux vues, on distingue les géométries épipolaires "calibrées" et "non calibrées" selon qu'on considère des caméras calibrées ou non.

Plusieurs techniques ont été proposées pour estimer la matrice fondametale à partir des points correspondents et on peut classifier les différentes méthodes en deux catégories :

- La méthode basée sur le critère linéaire : pour chaque paire de points correspondants, on a une équation linéaire dont il est possible de résoudre le système d'équations linéaires avec 8 paires de points correspondants (8-Points Algorithm ([151])). Dans un article paru récemment, Hartley ([149]) montre comment on peut rendre la solution linéaire encore plus robuste numériquement tout simplement en opérant un changement de repère sur les données. Une solution linéaire qui tient explicitement compte de la géométrie du problème et qui donne également des résultats très satisfaisants a été recemment proposée par Boufama et al. ([146]). Un des inconvénients de cette approche est qu'elle est très sensible à l'image bruitée. En plus, elle ne considère pas le fait que le rang de la matrice fondamentale doit être égale à 2. Quelques nouvelles méthodes en comptant la condition du rang de la matrice fondamentale ont été proposées pour éliminer ces inconvénients (voir [150, 147]).
- La méthode basée sur le critère non linéaire : le premier critère de minimisation le plus souvent utilisé est la somme des carrés des distances d'un point à la droite épipolaire qui est censée passer par ce point ([151]). Le deuxième critère de minimisation est le critère du Gradient. Chaque pixel est mesuré avec une certaine incertitude et la minimisation du

critère initial recoit de chacun de ses termes une contribution de variance différente, ce qui rend la minimisation de la distance algébrique initiale sous-optimale. En introduisant la matrice de covariance d'un point et de son correspondant, on montre que le critère à minimiser est similaire au critère de distance (voir [151, 153, 152] pour le détail). Ces minimisations impliquent l'utilisation de techniques d'optimisation non linéaires telles que Newton, Gauss-Newton ou Levenberg-Marquardt. On peut confirmer que l'estimation non linéaire de la matrice fondamentale est moins sensible au bruit que la méthode linéaire.

Dans ce contexte, nous approchons ce problème à partir d'un modèle non linéaire du critère de la somme des distances d'un point à la droite épipolaire en appliquant la méthode de région de confiance. A chaque itération, la méthode de région de confiance fournit le sous-problème et nous utilisons l'algorithme DCA pour résoudre ce sous problème. A la fin, nous comparons le résultat et la qualité de l'image obtenue à partir de cette combinaison avec celui obtenu à partir de la combinaison de la méthode de région de confiance et la méthode de gradient conjugué tronqué pour évaluer l'efficacité de chaque algorithme.

Après avoir présenté le problème ici, dans le reste du chapitre nous donnons la notation et la formulation du problème d'estimation de la matrice fondamentale dans la deuxième section. La troisième section présente notre nouvelle approche et notre algorithme en combinant la méthode de région de confiance avec DCA ou la méthode de gradient conjugué tronqué. Nous finissons le chapitre avec quelques résultaux numériques sur certaines images standards sélectionnées.

5.2 Préliminaire

Dans cette section, nous décrivons des éléments de base employés par Hartley ([151]) et Horaud ([152]). Tout d'abord, nous définissons les notations utilisées. Ensuite, la géométrie épipolaire ainsi que la matrice fondamentale, l'estimation de la matrice fondamentale sont décrits.

5.2.1 Notations

Pour tout vecteur $a = [a_x, a_y, a_z]^T$, il y a une matrice antisymmétrique $[a]_\times$ définie comme

$$a_\times = \begin{bmatrix} 0 & a_z & -a_y \\ -a_z & 0 & a_x \\ a_y & -a_x & 0 \end{bmatrix}$$

telle que $[a]_\times b$ représente le produit vectoriel $a \times b$.

Un point $M = [x, y, z]^T$ dans l'espace 3D est représenté par ses coordonnées homogènes $M = [x, y, z, 1]^T$. De la même façon, son image sur le plan image (où plan repère) $m = [x, y]^T$ est représenté par ses coordonnées homogènes $m = [x, y, 1]^T$.

Une droite l dans l'espace 2D est représenté par un vecteur 3D, i.e. $l = [a, b, c]$. Si un point m est sur une droite l alors $l^T m = 0$. En conséquence, il n'y a aucune différence entre la représentation d'une droite et la représentation d'un point. Ce fait est connu comme le principe de dualité. L'équation d'une droite l passant par deux points de m_1 et m_2 est donnée par $l = m_1 \times m_2$ et l'intersection entre deux droite l_1 et l_2 est un point $m = l_1 \times l_2$.

Un plan Π dans l'espace 3D est représenté par un vecteur 4D, i.e. $\Pi = [a, b, c, d]$. Si un point M est sur le plan Π alors $\Pi^T M = 0$.

5.2.2 Modèle projectif

Soit un point O dans le plan image appelé point principal et soit une droite perpendiculaire au plan image passant par O, l'axe optique. Soit un point F placé sur l'axe optique à une distance f du plan image. Le point F est le centre de projection et f est la distance focale. Les images sont formées par projection sur le plan image à travers le centre de projection comme illustré à la figure 5.3. Sa projection M dans le plan image est donnée par la matrice de projection

$$sm = PM,$$

où s est un rapport d'échelle et P est une matrice 3×4 appelé matrice de projection.

La matrice de projection P peut être décomposée en une suite de transformations ([151])

$$P = AP_cT,$$

où :
- A est la transformation affine reliant les repères image et les points.
- P_c est la projection perspective proprement dite.
- T est la transformation rigide entre le repère monde et le repère caméra.

5.2.3 Géométrie épipolaire

Un des objectifs de la vision par ordinateur est de reconstruire la structure tridimensionnelle de l'espace à partir d'une ou plusieurs images. La vision stéréoscopique utilise deux images prises avec deux caméras. Connaissant le modèle de projection de chaque caméra et la relation spatiale entre les deux caméras, il s'agit de calculer les coordonnées 3D d'un point à partir de ses deux projections dans les deux images.

La géométrie épipolaire est la contrainte de base surgissant dans la vision stéréoscopique. La figure 5.4 illustre les projections m_1 et m_2 du point M dans l'espace avec deux caméras C_1 et

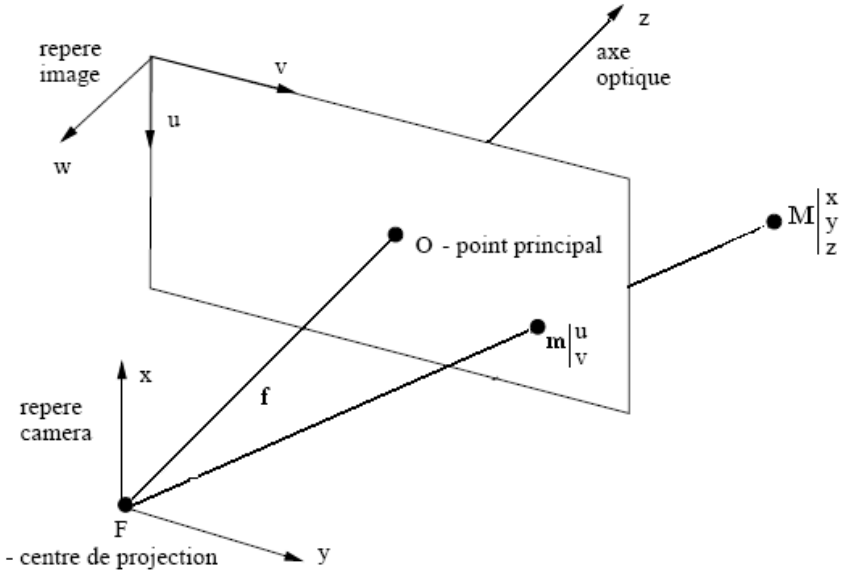

FIG. 5.1 – Le modèle géométrique d'une caméra.

C_2. Les droites qui sont les intersections du plan MC_1C_2, le plan épipolair, avec le plan Π_1 et Π_2 ont applé les droites épipolaires. La droite C_1C_2 passe par le point e_1 dans le premier repère image et par le point e_2 dans le deuxième repère image. Ce sont les épipoles, par lesquels passent les droites épipolaires. Pour chaque point m_1 dans le premier repère image, le point m_2 correspondant dans la deuxième repère image passe par la droite épipolaire l_2.

5.2.4 Matrice fondamentale

La matrice fondamentale contient toute l'information géométrique disponible et permet d'obtenir la géométrie épipolaire à partir de deux vues perspectives non calibrées. La matrice fondamentale F est une représentation algébrique de la géométrie épipolaire non calibrée et elle est la matrice d'une corrélation qui transforme des points dans une image sur des droites dans l'autre image. Plus précisément, il s'agit de la corrélation transformant un point image sur la droite épipolaire correspondante dans l'autre image.

Soit F_{12} la matrice fondamentale 3×3 passant de l'image 1 à l'image 2. La matrice fondamentale F_{21} pour le sens inverse est juste sa transposée $F_{12} = F_{21}^T$ dont nous utilisons les notations simplifiées $F = F_{12}$ et $F^T = F_{21}$. Pour un point m_1, la représentation projective de sa droite épipolaire est représentée $l_2 = Fm_1$. La contrainte épipolaire pour deux points de l'image m_1 et m_2 peut s'écrire comme

$$m_2^T F m_1 = 0,$$

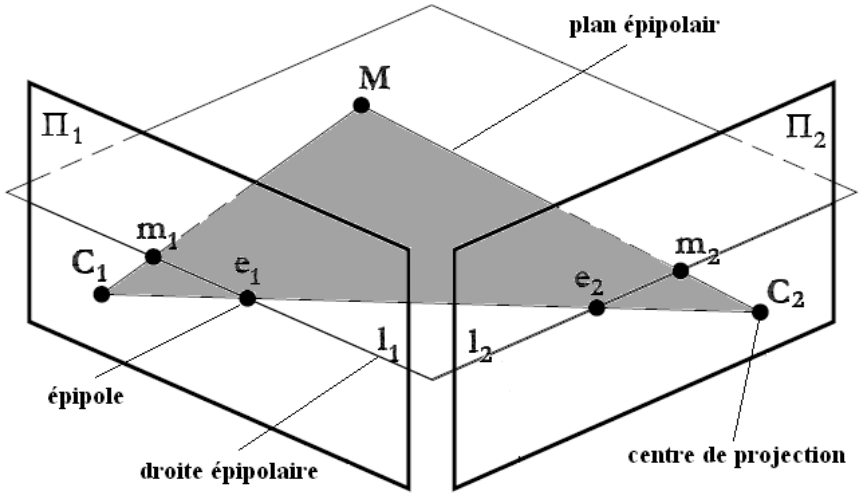

FIG. 5.2 – La géométrie épipolaire.

ou

$$m_1^T F^T m_2 = 0.$$

La connaissance de la matrice fondamentale F nous permet d'avoir une reconstruction projective. Si on considère les paramètres de la première caméra comme le premier repère image alors les paramètres de la deuxième caméra sont déninis par la matrice fondamentale liant de deux images :

$$P_1 = [I|0] \quad et \quad P_2 = [[e_2]_\times F|e_2].$$

5.2.5 Estimation de la matrice fondamentale

Plusieurs techniques ont été proposées pour estimer la matrice fondametale à partir des points correspondents. Ici, nous présentons les critères non linéaires pour cette estimasion.

5.2.5.1 Critère de la distance de droite épipolaire

Le premier critère de minimisation le plus souvent utilisé est la somme des carrés des distances d'un point à la droite épipolaire qui est censée passer par ce point. Considérons N points de chaque image, nous avons la somme de distance suivante :

$$\sum_{i=1}^{N} \left(d^2(m_{1i}, l_{1i}) + d^2(m_{2i}, l_{2i}) \right).$$

Puisque

$$l_{2i} = F m_{1i}, l_{1i} = F^T m_{2i} \quad et \quad m_{2i}^T F m_{1i} = m_{1i}^T F^T m_{2i},$$

dont, nous avons :

$$\sum_{i=1}^{N}\left(\frac{1}{(Fm_{1i})_1^2+(Fm_{1i})_2^2}+\frac{1}{(F^Tm_{2i})_1^2+(F^Tm_{2i})_2^2}\right)(m_{2i}^TFm_{1i})^2, \tag{5.1}$$

où $(Fm_{ki})_j^2$ est le carré de l'entité j^{ieme} du vecteur Fm_{ki} pour tout $k=1,2$.

5.2.5.2 Critère du Gradient

Dans ce méthode, chaque pixel est mesuré avec une certaine incertitude et la minimisation du critère initial recoit de chacun de ses termes une contribution de variance différente, ce qui rend la minimisation de la distance algébrique initiale sous-optimale. En introduisant la matrice de covariance d'un point et de son correspondant, on montre que le critère à minimiser est similaire au critère de distance suivant :

$$\sum_{i=1}^{N}\frac{(m_{2i}^TFm_{1i})^2}{(Fm_{1i})_1^2+(Fm_{1i})_2^2+F^Tm_{2i})_1^2+(F^Tm_{2i})_2^2}. \tag{5.2}$$

5.3 Méthode de région de confiance pour la résolution

5.3.1 Méthode de région de confiance ([1, 2])

Considérons le problème

$$(P) \quad \min\{f(x) : x \in \mathbb{R}^n\},$$

où f(x) est une fonction deux fois continûment différentiable sur $\mathbb{R}^n$. La plupart des méthodes itératives pour résoudre ce problème sont basées sur

- un modèle, qui est une approximation appropriée de la fonction objective qui permettra une prédiction de solution et une localisation de minima locaux,
- un prototype qui décrit la stratégie de l'utilisation du modèle considéré de manière à obtenir des propriétés satisfaisantes de convergence.

Le modèle le plus utilisé est quadratique, i.e.

$$q(d)=\frac{1}{2}d^THd+\omega^Td,$$

où $\omega \in \mathbb{R}^n$ et H est une matrice sysmétrique d'ordre n tel que

$$f(x+d)\approx f(x)+q(d)$$

dans une voisinage de F. Il est naturel de prendre ω comme le gradient de f en x et H comme le Hessien de f en x.

La méthode de région de confiance est introduite comme une amélioration de la méthode de Newton. Basée sur le modèle quadratique, elle génère la suite $\{x^k\}$ qui a plus de chances de converger ver une solution globale de (**P**). Cette approche est étudiée par plusieurs auteurs dans différents contextes, dont Levenberg ([80]) et Marquardt ([81]) sont les premiers, ensuite Moré et al. ([82, 83]), Sorensen et al. ([88]), Tao P.D et al. ([43, 41]), Wolkovicz et al. ([101]) ont des contributions importantes. Le méthode de région de confiance est actuellement reconnue comme la méthode la plus performance, robuste et fiable pour ce type de problème.

L'idée de la méthode de région de confiance est de calculer d^k appelé "direction de descente" en résolvant un problème de type

$$(P_k) \quad \min\left\{q_k(d) = \frac{1}{2}d^T H_k d + \omega_k^T d : \|d\| \leq r_k\right\}.$$

Le rayon r_k appelé rayon de confiance est choisi dynamiquement en fonction d'une mesure appropriée entre $q_k(d^k)$ et $f(x^k + d^k)$. Cette mesure est en général définie par la quantité suivante, appelée "coefficient de qualité"

$$\tau_k = \frac{\Delta f_k}{\Delta q_k} = \frac{f(x^k) - f(x^k + d^k)}{q_k(0) - q_k(d^k)}. \tag{5.3}$$

qui est le rapport entre la réduction actuelle de f à l'étape k et la réduction prédite correspondante du modèle quadratique.

Schéma de la méthode de région de confiance

– Etant données $x^0 \in \mathbb{R}^n$, $s_2 > s_1 > 0$ et $r_0 > 0$.
– Pour $k = 0, 1, ...,$
  k.1 Déterminer $f(x^k)$ et le modèle quadratique q_k.
  k.2 Calculer une solution d^k du problème (P_k).
  k.3 Déterminer τ_k via (5.3).
  k.3 Si $\tau_k \leq s_1$ alors $x^{k+1} = x^k$ et $r_{k+1} = \frac{r_k}{2}$, sinon $x^{k+1} = x^k + d^k$.
  Si $\tau_k \geq s_2$ alors $r_{k+1} = 2r_k$, sinon $r_{k+1} = r_k$.

La convergence globale de l'algorithme de la méthode de région de confiance peut être récapitulée dans le prochain théorème ([88, 82, 73]).

Théorème 5.1 *(Propriétés de la convergence) Soit $f : \mathbb{R}^n \to \mathbb{R}$ une fonction deux fois continûment différentiable et bornée inférieurement sur $\mathbb{R}^n$, et supposons que $\nabla^2 f$ est borné sur l'ensemble de niveau*

$$\left\{x \in \mathbb{R}^n : f(x) \leq f(x^0)\right\}.$$

Soit $\{x^k\}$ la suite généré par la méthode de région de confiance avec $H_k = \nabla^2 f(x^k)$ et d^k satisfaisant

$$q_k(d^k) \leq \beta_1 \min\{q_k(d) : \|d\| \leq r_k\}$$

et

$$\|d^k\| \leq \beta_2 r_k,$$

où les constantes β_1, β_2 *sont des nombres positifs. Alors*

1. $\lim_{k\to+\infty} \|\nabla f(x^k)\| = 0$.
2. *Si* $\{x^k\}$ *est bornée alors il existe un point limite* x^* *avec* $\nabla^2 f(x^*)$ *semi-définie positive.*
3. *Si* x^* *est un point limite isolé de* $\{x^k\}$ *alors* $\nabla^2 f(x^*)$ *est semi-définie positive.*
4. *Si* $\nabla^2 f(x^*)$ *est non singulière pour un point limite* x^* *de* $\{x^k\}$ *alors*
   a. $\nabla^2 f(x^*)$ *est définie positive,*
   b. $\lim_{k\to+\infty} = x^*$ *et il existe un* $\epsilon > 0$ *et* $K > 0$ *tel que* $r_k > \epsilon$, $\forall k > K$,
   c. *la convergence est superlinéaire.*

5.3.2 Formulation

Dans ce travail, nous utilisons le critère de la distance de droite épipolaire pour résoudre le problème d'estimation de la matrice fondametal. Considéron le problème d'optimisation suivant :

$$(\mathbf{P}) \quad \begin{cases} \min\limits_{F} \sum\limits_{i=1}^{N} d(m_{2i}, Fm_{1i})^2 + d(m_{1i}, F^T m_{2i})^2 \\ \text{tel que} \quad \|F\| = 1, \end{cases} \tag{5.4}$$

où la contrainte $\|F\| = 1$ pour éliminer la solution triviale.

Nous prenons les notations suivantes

$$F = \begin{pmatrix} f_1 & f_2 & f_3 \\ f_4 & f_5 & f_6 \\ f_7 & f_8 & f_9 \end{pmatrix}, \quad m_{1i} = \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix}, \quad m_{2i} = \begin{pmatrix} \bar{x}_i \\ \bar{y}_i \\ 1 \end{pmatrix}$$

qui nous permettent de détailler la mesure de distance de chaque point i

$$\frac{(m_{2i}^T F m_{1i})^2}{(f_1 x_i + f_2 y_i + f_3)^2 + (f_4 x_i + f_5 y_i + f_6)^2} + \frac{(m_{2i}^T F m_{1i})^2}{(f_1 \bar{x}_i + f_4 \bar{y}_i + f_7)^2 + (f_2 \bar{x}_i + f_5 \bar{y}_i + f_8)^2}.$$

5.3.3 Résolution par la méthode de région de confiance

Considérons la relaxation de la contrainte $\|F\| = 1$, la fonction objective dans (5.4) devient

$$\Phi(F) = \sum_{i=1}^{N} d(m_{2i}, Fm_{1i})^2 + d(m_{1i}, F^T m_{2i})^2 + \lambda(F^T F - 1)^2, \tag{5.5}$$

où λ est une constance de pénalisation sur F.

Clairement, la fonction $\Phi(F)$ est une fonction deux fois continue différentiable sur $\mathbb{R}^9$. Selon la description de la méthode de région de confiance ci-dessus, nous définissons la fonction $q(d)$

$$q(d) = \frac{1}{2} d^T H d + \omega^T d,$$

où $\omega \in \mathbb{R}^9$ est le vecteur du Gradient de Φ en F et H est une matrice symétrique du Hessien de Φ en F tel que

$$\Phi(F + d) \approx \Phi(F) + q(d)$$

dans une voisinage de F.

D'après le théorème de la méthode de région de confiance, à chaque itération, nous calculons la direction de descente d^k en résolvant le sous problème de type

$$(P_k) \quad \min\{q_k(d) = 1/2 d^T H_k d + \omega_k^T d : \|d\| \leq r_k\}, \tag{5.6}$$

et le rayon de confiance r_k

$$\tau_k = \frac{\Delta \Phi_k}{\Delta q_k} = \frac{\Phi(F^k) - \Phi(F^k + d^k)}{q_k(0) - q_k(d^k)}. \tag{5.7}$$

Algorithme 5.1 *Algorithme de région de confiance.*

Initialisation :
- *Choisir les tolérances ϵ_ω, ϵ_Φ, ϵ_r positives suffisamment petites.*
- *Calculer $F^0 \in R^9$ une solution linéaire de la matrice fondamentale utilisant la méthode de 8 points.*
- *Choisir $r_o \geq 0$ et poser $s_1 \leftarrow 0.25$, $s_2 \leftarrow 0.75$.*
- *Poser $k \leftarrow 0$*

Itération k = 0, 1,...

S.1. *Calculer $\Phi_k = \Phi(F^k)$*
Calculer le vecteur du Gradient ω_k de Φ en F^k
Calculer la matrice du Hessien H_k de Φ en F^k

S.2. *Si $\|\omega_k\| \leq \epsilon_\omega$ ou $\|\Phi_k\| \leq \epsilon_\Phi$ ou $\|r_k\| \leq \epsilon_r$ alors, F^k est une solution. STOP.*

S.3. *Résoudre le sous problème (P_k) par DCA (où la méthode de gradient conjugué tronqué) pour obtenir la solution d^k.*

S.4. *Déterminer τ_k via (5.7).*

S.5. *Déterminer F^{k+1} et r_{k+1}.*
*Si $\tau_k < s_1$ alors $r_{k+1} = r_k$ et retourner à **S.3**.*
Sinon, $F^{k+1} = F^k + d^k$.
Si $\tau_k \geq s_2$ alors $r_{k+1} = 2r_k$, sinon $r_{k+1} = r_k$.
*Poser $k + 1 \leftarrow k$ et retourner à **S.1**.*

La convergence de l'algorithme 5.1 peut être récapitulée dans la section de la méthode de région de confiance ([82, 83, 88]).

5.3.4 DCA pour résoudre le sous problème P_k

Dans cette section, nous considérons le sous problème d'optimisation quadratique non convexe suivant

$$\min\{q_k(d) = 1/2d^T H_k d + \omega_k^T d : \|d\| \leq r_k\}, \tag{5.8}$$

où $\omega_k \in \mathbb{R}^n$ et H_k est une matrice symétrique réelle d'ordre n. Nous montrons d'abord que (5.8) est une programmation DC et puis nous présentons le DCA appliqué à ce sous problème.

Soit $\mathcal{K} = \{d \in \mathbb{R}^n : \|d\| \leq r_k\}$. Soit $\chi_{\mathcal{K}}$ la fonction indicatrice sur $\mathcal{K}$, $\chi_{\mathcal{K}}(d) = 0$ si $d \in \mathcal{K}$, $+\infty$ sinon. Soit ρ le nombre positif tel que $\rho I - H_k$ est la matrice définite positive. Soit g_k et h_k les fonctions définies par

$$g_k(d) = 1/2\rho\|d\|^2 + \omega_k^T d + \chi_{\mathcal{K}}(d), \tag{5.9}$$

$$h_k(d) = \frac{1}{2} d^T(\rho I - H_k)d. \tag{5.10}$$

Par conséquent, g_k et h_k sont des fonctions convexes donc le problème P_k est une programmation DC sous la formule

$$\min\{g_k(d) - h_k(d) \ : \ d \in \mathcal{K}\}. \tag{5.11}$$

Selon la description de DCA dans ([27]), la résolution du problème P_k via cette décomposition par DCA consiste en la détermination de deux suites $\{y^l\}$ et $\{d^l\}$

$$y^l = (\rho I - H_k)d^l, \tag{5.12}$$

$$d^{l+1} \in argmin\left\{\frac{1}{2}\rho\|d\|^2 + d^T(\omega_k - y^l) + \chi_{\mathcal{K}}(d) : d \in \mathbb{R}^n\right\}. \tag{5.13}$$

Clairement, d^{l+1} est la projection de $(y^l - \omega_k)/\rho$ sur $\mathcal{K}$,i.e.

$$d^{l+1} = P_{\mathcal{K}}\left(d^l - \frac{1}{\rho}(H_k d^l + \omega_k)\right).$$

Algorithme 5.2 *Algorithme de DCA*

Initialisation :
- *Choisir $d^0 \in R^9$ tel que $d_i^0 = \frac{r_k}{3}$, $\quad \forall i = 1...9$.*
- *Calculer la valeur ρ de la matrice du Hessien H_k.*
- *Choisir une tolérance ϵ positive suffisamment petite.*
- *Poser $l \leftarrow 0$*

Répeter
- *Si $\|(\rho I - H_k)d^l - \omega_k\| \leq \rho r_k$ alors*

$$d^{l+1} = \frac{(\rho I - H_k)d^l - \omega_k}{\rho},$$

– *sinon,*

$$d^{l+1} = r_k \frac{(\rho I - H_k)d^l - \omega_k}{\|(\rho I - H_k)d^l - \omega_k\|}.$$

– $l + 1 \leftarrow l$

Jusqu'à $\|d^{l+1} - d^l\| \leq \epsilon$ *ou* $\|d^{l+1} - d^l\|/\|d^{l+1}\| \leq \epsilon$. *Dans ce cas,* d^l *est la* ϵ*-solution du problème* P_k.

La convergence de l'algorithme 5.3 peut être récapitulée dans le chapitre 1 ([1, 3, 27]).

5.3.5 Méthode de gradient conjugué tronqué pour résoudre le sous problème P_k

Considérons le sous problème d'optimisation quadratique non convexe (5.8) sous la forme générale

$$\min\{q(d) = 1/2 d^T H d + \omega^T d : \|d\| \leq r\}, \tag{5.14}$$

où $\omega \in \mathbb{R}^n$ et H est une matrice symétrique réelle d'ordre n.

A chaque itération de l'algorithme de région de confiance, le sous problème (5.14) doit être résolu exactement ou d'une façon approchée pour obtenir la solution d^k. Une façon de calculer la solution inexacte de (5.14) est la méthode de gradient conjuguée tronqué proposée par Toint ([92]) et Steihaug ([89]).

Selon la description de la méthode de gradient conjugué tronqué dans ([89]), la résolution de (5.14) consiste en la détermination de deux suites

$$x_{l+1} = x_l + \alpha_l d_l, \tag{5.15}$$

et

$$d_{l+1} = -\omega_l + \beta_l d_l, \tag{5.16}$$

où

$$\omega_l = \nabla q(x_l) = \omega + H x_l, \tag{5.17}$$

$$\alpha_l = \frac{-\omega_l^T d_l}{d_l^T H d_l}, \tag{5.18}$$

$$\beta_l = \frac{\|\omega_{l+1}\|^2}{\|\omega_l\|^2}. \tag{5.19}$$

Algorithme 5.3 *Algorithme de gradient conjugué tronqué*

Initialisation :

– *Poser* $x_0 = 0$, $\omega_0 = \omega$, $d_0 = -\omega$.

– *Poser* $l \leftarrow 0$

Répeter

1. Si $\|\omega_l\| = 0$ *alors* $d^* = x_l$*, arrêter.*
- *Si* $d_l^T H d_l \leq 0$*, passer à l'étape 3.*
- *Sinon, calculer* α_l *via (5.18).*

2. Si $\|x_l + \alpha_l d_l\| \geq r$ *alors, passer à l'étape 3.*
- *Calculer* x_{l+1} *via (5.15).*
- *Calculer* ω_{l+1} *via (5.17).*
- *Calculer* β_l *via (5.19).*
- *Poser* $d_{l+1} \leftarrow -\omega_l + \beta_l d_l$*.*
- $l \leftarrow l + 1$*, retourner à l'étape 1.*

3. Calculer $\alpha_l^* \geq 0$ *qui est satisfait de* $\|x_l + \alpha_l^* d_l\| = r$*.*
$d^* = x_l + \alpha_l^* d_l$*, arrêter.*

Dans ce cas, d^* *est la solution inexacte du sous problème* P_k*.*

5.4 Expériences numériques

Dans cette section, nous présentons quelques résultats obtenus en appliquant les méthodes proposées pour résoudre le problème de l'estimisation de la matrice fondamentale. Le but est d'étudier son efficacité par rapport à la méthode basée sur la critère linéaire. L'exactitude de la solution F sera évaluée en utilisant la mesure de la distance épipolaire moyenne

$$d = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{\sqrt{(Fm_{1i})_1^2 + (Fm_{1i})_2^2}} + \frac{1}{\sqrt{(F^T m_{2i})_1^2 + (F^T m_{2i})_2^2}}\right)(m_{2i}^T F m_{1i}).$$

Pour vérifier l'efficacité de notre méthode proposée, nous utilisons une image standard avec 50 points utilisés (matching point en Englais). La figure 5.4 (resp. 5.5, 5.6) affiche les droites épipolaires et les points utilisés de l'algorithme 8 points (resp. de l'algorithme de région de confiance en résolvant le sous problème par DCA ou par la méthode gradient conjugué tronqué). Le tableau 5.1 montre la distance épipolaire moyenne de chaque algorithme.

Tab. 5.1 – La distance épipolaire moyenne de chaque méthode.

Méthode	d	DET(F)
8-Points	17.919	4.5821e-12
RC-DCA	13.566	0.0022
RC-GC	1.7452	0.1929

Commentaires. A partir des résultats expérimentaux, nous constatons que :
- L'algorithme combiné de la méthode région de confiance et DCA donne une image de qualité supérieure par rapport à celle obtenue en appliquant l'algorithme 8 points améliorée.

- Le tableau 5.1 montre que l'algorithme combiné de la méthode de région de confiance avec la méthode de gradient conjugué tronqué donne le meilleur résultat. Il est alors facile de s'apercevoir que l'efficacité de l'algorithme combinant DCA n'est pas satisfaisante car le nombre de variables dans ce modèle est très petit. Cependant, cela pourrait être une nouvelle approche prometteuse si nous considérons le modèle qui compte les points utilisés comme les variables. Plus précisément, c'est une technique permettant de contraindre la position des points.

Conclusion. Dans ce chapitre, nous proposons une nouvelle méthode de l'estimation de la matrice fondamentale. A partir d'un modèle non linéaire existant, nous appliquons la méthode de région de confiance, à chaque itération, son sous problème est résolu par DCA ou par la méthode de gradient conjugué tronqué. Les résultats expérimentaux prouvent que la méthode basée sur la distance épipolaire a une stabilité meilleure que la méthode 8-points améliorée.

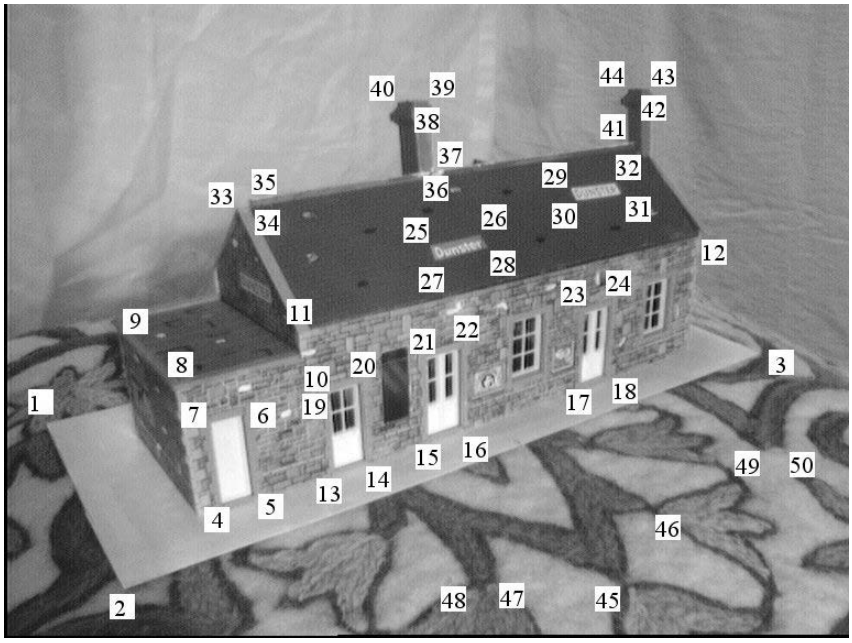

FIG. 5.3 – L'image originale et 50 points utilisés.

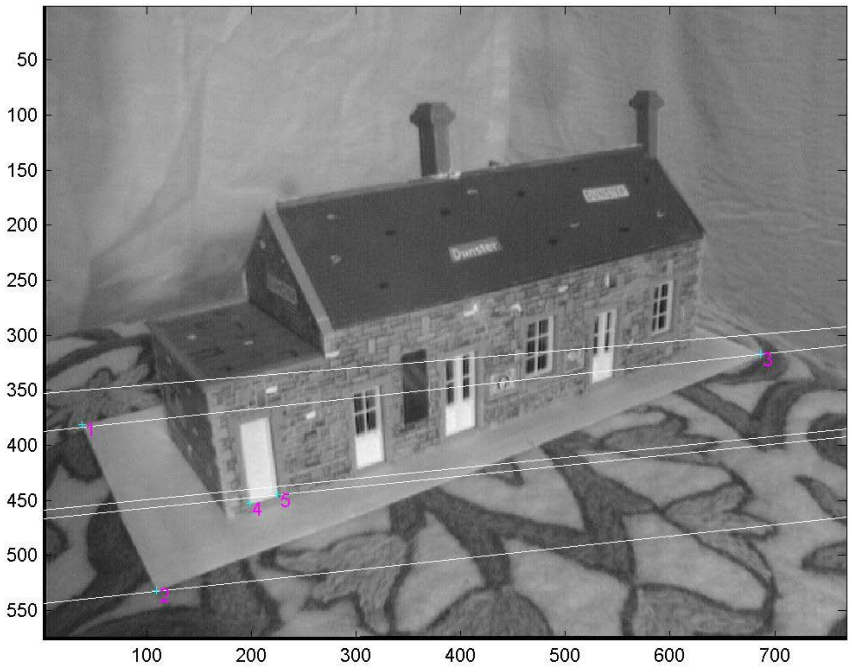

FIG. 5.4 – Le résultat de l'algorithme 8 points.

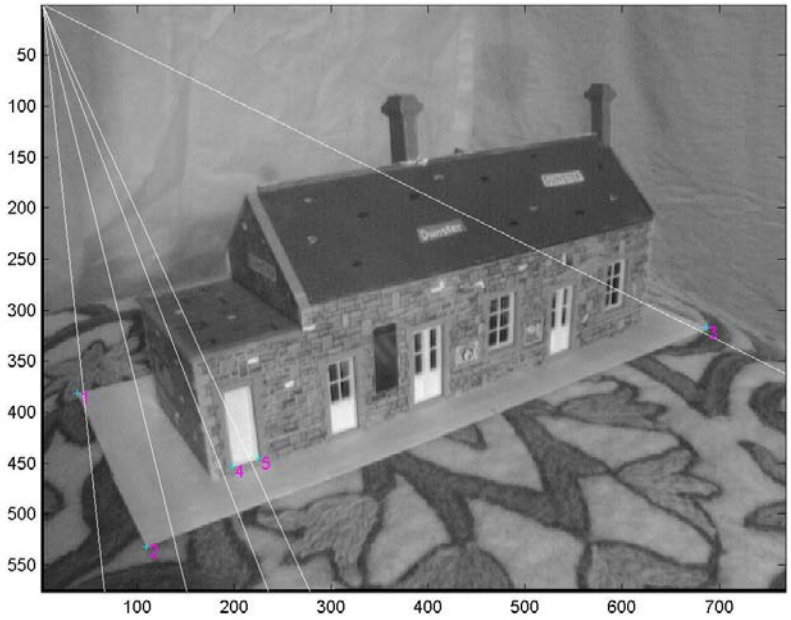

FIG. 5.5 – Le résultat de l'algorithme de région de confiance et DCA.

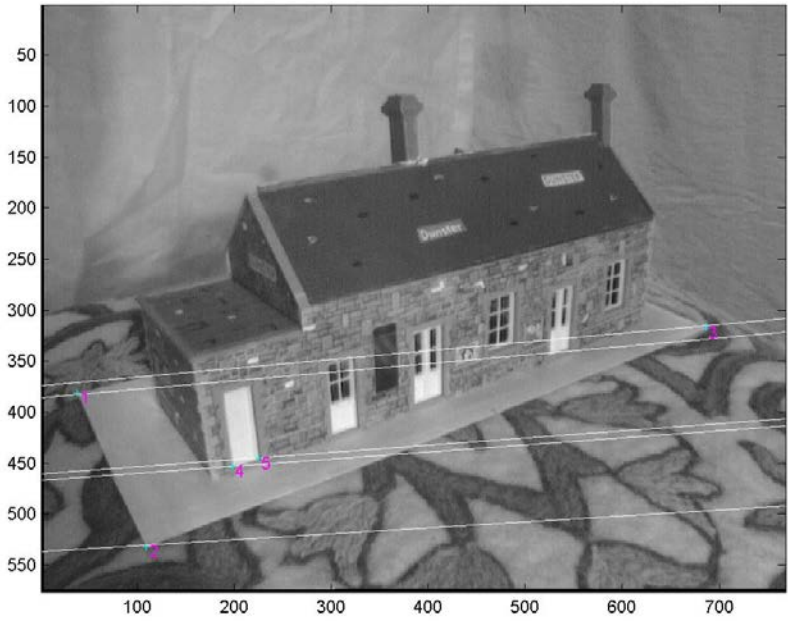

FIG. 5.6 – Le résultat de l'algorithme de région de confiance et de gradient conjugué tronqué.

Conclusion

Cette thèse s'intéresse particulièrement aux techniques d'optimisation en traitement d'image et vision par ordinateur et en transport logistique. Sur le plan méthodologique elle repose sur trois grandes lignes d'optimisation non convexe : méthodes globales de Séparation et Évaluation, la programmation DC et DCA, et la méthode de région de confiance, parmi lesquelles DCA joue le rôle crucial. Notre ambition est de proposer les nouveaux algorithmes combinés efficaces basées sur DCA étant capables de traiter des problèmes de grande taille. Sur le plan d'application nous concentrons à certains problèmes importants qui sont très étudiés et intéressés par les chercheurs en recherche opérationnelle et imagerie. Pour chacun de ces problèmes, nous avons proposé des algorithmes adaptés à sa structure spéciale.

Pour le problème de conception d'une chaîne d'approvisionnement multi-niveaux nous avons proposé une combinaison de l'algorithme de Séparation et Évaluation et DCA. Cette combinaison permet de trouver des bons points initiaux pour DCA et de prouver la globalité de DCA. Sa performance est largement supérieure aux schémas SE classiques et elle a été ainsi appliquée avec succès aux problèmes de taille réelle où le nombre des variables binaires est grande.

Quant au problème de reconstruction d'une image binaire, bien que nous l'avons considéré comme une programmation linéaire et/ou quadratique en varaiables mixtes binaires, il est difficile d'envisager les méthodes globales de type SE ou AE, vu le nombre des variables binaires et le nombre des contraintes. Nous avons donc utilisé seul arenal puissant de DCA et avons proposé une technique pour rechercher un bon paramètre de pénalité pour améliorer la qualité de l'image recontruite. Nous proposons également une nouvelle façon d'estimation inférieure de la fonction quadratique non convexe sur un rectangle. Cette borne nous semble intéressante pour la recherche d'un bon point initial de DCA.

De même pour le problème de segmentation d'image qui est une étape primaire dans la plupart des applications de la vision d'ordinateur à l'analyse d'image (essentiellement en imagerie médicale), il nous est impossible de développer les méthodes globales de type SE ou AE, vu la complexité de la fonction objectif et la taille très grande du modèle FCM de classification floue que nous avons utilisé pour la segmentation. Puisque la programmation

DC et DCA constitue un cadre théorique et algorithmique enrichissant pour la fouille de données ([18]-[20] et [49]-[62]), nous avons exploité un schéma de DCA extrêmement simple pour la classification floue ([20]) en incorporant les informations spatiales au modèle FCM standard. DCA appliqué au nouveau modèle est de même forme que DCA pour le modèle FCM standard, et avec les informations spatiales notre algorithme améliore nettement la segmentation des images brutées. Pour trouver un bon point initial de l'algorithme, nous avons mis en place un algorithme combiné de DCA et FCM.

Concernant le problème d'estimation non linéaire de la matrice fondamentale qui joue un rôle très important en vision par ordinateur nous avons appliqué la méthode région de confiance sur un modèle existant pour améliorer les méthodes classiques de type Levenbert Marquard. Nous avons proposé deux méthodes de résolution (dont DCA) de sous-problème de minimisation d'une forme quadratique non convexe sur une boulle de la norme euclidenne.

Finalement, dans les expériences numériques, nous avons montré la superiorité des algorithmes proposés par rapport aux algorithmes standards.

Perspectives

Une question pertinente lors d'utilisation de DCA est "comment trouver le bon point initial" ? Dans les problèmes de conception d'une chaîne d'approvisionnement multi-niveaux du chapitre 2 et de segmentation d'image du chapitre 4, nos techniques proposées répondent efficacement à cette question, pourtant celle-ci reste ouverte dans la reconstruction d'image binaire et la résolution des sous problèmes issus de la méthode de région de confiance pour l'estimation de la matrice fondamentale.

Pour la résolution des programmations linéaire et/ou quadratique en variables mixte 0-1 dans les chapitre 2 et 3, le paramètre de pénalité influence considérablement sur la vitesse de la convergence de l'algorithme et la qualité de la solution obtenue. Il est donc très intéressant d'étudier d'une façon plus approfondie sur le choix d'un tel paramètre.

Par ailleurs dans le problème de conception d'une chaîne d'approvisionnement multi-niveaux, nous avons combiné DCA avec une méthode d'optimisation globales de type SE en utilisant la relaxation linéaire classique. Nous constatons que DCA fournit rapidement une solution optimale mais l'algorithme SE doit continuer davantage plusieurs itérations pour prouver la globalité de DCA en améliorant lentement la borne inférieure. La question est de trouver d'autres bornes inférieures et/ou d'autres méthodes (méthodes de coupe par exemple) qui pourraient être combinées avec DCA de façon plus efficaces ?

Enfin, la reconstruction d'image binaire ou plus généralement la discrète tomographie a été

considérée dans ce travail sous la forme de trois problèmes d'optimisation (dont la dimension est différente), or il existe d'autres modèles d'optimisation équivalents pour lesquels nous pourrions appliquer DCA. La question de trouver le meilleur schéma de DCA reste ouverte.

Tous ces questions feront l'objet de nos travaux dans une future proche.

ANNEXE

ARTICLE IN PRESS

Available online at www.sciencedirect.com
SCIENCE DIRECT®

European Journal of Operational Research xxx (2006) xxx–xxx

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/ejor

A continuous DC programming approach to the strategic supply chain design problem from qualified partner set ☆

Le Thi Hoai An [a,*], Nguyen Trong Phuc [a], Pham Dinh Tao [b]

[a] *Laboratory of Theoretical and Applied Computer Science (LITA), UFR MIM, University of Paul Verlaine, Metz, Ile du Saulcy, 57045 Metz, France*

[b] *Laboratory of Modeling, Optimization and Operations Research, LMI, National Institute for Applied Sciences, Rouen BP 08, Place Emile Blondel F 76131, Mont Saint Aignan Cedex, France*

Received 15 October 2004; accepted 15 June 2005

Abstract

We present a new continuous approach based on the DC (difference of convex functions) programming and DC algorithms (DCA) to the problem of supply chain design at the strategic level when production of a new market opportunity has to be launched among a set of qualified partners. A well known formulation of this problem is the mixed integer linear program. In this paper, we reformulate this problem as a DC program by using an exact penalty technique. The proposed algorithm is a combination of DCA and Branch and Bound scheme. It works in a continuous domain but provides mixed integer solutions. Numerical simulations on many empirical data sets show the efficiency of our approach with respect to the standard Branch and Bound algorithm.

Keywords: DC Programming; DCA; Mixed Integer programming; Exact penalty techniques; Branch and bound techniques; Supply chain design

1. Introduction

In the today's competitive marketplace, the analysis and design of production–distribution systems have been an active area of research for higher efficiency and lower operational costs. The objective of these systems is the management of material and information flows both in and between facilities, such as vendors, manufacturing and assembly plants and distribution centers. Companies continuously search for ways to improve their operations based on the approaches such as the optimization models and algorithms, decision support systems or computerized analysis tools.

☆ This work is partially supported by the Conseil Régional Lorraine via the project "Optimisation et Aide à la Décision dans les Systèmes d'Information".

* Corresponding author.
E-mail addresses: lethi@univ-metz.fr (L.T. Hoai An), nguyen@sciences.univ-metz.fr (N.T. Phuc), pham@insa-rouen.fr (P.D. Tao).

0377-2217/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2005.06.079

Various types of problems have been considered in many bodies of research: a single component of the overall production–distribution systems, the coordination of two different functions of the system such as buyer–vendor, production–distribution and inventory–distribution [11–13,19], the distinction between the integration of different function [6], and/or the coordination of different levels of the company within the same function.

Three levels of planning are distinguished via the time horizon [7]. They are strategic, tactical and operational levels. The strategic level considers long term decisions and requires approximate and aggregated data. The operational level involves short term decisions and requires transactional data. The tactical one falls in between the two previous extremes levels. In this paper we are interested in a strategic design of supply chains. Many models have been formulated for this problem. One of them is developed with emphasis on mixed integer programming models [20]. For an extensive review on strategic production–distribution models in a global supply environment, the reader is referred to [20,8,15,18].

The model that we consider is formulated in [9,10] as a mixed integer programming problem, and the difficulty arises when the number of binary variables increases. There are a lot of algorithms designed for mixed integer programming problem such as Branch and Bound method, cutting plane algorithm and dynamic programming [14]. Some methods are developed based on the conjunction with others. One of them is known as the branch and cut by combining the cutting plane algorithm with Branch and Bound method. Among approaches based on the key properties of the problems being considered, heuristic methods are proposed to find a good solution.

In this paper, by using an exact penalty technique we treat this problem as a DC program in the context of continuous optimization. Further, we combine the DCA with the classical Branch and Bound method for finding global solutions.

The remainder of the paper is organized as follows. In Section 2, we report the description and the formulation of this problem [10] as a mixed integer program. Section 3, describes how to reformulate the problem in the continuous form via an exact penalty technique. Section 4, is devoted to the DC programming and DCA for solving the penalty equivalent. The combined DCA–Branch and Bound algorithm is presented in Section 5, while the computational results are reported in Section 6.

2. Problem description and formulation

2.1. Description

The model that we consider (see [9,10]) is composed φ required operations (noted $E_1, E_2, \ldots, E_\varphi$) to manufacture the product. Each qualified partner has a unit production cost and holding costs. There is a transportation cost between two successive qualified partners and a fixed cost to make a collaboration between them. These costs may vary from echelon to echelon, from partner to partner and possibly from period to period. The problem is represented as a network $G = (N, A)$, where N is the set of nodes and A is the set of arcs. Each node represents a location of qualified partner and each arc represents a possible connection between two partners. The objective is searching the ''best'' chain connecting a node in the first echelon to a node in the last echelon. The problem's formulation is considered in [9,10] with the following assumptions:

- Single market opportunity.
- Demand is deterministic but not constant over the planning horizon, and is fulfilled only by nodes in the final echelon of the network.
- Backorders are not permitted at the last echelon.
- There are capacity constraints at every node and the associated capacities may differ from node to node within an echelon and from echelon to echelon.
- Operation time for the total units produced in a period plus transportation time to the next node in sequence is one period.
- The transportation system is uncapacitated.

2.2. Mixed integer formulation

The problem is formulated in [9,10] as a mixer integer problem that we describe below.

Parameters given:

$P = \langle 1, \ldots, \varphi \rangle$ sequence of required operations to manufacture the product.
N_α number of nodes in echelon α.
$F_{i,\alpha,j,\alpha+1}$ fixed setup cost making the connection between node i in echelon α to node j in echelon $\alpha+1$.
$C_{i,\alpha,j,\alpha+1,t}$ transportation unit cost from node i in echelon α to node j echelon $\alpha+1$ in period t.
$H^f_{i,\alpha,t}$ holding cost at the exit of node i in echelon α in period t.
$H^r_{i,\alpha,t}$ holding cost at the entrance of node i in echelon α in period t.
$U_{i,\alpha,t}$ unit processing cost at node i in echelon α in period t.
T last period in the planning horizon when the demand exists.
d_t forecasted demand in period t.
M a very large number, greater than the sum of the forecasted demand.
$\Phi_{i,\alpha,t}$ production capacity available of node i in echelon α in period t.

Variables:

$h^f_{i,\alpha,t}$, $h^r_{i,\alpha,t}$, $z_{i,\alpha,t}$: amount of finished goods, amount of raw material and amount of produced goods at node i in echelon α in period t.
$x_{i,\alpha,j,\alpha+1,t}$: amount of product shipped from node i in echelon α to node j in echelon $\alpha+1$ in period t.

$$W_{i,\alpha} := \begin{cases} 1 & \text{if node } i \text{ in echelon } \alpha \text{ is included in the chain,} \\ 0 & \text{otherwise.} \end{cases}$$

$$Y_{i,\alpha,j,\alpha+1} := \begin{cases} 1 & \text{if both of two nodes are included in the chain,} \\ 0 & \text{otherwise.} \end{cases}$$

Mixed Integer formulation:

$$(\text{MIP}) \sum_{\alpha=1}^{\varphi-1} \sum_{i=1}^{N_\alpha} \sum_{j=1}^{N_{\alpha+1}} F_{i,\alpha,j,\alpha+1} Y_{i,\alpha,j,\alpha+1} + \sum_{\alpha=1}^{\varphi-1} \sum_{t=1}^{T} \sum_{i=1}^{N_\alpha} \sum_{j=1}^{N_{\alpha+1}} C_{i,\alpha,j,\alpha+1,t} x_{i,\alpha,j,\alpha+1,t} + \sum_{\alpha=1}^{\varphi} \sum_{t=1}^{T} \times \sum_{i=1}^{N_\alpha} \left(H^f_{i,\alpha,t} h^f_{i,\alpha,t} + H^r_{i,\alpha,t} h^r_{i,\alpha,t} + U_{i,\alpha,t} z_{i,\alpha,t} \right)$$

Subject to:

$$\sum_{i=1}^{N_\alpha} W_{i,\alpha} = 1 \quad \forall \alpha \in P \tag{1}$$

$$Y_{i,\alpha,j,\alpha+1} \geqslant W_{i,\alpha} + W_{j,\alpha+1} - 1 \tag{2}$$

$$\alpha = 1, 2, \ldots, \varphi - 1 \quad i = 1, \ldots, N_\alpha \quad j = 1, \ldots, N_{\alpha+1}$$
$$z_{i,\alpha,t} \leqslant \Phi_{i,\alpha,t} W_{i,\alpha} \quad \forall \alpha \in P \quad i = 1, \ldots, N_\alpha \quad t = 1, \ldots, T \tag{3}$$

$$\sum_{t=1}^{T} \sum_{j=1}^{N_{\alpha+1}} x_{i,\alpha,j,\alpha+1,t} \leqslant W_{i,\alpha} M \quad \alpha = 1, 2, \ldots, \varphi - 1 \quad i = 1, 2, \ldots, N_\alpha \tag{4}$$

$$\sum_{t=1}^{T} \sum_{i=1}^{N_\alpha} x_{i,\alpha,j,\alpha+1,t} \leqslant W_{j,\alpha+1} M \quad \alpha = 1, 2, \ldots, \varphi - 1 \quad j = 1, 2, \ldots, N_{\alpha+1} \tag{5}$$

$$h^f_{i,\alpha,t} = h^f_{i,\alpha,t-1} + z_{i,\alpha,t} - \sum_{j=1}^{N_{\alpha+1}} x_{i,\alpha,j,\alpha+1,t} \tag{6}$$

$$\alpha = 1, 2, \ldots, \varphi - 1 \quad i = 1, 2, \ldots, N_\alpha \quad t = 1, 2, \ldots, T$$
$$h^f_{i,\varphi,t} = h^f_{i,\varphi,t-1} + z_{i,\varphi,t} - d_t W_{i,\varphi} \quad i = 1, 2, \ldots, N_\varphi \quad t = 1, 2, \ldots, T \tag{7}$$

$$h^r_{j,\alpha+1,t} = h^r_{j,\alpha+1,t-1} - z_{j,\alpha+1,t} + \sum_{i=1}^{N_\alpha} x_{i,\alpha,j,\alpha+1,t} \tag{8}$$

$$\alpha = 1,2,\ldots,\varphi-1 \quad j = 1,2,\ldots,N_{\alpha+1} \quad t = 1,2,\ldots,T$$

$$h^f_{i,\alpha,0} = 0 \quad i = 1,2,\ldots,N_\alpha \quad \forall \alpha \in P \tag{9}$$

$$h^r_{i,\alpha,0} = 0 \quad i = 1,2,\ldots,N_\alpha \quad \forall \alpha \in P \tag{10}$$

$$h^f_{i,\alpha,t}, h^f_{i,\alpha,t}, z_{i,\alpha,t} \geqslant 0 \quad i = 1,2,\ldots,N_\alpha \quad \forall \alpha \in P \quad t = 1,2,\ldots,T \tag{11}$$

$$x_{i,\alpha,j,\alpha+1,t}, Y_{i,\alpha,j,\alpha+1} \geqslant 0, \quad i = 1,2,\ldots,N_\alpha \quad j = 1,2,\ldots N_{\alpha+1} \quad \alpha = 1,2,\ldots,\varphi-1 \quad t = 1,2,\ldots,T \tag{12}$$

$$W_{i,\alpha} \in \{0,1\} \quad i = 1,2,\ldots,N_\alpha \quad \alpha = 1,2,\ldots,\varphi. \tag{13}$$

The objective function to be minimized is the integrated cost for production, inventory holding and transportation. Constraints (1) and (2) ensure that at each stage, one and only one qualified partner is selected in the chain. Constraint (3) deals with the capacity of production of each partner while constraints (4) and (5) express the collaboration between two partners in the successive stages. To ensure a relation between the transportation, the production and the stocking, the constraints (6)–(8) are necessary. Constraints (9) and (10) provide the initial conditions of inventory at the partners. Constraints (11) and (12) indicate that the quantities of production, transportation, and stocking are nonnegative. Finally, constraint (13) means that the variables W_i are binary. Note that, the time horizon does not start in period 1 and does not finish in period T for all echelons. In this model, we consider that all variables outside of the production is null.

This problem is a mixed-integer problem and so a nonconvex program. Its difficulty depends on the number of stages, the number of qualified partners in each stage and the number of periods. The number of variables, the number of binary variables and the number of constraints are calculated by following formulas.

The number of variables:

$$(T+1) * \sum_{\alpha=1}^{\varphi-1} N_\alpha N_{\alpha+1} + (3T+4) * \sum_{\alpha=1}^{\varphi} N_\alpha.$$

The number of binary variables:

$$\sum_{\alpha=1}^{\varphi} N_\alpha.$$

The number of constraints:

$$(T+2) * \sum_{\alpha=1}^{\varphi-1} N_\alpha N_{\alpha+1} + (6T+5) * \sum_{\alpha=1}^{\varphi} N_\alpha + \varphi - N_\varphi - (T+1)N_1.$$

3. Concave minimization reformulation

In this section, using the well known results concerning the exact penalty we will formulate (MIP) in the form of concave minimization programming. Consider now the mixed integer linear program in a general form:

$$(\text{GMIP}) \ \alpha = \min\left\{c^{\mathrm{T}}x + d^{\mathrm{T}}y : (x,y) \in D, y \in \{0,1\}^m\right\},$$

where $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$ are continuous and binary variables respectively, and D is a nonempty bounded polyhedral convex set in $\mathbb{R}^n \times \mathbb{R}^n$ defined by a finite number of linear constraints.

In [1], problem (GMIP) is reformulated as a concave quadratic program via the penalty function $\theta(y) = \sum_{i=1}^m y_i(1-y_i)$. In this paper, we use another penalty function that gives a more suitable DC program and then makes efficiency of our algorithm. Let us consider the function p defined by

$$p(x,y) = \theta(y) := \sum_{i=1}^{m} \min(y_i, 1-y_i).$$

Set $K:=\{(x,y)\in D: y\in[0,1]^m\}$. Clearly, p is concave and finite on K, $p(x,y)\geqslant 0$ for all $(x,y)\in K$, and

$$\{(x,y)\in D,\ y\in\{0,1\}^m\}=\{(x,y)\in K,\ p(x,y)\leqslant 0\}.$$

Hence problem (GMIP) can be rewritten as

$$\alpha=\min:\left\{c^{\mathrm{T}}x+d^{\mathrm{T}}y:(x,y)\in K,\ p(x,y)\leqslant 0\right\}. \tag{14}$$

From Theorem 1 below we get, for a sufficiently large value of t ($t\geqslant t_0$), the equivalent concave minimization problem to (GMIP):

$$\min:\left\{c^{\mathrm{T}}x+d^{\mathrm{T}}y+tp(x,y):(x,y)\in K\right\}. \tag{15}$$

Theorem 1 (Theorem 1, [5]). *Let K be a nonempty bounded polyhedral convex set, f be a finite concave function on K and p be a finite nonnegative concave function on K. Then there exists $t_0\geqslant 0$ such that the following problems have the same solution set and the same optimal value*:

$$(P_t)\quad \alpha(t)=\inf\{f(x)+tp(x):x\in K\}$$
$$(P)\quad \alpha=\inf\{f(x):x\in K,p(x)\leqslant 0\}.$$

More precisely if the vertex set of K, denoted by $V(K)$, is contained in $\{x\in K,p(x)\leqslant 0\}$, then $t_0=0$, otherwise $t_0=\min\left\{\frac{f(x)-\alpha(0)}{s}:x\in K,p(x)\leqslant 0\right\}$, where $S:=\min\{p(x):x\in V(K),p(x)>0\}>0$.

Now let $x\in\mathbb{R}^n$ be the continuous variables and $y\in\mathbb{R}^m$ be the binary variables in problem (MIP). Clearly, the set D of feasible points (x,y) determined by the system of the constraints $\{(1),\ldots,(12)\}$ is a nonempty, bounded polyhedral convex set in $\mathbb{R}^n\times\mathbb{R}^m$. Problem (MIP) can be expressed in the form of (GMIP) and so the above result is available for (MIP).

4. A DCA scheme for solving problem (15)

4.1. DC programming

In this section, we investigate a DC programming approach for solving (15). A general DC program is of the form:

$$\alpha:=\inf\{f(x):=g(x)-h(x):x\in\mathbb{R}^n\}, \tag{16}$$

with g, h being lower semicontinuous proper convex functions on $\mathbb{R}^n$. It should be noted that in (16) the convex constraint set C is incorporated in the convex DC component g with the help of its indicator function χ_C

$$\chi_C(x):=0\quad\text{if}\quad x\in C,+\infty\quad\text{otherwise.}$$

The dual of (16) is the DC program

$$\alpha:=\inf\{h^*(y)-g^*(y):y\in\mathbb{R}^n\},$$

where g^* is the conjugate function of g:

$$g^*(y):=\sup\{\langle x,y\rangle-g(x):x\in\mathbb{R}^n\}.$$

A function g is polyhedral convex function on $\mathbb{R}^n$ if it is of the form:

$$g(x):=\max\{\langle a_i,x\rangle-\gamma_i:i=1,\ldots,m\}+\chi_C(x),$$

where $a_i\in\mathbb{R}^n,\gamma_i\in\mathbb{R}$, $i=1,\ldots,m$, $C\subset\mathbb{R}^n$ is a nonempty polyhedral convex set and χ_C is the indicator function of C.

If g or h are polyhedral convex functions then (15) is called a polyhedral DC program. The DCA (see [1–4,16,17], and references therein) is an efficient method which has been successfully applied to a lot of various large-scale nonconvex programs. It is a descent method without linesearch, consisting of the construction of the two sequences $\{x^k\}$ and $\{y^k\}$, (candidates for being primal and dual solutions, respectively), such that their corresponding limit points x^∞ and y^∞ satisfy local optimality conditions. Recall that the optimality conditions for DC programming are given by:

(i) If x^* is a local minimizer for (16) then

$$\emptyset \neq \partial h(x^*) \subset \partial g(x^*), \tag{17}$$

where $\partial h(x^*) := \{y^* \in \mathbb{R}^n : h(x) \geqslant h(x^*) + \langle x - x^*, y^* \rangle \quad \forall x \in \mathbb{R}^n\}$ is the subdifferential of h at x^*. Recall that $\partial h(x^*)$ is the extension of the derivative to nondifferentiable convex function and each element $y^* \in \partial h(x^*)$ is a subgradient of h at x^*.

Its converse is true for some class of DC programs, in particular for polyhedral ones in which the second DC component h is a polyhedral convex function ([2,4,16]).

(ii) x^* is called a critical point of $g - h$ or for (16) if

$$\emptyset \neq \partial g(x^*) \cap \partial h(x^*). \tag{18}$$

There are two forms of DCA: the simplified DCA (or simply DCA) and the complete DCA [2,4,16]. In practice the first is preferred to the latter because it is less expensive.

DCA (DC Algorithm) ([2,4,16,17])

1. Let $x^1 \in \mathbb{R}^n$. Set $k = 1$ and let ϵ_1, ϵ_2 be sufficiently small positive numbers.
2. Compute $y^k \in \partial h(x^k)$.
3. Compute $x^{k+1} \in \partial g^*(y^k)$, i.e., x^{k+1} is a solution of the convex program

$$\min\{g(x) - \langle x, y^k \rangle : x \in \mathbb{R}^n\}.$$

4. If either $\|x^{k+1} - x^k\| \leqslant \epsilon_1(\|x^k\| + 1)$ or $|f(x^{k+1}) - f(x^k)| \leqslant \epsilon_2(|f(x^k)| + 1)$, then stop and x^k is the computed solution, otherwise, set $k = k + 1$ and go to Step 2.

4.2. DCA for solving problem (15)

We first prove that (15) is a DC program and then present the DCA applied to the resulting DC program. Denote by χ_K the indicator function on K. Let g and h be the functions defined by

$$g(x,y) = \chi_K(x,y) \quad \text{and} \quad h(x,y) = -c^{\mathrm{T}}x - t\sum_{i=1}^{m} \min(y_i, 1 - y_i). \tag{19}$$

Hence g and h are convex functions, and so problem (15) is a DC program in the form

$$\min\{g(x,y) - h(x,y) : (x,y) \in \mathbb{R}^n \times \mathbb{R}^m\}. \tag{20}$$

By the very definition of h, a subgradient $(u,v) \in \partial h(x,y)$ can be chosen as follows:

$$(u,v) \in \partial h(x,y) \Leftarrow u = -c, \quad v = (v_i), \quad \text{with} \quad v_i := \begin{cases} -t & \text{if} \quad y_i \geqslant 0.5, \\ t & \text{otherwise.} \end{cases} \tag{21}$$

Algorithm 1 (*DCA applied to* (15)).

1. Let $(x^1, y^1) \in \mathbb{R}^n \times \mathbb{R}^m$. Set $k = 1$ and let ϵ_1, ϵ_2 be sufficiently small positive numbers.
2. Compute $(u^k, v^k) \in \partial h(x^k, y^k)$ via (21).
3. Solve the linear program:

$$\min\{-\langle (u^k, v^k), (x,y) \rangle : (x,y) \in K\} \tag{22}$$

to obtain (x^{k+1}, y^{k+1}).

4. If either $\|(x^{k+1}, y^{k+1}) - (x^k, y^k)\| \leqslant \epsilon_1(\|(x^k, y^k)\| + 1)$ or

$$\left|\langle c, x^{k+1}-x^k\rangle + t\sum_{i=1}^{m}\left(\min\left(y_i^{k+1}, 1-y_i^{k+1}\right) - \min\left(y_i^k, 1-y_i^k\right)\right)\right|$$
$$\leqslant \epsilon_2\left(\left|\langle c, x^k\rangle + t\sum_{i=1}^{m}\min\left(y_i^k, 1-y_i^k\right)\right| + 1\right)$$

then stop, (x^k, y^k) is the computed solution, otherwise, set $k = k + 1$ and go to Step 2.

The convergence of Algorithm 1 can be summarized in the next theorem whose proof is essentially based on the DCA's Convergence Theorem for a polyhedral DC program [2,4,16].

Theorem 2 Theorem 2, Convergence properties of Algorithm 1.

(i) *Algorithm 1 generates a sequence* $\{(x^k, y^k)\}$ *contained in* $V(K)$ *such that the sequence* $\{c^{\mathrm{T}}x^k + t\theta(y^k)\}$ *is decreasing.*
(ii) *There is a nonnegative number* t_1 *such that for every* $t > t_1$ *the sequence* $\{\theta(y^k)\}$ *is decreasing. In particular, if* (x^r, y^r) *is a feasible solution of* (GMIP) *then* (x^k, y^k), *for* $k \geqslant r$, *is feasible too.*
(iii) *The sequence* $\{(x^k, y^k)\}$ *converges to* $(x^*, y^*) \in V(K)$ *after a finite number of iterations. The point* (x^*, y^*) *is a critical point of problem (15). Moreover if* $y_i^* \neq \frac{1}{2}$ *for* $i = 1,\ldots,m$, *then* (x^*, y^*) *is a local minimizer to problem (15).*

Proof

(i) Is a consequence of DCA's convergence theorem for a general DC program (see [1–4,16,17]).
(ii) As mentioned above, if $V(K)$ is contained in the feasible solution set of (GMIP) then the assertion is trivial with $t_1 = 0$. Otherwise let

$$\xi := \min\{\theta(y') - \theta(y) : ((x,y),(x',y')) \in V(K)\times V(K), \theta(y') > \theta(y)\};$$
$$\eta := \max\{c^{\mathrm{T}}x' - c^{\mathrm{T}}x : ((x,y),(x',y')) \in V(K)\times V(K)\};$$

then $0 < \xi < +\infty$ and $0 \leqslant \eta < +\infty$ since $V(K)$ is a finite. Consider now the nonnegative number t_1 defined by

$$t_1 := \frac{\xi}{\eta}$$

and $t > t_1$. Let $\{(x^k, y^k)\}$ be the sequence generated by Algorithm 1 applied to (15) with this value t. Assume for contradiction that there is $r \geqslant 1$ such that $\theta(y^{r+1}) > \theta(y^r)$. Then

$$t[\theta(y^{r+1}) - \theta(y^r)] > t_1[\theta(y^{r+1}) - \theta(y^r)] = \frac{\xi}{\eta}[\theta(y^{r+1}) - \theta(y^r)] \geqslant \xi.$$

Hence

$$t[\theta(y^{r+1}) - \theta(y^r)] \geqslant c^{\mathrm{T}}x^r - c^{\mathrm{T}}x^{r+1}$$

i.e.,

$$c^{\mathrm{T}}x^{r+1} + t\theta(y^{r+1}) > c^{\mathrm{T}}x^r + t\theta(y^r),$$

that contradicts the decrease of the sequence $\{c^{\mathrm{T}}x^k + t\theta(y^k)\}$.
(iii) Problem (15) with the DC decomposition (19) is a polyhedral DC program of the form (16) since the second DC component h is a polyhedral convex function. (In fact the first DC component g is polyhedral convex too). So DCA applied to (16) has a finite convergence ([2,4,16]). According to the DCA's convergence theorem, for any DC program, the solution computed by DCA is a critical point of $g - h$, i.e.,

$$\partial g(x^*, y^*) \cap \partial h(x^*, y^*) \neq \emptyset. \tag{23}$$

If $y_i^* \neq 1/2, \quad \forall i = 1, \ldots, m$, then h is differentiable at (x^*, y^*) and then the condition (23) becomes $\partial h(x^*, y^*) \subset \partial g(x^*, y^*)$. This subdifferential inclusion (which is a necessary condition of local optimality in DC programming) is also sufficient in the case of a polyhedral DC program whose second DC component h is a polyhedral convex function ([2,4,16]). The proof is then complete. □

Remark. According to Theorem 2, let us emphasize the key features of DCA applied to (15): ($\{(x^k, y^k)\}$ being generated by DCA applied to (15) with $t > \max\{t_0, t_1\}$)

(i) Both sequences $\{c^{\mathrm{T}}x^k + t\theta(y^k)\}$ and $\{\theta(y^k)\}$ are decreasing.
(ii) If (x^r, y^r) is feasible for (GMIP) then (x^k, y^k), for $k \geqslant r$, is feasible too. In this case the sequence $\{(x^k, y^k)\}$ moves in the feasible solution set of (GMIP), $((x^k, y^k) \in V(K), y^k \in \{0,1\}^m)$, while decreasing the objective function.

These nice properties have great impacts on the search for mixed integer solutions.

5. Restart DCA: A combined DCA–Branch and Bound algorithm

For globally solving the strategic supply chain design problem we combine the DCA with the classical Branch and Bound method applied to (MIP). The linear relaxation is used for computing lower bounds while the upper bounds are determined by applying DCA to (15). Our combined algorithm can be summarized as follows: starting with the rectangle $R_0 := [0,1]^m$, we consider at each iteration $k \geqslant 0$ the rectangle R_k corresponding to the smallest lower bound β_k. The selected rectangle R_k is divided into two subrectangles R_k,$i = 0,1$ and the lower bound is improved by solving the corresponding linear programs. The upper bound γ_k is determined by applying the DCA to (15). The procedure is determined when $\gamma_k - \beta_k \leqslant \epsilon$ and it provides an ϵ-optimal solution of (MIP).

The combined algorithm.

Let $R_0 := [0,1]^m$. Set $\gamma_0 := +\infty$, $\beta_0 := -\infty$, *restart* := *true*, $\mathscr{R} := \{R_0\}$, and $k = 0$. Let ϵ be sufficiently small positive number.

1. Let R_k be the rectangle such that $\beta_k = \beta(R_k) = \min\{\beta(R) : R \in \mathscr{R}\}$. Bisect R_k into two subrectangles R_{k_0} and R_{k_1} via the index j^*

$$R_{k_i} = \{y \in R_k : y_{j^*} = i, \ i = 0, 1\}.$$

2. Compute lower bounds β_{k_i} $(i = 1, 2)$ by solving the linear relaxation problems corresponding to the set R_{k_i}.
3. If (*restart* = *true*) then update γ_k, the best upper bound of the optimal value of (MIP) by applying DCA to problem (15) from a suitable starting point discovered in Step 2.
4. If $\mathscr{R} =$ (i.e., $\gamma_k - \beta_k \leqslant \epsilon$), then STOP, the optimal solution is (x^k, y^k) that verify $c^{\mathrm{T}}x^k + d^{\mathrm{T}}y^k = \gamma_k$, otherwise update

$$\mathscr{R} \leftarrow \mathscr{R} \cup \{R_{k_i} : \beta(R_{k_i}) < \gamma_k - \epsilon, i = 0, 1\} \setminus R_k$$

and go to Step 1.

The combined algorithm differs from the classical Branch and Bound scheme by Step 3 in which DCA is investigated. Here *restart* is a boolean variable which takes value *true* when we decide to restart DCA. The question *when DCA is restarted* is interesting from numerical points of views and it will be studied in Section 5.2. As in several DC programs, DCA provides a global solution to (15) (and so is to (MIP)) from a *good* starting point. Such a point can be found efficiently while computing lower bounds (see Section 5.1 for the details). We will see in the computational experiments that DCA provides a global solution after the first two iterations of the brand and bound algorithm. Nevertheless we must continue the brand and bound process to ameliorate the lower bound until *near* the best upper bound. In fact, the Branch and Bound algorithm is introduced to find a good starting point for DCA and check the globality of DCA.

5.1. Finding a good starting point for DCA

From Theorem 2 we see that, starting with a feasible solution to (MIP) DCA provides a better feasible solution, although it works on a continuous feasible set of (15). It is so important to find a good feasible point to (MIP) for restarting DCA.

During the Branch and Bound procedure we can restart DCA from the best feasible solution to (MIP) that is discovered while computing lower bounds. This is motivated by a similar and efficient way introduced in the combined DCA–Branch and Bound algorithm for nonconvex quadratic programming [1].

On the other hand, for obtaining rapidly a good feasible point to (MIP) we also investigate a fast procedure to compute an optimal solution of the concave quadratic program

$$0 = \min\left\{\sum_{i=1}^{m} y_i(1-y_i) : (x,y) \in K\right\}. \tag{24}$$

That is the DCA developed in [1].

In our algorithm, a suitable starting point is computed by choosing one of the two procedures according to the current situation.

5.2. When DCA is restarted?

During the Branch and Bound process we restart DCA when a feasible solution to (MIP) which improves the best current upper bound is pointed out. In such a case, the starting point of DCA is the just mentioned feasible solution to (MIP).

On the other hand, DCA is also restarted when the number of the $0-1$ components of the binary variables (denoted $N_{y^{R_k}}$) of the solution (x^{R_k}, y^{R_k}) to the corresponding linear relaxation problem is sufficiently large, namely $N_{y^{R_k}} \geqslant m/2$. The starting point of DCA in this case is the solution of problem (24).

We now describe the combined DCA–Branch and Bound algorithm for globally solving problem (MIP).

5.3. Algorithm 2 (DCA–BB)

Let $R_0 := [0,1]^n$.

Solve the linear relaxation problem of (MIP) to obtain an optimal solution (x^{R_0}, y^{R_0}) and the first lower bound $\beta_0 := \beta(R_0)$.
Solve (15) by DCA from the starting point (x^{R_0}, y^{R_0}) to obtain $(x_t^{R_0}, y_t^{R_0})$.
If $(x_t^{R_0}, y_t^{R_0})$ is feasible to (MIP) **then** set $\gamma_0 := c^{\mathrm{T}} x_t^{R_0}$ and set $(x^0, y^0) := (x_t^{R_0}, y_t^{R_0})$ **else** set $\gamma_0 := +\infty$.
If $(\gamma_0 - \beta_0) \leqslant \epsilon|\gamma_0|$ **then** (x^0, y^0) is an ϵ-optimal solution of (MIP) **else** set $\mathscr{R} \leftarrow \{R_0\}, k \leftarrow 0$.
While stop = false **do**
Select a rectangle R_k such that $\beta_k = \beta(R_k) = \min\{\beta(R) : R \in \mathscr{R}\}$.
Bisect R_k into two subrectangles R_{k_0} and R_{k_1} via the index $j*$

$$R_{k_i} = \{y \in R_k : y_{j*} = i, \quad i = 0,1\}$$

Solve the subproblems (P_{k_i}) to obtain $\beta(R_{k_i})$ and $(x^{R_{k_i}}, y^{R_{k_i}})$:

$$(P_{k_i})\ \beta(R_{k_i}) := \min\{c^{\mathrm{T}} x : (x,y) \in K, \quad y \in R_{k_i}\} \quad (i = 0,1).$$

If $(x^{R_{k_i}}, y^{R_{k_i}})$ is the best feasible solution to (MIP) **then** update γ_k and the best feasible solution (x^k, y^k) by applying DCA to (15) from $(x^{R_{k_i}}, y^{R_{k_i}})$.
else if $(N_{y^{R_{k_i}}} \geqslant m/2)$ **then** solve (24) by DCA to obtain $(x_t^{R_{k_i}}, y_t^{R_{k_i}})$. Apply DCA to (15) from $(x_t^{R_{k_i}}, y_t^{R_{k_i}})$.
Update γ_k and the best feasible point (x^k, y^k).
Endif
Endif
Set $\mathscr{R} \leftarrow \mathscr{R} \cup \{R_{k_i} : \beta(R_{k_i}) < \gamma_k - \epsilon, i = 0,1\} \setminus R_k$

If $\mathscr{R} = \emptyset$ **then** STOP, (x^k, y^k) is ϵ-optimal solution,
else $k \leftarrow k + 1$.
Endwhile

6. Computational results

The algorithm was implemented on the Dell computer in C++ with double precision. To solve resulting linear programs, we used the software CPLEX version 7.0. In order to evaluate the performance of the proposed algorithm, we randomly generated examples in which the number of binary variables is increased by changing the number of echelons and the number of nodes. All the generated examples had at least one feasible solution. In this experiment, we compare the efficiency of our DCA–BB algorithm with the classical Branch and Bound algorithm. For all test problems, we always obtained an ϵ-optimal solution, with $\epsilon \leqslant 5.10^{-2}$. The results of our algorithm in each case are summarized in the following tables with 10 test problems for each case. We use the following notations:

- N^oit: number of iterations of each algorithm.
- UB, LB: the last upper bound and the last lower bound of each algorithm.
- CPU: the total time for each algorithm and is given in seconds.
- R: the number of restarting DCA.
- N^oF: the 'th' – restarting DCA in Algorithm 2 at which DCA provides the first feasible solution to (MIP).

Table 1 contains the results of the problems with 5 stages, 6 nodes for each stage and 8 periods. There are 2280 variables with 30 binary variables and 929 constraints. Tables 2–4 contain the results of the problems with 10 stages, 10 nodes for each stage and 2, 4 and 6 corresponding periods. There are 100 binary variables. The last row of each table indicates the average results on 10 test problems.

Comments on the numerical simulations: from the results in the tables below we observe that

- DCA without restart (Algorithm 1) found an ϵ-optimal (mixed integer) solution to (MIP) in several cases (13/40 problems). In such a case Algorithm 2 (BB-DCA) needs more iterations only for improving lower bounds.
- The DCA with restartings provides a first feasible solution to (MIP) very rapidly: the average number of restarting DCA is 1.6 on 40 test problems.
- DCA is inexpensive and can so handle problems with large number of binary variables. The superiority of DCA–BB relative to the Branch and Bound algorithm increases when the number of binary variables increases. DCA–BB is fast for large-scale problems while Branch and Bound algorithm is quite slow or it cannot solve some problems in reasonable times.

Table 1
The performance of the algorithms with 5 echelons, 6 nodes for each echelon and 8 periods in the planning horizon

No.	Branch and Bound				DCA and Branch and Bound					
	N^oit	UB	LB	CPU	N^oit	R	UB	LB	CPU	N^oF
01	23	12751540	12749999	17.250	6	3	13009398	12527906	27.703	2
02	6	13096550	12936516	4.656	1	1	13096550	12886344	3.875	1
03	26	12311228	12274897	20.797	10	3	12311228	12076966	31.500	1
04	7	11041377	10994134	8.687	1	1	11041377	10912780	5.906	1
05	12	12254463	12246380	16.016	8	2	12529261	12192327	35.969	2
06	27	12173214	12160131	32.062	10	2	12173214	11881686	36.141	2
07	143	13354261	13353492	93.000	39	7	13694028	13155286	115.00	2
08	6	11399543	11280127	7.454	1	1	11399543	11245948	6.813	1
09	19	12024911	11974646	16.422	11	2	12239793	11920423	23.735	2
10	51	12943749	12829760	48.469	17	3	13140308	12626941	57.391	3
T	*32.0*	–	–	*26.481*	*10.4*	*2.4*	–	–	*34.403*	*1.9*

There are 2280 variables with 30 binary variables and 929 constraints.

Table 2
The performance of the algorithms with 10 echelons, 10 nodes for each echelon and 2 periods in the planning horizon

No.	Branch and Bound				DCA and Branch and Bound					
	N^oit	UB	LB	CPU	N^oit	R	UB	LB	CPU	N^oF
01	28	8 104 834	8 104 460	32.05	10	4	8 396 799	8 006 590	35.28	2
02	24	8 027 372	7 989 584	22.16	9	3	8 196 882	7 951 863	34.53	3
03	11	5 833 312	5 816 744	12.61	1	1	5 851 223	5 784 518	4.20	1
04	171	6 428 269	6 425 561	128.49	31	8	6 609 466	6 462 683	67.39	2
05	20	6 439 370	6 437 419	16.94	1	1	6 495 153	6 416 677	3.99	1
06	250	7 160 614	7 052 829	225.95	19	7	7 268 901	6 923 530	65.34	2
07	65	8 480 361	8 386 595	53.99	44	4	8 480 361	8 310 454	65.25	3
08	145	8 937 962	8 800 644	118.84	20	7	8 973 170	8 611 216	73.00	7
09	28	8 281 668	8 095 728	27.11	10	4	8 227 518	8 030 711	39.84	2
10	113	8 494 666	8 366 524	110.06	54	3	8 729 058	8 334 181	79.70	3
T	*85.5*	–	–	*74.82*	*19.9*	*4.2*	–	–	*46.85*	*2.6*

There are 4600 variables with 100 binary variables and 1870 constraints.

Table 3
The performance of the algorithms with 10 echelons, 10 nodes for each echelon and 4 periods in the planning horizon

No.	Branch and Bound				DCA and Branch and Bound					
	N^oit	UB	LB	CPU	N^oit	R	UB	LB	CPU	N^oF
01	342	13 775 283	13 668 217	1024.03	16	1	13 931 102	13 280 286	126.64	1
02	174	12 587 549	12 517 932	428.41	45	8	12 694 792	12 421 078	306.31	1
03	77	15 176 654	15 008 953	211.75	1	1	15 291 922	14 593 939	20.52	1
04	78	14 457 193	14 301 062	317.97	1	1	14 554 540	13 904 302	17.38	1
05	1319	16 762 655	16 555 092	3444.72	189	4	17 085 218	16 301 870	1012.21	1
06	67	14 435 312	14 271 973	165.99	17	1	14 846 049	14 184 674	94.41	1
07	33	15 021 565	14 911 221	101.05	20	4	15 467 196	14 852 319	150.13	1
08	17	13 164 280	13 120 270	63.45	1	1	13 164 280	13 018 489	16.72	1
09	23	13 776 803	13 404 263	70.47	1	1	13 776 803	13 267 787	17.94	1
10	620	14 959 440	14 696 923	1342.77	118	22	14 934 627	14 457 342	747.49	1
T	*275.0*	–	–	*717.06*	*40.9*	*4.4*	–	–	*250.96*	*1*

There are 7000 variables with 100 binary variables and 2450 constraints.

Table 4
The performance of the algorithms with 10 echelons, 10 nodes for each echelon and 6 periods in the planning horizon

No.	Branch and Bound				DCA & Branch and Bound					
	N^oit	UB	LB	CPU	N^oit	R	UB	LB	CPU	N^oF
01	482	20 333 800	19 896 506	3672.70	136	20	20 326 982	19 632 766	2321.84	1
02*	144	20 750 750	20 635 325	1506.38	5	1	20 859 678	19 883 544	196.88	1
03	622	21 501 251	21 199 952	4163.52	186	29	21 855 651	20 965 148	2936.67	1
04*	52	20 490 694	20 281 185	804.86	28	3	21 233 824	20 303 946	661.95	1
05*	99	21 419 045	21 203 578	1080.16	4	1	21 727 324	20 700 557	170.92	1
06	301	22 394 747	22 024 054	2241.97	102	14	22 881 899	21 825 177	1633.05	1
07*	352	21 659 415	21 539 517	3868.56	138	14	21 667 224	21 346 899	2715.89	1
08	333	21 616 715	21 367 102	3312.48	76	26	22 120 621	21 152 122	1801.58	1
09*	259	21 999 296	21 737 766	2085.94	101	13	22 655 609	21 734 990	1869.78	1
10	425	20 551 240	20 281 236	2928.49	156	26	21 043 910	20 084 409	2474.47	1
T	*306.9*	–	–	*2566.51*	*93.2*	*14.7*	–	–	*1678.30*	*1*

There are 9400 variables with 100 binary variables and 3030 constraints.

7. Conclusion

We proposed an efficient approach for the supply chain design problem from qualified partner sets. Our method is based on DCA and Branch and Bound decomposition method. Preliminary numerical simulations show that the DCA–BB is very interesting. The DCA is original because it can give a mixed integer solution while it works in a continuous domain. Its inexpensiveness is crucial to high-dimensional problems that we address in future work.

References

[1] Le Thi Hoai An, Pham Dinh Tao, A Continuous Approach for Globally Solving Linearly Constrained Quadratic Zero—One Programming Problems, Optimization 50 (2001) 93–120.
[2] Le Thi Hoai An, Pham Dinh Tao, DC programming: Theory, algorithms and applications. The State of the Art. In: Proceedings (containing the refereed contributed papers) of The First International Workshop on Global Constrained Optimization and Constraint Satisfaction (Cocos' 02), p. 28, Valbonne–Sophia Antipolis, France, October 2–4, 2002.
[3] Le Thi Hoai An, Pham Dinh Tao, Large scale molecular optimization from distances matrices by a DC optimization approach, SIAM Journal of Optimization 14 (1) (2003) 77–116.
[4] Le Thi Hoai An, Pham Dinh Tao, The DC (difference of convex functions) Programming and DCA revisited with DC models of real world nonconvex optimization problems, Annals of Operations Research 133 (2005) 23–46.
[5] Le Thi Hoai An, Pham Dinh Tao, Le Dung Muu, Exact Penalty in DC. Programming, Vietnam Journal of Mathematics 2 (2) (1999) 169–178.
[6] R. Bhatnagar, P. Chandra, S.K. Goyal, Models form ulti-plant coordination, European Journal of Operational Research 67 (1993) 141–160.
[7] R.H. Ballou, Business Logistics Management, 3rd ed., Prentice-Hall, 1992.
[8] B.M. Beamon, Supply chain design and Analysis: Models and Methods, International Journal of Production Economics 55 (1998) 281–294.
[9] Satyaveer Singh Chauhan, Chaînes d'approvisionnement: Approches stratégique et tactique, Thèse de Doctorat, Université de Metz, Septembre 2003.
[10] Satyaveer Singh Chauhan, Jean-Marie Proth, Ana Maria Sarmiento, Rakesh Nagi, Strategic Supply Chain Design for a New Market Opportunity from Qualified Partner Sets, INRIA Research Report, RR-4508, July 2002.
[11] M.A. Cohen, H.L. Lee, Strategic analysis of integrated production–distribution systems: Models and methods, Operation Research 36 (1988) 216–228.
[12] M.A. Cohen, H.L. Lee, Resource deployment analysis of global manufacturing and distribution networks, Journal of Manufacturing Operations Management 2 (1989) 81–104.
[13] M.A. Cohen, Al, International manufacturing and distribution networks: A normative model frameworkManaging International Manufacturing, North-Holland, Amsterdam, 1989, pp. 67–93.
[14] A.M. Geoffrion, G.W. Graves, Multicommodity distribution system design by Bender's decomposition, Management Science 2 (5) (1974) 822–844.
[15] M. Govil, J.M. Proth, Supply Chain Design and Management: Strategic and Tactical Perspectives, London Academic Press, 2002.
[16] Pham Dinh Tao, Le Thi Hoai An, Convex analysis approach to DC programming: Theory, algorithms and applications, Acta Mathematica Vietnamica, dedicated to Professor Hoang Tuy on the occasion of his 70th birthday, 22(1) (1997) 289–355.
[17] Pham Dinh Tao, Le Thi Hoai An, DC optimization algorithms for solving the trust region subproblem, SIAM Journal of Optimization 8 (2) (1998) 476–505.
[18] A.M. Sarmiento, R. Nagi, A review of Integrated Analysis of Production–Distribution Systems, IIE Transactions Special Issue on Manufacturing Logistics 11 (1999) 1061–1074.
[19] D.J. Thomas, P.M. Griffin, Coordinated supply chain management, European Journal of Operation Research 94 (1996) 1–15.
[20] J.C. Vidal, M. Goetschalckx, Strategic production–distribution models: A critical review with emphasis on global supply chain models, European Journal of Operation Research 98 (1997) 1–18.

A Brand and Bound method for Multivariate global optimization with box constraints

Le Thi Hoai An[1], Ouanes Mohand[1,2], and Nguyen Trong Phuc[1]

[1] Laboratoire de l'Informatique Théorique et Appliquée.
UFR MIM, Université Paul Verlaine - Metz, Ile du Saulcy, 57045 Metz, France.
lethi@univ-metz.fr; nguyen@univ-metz.fr
[2] Département de Mathématiques.
Faculté des Sciences, Université de Tizi-Ouzou, Algérie.
ouanes_mohand@yahoo.fr

Abstract. The paper is concerned with the multivariate global optimization with box constraints. A new branch and bound method is investigated which is a generalization of the approach developed in [12] for univariate global optimization. In case of nonconvex quadratic programming it is proved that, under some conditions, our lower bound is better than the classical lower bound provided by DC (Difference of Convex functions) programming approach. Numerical results on several test problems are presented which show that the algorithm is more efficient than some standard methods ([6]).
Keywords. Global optimization, branch and bound, quadratic underestimation, w-subdivision.

1 Introduction

We consider the following problem

$$(P)\begin{cases}\alpha := \min f(s)\\ s \in B\end{cases}$$

where B is a box in $\mathbb{R}^n$ and $f : O \subset \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable on an open convex set O containing B. Several methods have been investigated for this problem (see e.g. [5], [6], [7], [9], [10], [21]). Although the constraints are simple, the problem is still very hard in view of finding a global solution.

Basing on the idea developed in [12] for univariate global optimization we propose in this work a convex underestimating function for f and a branch and bound algorithm for solving (P). This bounding procedure suggests us an interesting w-subdivision in the branching which has been shown to be efficient in previous works (see e.g. [11], [17]). Focusing on the quadratic function we prove that, in some cases, this new underestimating function is better than the classical underestimator in DC programming approach.

The paper is organized as follows. In Section 2 we present main new results concerning lower bounding procedure. The case of quadratic programming is discussed in Section 3. The algorithm and its convergence are studied in Section 4 while numerical results are reported in the last section.

2 Main new results

Let $S := [p, q]$ is a bounded closed interval in $\mathbb{R}$ and f is a continuously twice differentiable function on S on which their second derivative is bounded. Let $p \leq a < b \leq q$ and $w_j : \mathbb{R} \to \mathbb{R}$ $(j = 0, 1)$ the functions defined by

$$w_0(s) = \begin{cases} \frac{b-s}{b-a} & \text{if } a \leq x \leq b \\ 0 & \text{otherwise} \end{cases}, \quad w_1(s) = \begin{cases} \frac{s-a}{b-a} & \text{if } a \leq x \leq b \\ 0 & \text{otherwise} \end{cases}. \tag{1}$$

Clearly, $\sum_{i=1}^{1} w_i(s) = 1, \forall s \in [a, b]$ and $w_i(s_j) = 0$ if $i \neq j$, 1, otherwise. Let $L_h f$ be the piecewise linear interpolant to f at points a, b ([3], [4])

$$L_h f(s) = \sum_{i=0}^{1} f(s_i) w_i(s). \tag{2}$$

In [12] we proposed a quadratic underestimator of f on the interval $[a, b]$ which is better than the well-known linear underestimator of f ([3]). It is given by $(h = b - a)$

$$LB(s) = L_h f(s) - \frac{1}{2} K (s - a)(b - s)$$

where K is positive number such that $|f"(s)| \leq K$, $\forall s \in [p, q]$.

We can generalize this result to multivariate global optimization problem (P). Let B be the box $\Pi_{i=1}^{n} \left[s_i^0, s_i^1\right]$ whose vertex set is denoted $V(B)$. An element in $V(B)$ is denoted as $v := (s_1^{i_1}, ..., s_n^{i_n})$ with $i_k = 0$ or 1, for $k = 1, ..., n$. We express (P) in the form

$$(P) \begin{cases} \min f(s_1, ..., s_n) \\ (s_1, ..., s_n) \in B \end{cases}$$

and define the next function $\varphi : \mathbb{R}^n \to \mathbb{R}$ as

$$\varphi(s_1, ..., s_n) = L_{h_n}(...(L_{h_1} f(s_1, ..., s_n))...) - \frac{1}{2} K (\sum_{i=1}^{n} (s_i - s_i^0)(s_i^1 - s_i))$$

$$= \sum_{i_n=0}^{1} (...(\sum_{i_1=0}^{1} f(s_1^{i_1}, ..., s_n^{i_n}) w_{i_1}(s_1))...) w_{i_n}(s_n) - \frac{1}{2} K (\sum_{i=1}^{n} (s_i - s_i^0)(s_i^1 - s_i))$$

where K is a positive number such that $K \geq ||H_f(s_1, ..., s_n)||$, $\forall (s_1, ..., s_n) \in B$ and $H_f(s_1, ..., s_n)$ is the Hessian matrix of the function f. Here the matrix norm of a $A = (a_{ij})$ is defined by

$$||A|| := \max_{i=1,...,n} \sum_{j=1}^{n} |a_{ij}|.$$

Theorem 1. *i) The functions φ and f agree on the vertex set $V(B)$ of B.*

ii) φ is a minorization of f on B, say $\varphi(s) \leq f(s)$, $\forall s \in B$ if $K \geq \max_{s \in B} |f''_{s_i s_i}(s)|$

Proof. i) By the very definition

$$\varphi(s) := \sum_{i_n=0}^{1}(...(\sum_{i_1=0}^{1} f(s_1^{i_1},...,s_n^{i_n})w_{i_1}(s_1))...)w_{i_n}(s_n) - \frac{1}{2}K(\sum_{i=1}^{n}(s_i - s_i^0)(s_i^1 - s_i)).$$

Let $(s_1^{j_1},...,s_n^{j_n})$ be a vertex of B. We have

$$\begin{aligned}&\varphi(s_1^{j_1},...,s_n^{j_n})\\ &= \sum_{i_n=0}^{1}(...(\sum_{i_1=0}^{1} f(s_1^{i_1},...,s_n^{i_n})w_{i_1}(s_1^{j_1}))...)w_{i_n}(s_n^{j_n}) - \frac{1}{2}K(\sum_{i=1}^{n}(s_i^{j_i} - s_i^0)(s_i^1 - s_i^{j_i})).\end{aligned}$$

Clearly,

$$\frac{1}{2}K(\sum_{i=1}^{n}(s_i^{j_i} - s_i^0)(s_i^1 - s_i^{j_i})) = 0. \tag{3}$$

On the other hand, from the definition of w_{i_j} $(i_j = 0, 1)$ it follows that

$$w_0(s_1^0) = w_1(s_1^1) = 1 \text{ and } w_0(s_1^1) = w_1(s_1^0) = 0,$$

therefore $\sum_{i_1=0}^{1} f(s_1^{i_1},...,s_n^{i_n})w_{i_1}(s_1^{j_1}) = f(s_1^{j_1},...,s_n^{i_n})$.
Likewise $\sum_{i_2=0}^{1} f(s_1^{j_1},...,s_n^{i_n})w_{i_2}(s_2^{j_2}) = f(s_1^{j_1},...,s_n^{i_n})$,
and so on $\sum_{i_n=0}^{1} f(s_1^{j_1},...,s_n^{i_n})w_{i_n}(s_n^{j_n}) = f(s_1^{j_1},...,s_n^{j_n})$.
Hence $\varphi(s_1^{j_1},...,s_n^{j_n}) = f(s_1^{j_1},...,s_n^{j_n})$.

ii) We have

$$\begin{aligned}&|f(s_1,s_2,...,s_n) - \sum_{i_n=0}^{1}(...(\sum_{i_1=0}^{1} f(s_1^{i_1},s_2^{i_2},...,s_n^{i_n})w_{i_1}(s_1))...)w_{i_n}(s_n)|\\ &= |(f(s_1,s_2,...,s_n) - \sum_{i_1=0}^{1} f(s_1^{i_1},s_2,...,s_n)w_{i_1}(s_1)) +\\ &(\sum_{i_1=0}^{1} f(s_1^{i_1},s_2,...,s_n)w_{i_1}(s_1) - \sum_{i_2=0}^{1}(\sum_{i_1=0}^{1} f(s_1^{i_1},s_2^{i_2},...,s_n)w_{i_1}(s_1))w_{i_2}(s_2)) +\\ &(\sum_{i_2=0}^{1}(\sum_{i_1=0}^{1} f(s_1^{i_1},s_2^{i_2},...,s_n)w_{i_1}(s_1))w_{i_2}(s_2) - ...) + ... +\\ &(\sum_{i_{n-1}=0}^{1}(...(\sum_{i_1=0}^{1} f(s_1^{i_1},s_2^{i_2},...,s_{n-1}^{i_{n-1}},s_n)w_{i_1}(s_1))...)w_{i_{n-1}}(s_{n-1}) -\\ &\sum_{i_n=0}^{1}(...(\sum_{i_1=0}^{1} f(s_1^{i_1},s_2^{i_2},...,s_n^{i_n})w_{i_1}(s_1))...)w_{i_n}(s_n))|\end{aligned}$$

$$
\begin{aligned}
&\leq |(f(s_1,s_2,...,s_n)-\sum_{i_1=0}^{1} f(s_1^{i_1},s_2,...,s_n)w_{i_1}(s_1))| + \\
&|\sum_{i_1=0}^{1} f(s_1^{i_1},s_2,...,s_n)w_{i_1}(s_1)-\sum_{i_2=0}^{1}(\sum_{i_1=0}^{1} f(s_1^{i_1},s_2^{i_2},...,s_n)w_{i_1}(s_1))w_{i_2}(s_2)| + \\
&|\sum_{i_2=0}^{1}(\sum_{i_1=0}^{1} f(s_1^{i_1},s_2^{i_2},...,s_n)w_{i_1}(s_1))w_{i_2}(s_2)-...| + ... + \\
&|\sum_{i_{n-1}=0}^{1}(...(\sum_{i_1=0}^{1} f(s_1^{i_1},s_2^{i_2},...,s_{n-1}^{i_{n-1}},s_n)w_{i_1}(s_1))...)w_{i_{n-1}}(s_{n-1}) - \\
&\sum_{i_n=0}^{1}(...(\sum_{i_1=0}^{1} f(s_1^{i_1},s_2^{i_2},...,s_n^{i_n})w_{i_1}(s_1))...)w_{i_n}(s_n))| \\
&\leq \frac{1}{2}K(s_1-s_1^0)(s_1^1-s_1)+\frac{1}{2}K(s_2-s_2^0)(s_2^1-s_2)+...+\frac{1}{2}K(s_n-s_n^0)(s_n^1-s_n) \\
&=\frac{1}{2}K(\sum_{i=1}^{n}(s_i-s_i^0)(s_i^1-s_i))
\end{aligned}
$$

Thus $\varphi(s)\leq f(s)\ ,\ \forall s\in B$. As an immediate consequence of this theorem we have

$$
\begin{aligned}
&|\sum_{i_n=0}^{1}(...(\sum_{i_1=0}^{1} f(s_1^{i_1},...,s_n^{i_n})w_{i_1}(s_1))...)w_{i_n}(s_n)-f(s_1,...,s_n)| \\
&\qquad\leq \frac{1}{2}K\left(h_1^2+...+h_n^2\right) \quad with \quad h_i=s_i^1-s_i^0. \qquad (4)
\end{aligned}
$$

Theorem 2. *The function φ is convex on B if*

$$K\geq \max_{s\in B}\max_{i=1,...,n}\sum_{j=1,j\neq i}^{n}|f''_{s_is_j}(s)|.$$

Proof. We express φ in the form

$$
\begin{aligned}
\varphi(s)&=\sum_{i_n=0}^{1}(...(\sum_{i_1=0}^{1} f(s_1^{i_1},...,s_n^{i_n})w_{i_1}(s_1))...)w_{i_n}(s_n)-\frac{1}{2}K(\sum_{i=1}^{n}(s_i-s_i^0)(s_i^1-s_i)) \\
&=\sum_{i_n=0}^{1}(...(\sum_{i_1=0}^{1} f(s_1^{i_1},...,s_n^{i_n})w_{i_1}(s_1))...)w_{i_n}(s_n) \\
&\quad+\frac{1}{2}Ks_i^2-\frac{1}{2}K\sum_{i=1}^{n}((s_i^1+s_i^0)s_i-s_i^0s_i^1).
\end{aligned}
$$

Since the part

$$\frac{1}{2}K\sum_{i=1}^{n}((s_i^1+s_i^0)s_i-s_i^0s_i^1)$$

is linear, it suffices to prove that the function Φ defined by

$$\Phi(s_1,...,s_n) := \sum_{i_n=0}^{1}(...(\sum_{i_1=0}^{1} f(s_1^{i_1},...,s_n^{i_n})w_{i_1}(s_1))...)w_{i_n}(s_n) + \frac{1}{2}Ks_i^2$$

is convex. This amounts to show that the Hessian matrix of Φ, denoted H_Φ, is semi-definite positive. Let

$$L_h f(s_1,...,s_n) := \sum_{i_n=0}^{1}(...(\sum_{i_1=0}^{1} f(s_1^{i_1},...,s_n^{i_n})w_{i_1}(s_1))...)w_{i_n}(s_n).$$

From the definition of w_{i_j}, it is easy to see that all elements $(L_h f)''_{s_i s_i}$ are zero, for $i = 1,...,n$. Hence H_Φ takes the form

$$H_\Phi = \begin{pmatrix} K & L_{12} & . & . & L_{1n} \\ L_{21} & K & L_{23} & . & L_{2n} \\ . & . & . & . & . \\ . & . & . & . & . \\ L_{n1} & . & . & . & K \end{pmatrix}$$

with $L_{ij} := (L_h f)''_{s_i s_j}(s_1,...,s_n)$ is the second mixed derivatives of $L_h f$ with respect to the variables s_i and s_j.

The second mixed derivatives of $L_h f$, for example the second mixed derivative with respect to variables s_1 and s_2, can be computed as follows. First, the derivative with respect to one variable, for example, s_1, can be expressed as

$$\begin{aligned}
&L_h f'_{s_1}(s_1,...,s_n) = (\sum_{i_n=0}^{1}(...(\sum_{i_1=0}^{1} f(s_1^{i_1},...,s_n^{i_n})w_{i_1}(s_1))...)w_{i_n}(s_n))'_{s_1} \\
&= \sum_{i_n=0}^{1}(...(\sum_{i_2=0}^{1}(f(s_1^0,s_2^{i_2},...,s_n^{i_n})w'_0(s_1) + f(s_1^1,s_2^{i_2},...,s_n^{i_n})w'_1(s_1))w_{i_2}(s_2))...)w_{i_n}(s_n) \\
&= \sum_{i_n=0}^{1}(...(\sum_{i_2=0}^{1}(f(s_1^0,s_2^{i_2},...,s_n^{i_n})\frac{-1}{s_1^1-s_1^0} + f(s_1^1,s_2^{i_2},...,s_n^{i_n})\frac{1}{s_1^1-s_1^0})w_{i_2}(s_2))...)w_{i_n}(s_n) \\
&= \sum_{i_n=0}^{1}(...(\sum_{i_2=0}^{1}(\frac{f(s_1^1,s_2^{i_2},...,s_n^{i_n}) - (f(s_1^0,s_2^{i_2},...,s_n^{i_n})}{s_1^1-s_1^0})w_{i_2}(s_2))...)w_{i_n}(s_n) \\
&= \sum_{i_n=0}^{1}(...(\sum_{i_2=0}^{1}(f'_{s_1}(\xi,s_2^{i_2},...,s_n^{i_n}))w_{i_2}(s_2))...)w_{i_n}(s_n) \quad with \quad s_1^0 < \xi < s_1^1.
\end{aligned}$$

Afterwards the second derivative with respect to the first two variables is computed as:

$$L_h f''_{s_1 s_2}(s_1,...,s_n) \; = \sum_{i_n=0}^{1}(...(\sum_{i_2=0}^{1}(f'_{s_1}(\xi,s_2^{i_2},...,s_n^{i_n}))w_{i_2}(s_2))...)w_{i_n}(s_n))'_{s_2}$$

$$
\begin{aligned}
&= \sum_{i_n=0}^{1}(...(\sum_{i_3=0}^{1}(f'_{s_1}(\xi, s_2^0, s_3^{i_3}, ..., s_n^{i_n})w'_0(s_2) \\
&\quad + f'_{s_1}(\xi, s_1^1, s_2^{i_2}, ..., s_n^{i_n})w'_1(s_2))w_{i_3}(s_3))...)w_{i_n}(s_n) \\
&= \sum_{i_n=0}^{1}(...(\sum_{i_3=0}^{1}(f'_{s_1}(\xi, s_2^0, s_3^{i_3}, ..., s_n^{i_n})\frac{-1}{s_2^1 - s_2^0} \\
&\quad + f'_{s_1}(\xi, s_1^1, s_2^{i_2}, ..., s_n^{i_n})\frac{1}{s_2^1 - s_2^0})w_{i_3}(s_3))...)w_{i_n}(s_n) \\
&= \sum_{i_n=0}^{1}(...(\sum_{i_3=0}^{1}(\frac{f'_{s_1}(\xi, s_1^1, s_2^{i_2}, ..., s_n^{i_n}) - f'_{s_1}(\xi, s_2^0, s_3^{i_3}, ..., s_n^{i_n})}{s_2^1 - s_2^0})w_{i_3}(s_3))...)w_{i_n}(s_n) \\
&= \sum_{i_n=0}^{1}(...(\sum_{i_3=0}^{1}(f''_{s_1 s_2}(\xi, \eta, s_3^{i_3}, ..., s_n^{i_n}))w_{i_3}(s_3))...)w_{i_n}(s_n)
\end{aligned}
$$

with $s_1^0 < \xi < s_1^1$ and $s_2^0 < \eta < s_2^1$.

Since $w_0(x) + w_1(x) = 1$ for all $x \in \left[s_i^0, s_i^1\right], i = 1, ..., n$ we have

$$\sum_{i_3=0}^{1} f''_{s_1 s_2}(\xi, \eta, s_3^{i_3}, ..., s_n^{i_n})w_{i_3}(s_3) \leq \max_{s_3 \in \left[s_3^0, s_3^1\right]} f''_{s_1 s_2}(\xi, \eta, s_3, ..., s_n^{i_n}),$$

and so on

$$\sum_{i_n=0}^{1}(...(\sum_{i_3=0}^{1} f''_{s_1 s_2}(\xi, \eta, s_3^{i_3}, ..., s_n^{i_n})w_{i_3}(s_3))...)w_{i_n}(s_n)) \leq \max_{(s_1,...,s_n)\in B} f''_{s_1 s_2}(s_1, ..., s_n).$$

Hence

$$|L_h f''_{s_1 s_2}(s_1, ..., s_n)| \leq \max_{(s_1,...,s_n)\in B} |f''_{s_1 s_2}(s_1, ..., s_n)|.$$

Likewise, for any pair $i \neq j$ we have

$$\left|(L_h f)''_{s_i s_j}\right| \leq \max_{(s_1,...,s_n)\in B} |f''_{s_i s_j}(s_1, ..., s_n)|.$$

Consequently, if

$$K \geq \max_{s\in B} \max_{i=1,...,n} \sum_{j=1, j\neq i}^{n} |f''_{s_i s_j}(s)|,$$

then

$$K \geq \max_{s\in B} \max_{i=1,...,n} \sum_{j=1, j\neq i}^{n} |L_h f''_{s_i sj}(s_1, ..., s_n)|$$

and therefore H_Φ is semi-definite positive. □

Notes and Comments. The inequality $K \geq \max_{s\in B} \max_{i=1,...,n} \sum_{j=1, j\neq i}^{n} |f''_{s_i s_j}(s)|$ implies that $K \geq \max_{s\in B} |f''_{s_i s_i}(s)|$, a sufficient condition for φ to be a underestimator of f (see Theorem 1).

3 Underestimator of quadratic functions

Nonconvex quadratic programs constitute an important and difficult class of problems in global optimization. Many applications in operations research and engineering sciences can be formulated as nonconvex quadratic programs. In this section we are interested in the case where f is a quadratic function

$$f(s) = \frac{1}{2}s^T As + c^T s$$

with $A = (a_{ij})_{1\leq i\ ,j\ \leq n}$ being a symmetric real $n \times n$ matrix and $c \in \mathbb{R}^n$.

Theorem 3. *If the function f is quadratic, then the function φ is quadratic too.*

Proof. By the very definition

$$\varphi(s) := L_h f(s_1, ..., s_n) - \frac{1}{2}K(\sum_{i=1}^{n}(s_i - s_i^0)(s_i^1 - s_i).$$

Since the function $\frac{1}{2}K\sum_{i=1}^{n}(s_i - s_i^0)(s_i^1 - s_i)$ is quadratic, it suffices to prove that $L_h f$ is quadratic. Indeed, since $w_{i_j}(.)$ are linear, we have

$$\begin{aligned}(L_h f(s))''_{s_1 s_2} &= \sum_{i_n=0}^{1}(...(\sum_{i_3=0}^{1} f''_{s_1 s_2}(\xi, \eta, s_3^{i_3}, ..., s_n^{i_n})w_{i_3}(s_1))...)w_{i_n}(s_n)) \\ &= \sum_{i_n=0}^{1}(...(\sum_{i_3=0}^{1} a_{12}w_{i_3}(s_1))...)w_{i_n}(s_n)) = a_{12}.\end{aligned}$$

On the other hand, as seen above, all elements on the diagonal of $H_{L_h f}$ are zero, hence $H_{L_h f}$ and H_φ take the form

$$H_{L_h f} = \begin{pmatrix} 0 & a_{12} & . & . & a_{1n} \\ a_{21} & 0 & a_{23} & . & a_{2n} \\ . & . & . & . & . \\ . & . & . & . & . \\ a_{n1} & . & . & . & 0 \end{pmatrix}, H_\varphi = \begin{pmatrix} K & a_{12} & . & . & a_{1n} \\ a_{21} & K & a_{23} & . & a_{2n} \\ . & . & . & . & . \\ . & . & . & . & . \\ a_{n1} & . & . & . & K \end{pmatrix}.$$

Therefore φ is a quadratic function. □

Let D be the diagonal matrix with $D_{ii} := a_{ii}$ and I be the $(n \times n)$ identity matrix. φ can be expressed as

$$\varphi(s) = \frac{1}{2}s^T(A - D + KI)s + \sum_{i=1}^{n}\alpha_i s_i + \alpha_0$$

with $\alpha_i \in \mathbb{R}$. For determining $(\alpha_i)_{i=0,1,...,n}$ we consider $n+1$ vertices $v^1, ..., v^n$ of B and solve the next linear system

$$\begin{cases} \varphi(v^1) = f(v^1) \\ \varphi(v^2) = f(v^2) \\ \quad ... \\ \varphi(v^{n+1}) = f(v^{n+1}) \end{cases}. \tag{5}$$

Notes and Comments. Computing the underestimator φ of a quadratic function f on B is particularly simple:

i) it suffices to take $K \geq \max\{\max_{i=1,\ldots,n}|a_{ii}|, \lambda_1(H_{L_h f})\}$ to obtain the convex underestimating function φ of f, where $\lambda_1(.)$ denotes the smallest eigenvalue of the matrix (.). From the numerical point of view, smaller K is, more efficient the algorithm is.

ii) For computing the underestimator φ of f on B we need only $n+1$ vertex of B (while in general case all vertices are to be considered).

3.1 Comparison with the classical underestimator of quadratic function by DC programming approach

It is well known that a quadratic function f can be expressed as a Difference of Convex function (DC), and the following underestimation of f has been proposed in [11], [17]:

$$LB_{dc}(s) = \frac{1}{2}s^T(A+\rho I)s - \frac{1}{2}\rho\sum_{i=1}^{n}((s_i^1+s_i^0)s_i - s_i^1 s_i^0) \quad where \quad \rho \geq -\lambda_1(A).$$

Theorem 4. *If* $min_{i=1,\ldots,n}a_{ii} \geq K-\rho$, *then* $\varphi(s) \geq LB_{dc}(s)$, $\forall s \in B$.

Proof. Let

$$\begin{aligned}\psi(s) \ : = LB_{dc}(s) - \varphi(s) &= \frac{1}{2}s^T(A+\rho I)s - \frac{1}{2}\rho\sum_{i=1}^{n}((s_i^1+s_i^0)s_i - s_i^1 s_i^0) \\ &\quad - \frac{1}{2}s^T(A-D+KI)s + \sum_{i=1}^{n}\alpha_i s_i + \alpha_0 \\ &= \frac{1}{2}s^T(\rho I + D - KI)s - \frac{1}{2}\rho\sum_{i=1}^{n}((s_i^1+s_i^0)s_i - s_i^1 s_i^0) - \sum_{i=1}^{n}\alpha_i s_i + \alpha_0.\end{aligned}$$

Clearly that $\psi(s) = 0, \ \forall s \in V(B)$. If $\min_{i=1,\ldots n}a_{ii} \geq K-\rho$, then the matrix $\rho I + D - KI$ is semi definite positive, consequently ψ is convex. In this case

$$\max\{\psi(s): \quad s \in B\} = \max\{\psi(s): \quad s \in V(B)\} = 0$$

which implies $LB_{dc}(s) \leq \varphi(s); \forall s \in B$. □

4 A Branch and Bound algorithm and its convergence

We can now describe the branch and bound algorithm for solving (P). Denote by, LB_k, UB_k and s^k respectively the best lower bound, the best upper bound of α and the best solution to (P) at iteration k.

Algorithm BB:

– **Initialization:** Let ε be a given sufficiently small number. Compute K, a upper bound of $\|H_f\|$ on B. Set $T^0 = B; h_i^0 = s_i^1 - s_i^0$. Solve the convex program

$$\min\left\{\varphi(s) : s \in T^k\right\} \tag{6}$$

to obtain an optimal solution $\widetilde{\mathbf{s}}^0$.
Set $UB_0 := \min\left\{\min_{v\in V(B)} f(v), f(s^*)\right\}$, $LB(T^0) := \varphi(\widetilde{\mathbf{s}}^0)$.
If $UB_0 - LB_0 \le \varepsilon$ then STOP $\widetilde{\mathbf{s}}^0$ is an ε - optimal solution,
else set $M \leftarrow \{T^0\}$, $k \leftarrow 1$, and go to iteration k.

– **Iteration k**

k1. Let $T^k = \prod_{k=1}^n [a_k, b_k] \in M$ be the rectangle such that $LB_k = LB(T^k)$,and $\widetilde{\mathbf{s}}^k$ be the solution of Problem (6).
Set K_k such that $\|H_f(s)\| \le K_k$, $\forall s \in T^k$.

k2. Bisect T^k into two subrectangles T^{k1}, T^{k2} by w-subdivision procedure via $\widetilde{\mathbf{s}}^k$.

k3. For $i = 1, 2$ do
Solve the convex program (6) where k is replaced by k_i to obtain an optimal solution $\widetilde{\mathbf{s}}^{ki}$.

k4. Update the upper bound $UB_k = \min\left\{UB_k, f(\widetilde{\mathbf{s}}^{k1}), f(\widetilde{\mathbf{s}}^{k2})\right\}$. Let s^k be the best current solution, i.e. $f(s^k) = UB_k$.
Set $M \leftarrow M \cup \{T_i^k : LB(T_i^k) < UB^k - \varepsilon, i = 1, 2\}\backslash\{T^k\}$.
Update the lower bound: $LB_k = \min\{LB(T) : T \in M\}$.

k5. If $M = \emptyset$ then STOP, s^k is an optimal solution.
else set $k \leftarrow k + 1$, and return to **k1**.

Theorem 5. *[Convergence of the algorithms]*
Either the algorithm is finite or it generates a bounded sequence $\{s^k\}$, every accumulation point of which is a global optimal solution of (P), and

$$UB_k \searrow \alpha, \quad LB_k \nearrow \alpha.$$

Proof. We have

$$\begin{aligned}
&\min f(s_1^{i_1}, ..., s_n^{i_n}) - \frac{1}{2}K\left(h_{1k}^2 + ... + h_{nk}^2\right)\\
&\le \min \sum_{i_n=0}^{1}(...(\sum_{i_1=0}^{1} f(s_1^{i_1}, ..., s_n^{i_n})w_{i_1}(s_1))...)w_{i_n}(s_n) - \frac{1}{2}K(\sum_{i=1}^{n}(s_i - s_i^0)(s_i^1 - s_i))\\
&\le \min f(s_1, ..., s_n) \le \min f(s_1^{i_1}, ..., s_n^{i_n}) + \frac{1}{2}K\left(h_{1k}^2 + ... + h_{nk}^2\right)
\end{aligned}$$

Let $\quad LB_k^1 = \min f(s_1^{i_1}, ..., s_n^{i_n}) - \frac{1}{2}K\left(h_{1k}^2 + ... + h_{nk}^2\right)$

$$LB_k = \min \sum_{i_n=0}^{1}(...(\sum_{i_1=0}^{1} f(s_1^{i_1}, ..., s_n^{i_n})w_{i_1}(s_1))...)w_{i_n}(s_n) - \frac{1}{2}K(\sum_{i=1}^{n}(s_i - s_i^0)(s_i^1 - s_i))$$

$$UB_k^1 = \min f(s_1^{i_1}, ..., s_n^{i_n}) + \frac{1}{2}K\left(h_{1k}^2 + ... + h_{nk}^2\right) \quad \textit{and} \quad UB_k = \min f(\, s_1^{i_1}, ..., s_n^{i_n})$$

with $h_{jk} = s^1_{jk} - s^0_{jk}$; for $j = 1, ..., n$. Then

$$0 \leq \lim_{k\to\infty} (UB_k - LB_k) \leq \lim_{k\to\infty} \left(UB^1_k - LB^1_k\right) \leq \lim_{k\to\infty} K\left(h^2_{1k} + ... + h^2_{nk}\right) = 0$$

that implies $\lim_{k\to\infty} UB_k - LB_k = 0$.
Moreover, since $s^k \in T^0$ and $UB_k = f(s^k)$, any cluster point of the sequence $\{s^k\}$ belongs to T^0 and has the function value α, i.e., solves problem (P). The theorem is proved. □

5 Computational results

The algorithm was implemented in C++ with double precision and run on a PC Inter Pentium 4, 3.0 GHz, 1 Go of RAM. For solving the subproblem (6) we used the subgradient projection method [16]. In this experiment, we compare the efficiency of our BB algorithm with the algorithm developed in [6] for Hölder test functions in term of number of function evaluations. These test problems are in two-dimensions, then at each iteration, we have to evaluate 5 times the objective function (four times at four vertex of each rectangle and one time while computing upper bound). In Table 1 we describe the expression of the objective function, the interval $[a, b]$ of test problems given in [6]. We also indicate here the value of K used in our algorithm. The results of the BB algorithm are summarized in Table 2 with different value of the tolerance ϵ: $\epsilon = 10^{-3}$ (right), and ϵ considered in [6] (left). In Table 3 we report the comparative results of the BB algorithm and the HOL^n algorithm developed in [6].

We use the following notations:

- Iter: the number of iterations of each algorithm.
- F*opt*, LB: the last upper bound, the last lower bound given by the algorithm.
- CPU : the computing time in seconds.
- N^oF_i, for $i = 1, 2, 3$): the number of function evaluations of the HOL^n algorithm in three cases with different Lipschitz constant values (see [6]).
- N^oF_{ave}: the average of the numbers of function evaluations in three cases of the HOL^n algorithm
- N^oF: the number of function evaluations of the BB algorithm.

Conclusion. We have proposed in this paper a new convex underestimation for twice differentiable functions on a box and a branch and bound method for minimizing such functions on box constraints. In case of quadratic functions, our underestimating function is also quadratic and can be easily determined. Moreover, under some conditions, this quadratic underestimation is better than a classical lower bounding in DC programming approach. Preliminary computational results in bidimensional case show the superiority of our algorithm compared with some efficient existing algorithms. They suggest to us extending the numerical experiments in higher dimension, and exploiting the efficiency of the quadratic underestimation to develop the branch and bound framework for box constrained quadratic programming Work in these directions is currently in progress.

Table 1. Holder test functions and the K value.

Problem	Function(**min**)	Interval$[a,b]$	K
1	$-sin(x)sin(xy)$	$[0,4]$x$[0,4]$	12
2	$-sin(2x+y)/(sin(y)+2)$	$[-5,5]$x$[-5,5]$	5
3	$sin(x+y)+(x-y)^2-1.5x+2.5y+1$	$[-1.5,4]$x$[-3,3]$	100
4	$-sin((x-1)(x-2)(y+1))$	$[-1,1]$x$[-2,0]$	18
5	$(x-2)^2+(y-1)^2+$ $0.04/(1-x^2/4-y^2)+(x-2y+1)^2/0.2$	$[1,2]$x$[1,2]$	80
6	$(y-5x^2/(4\pi^2)+5x/\pi-6)^2+$ $10(1-1/(8\pi))cos(x)+10$	$[-5,10]$x$[0,15]$	40
7	$100(y-x^2)^2+(x-1)^2$	$[-3,3]$x$[-1.5,4.5]$	6000
8	$0.1(12+x^2+(1+y^2)/x^2)+$ $(x^2y^2+100)/(x^4y^4)$	$[1,3]$x$[1,3]$	100
9	$0.5(x^2+y^2)-cos(10ln(2x))cos(10ln(3y))+1$	$[0.01,1]$x$[0.01,1]$	800000
10	$(1+(x+y+1)^2(19-14x+3x^2-14y+6xy+3y^2))$ $(30+(2x-3y)^2(18-32x+12x^2+48y-36xy+27y^2))$	$[-2,2]$x$[-2,2]$	80000

Table 2. Computational results for Holder test functions

Problem	ϵ-selected						ϵ=1e-03				
	Fopt	LB	ϵ	N^oIter	N^oF	T_{cpu}	Fopt	LB	N^oIter	N^oF	T_{cpu}
1	-0.999999	-1.000010	5.0e-05	222	889	0.000	-0.999953	-1.001022	162	649	0.000
2	-0.999952	-1.000168	2.0e-04	227	909	0.000	-0.999952	-1.001328	224	897	0.000
3	-1.913223	-1.913223	1.0e-08	1698	6793	0.015	-1.913221	-1.914226	645	2581	0.000
4	-1.000000	-1.002037	2.0e-03	135	541	0.000	-1.000000	-1.001033	182	729	0.000
5	0.169043	0.169043	1.0e-08	201	805	0.000	0.169055	0.167986	62	249	0.000
6	0.399250	0.379702	2.0e-02	250	1001	0.000	0.397899	0.396868	306	1225	0.000
7	0.000000	-0.000003	3.0e-06	4406	17625	0.093	0.000000	-0.001011	2411	9645	0.046
8	1.744158	1.728338	9.0e-03	547	2189	0.000	1.744155	1.742410	1642	6569	0.016
9	———	———	1.0e-04	—	—	—	———	———	$\geq$20000	—	$\geq$10.0
10	3.000003	2.998501	5.0e-04	3103	12201	0.059	3.000003	2.996983	2822	10689	0.062

Table 3. Compare the efficiency of our BB algorithm with BB HOL^n Algorithm.

Problem	$BB-HOL^n$				OUR $BB-Algorithm$
	N^oF_1	N^oF_2	N^oF_3	N^oF_{ave}	N^oF
1	1013	2527	6101	**3213**	**889**
2	1089	2939	6775	**3601**	**909**
3	1123	2333	5215	**2890**	**6793**
4	1821	3123	5541	**3495**	**541**
5	2673	4591	7355	**4873**	**805**
6	3519	6133	9735	**6462**	**1001**
7	7967	11249	14301	**11172**	**17625**
8	12643	18253	19133	**16676**	**2189**
9	15687	19683	19683	**18351**	**20000**
10	13945	16139	17771	**15951**	**12201**
Average	**6148**	**8697**	**11161**	**8668.6**	**6295.3**

References

1. W. Baritompa and A.Cutler (1994), *Accelerations for global optimization covering methods using second derivates*, Journal of Global optimization, 4:329-341
2. E.Baumann (1988), *Optimal centered forms,* BIT, 28:80-87.
3. C de Boor, *A practical Guide to Splines Applied Mathematical Sciences*, Springer Verlag (1978).
4. Ciarlet P.G., *The Finite Element Method for Elliptic Problems,* Studies in Math.and its Appl.,1979.
5. Gergel, V.P., *A Global Optimization Algorithm for Multivariate functions with Lipschitzian First Derivatives,* Journal .of Global Optimization, Vol.10, N° 3, (1997), pp. 257-281.
6. Gourdin, E., Jaumard, B., and Ellaia, R., *Global Optimization of Hölder Functions,* Journal of Global Optimization, Vol.8, N° 4, (1996), pp. 323-348.
7. Hansen, E., *Global optimization using interval analysis the multi-dimensional case.* Numer. Math., 34:247270, 1980.
8. Hansen, E., *Global Optimization using Interval Analysis,* volume 165 of Pure and Applied Mathematics, Marcel Dekker,New York, 1992.
9. Horst, R. and Pardalos, P.M. (1995), *Handbook of Global Optimization,* Kluwer Academic Publishers, Dordrecht.
10. Horst, R. and Tuy, H. (1993), *Global Optimization - Deterministic Approaches*, Springer-Verlag, Berlin.
11. Le Thi Hoai An and Pham Dinh Tao, *A Branch-and-Bound method via D.C. Optimization Algorithm and Ellipsoidal technique for Box Constrained Nonconvex Quadratic Programming Problems*, Journal of Global Optimization, **13** (1998), pp. 171-206.
12. Le Thi Hoai An and Ouanes, M., *Convex quadratic underestimation and Branch and Bound for univariate global optimization with one nonconvex constraint,* revised version in RAIRO Recherche Opérationnelle.
13. Messine, F. and Lagouanelle, J. (1998), Enclosure methods for multivariate differentiable functions and application to global optimization, Journal of Universal Computer Science 4(6): 589-603.
14. Moore, R., *Interval Analysis,* Prentice-Hall, Englewood Cliffs, New Jersey, 1966.
15. Pardalos, P. M., and Romeijn, H. E.: *Handbook of Global Optimization* Volume 2. Nonconvex Optimization and Its Applications, Springer, Boston/Dordrecht/London (2002).
16. B. T. Polyak, *Introduction to optimization,* Inc., Publications Division, 1987.
17. Thai Quynh Phong, Le Thi Hoai An and Pham Dinh Tao, *On the global solution of linearly constrained indefinite quadratic minimization problems by decomposition branch and bound method.* RAIRO, Recherche Opérationnelle, **30 (1)** (1996), 31-49.
18. Pijavskii S.A. (1972), *An Algorithm for Finding the Absolute Extremum of a Function,* USSR, Comput. Math. and Math. Physics, 12: 57-67.
19. Ratschek, H. and J. Rokne, J., *New Computer Methods for Global Optimization,* Wiley, New York, 1982.
20. Rinnooy, A.H.G. and Timmer, G.H., *Global Optimization: A Survey,* In: Nemhauser G.L. et al. eds., Handbooks of Operations Research, North-Holland, Elsevier Science Publishers (1989), pp. 631-662.
21. Xiaojun Wang, Tsu-Shuan Chang: *A Multivariate Global Optimization Using Linear Bounding Functions.* Journal of Global Optimization. **12** (1998) 383-404.

Bibliographie

[1] H.A Le Thi *Analyse numérique des algorithmes de l'optimisation DC. Approches locale et globale. Codes et simulations numériques en grande dimension. Applications.* Thèse de Doctorat de l'Université de Rouen, 1994.

[2] H.A Le Thi *Contribution à l'optimisation non convexe et l'optimisation globale : Théorie, Algorithmes et Applications* Habilitation à Diriger des Recherches, Université de Rouen, 1997.

[3] H.A Le Thi *An efficient algorithm for globally minimizing a quadratic function under convex quadratic constraints* Mathematical Programming, Ser. A, Vol.87, N°.3, 401–426, 2000.

[4] H.A Le Thi *Solving large scale Molecular distance geometry problem by a smoothing technique via the Gaussian transform an DC programming* Journal of Global Optimization, Vol.27, 375–397, 2003.

[5] H.A Le Thi and T. Pham Dinh *Solving a class of linearly constrained indefinite quadratic problems by DC algorithms* Journal of Global Optimization, Vol.11, 253–285, 1997.

[6] H.A Le Thi and T. Pham Dinh *A Branch-and-Bound method via DC Optimization Algorithm and Ellipsoidal techniques for Box Constrained Nonconvex Quadratic Programming Problems* Journal of Global Optimization, Vol.13, 171–206, 1998.

[7] H.A Le Thi and T. Pham Dinh *DC programming approach for large-scale molecular optimization via the general distance geometry problem* Nonconvex Optimization and Its Applications 40, in Optimization in Computational Chemistry and Molecular Biology : Local and Global Approaches, Kluwer Academic Publishers, 301–339, 2000.

[8] H.A Le Thi and T. Pham Dinh *Large Scale Molecular Conformation via the Exact Distance Geometry Problem* In Optimization, Lecture Notes in Economics and Mathematical Systems, Vol.481, Heidelberg, Springer-Verlag, 260-277, 2000.

[9] H.A Le Thi and T. Pham Dinh *A continuous Approach for Globally Solving Linearly Constrained Quadratic Zero - One Programming Problems* Optimization, Vol.50, 93–120, 2001.

[10] H.A Le Thi and T. Pham Dinh *DC optimization approaches via Markov models for restoration of signals (1-D) and (2-D)* Nonconvex Optimization and Its Applications 54 :

In Advances in Convex Analysis and Global Optimization, Kluwer Academic Publishers, 300–317, 2001.

[11] H.A Le Thi and T. Pham Dinh *DC programming approach and solution algorithm to the multidimensional scaling problem* Nonconvex Optimization and Its Applications 53 : In From Local to Global Optimization, Kluwer Academic Publishers, 231–276, 2001.

[12] H.A Le Thi and T. Pham Dinh *DC Programming Approach for Multicommodity Network Optimization Problems with Step Increasing Cost Functions* Special Issue of Journal of Global Optimization (dedicated to Professor R. Horst on the occasion of his 60 th birthday), Vol.22, 204–233, 2002.

[13] H.A Le Thi and T. Pham Dinh *Dc Programming. Theory, Algorithms, Applications : The State of the Art* First InternationalWorkshop on Global Constrained Optimization and Constraint Satisfaction, October 2-4, 2002, Valbonne-Sophia Antipolis, France, Research Report, Laboratory of Modeling, Optimization & Operations Research, Insa-Rouen, France, 2002.

[14] H.A Le Thi and T. Pham Dinh *Large scale molecular optimization from distances matrices by a DC optimization approach* SIAM Journal of Optimization, Vol.14, N°.1, 77–116, 2003.

[15] H.A Le Thi and T. Pham Dinh *A new algorithm for solving large scale molecular distance geometry problems, Applied Optimization : in Hight Performance Algorithms and Software for Nonlinear Optimization* Kluwer Academic Publishers, 276–296, 2003.

[16] H.A Le Thi and T. Pham Dinh *The DC (Difference of Convex functions) Programming and DCA Revisited with DC Models of Real World Nonconvex Optimization Problems* Annals of Operations Research, Vol.133, 23–46, 2005.

[17] H.A Le Thi and T. Pham Dinh *A continuous approach for the concave cost supply problem via DC Programming and DCA* to appear in Discrete Applied Mathematics.

[18] H.A Le Thi, T. Belghiti and T. Pham Dinh *A new efficient algorithm based on DC programming and DCA for Clustering* In Press, Available July 2006, Journal of Global Optimization.

[19] H.A Le Thi, M. Le Hoai and T. Pham Dinh *Optimization based DC programming and DCA for Hierarchical Clustering* In Press, Available Online June 2006, European Journal of Operational Research.

[20] H.A Le Thi, M. Le Hoai and T. Pham Dinh *Une nouvelle approche basée sur la programmation DC et DCA pour la classification floue* EGC 2007, RNTI, 703–714, 2007.

[21] H.A Le Thi, T.P Nguyen and T. Pham Dinh *A Continuous DC Programming Approach To The Strategic Supply Chain Design Problem From Qualified Partner Set* In Press, Available Online June 2006, European Journal of Operational Research.

[22] H.A Le Thi, M. Le Hoai, T.P Nguyen and T. Pham Dinh *Noisy Image Segmentation by Fuzzy C-Means Clustering based DCA* Submitted, 2007.

[23] H.A Le Thi, T. Pham Dinh and H. Dinh Nho *Towards Tikhonov regularization of nonlinear ill-posed problems : a DC programming approach* C.R. Acad. Sci. Paris, Ser. I, Vol.335, 1073–1078, 2002.

[24] H.A Le Thi, T. Pham Dinh and H. Dinh Nho *Solving inverse problems for an elliptic equations by DC (Difference of Convex functions) programming* Journal of Global Optimization, Vol.25, 407–423, 2003.

[25] H.A Le Thi, T. Pham Dinh and H. Dinh Nho *On the ill-posedness of the trust region Subproblem* Journal of Inverse and Ill-posed Problems, Vol.11, 545–577, 2003.

[26] H.A Le Thi, T. Pham Dinh and N. Huynh Van *Exact penalty techniques in DC Programming* Submitted, 2004.

[27] H.A Le Thi, T. Pham Dinh and M. Le Dung *Numerical solution for optimization over the efficient set by DC optimization algorithm* Operations Research Letters, Vol.19, 117–128, 1996.

[28] H.A Le Thi, T. Pham Dinh and M. Le Dung *A combined DC Optimization : Ellipsoidal Branch-and-Bound Algorithm for Solving Nonconvex Quadratic Programming Problems* Journal of Combinatorial Optimization, Vol.2, N°.1, 9–28, 1998.

[29] H.A Le Thi, T. Pham Dinh and M. Le Dung *Exact Penalty in DC. Programming* Vietnam Journal of Mathematics, Vol.27, N°.2, 169–178, 1999.

[30] H.A Le Thi, T. Pham Dinh and M. Le Dung *Simplicially Constrained DC Optimization over the Efficient Set and Weakly Efficient Sets* Journal of Optimization, Theory and Applications, Vol.117, N°.3, 503-531, 2003.

[31] H.A Le Thi, T. Pham Dinh and T. Nguyen Van *Combination between Local and Global Methods for Solving an Optimization Problem over the Efficient Set* European Journal of Operational Research, Vol.142, 257–270, 2002.

[32] T. Pham Dinh *Elements homoduaux relatifs à un couple de normes (φ, ψ). Applications au calcul de $S_{\varphi\psi}(A)$* Technical Report, Grenoble, 1975.

[33] T. Pham Dinh *Calcul du maximum d'une forme quadratique définie positive sur la boule unité de la norme du max* Technical Report, Grenoble, 1976.

[34] T. Pham Dinh *Contribution à la théorie de normes et ses applications à l'analyse numérique* Thèse de Doctorat d'Etat Es Science, Université Joseph Fourier- Grenoble, 1981.

[35] T. Pham Dinh *Convergence of subgradient method for computing the bound norm of matrices* Linear Alg. and Its Appl., Vol.62, 163–182, 1984.

[36] T. Pham Dinh *Algorithmes de calcul d'une forme quadratique sur la boule unité de la norme maximum* Numer. Math., Vol.45, 377–440, 1985.

[37] T. Pham Dinh *Algorithms for solving a class of non convex optimization problems. Methods of subgradients* Fermat days 85. Mathematics for Optimization, Elsevier Science Publishers B.V. North-Holland, 1986.

[38] T. Pham Dinh *Duality in DC (difference of convex functions) optimization. Subgradient methods* Trends in Mathematical Optimization, International Series of Numer Math., Vol 84, 277–293, 1988.

[39] T. Pham Dinh and H.A Le Thi *Stabilité de la dualité lagrangienne en optimisation DC (différence de deux fonctions convexes)* C.R. Acad. Paris, P.318, Série I, 379–384, 1994.

[40] T. Pham Dinh and H.A Le Thi *Lagrangian stability and global optimality in nonconvex quadratic minimization over Euclidean balls and spheres* Journal of Convex Analysis, Vol.2, 263–276, 1995.

[41] T. Pham Dinh and H.A Le Thi *DC optimization algorithms for globally minimizing nonconvex quadratic forms on Euclideans balls and spheres* Operations Research Letters, Vol.19, 207–216, 1996.

[42] T. Pham Dinh and H.A Le Thi *Convex analysis approach to d.c. programming : Theory, Algorithms and Applications* Acta Mathematica Vietnamica, dedicated to Professor Hoang Tuy on the occasion of his 70th birthday, Vol.22, N°.1, 289–355, 1997.

[43] T. Pham Dinh and H.A Le Thi *DC optimization algorithms for solving the trust region subproblem* SIAM Journal of Optimization, Vol.8, N°.2, 476–505, 1998.

[44] F.B Akoa *Approches de points intérieurs et de programmation DC en optimisation non convexe. Code et simulations numériques industrielles.* Thèse de Doctorat de l'Université de Rouen, 2005.

[45] T.Q Phong *Analyse Numerique des Méthodes d'Optimisation Globale* Thèse de Doctoral, Université de Rouen, 1994.

[46] T.Q Phong, T. Pham Dinh and H.A Le Thi *A New Method for Solving DC Programming Problems. Application to Fuel Mixture Nonconvex Optimization Problem* Journal of Global Optimal, Vol.6, 87–105, 1995.

[47] P.T Thach *DC sets, DC functions and nonlinear equations* Mathematical Programming, Vol.58, 415–428, 1993.

[48] N.V Vinh *Méthodes exactes pour l'optimisation DC polyédrale en variables mixtes 0-1 basées sur DCA et des nouvelles coupes* Thèse de Doctorat de l'Université de Rouen, 2006.

[49] R. Collobert, F. Sinz, J. Weston and L. Bottou *Trading Convexity for Scalability* Proceedings of the 23rd ICML, 2006.

[50] N. Krause and Y. Singer *Leveraging the margin more carefully* International Conference on Machine Learning ICML, 2004.

[51] Y. Liu, X. Shen and H. Doss *Multicategory ψ-Learning and Support Vector Machine : Computational Tools* Journal of Computational and Graphical Statistics, Vol.14, 219–236, 2005.

[52] Y. Liu and X. Shen *Multicategory ψ-Learning* Journal of the American Statistical Association, Vol.101, 500–509, 2006.

[53] J. Neumann, C. Schnörr and G. Steidl *SVM-based Feature Selection by Direct Objective Minimisation* Pattern Recognition, Proc. of 26th DAGM Symposium, LNCS, Springer, August 2004.

[54] J. Neumann, C. Schnörr and G. Steidl *Combined SVM-Based Feature Selection and Classification Machine Learning* in Press (published online) http ://www.cvgpr.uni-mannheim.de/Publications/ML-05.pdf.

[55] C. Ronan, S. Fabian, W. Jason and B. Léon *Trading Convexity for Scalability* International Conference on Machine Learning ICML, 2006.

[56] X. Shen *From large margin classification to ψ-learning* School of Statistics, University of Minnesota.

[57] X. Shen, G.C Tseng, X. Zhang, and W.H Wong *On ψ-Learning* Journal of American Statistical Association, Vol.98, 724–734, 2003.

[58] T. Schüle, C. Schnörr, S. Weber and J. Hornegger *Discrete Tomography by convex-concave regularization and DC programming* Discrete Applied Mat., Vol.151, 229–243, 2005.

[59] T. Schüle, S. Weber and C. Schnörr *Adaptive Reconstruction of Discrete-Valued Objects from few Projections* Electr. Notes in Discr. Math., Vol.20, 365–384, 2005.

[60] S. Weber, T. Schüle, J. Hornegger and C. Schnörr *Binary Tomography by Iterating Linear Programs from Noisy Projections* IWCIA, LNCS 3322, 38–51, 2004.

[61] S. Weber, C. Schnörr, T. Schüle and J. Hornegger *Binary Tomography by Iterating Linear Programs* R. Klette, R. Kozera, L. Noakes and J. Weickert (Eds.), Computational Imaging and Vision - Geometric Properties from Incomplete Data, Kluwer Academic Press 2005.

[62] S. Weber, T. Schüle and C. Schnörr *Prior Learning and Convex-Concave Regularization of Binary Tomography* Electr. Notes in Discr. Math., Vol.20, 313–327, 2005.

[63] S. Weber, A. Nagy, T. Schüle, C. Schnörr and A. Kuba *A Benchmark Evaluation of Large-Scale Optimization Approaches to Binary Tomography* DGCI 2006, LNCS 4245, 146–156, 2006.

[64] S. Weber, T. Schüle, A. Kuba and C. Schnörr *Binary Tomography with Deblurring* IWCIA 2006, LNCS 4040, 375–388, 2006.

[65] A.L Yuille and A. Rangarajan *The Convex Concave Procedure (CCCP)* Advances in Neural Information Processing System 14, Cambrige MA : MIT Press.

[66] A. Auslender *Optimisation Méthodes Numériques* Paris : Masson, 1976.

[67] H.P Benson *Concave minimization : theoty, applications and algorithms* in Handbook of Global Optimisation, R. Horst and P. Padalos eds., Kluwer Academic Publishers, 43–148, 1995.

[68] H.P Benson *Deterministic algorithm for constrained concave minimization : a unified critical survey* Naval Research Logistics, Vol.43, 765–795, 1996.

[69] H.P Benson and R. Horst *A branch and bound - outer approximation for concave minimization over a convex set* Journal of Computers and Mathematics with applications, Vol.21, 67–76, 1991.

[70] R.W Cottle and G.B Dantzig *Complementary pivot theory of mathematical programming* Linear algebra and its applications, Vol.1, 103–125, 1968.

[71] G.B Dantzig *Linear Programming and Extensions* Princeton University Press, Princeton, New Jersey, 1963.

[72] G.B Dantzig, D.R Fulkerson and S.M Johnson *Solution of a large scale traveling salesman problem* Operations Recheach, Vol.2, 393–410, 1954.

[73] R. Fletcher *Pratical methods of Optimization* John Wiley, New York, 1980.

[74] R. Horst, N.V Thoai and H. Tuy *On a outer approximation concept in global optimisation* Optimization, Vol.20, 255–264, 1989.

[75] R. Horst, N.V Thoai and H.P Benson *Concave minimization via conical partitions and polyhedral outer approximation* Mathematical Programming, Vol.50, 259–276, 1991.

[76] R. Horst, P.M Pardalos and N.V Thoai *Introduction to Global Optimization* Kluwer Academic Publishers, 1995.

[77] R. Horst and H. Tuy *Global Optimization : Deterministic Approaches* Third edition, Springer-Verlag, 1996.

[78] R. Horst and N.V Thoai *DC Programming : Overview* Journal of Optimization Theory and Applications, Vol.2103, 1–43, 1999.

[79] P.J Laurent *Approximation et optimisation* Paris : Hermann, 1972.

[80] K. Levenberg *A method for the solution or certain nonlinear problems in least quares* Quart. Appl. Math., Vol.2, 1944.

[81] D.W. Marquardt *An algorithm for least squares estimation of nonlinear parameters* SIAM J. Appl. Math., Vol.11, 1963.

[82] J.J Moré *Recent developpements in algorithm and software for trust region methods* Mathematique Programming, The state of the art, Springer-Verlag, Berlin, 258–287, 1983.

[83] J.J Moré and D.C Sorensen *Computing a trust region step* SIAM J. Sci. Statist. Comput., Vol.4, 553–572, 1983.

[84] B. T Polyak *Introduction to optimization* Inc., Publications Division, 1987.

[85] R.T Rockafellar *Convex analysis* Princeton University Press, 1970.

[86] R.T Rockafellar *Monotone operators and the proximal point algorithm* SIAM Journal on Control and Optimization, Vol.14, 877–898, 1976.

[87] S.J Sadjadi and K. Ponnambalam *Advances in trust region algorithms for constrained optimization* Applied Numerical Mathematics, Vol.29, 423–443, 1999.

[88] D.C Sorensen *Newton's method with a model trust region modification* SIAM J. Numer. Anal., Vol.19, N°.2, 409–426, 1982.

[89] S. Steihaug *The conjugate gradient method and trust region in large scale optimization* SIAM J. Numer. Anal., Vol.20, 626–637, 1983.

[90] J.F Toland *Duality in nonconvex optimization* Journal of Mathematical Analysis and Applications, Vol.66, 399-415, 1978.

[91] J.F Toland *On subdifferential calculus and duality in nonconvex optimization* Bull. Soc. Math. France, Mémoire 60, 177–183, 1979.

[92] P.L Toint *Towards an efficient sparsity exploiting Newton method for minimization* Duff, I., ed., Sparse Matrices and Their Uses, 57–88, Academic Press, 1981.

[93] H. Tuy *Concave programming under linear constraints* Translated Soviet Mathematics, Vol.5, 1437–1440, 1964.

[94] H. Tuy *On outer approximation methods for solving concave minimization problems* Acta Mathematica Vietnamica, Vol.8, 3–34, 1983.

[95] H. Tuy *Global Minimization of a Difference of Two Convex Functions* Mathematics Programming Study, Vol.30, 150–182, 1987.

[96] H. Tuy *Global Optimization : Deterministic Approaches* 2nd revised edition, Springer-Verlag, Berlin, 1993.

[97] H. Tuy *DC Optimisation : Theory, Methods and Algorithms, Handbook of Global Optimisation* Horst and Pardalos eds, Klaver Academic Publishers, 149–216, 1995.

[98] H. Tuy *Convex Analysis and Global Optimization* Kluwer Academic Publishers, 1998.

[99] J.B.H Urruty *Generalized differentiability, duality and optimization for problem dealing with differences of convex functions*Lecture Notes in Economics and Mathematical Systems, Vol.256, Heidelberg, Springer-Verlag, 260–277, 1985.

[100] J.B.H Urruty *Conditions nécessaires et suffisantes d'optimalité globale en optimisation de différences de deux fonctions convexes* CRAS, Vol.309, Série I, 459–462, 1989.

[101] F. Rendl and H. Wolkovicz *A semidefinite framework to trust region subproblems with application to large scale minimization* CORR Report 94-32, University of Waterloo, 1994.

[102] B.C Arntzen, G.G Brown, T.P Harrison and L.L Trafton *Global supply chain management at Digital Equipment Corporation* Interfaces, Vol.25, N°.1, 69–93, 1995.

[103] J. Ashayeri, A.J Westerhof and P.V Alst *Application of mixed integer programming to a largescale logistics problem* International Journal of Production Economics, Vol.36, 133–152, 1994.

[104] R.H Ballou *Business Logistics Management* 3rd Ed. Prentice-Hall, 1992.

[105] B.M Beamon *Supply chain design and Analysis : Models and Methods* International Journal of Production Economics, Vol. 55, 281–294, 1998.

[106] G.G Brown, G.W Graves and M.D Honczarenko *Design and operation of a multicommodity production/distribution system using primal goal decomposition* Management Science, Vol. 33, 1469–1480, 1987.

[107] S.S Chauhan, J.M Proth, A.M Sarmiento and R. Nagi *Strategic Supply Chain Design for a New Market Opportunity from Qualifed Partner Sets* INRIA Research Report, RR-4508, July 2002.

[108] S.S Chauhan *Chaînes d'approvisionnement : approches stratégique et tactique* Thèse de Doctorat, Université de Metz, Septembre 2003.

[109] M.A Cohen and H.L Lee *Strategic analysis of integrated production-distribution systems : models and methods* Operation Research, Vol.36, 216–228, 1988.

[110] M.A Cohen and H.L Lee *Resource deployment analysis of global manufacturing ans distribution networks* Journal of Manufacturing Operations Management, Vol.2, 81–104, 1989.

[111] M.A Cohen, M. Fisher and R. Jaikumar *International manufacturing and distribution networks : A normative model framework* Managing International Manufacturing, North-Holland Amsterdam, 67–93, 1989.

[112] A.M Geoffrion and G.W Graves *Multicommodity distribution system design by Bender's decomposition* Management Science, Vol.20, N°.5, 822–844, 1974.

[113] M. Govil and J.M Proth *Supply Chain Design and Management : Strategic and Tactical Perspectives* London Academi Press, 2002.

[114] A.M Sarmiento and R. Nagi *A review of Integrated Analysis of Production-Distribution Systems* IIE Transactions Special Issue on Manufacturing Logistics, Vol.11, 1061–1074, 1999.

[115] D.J Thomas and P.M Griffin *Coordinated supply chain management* European Journal of Operation Research, Vol.94, 1–15, 1996.

[116] T.J.V Roy *Multi-level production and distribution planning with transportation fleet optimization* Management Science Vol.35, N°.12, 1443–1453, 1989.

[117] J.C Vidal and M. Goetschalckx *Strategic production-distribution models : A critical review with emphasis on global supply chain models* European Journal of Operation Research, Vol.98, 1–18, 1997.

[118] A. Alpers, P. Gritzmann and L. Thorens *Stability and Instability in Discrete Tomography* Lecture Notes in Computer Sciece, Vol.2243, 175–186, 2002.

[119] T. Frese, C.A Bouman and K. Sauer *Multiscale Bayesian methods for Discrete Tomography* in Discrete Tomography : Foundations, Algorithms, and Applications, Birkhäuser, Boston, 237–264, 1999.

[120] P. Fishburn, P. Schwander, L. Shepp and R. Vanderbei *The discrete Radon transform and its approximate inversion via linear programming* Discrete Appl. Math., Vol.75, 39–61, 1997.

[121] D. Gale *A theorem on flows in networks* Pacific. J. Math., Vol.7, 1073–1082, 1957.

[122] R.J Gardner, P. Gritzmann and D. Prangenberg *On the computational complexity of reconstructing lattice sets from their X-rays* Discrete Mathematics, Vol.202, 45–71, 1999.

[123] R.J Gardner and P. Gritzmann *Discrete tomography : determination of finite sets by X-rays* Trans. Amer. Math. Soc., Vol.349, 2271–2295, 1997.

[124] P. Gritzmann, S. Vries and M. Wingelmann *Approximating binary images from discrete X-rays* SIAM J. Optimizat., Vol.11, N°.2, 522–546, 2000.

[125] G. T. Herman *Image Reconstruction from Projections : the Fundamentals of Computerized Tomography* Academic Press, NewYork, 1980.

[126] G. T. Herman and A. Kuba *Discrete Tomography : Foundations, Algorithms, and Applications* Birkhäuser, Boston, 1999.

[127] G. T. Herman and A. Kuba *Discrete Tomography : A historical overview* in Discrete Tomography : Foundations, Algorithms, and Applications, Birkhäuser, Boston, 3–34, 1999.

[128] A.D Lungo and M. Nivat *Reconstruction of connected sets from two projections* in Discrete Tomography : Foundations, Algorithms and Applications, Birkhäuser, Boston, 85–113, 1999.

[129] S. Matej, A. Vardi, G.T Herman, and E. Vardi *Binary Tomography using Gibbs priors* in Discrete Tomography : Foundations, Algorithms, and Applications, Birkhäuser, Boston, 191–212, 1999.

[130] H. Ryser *Combinatorial properties of matrices of zeros and ones* Canad. J. Math., Vol.9, 371–377, 1957.

[131] R. Tijdeman and L. Hajdu *An algorithm for discrete tomography* Linear Algebra and Its Applications, Vol.339, 147–169, 2001.

[132] Y. Vardi and D. Lee *Discrete Radon transform and its approximate inversion via the EM algorithm* Int. J. Imaging Sci. Tech., Vol.9, 155–173, 1998.

[133] M.N Ahmed, S.M Yamany, N. Mohamed, A.A Farag and T. Moriarty *A modified fuzzy C-means algorithm for bias field estimation and segmentation of MRI data* IEEE Trans. on Medical Imaging, Vol.21, 193–199, 2002.

[134] J.C. Bezdek *Pattern Recognition with Fuzzy Objective Function Algorithm* New York, NY. Plenum Press, 1981.

[135] J.C. Bezdek, L.O. Hall and L.P. Clake *Review of MR image segmentation techniques using pattern recognition* Medical Physics, Vol.20, 1033–1048, 1993.

[136] C. Bouman and B. Liu *Segmentation of Textured Images Using a Multiple Resolution Approach* Proc. IEEE Int'l Conf. on Acoust., Speech and Sig. Proc., NewYork, NY, April 11-14, 1124–1127, 1988.

[137] P. Campadelli, D. Medici and R. Schettini *Color Image Segmentation using Hopfield Networks* Image and Vision Computing, Vol.15, N°.3, 161–166, 1997.

[138] J.C Dunn *A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters* Journal of Cybernetics, Vol.3, 32–57, 1973.

[139] W.L Hung, M.S Yang and D.H Chen *Parameter selection for suppressed fuzzy c-means with an application to MRI segmentation* Pattern Recognition Letters, Vol.27, 424–438, 2006.

[140] E. Littman and E. Ritter *Adaptative Color Segmentation - A Comparison of Neural and Statistical Methods* IEEE Transactions on Neural Networks, Vol.8, N°.1, 175–185, 1997.

[141] J.P. Pal and S.K. Pal *A review on image segmentation techniques* Pattern Recognition, Vol.26, 1277–1294, 1993s.

[142] D.L Pham *Fuzzy Clustering with spatial constraints* Proc.IEEE Intern. Conf. on Image Processing, New Yord, USA.

[143] J.C Rajapakse, J.N Giedd and J.L Rapoport *Statistical Approach to Segmentation of Singke-Chanel Cerebral MR Images* IEEE Trans. On Medical Imaging, Vol.16, 2004.

[144] D.Q Zhang and S.C Chen *A novel kernelized fuzzy C-means algorithm with application in medical image segmentation* Artificial Intelligence in Medicine, Vol.32, 37–50, 2004.

[145] A. Bartoli and P. Sturm *Nonliner estimation of the fundamental matrix with minimal parameters* IEEE Trans. on Part. Analysis and Machine Inte., Vol.26, N°.3, 426–432, 2004.

[146] B. Boufama and R. Mohr *Epipole and fundamentalmatrix estimation using virtual parallax* In Proceedings Fifth International Conference on Computer Vision, Cambridge, Mass., IEEE Computer Society Press, Los Alamitos, Ca., June 1995.

[147] G. Chesi, A. Vicino and R. Cipolla *Estimating the fundamental matrix via contrained Least-Squares : A convex approach* IEEE Trans. on Part. Analysis and Machine Inte., Vol.24, N°.3, 397–401, 2002.

[148] R. Elias and R. Laganière *Projective Geometry for Three-Dimensional Computer Vision* Technical Report, School of Information Technology and Engineering, University of Ottawa, Ottawa, Ontario, Canada, K1N 6N5.

[149] R. Hartley *In defence of the 8 point algorithm* In Proceedings Fifth International Conference on Computer Vision, Cambridge, Mass., IEEE Computer Society Press, Los Alamitos, Ca., June 1995.

[150] R. Hartley *Minimizing algebraic error in geometry estimation problem* Proc. Sixth Int'l Conf. Computer Vision, 469–476, 1998.

[151] R. Hartley and A. Zisserman *Multiple View Geometry in computer vision* Cambridge University Press, United Kingdom, 2000.

[152] R. Horaud and O. Monga *Vision par ordinateur : outils fondamentaux* Editions Hermes, Paris, 1993.

[153] Z. Zhang *Determining the Epipolar Geometry and its Uncertainty : A Review* I. J. Computer Vision, Vol.27, N°.2, 161–195, 1998.

[154] Z. Zhang and C. Loop *Estimating the Fundamental Matrix by transforming image points in projective space* Computer Vision & Image Understanding, Vol.82, N°.2, 174–180, 2001.

Résumé

Les travaux présentés dans cette thèse concernent les nouvelles techniques d'optimisation pour la résolution de quatre problèmes importants issus de deux domaines : transport logistique et vision et traitement d'image. Il s'agit des problèmes d'optimisation non convexe de très grande dimension pour lesquels la recherche des bonnes méthodes de résolution est toujours d'actualité.

Notre travail s'appuie principalement sur la programmation DC et DCA. Cette démarche est motivée par la robustesse et la performance de la programmation DC et DCA comparées à des méthodes existantes, leur adaptation aux structures des problèmes traités et leur capacité de résoudre des problèmes industriels de grande dimension.

La thèse est composée de cinq chapitres. Le premier chapitre, servant des références aux autres, présente les outils de bases de la programmation DC et DCA. Le chapitre deux porte sur le problème de conception d'une chaîne d'approvisionnement. Nous traitons ce problème par DCA via la pénalité exacte en basant sur les décompositions DC appropriées et proposons une technique de recherche de bon point initial. Le chapitre trois concerne la tomographie discrète appliquée au problème de reconstruction d'image binaire. Ce problème est traité par les trois approches différentes basées sur DCA. Dans le chapitre quatre, nous étudions le problème de segmentation d'image via la classification floue par DCA. L'estimation non linéaire de la matrice fondamentale en vision par ordinateur par la méthode de région de confiance basée sur la méthode de gradient conjugué tronqué ou DCA est développée dans le dernier chapitre.

Mots-clés : Programmation DC, DCA, Optimisation globale, Programmation linéaire en variables mixtes 0-1, Chaîne d'approvisionnement, Reconstruction d'image, Segmentation d'image, Matrice fondamentale.

Abstract

The works presented in this thesis are related to new techniques of optimization for the solution of four important problems of two fields : transport logistics, computer vision and image processing. They are nonconvex optimization problems of very large dimensions for which the research of good solution methods is always of actuality.

Our work is based mainly on the DC programming and DCA that have been successfully applied in various fields of applied sciences, including machine learning. It is motivated and justified by the robustness and the good performance of DC programming and DCA in comparison with the existing methods.

The thesis is divided into five chapters. The first chapter, which serves as a reference for other chapters, presents some tools from the bases of DC programming and DCA. The second chapter discusses the problem of supply chain design at the strategic level, which can be formulated as a mixed integer linear program. We treat this problem by DC algorithms via exact penalty on the basis of suitable DC decompositions and a technique for finding a good starting point. The third chapter is concerned with the discrete tomography applied to the problem of binary image reconstruction. This problem is solved by three different approaches based on DCA. In the fourth chapter, we study the problem of image segmentation using Fuzzy C-Means clustering via DCA. A nonlinear estimate of the fundamental matrix in computer vision by the trust-region method based on the truncated conjugate gradient method or DCA is developed in the final chapter.

Keywords : DC programming, DCA, Global optimization, Mixed integer programming, Supply chain design, Image reconstruction, Image segmentation, Fundamental matrix.