Approches locales et globales basées sur la programmation DC et DCA pour des problèmes combinatoires en variables mixtes 0-1 : applications à la planification opérationnelle

Thuan Nguyen Quang

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 QUANG THUAN

Titre de la thèse :

Approches locales et globales basées sur la programmation DC et DCA pour des problèmes combinatoires en variables mixtes 0-1. Applications à la planification opérationnelle.

Date de soutenance : le 10 Novembre 2010

Composition du Jury :

Président	PHAM DINH Tao	*Professeur, INSA-Rouen*
Rapporteurs	Mohamed DIDI BIHA	*Professeur, Université de Caen*
	Philippe MICHELON	*Professeur, Université d'Avignon et des Pays de Vaucluse*
Examinateurs	Jean-Marie PROTH	*Directeur de Recherche, INRIA Nancy-Grand Est*
	El-Houssaine AGHEZZAF	*Professeur, Université de Ghent, Belgique*
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 mon endettement à Madame le Professeur LE THI Hoai An, qui a accepté d'être ma directrice de thèse, pour son aide inestimable, les conseils pertinents et formateurs, les encouragements qu'elle m'a donnés. Je lui adresse toute ma gratitude pour la confiance et le soutien dont elle m'a témoigné.

Je tiens à remercier particulièrement Monsieur le Professeur PHAM DINH Tao, l'équipe Modélisation et Optimisation Appliquée de l'INSA de Rouen pour ses conseils, 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 Philippe MICHELON de l'Université d'Avignon et des Pays de Vaucluse et à Monsieur le Professeur Mohamed DIDI BIHA de l'université de Caen d'avoir accepté la charge de rapporteur de ma thèse, et d'avoir jugé mon travail.

Je tiens aussi à remercier Monsieur Jean-Marie PROTH, le Directeur de Recherche de INRIA Nancy-Grand Est, à Monsieur El-Houssaine AGHEZZAF, Professeur de l'Université de Ghent, Belgique 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'Institut Polytechnique d'Hanoï de m'avoir apporté de soutien. Je tiens à remercier particulièrement Madame Professeur NGUYEN Thi Bach Kim pour sa soutien, sa confiance et son encouragement.

Je me permets également de remercier la Région Lorraine et le Laboratoire LITA de l'Université de Paul Verlaine-Metz pour l'aide financière qu'ils m'ont attribuée.

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

Je remercie tous mes collègues français et vietnamiens rencontrés à Metz pour les moments agréables lors de mon séjour en France. Enfin je remercie tous ceux qui m'ont aidé de près ou de loin et tous ceux qui m'ont motivé même inconsciemment.

Table des matières

Table des figures

Liste des tableaux

Liste des Publications et Conférences

NGUYEN QUANG THUAN

Article avec comité de lecture

- Le Thi Hoai An, Nguyen Quang Thuan, *A Robust Approach for Nonlinear UAV Task Assignment Problem under Uncertainty*. To appear in : Transactions on Computational Collective Intelligence, LNCS, Springer Pub.

- Le Thi Hoai An, Nguyen Quang Thuan, Nguyen Huynh Tuong and Pham Dinh Tao, *Solving the earliness tardiness scheduling problem by DC programming and DCA*. Math. Balkanica (N.S.) 23 (2009), no. 3-4, pp. 271-288.

- Le Thi Hoai An, Nguyen Quang Thuan, Nguyen Huynh Tuong and Pham Dinh Tao, A *time-indexed formulation of earliness tardiness scheduling via DC programming and DCA*, IEEE Proceeding of the international Multi-conference on Computer Science and Information Technology, Mragowo, Poland, October 2009- ISBN978-83-60810-22-4, IEEE Catalog Number CFP0964E, pp.779-784.

- Le Thi Hoai An, Nguyen Quang Thuan, Phan Tran Khoa and Pham Dinh Tao, *Cross-layer Optimization in Multi-hop TDMA Networks using DCA*, IEEE Proceedings of the 17th international conference on computer communications and networks (ICCCN08), St Thomas, U.S. Virgin Islands, August 3 - 7, ISBN978-1-4244- 2390-3/08/ ©2008 IEEE (6 pages).

- Le Thi Hoai An, Nguyen Quang Thuan, Phan Tran Khoa and Pham Dinh Tao, *Energy Minimization-based Cross-layer Design in Wireless Networks*, IEEE Proceedings of the 2008 High Performance Computing & Simulation Conference (HPCS 2008) Nicosia, Cyprus, June 3 - 6, 2008, pp. 283-289.

- Nguyen Quang Thuan, Le Thi Hoai An, *Solving an inventory routing problem in supply chain by DC programming and DCA*. Submitted to LNCS, 2010

Article en cours de préparation

- Le Thi Hoai An, Nguyen Quang Thuan, *An efficient approach for an inventory routing problem.*

- Le Thi Hoai An, Nguyen Quang Thuan, *Cross-layer optimization via DCA programming and DCA.*

Communications aux colloques internationaux avec actes publiés

- Le Thi Hoai An, Nguyen Quang Thuan, Nguyen Huynh Tuong and Pham Dinh Tao, *A new approach to solve Just-in-Time Scheduling Problem*, the 4th International Conference on High Performance Scientific Computing, Hanoi, Vietnam, March, 3-6, 2009

- Le Thi Hoai An, Nguyen Quang Thuan, *A Robust Approach for Solving Nonlinear Task Assignment Problem under Uncertainty*, Workshop on Optimization and Learning : Theory, Algorithms and Applications, Metz June 17-18, 2010

Introduction

Nous nous intéressons dans cette thèse à la programmation linéaire et/ou nonlinéaire en variables pures/mixtes 0-1. De nombreux problèmes de la vie courante sont modélisés sous ces formes. C'est un modèle fondamental en Aide à la Décision où les variables 0 - 1 sont appelées variables de décision, et une solution optimale correspond à une décision optimale pour la situation considérée. Il est connu que ce problème est NP - difficile. Trouver une méthode efficace pour sa résolution est un grand défi en Optimisation et Recherche Opérationnelle. Pour simplifier la présentation, nous considérons dans ce qui suit la programmation en variables mixtes 0 - 1, sachant que le cas des variables "pures" 0 - 1 est un cas particulier du premier.

La programmation en variables mixtes 0-1 appartient à la classe de la programmation non convexe pour laquelle on a deux types d'approches différentes mais complémentaires :

i. Les 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) ([18]), R. Horst, P. Pardalos et N. V. Thoai ([27, 29]),... L'inconvénient majeur des méthodes globales est leur lourdeur (encombrement en places mémoires) et leur coût trop important. Par conséquent, elles ne sont pas applicables aux problèmes d'optimisation non convexes réels qui sont souvent de très grande dimension.

ii. Les 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 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 [36]-[69] et [82]-[87]).

Les travaux de cette thèse se situent tous dans ces deux approches : globales et DCA. Ils concernent les deux catégories de la programmation en variables mixtes 0-1 : la fonction ob-

jectif est linéaire et/ou non linéaire. Bien que la programmation en variables mixtes 0-1 soit un problème d'optimisation combinatoire, notre méthodologie est basée sur les outils d'optimisation continue dont l'épine dorsale est la programmation DC et DCA. Cette démarche est motivée par la robustesse et la performance de la programmation DC et DCA comparée à des méthodes existantes, leur adaptation aux structures des problèmes traités et leur capacité à résoudre des problèmes industriels de très grande dimension. A notre connaissance, DCA est actuellement parmi les rares algorithmes de la programmation non convexe capables de traiter des problèmes (différentiables ou non) de très grande dimension [39, 48, 49, 51, 57].

Nos contributions.

A l'aide des techniques de pénalité exacte nous reformulons la programmation en variables mixtes 0-1 en terme d'un problème d'optimisation continue qui est en fait une programmation DC. Dès lors, DCA est applicable. Pour résoudre globalement la Programmation Linéaire en variables Mixtes 0-1 - une large classe de problèmes importants en Aide à la Décision - nous proposons différentes méthodes globales de type Séparation et Evaluation (SE) et/ou Approximation de l'Extérieur (AE) basées sur DCA. La combinaison de DCA et de ces méthodes globales a pour multiple objectifs :
- de trouver une bonne solution réalisable du problème original via DCA et de réduire ainsi l'écart entre la borne inférieure et la borne supérieure de la valeur optimale ;
- de trouver un bon point initial pour DCA, grâce à la résolution du problème relaxé à l'étape courante ;
- de déterminer une coupe à partir d'une solution fournie par DCA dans les algorithmes de type DCA-AE ;
- de prouver la globalité d'une solution obtenue par DCA grâce à la réduction du gap (l'écart entre la borne inférieure et la borne supérieure) dans les méthodes globales.

Pour la Programmation Linéaire en variables Mixtes 0-1 (appelée PLM01 dans cette thèse), Nguyen et al [47, 78] ont introduit une combinaison de AE et DCA, appelée DCACUT. Ces auteurs ont proposé une nouvelle coupe basée sur un minimum local d'une fonction pénalité et développé une méthode de coupe utilisant DCA appliqué au problème pénalisé. Cette méthode est très efficace dans le cas où la solution fournie par ce DCA est un minimum local de la fonction pénalité [44, 45, 46, 75], car on peut introduire une "bonne" coupe dans un tel cas. Cependant elle n'est pas applicable dans le cas contraire, i.e, quand la solution obtenue par DCA n'est pas un minimum local de la fonction pénalité. Dans cette thèse, étant particulièrement intéressés par cette dernière situation, nous avons amélioré DCACUT pour qu'elle s'applique dans tous les cas.

En plus de DCACUT, nous proposons différentes combinaisons de DCA avec les approches globales : Méthode de coupes de Gomory, Séparation et Evaluation (SE) ou Branch and Bound (B&B) en Anglais, et Branch and Cut. Une étude algorithmique comparative sur les "Benchmark problems" est ainsi réalisée. Il s'agit de la comparaison entre
- les algorithmes d'AE : DCACUT, Méthode de coupe mixte de Gomory,
- les algorithmes d'AE et les algorithmes de type B&B,

- les algorithmes de type B&B comme B&B classique, B&B combiné avec DCA (noté BBDCA), et Branch-and-Cut.

Sur le plan des applications, nous nous sommes intéressés aux deux domaines suivants :
- Réseaux de télécommunication sans fil : nous considérons deux problèmes fondamentaux en optimisation inter-couches pour les réseaux sans fil dont l'un a pour objectif de maximiser la durée de vie d'un réseau et l'autre cherche à minimiser la consommation d'énergie.
- Ordonnancement : un problème d'ordonnancement consiste à organiser dans le temps la réalisation de tâches, compte tenu de contraintes temporelles (délais, contraintes d'enchaînement, etc.) et de contraintes portant sur la disponibilité des ressources requises. Un ordonnancement est défini par le planning d'exécution des tâches et d'allocation des ressources et vise à satisfaire un ou plusieurs objectifs. Nous traitons deux problèmes d'ordonnancement dont l'un a pour objectif de minimiser la pénalité d'avance et la pénalité de retard, et l'autre cherche à minimiser la pénalité d'avance-retard et la pénalité sur la date d'échéance.

Pour la programmation non linéaire en variables mixtes 0-1, nous étudions une large classe des problèmes souvent rencontrés en pratique : la fonction objectif est DC. Utilisant un résultat introduit très récemment dans [64] nous transformons le problème original en une programmation DC et développons ensuite DCA pour sa résolution.

Deux applications sont considérées :
- Affectation des tâches aux véhicules aériens non-pilotés (UAV) : l'utilisation des UAVs pour les différentes missions militaires attire de plus en plus l'attention. Nous cherchons à affecter un ensemble de UAVs à un ensemble de tâches de manière optimale. L'efficacité est basée sur les scores de cibles. La mission est de maximiser les scores de cibles tout en respectant des contraintes de capacité des UAV et des tâches. Nous développons un schéma DCA et un algorithme combiné DCA et B&B pour résoudre ce problème.
- Tournées de véhicules dans une chaîne d'approvisionnement : il s'agit d'affecter des véhicules à partir d'un dépôt central aux points de vente et de déterminer ses cycles pour minimiser le coût de transport, le coût de stockage, le coût de traitement... C'est un problème important et très difficile à cause de la forme complexe de la fonction objectif, en plus des variables binaires. Nous proposons un schéma DCA pour sa résolution numérique.

Organisation de la thèse.
La thèse est divisée en trois parties et est composée de 8 chapitres.

i. Dans la première partie intitulée "Outils de base" nous présentons des outils théoriques et algorithmiques servant des références aux autres parties. Après une présentation complète de la programmation DC et DCA dans le Chapitre 1, nous explorerons la technique de Séparation et Evaluation dans le Chapitre 2. Le Chapitre 3 est dédié à la présentation de la méthode de AE.

ii. La seconde partie concerne la programmation linéaire en variables mixtes 0-1 (PLM01). Elle est composée de trois chapitres. Le Chapitre 4 est consacré aux méthodes numériques

pour PLM01 : après la présentation de l'algorithme DCACUT amélioré, nous proposons différentes méthodes globales basées sur DCA et les expériences numériques comparatives pour ces méthodes. Les deux applications (les réseaux de télécommunication et l'ordonnancement) sont présentées, respectivement, dans le Chapitre 5 et le Chapitre 6.

iii. La troisième partie est consacrée à la programmation non linéaire en variables mixtes 0-1 et sa mise en oeuvre pour la résolution de deux problèmes concrets. Le Chapitre 7 porte sur l'affectation des tâches aux véhicules aériens non-pilotés alors que le Chapitre 8 concerne le problème des tournées de véhicules dans une chaîne d'approvisionnement.

Première partie

Outils de base

Chapitre 1

Introduction à la programmation DC et DCA

Résumé *Nous reportons dans ce chapitre les principaux résultats relatifs à la programmation DC et DCA qui nous seront les plus utiles dans la suite.*

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 relativement 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 les classes de 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 dimension raisonnable 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 ([18, 19]) qui présentent la théorie, les algorithmes et les 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'École Vietnamienne (Phan Thien Thach, Le Dung Muu, ...).

La seconde approche repose sur l'arsenal puissant d'analyse et d'optimisation convexe. Le 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) ([105]) 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 [36]-[69] et [82]-[87]) pour qu'il s'impose définitivement en optimisation non convexe comme étant un des algorithmes les plus simples et performants, capable 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 ([36]), H. A. Le Thi 1997 ([37]). Pour une étude détaillée nous nous référons à ces deux références (voir également [36]-[69] et [82]-[87]).

1.1 Éléments 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 ([33]), de R.T Rockafellar ([93]) et d'A. Auslender ([2]). 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 $-\infty - (+\infty) = +\infty$ ([93]). 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 $\partial f(x^0)$.

Étant 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 $\partial_\epsilon f(x^0)$.

La fonction $f : S \longrightarrow \mathbb{R}$ est dite semi-continue inférieure (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 differentiable 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 *([93])*
- *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 Fonctions 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 relativement 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 $\mathbb{R} \cup \{+\infty\}$ ([37]). 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 differentiable 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'application rencontrés.

Ainsi on ne peut d'emblée traiter tout problème d'optimisation non convexe et non differentiable. 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 équivalente 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).

Le 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 l'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 ([79, 80, 105, 106, 36, 37, 48]) qui, comme en optimisation convexe, a de profondes répercussions pratiques sur les méthodes numériques.

Les algorithmes de l'optimisation DC (DCA) dus à Pham Dinh Tao ([81, 82]) 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 ([79, 81]). Cependant, il a fallu attendre les travaux communs de Le Thi et Pham au cours de ces dix dernières années (voir [36]-[69] et [82]-[87]) 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 ([93]). 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 [48]).

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 ([105]).

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 ([105]).

□

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 ([107]) 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 [37] 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 ([105]). 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 ([37, 48]))*** *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. ([37, 48])). □

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 ([37, 48]))*** *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 définitive 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 *([48]) 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 ([48]) :

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 *([48])**(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 *([48]) 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 DCA. Pour rendre les composantes convexes 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 différentiables.

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 comme suit :

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

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 problèmes d'optimisation DC se rencontre fréquemment en pratique et possède d'intéressantes propriétés. Nous allons voir que la description de DCA y est particulièrement simple ([36, 37, 48]).

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 que 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 linéarisation 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 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 global 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 [36]-[69] et [82]-[87] et références incluses. Le traitement d'un programme non convexe par une approche DC et DCA devrait donc comporter 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

Méthode par Séparation et Evaluation (SE)

Résumé *Ce chapitre est consacré à la méthode par Séparation et Evaluation qui, en anglais, est appelée Branch-and-Bound.*

On considère le problème de minimisation d'une fonction continue sur un compact

$$\min\{f(x) : x \in S \subset \mathbb{R}^n\}. \tag{2.1}$$

Il est bien connu que si $S \neq \emptyset$ alors le problème admet une solution. On veut trouver une solution dite optimale $x^* \in S$ telle que

$$f(x^*) \leq f(x) \quad \forall x \in S.$$

L'idée de base de la méthode SE consiste en la division successive d'un ensemble qui contient S en sous-ensembles de plus en plus petits. A chaque sous-ensemble contenant une partie de S, on associe une borne inférieure de la valeur de la fonction objectif sur cet ensemble afin d'éliminer les parties non prometteuses et de sélectionner un ensemble que l'on devrait diviser par la suite.

Définition 2.0.1 *Soit M un compact dans $\mathbb{R}^n$ et soit I un ensemble fini d'indices. Un ensemble $M_i : i \in I$ de sous-ensembles compacts est appelé partition de M si*

$$M = \bigcup_{i \in I} M_i, \quad M_i \cap M_j = \partial M_i \cap \partial M_j, \quad \forall i, j \in I : i \neq j,$$

où ∂M_i dénote la frontière relative à M de M_i. □

2.1 Méthode de résolution et convergence

Adoptons la notation $\min f(S) = \min\{f(x) : x \in S\}$. Le schéma général de SE se résume de la manière suivante :

Prototype SE ([23, 102])

- **Initialisation**
  1. Choisir un compact $M_0 \supset S$, un ensemble fini des indices I_0, une partition $\mathcal{M}_0 = \{M_{0,i} : i \in I_0\}$ de M_0 satisfaisant $M_{0,i} \cap S \neq \emptyset, i \in I_0$.
  2. Pour chaque $i \in I_0$, déterminer
  $$S_{0,i} \subset M_{0,i} \cap S, S_{0,i} \neq \emptyset$$
  et
  $$\gamma_{0,i} = \gamma(M_{0,i}) := \min f(S_{i,0}), \quad x^{0,i} \in \arg\min f(S_{0,i}).$$
  3. Pour chaque $i \in I_0$ déterminer
  $$\beta_{0,i} = \beta(M_{0,i}) \leq \min f(S \cap M_{0,i}).$$
  4. Calculer
  $$\gamma_0 = \min_{i \in I_0} \gamma_{0,i}, \tag{2.2}$$
  $$x^0 \in \arg\min\{f(x^{0,i}, i \in I_0)\}, \tag{2.3}$$
  $$\beta_0 = \min_{i \in I_0} \beta_{0,i}. \tag{2.4}$$

- **Itération** k

  k.1 Supprimer tout $M_{k,i} \in \mathcal{M}_k$ vérifiant
  $$\beta_{k,i} \geq \gamma_0$$
  ou pour lequel on sait que $\min f(S)$ ne peut pas avoir lieu dans $M_{k,i}$. Soit $\mathcal{R}_k$ le collection des élements restant $M_{k,i} \in \mathcal{M}_k$.
  Si $\mathcal{R}_k = \emptyset$ **alors** STOP, x^k est une solution.

  k.2 Sélectionner $M_{k,i_k} \in \mathcal{R}_k$, choisir un ensemble fini des indices J_{k+1} et construire une partition
  $$\mathcal{M}_{k,i_k} = \{M_{k+1,i} : i \in J_{k+1}\}$$
  de M_{k,i_k} telle que $M_{k+1,i} \cap S \neq \emptyset$.

  k.3 Pour chaque $i \in J_{k+1}$ déterminer
  $$S_{k+1,i} \in M_{k+1,i} \cap S, S_{k+1,i} \neq \emptyset$$
  et
  $$\gamma_{k+1,i} = \gamma(M_{k+1,i}) := \min f(S_{k+1,i}), x^{k+1,i} \in \arg\min f(S_{k+1,i}).$$

k.4 Pour chaque $i \in J_{k+1,i}$ déterminer $\beta_{k+1,i}$ tel que

$$\beta_{k,i_k} \leq \beta_{k+1,i} \leq \min f(S \cap M_{k+1,i}).$$

k.5 Poser

$$\mathcal{M}_{k+1} = (\mathcal{R}_k \setminus M_{k,i_k}) \cup \mathcal{M}_{k,i_k}.$$

Soit I_{k+1} l'ensemble des indices tels que

$$\mathcal{M}_{k+1} = \{M_{k+1,i} : I \in I_{k+1}\}$$

est la partition actuelle.

Soient $\gamma_{k+1,i}, \beta_{k+1,i}, x^{k+1,i}$ les quantités correspondant à $M_{k+1,i}, i \in I_{k+1}$.

k.6 Calculer

$$\gamma_{k+1} = \min_{i \in I_{k+1}} \gamma_{k+1,i} \tag{2.5}$$

$$x^{k+1} \in \arg\min\{f(x^{k+1,i}, i \in I_{k+1})\} \tag{2.6}$$

$$\beta_{k+1} = \min_{i \in I_{k+1}} \beta_{k+1,i} \tag{2.7}$$

et retourner à l'itération $k+1$.

Remarque 2.1 *(i) Il faudrait déterminer $S_{k,i}, x^{k,i}, \beta_{k,i}$ de telle façon que ces bornes autant serrées que possible, avec un effort de calcul raisonnable. On parvient donc à un certain compromis.*

(ii) $\gamma_{k,i}, \beta_{k,i}$ sont des bornes inférieures et des bornes supérieures pour $\min f(S \cap M_{k,i})$ associées à chaque ensemble $M_{k,i}$ et γ_k, β_k sont des bornes inférieures et des bornes supérieures à $\min f(S)$ étant décroissantes et croissantes respectivement.

(iii) $\beta_{k,i} \geq \gamma_k$ indique que la solution x^k ne peut pas s'améliorer dans $M_{k,i}$ donc cet élément peut être éliminé.

Conditions de la convergence

La méthode SE converge dans le sens que chaque point d'accumulation de $\{x^k\}$ est une solution de (P). Évidemment, par la construction, on a

$$x^k \in S, \quad k = 0, 1, \ldots \tag{2.8}$$

$$\gamma_k \geq \gamma_{k+1} \geq \min f(S) \geq \beta_{k+1} \geq \beta_k \tag{2.9}$$

$$f(x^k) \geq f(x^{k+1}), \quad k = 0, 1, \ldots \tag{2.10}$$

Définition 2.1.1 *Une estimation de borne est dite cohérente (consistent) si, pour une suite décroissante quelconque $M_{k_q,i_{k_q}}$ générée par la procédure de séparation, i.e.*

$$M_{k_{q+1},i_{q+1}} \subset M_{k_q,i_q},$$

on a

$$\lim_{q\to\infty}(\gamma_{k_q,i_{k_q}} - \beta_{k_q,i_{k_q}}) = 0 \tag{2.11}$$

□

Puisque $\beta_{k_q,i_{k_q}} \leq \gamma_{k_q} \leq \gamma k_q, i_{k_q}$, la condition (2.11) peut s'écrire

$$\lim_{q\to\infty}(\alpha_{k_q} - \beta_{k_q,i_{k_q}}) = 0 \tag{2.12}$$

Par la monotonie et la bornitude des suites $\{\gamma_k\}, \{\beta_k\}$ on a

$$(f(x^k) = \gamma_k) \to \alpha, \quad \beta_k \to \beta, \quad \gamma \geq \min f(S) \geq \beta$$

Définition 2.1.2 *Une sélection est dite complète si pour chaque*

$$M \in \bigcup_{p=1}^{\infty}\bigcap_{k=p}^{\infty}\mathcal{R}_k$$

on a

$$\inf f(M \cap S) \geq \alpha.$$

Une sélection est dite ''borne-améliorante" (bound improving), si au moins après chaque nombre fini d'itérations, on a

$$M_{k,i_k} \in \arg\min\{\beta(M) : M \in \mathcal{R}_k\} \tag{2.13}$$

□

Théorème 2.1 *([23, 28]) Supposons que S est fermé et que* $\min f(S)$ *existe. Soit, dans le prototype, l'opération d'estimation de borne est cohérente. On a :*
(i) Si la sélection est complète, alors

$$\gamma := \lim_{k\to\infty}\alpha_k = \lim_{k\to\infty} f(x^k) = f(S) \tag{2.14}$$

(ii) Si la sélection est borne-améliorante, alors

$$\beta := \lim_{k\to\infty}\beta_k = \min f(S) \tag{2.15}$$

(iii) Si la sélection est complète, f est continue, et $\{x \in S : f(x) \leq f(x^0)\}$ est borné, alors chaque point d'accumulation de $\{x^k\}$ résout le problème (P).

Preuve : Voir [23, 28]. □

Noter que si l'estimation de borne est cohérente alors la sélection qui améliore des bornes est aussi complète, parce que $f(x) \leq \beta_{k_q}, \forall x \in S$ et lorsque (2.13) est appliqué, on a

$$\inf f(M \cap S) \geq \beta = \gamma$$

pour tout l'ensemble M.

2.2 Réalisation

Bien entendu, la réalisation d'un algorithme SE dépend du choix des opérations suivantes :
– Diviser M_{k,i_k}.
– Sélectionner M_{k,i_k}.
– Estmimer les bornes inférieures $\beta_{k,i}$.

2.2.1 Stratégie de division

Les éléments de partition M_k doivent être très simples pour qu'on puisse les manipuler facilement. Naturellement, on utilise les plus simples polyèdres comme des simplexes, des rectangles, des cônes (polyédraux) et des prismes. Il faut également diviser ces polyèdres de telle manière que la procédure de division soit exhaustive, ce qui est nécessaire pour assurer la convergence de la méthode SE.

Cette sous section détaille une division que nous utilisons dans nos algorithmes : la subdivision rectangulaire.

M_0 et toute partie de subdivision sont des n-rectangles dans $\mathbb{R}^n$. Le rectangle

$$M_0 = \prod_{i=1}^{n} [l_i, L_i]$$

le plus petit qui contient S (convexe) peut être déterminé en résolvant $2n$ problèmes convexes

$$l_i = \min\{x_i : x \in S\}, \quad L_i = \max\{x_i : x \in S\}, \quad i = 1, 2, \ldots, n.$$

Les processus de subdivision rectangulaire jouent un rôle important dans des méthodes de Séparation et Evaluation. L'approche de Phillips et Rosen [89] (voir également Kalantari et Rosen [31]) emploie la subdivision exhaustive, i.e. toutes les suite décroissantes des rectangles générées par l'algorithme tendra vers un point. Une bissection rectangulaire adaptative prétendue proposée dans Muu L.D. [34] semble être plus efficace parce que l'exhaustivité

n'est pas nécessaire pour la convergence. Dans Horst et Tuy [27], un concept de la subdivision rectangulaire normale (normal rectangular subdivision - NRS) a été présenté pour la classe des problèmes concaves séparables de minimisation qui inclut l'approche de Kalantari-Rosen et une subdivision proposées plus tôt dans Falk et Soland [13]. Intuitivement, la variante des algorithmes rectangulaires en utilisant w-subdivision et subdivision adaptative devrait converger plus rapidement que ceux qui emploient la subdivision exhautive, parce qu'ils tiennent compte des conditions du sous-problème relaxé courant.

2.2.2 Règle de sélection

Naturellement, on peut choisir à chaque itération

$$M_{k,i_k} \in \arg\min\{\beta(M) : M \in \mathcal{R}_k\}$$

qui satifait (2.13), i.e. cette sélection améliore des bornes. Pourtant, il y a bien d'autres règles qui n'utilisent pas explicitement cette propriété. Par exemple (cf. Tuy et al. [21]) :

(S1) Pour chaque M on définit $\mathcal{G}(M)$ - l'index d'étape où M est créé et à chaque itération, on choisit le plus ''vieil" ensemble, c'est-à-dire

$$M_{k,i_k} \in \arg\min\{\mathcal{G}(M) : M \in \mathcal{R}_k\}$$

(S2) Pour chaque M on définit une quantité $\delta(M)$ liée à la taille de M (e.g. le diamètre, le volume, etc.). Supposons que la division soit telle que, étant donné $\epsilon > 0$, on puisse toujours obtenir M avec $\delta(M) \leq \epsilon$ après un nombre fini de divisions de M. Alors, on choisit

$$M_{k,i_k} \in \arg\max\{\delta(M) : M \in \mathcal{R}_k\}$$

2.2.3 Estimation de borne

Étant donné un ensemble M_k. Pour estimer une borne inférieure, on va construire T_k tel que $M_k \cap S \subset T_k \subset M_k$ de manière que la borne $\beta(M_k) = \min f(T_k)$ soit estimée par des efforts raisonnables.

Définition 2.2.1 *Soit $\{T_k\}$ une suite d'ensembles de $\mathbb{R}^n$. Alors*

$$\overline{\lim_{k\to\infty}} T_k := \{x \in \mathbb{R}^n : x = \lim_{j\to\infty} x_{n_j}, x_{n_j} \in T_{n_j}\}$$

$$\underline{\lim_{k\to\infty}} T_k := \{x \in \mathbb{R}^n : x = \lim_{n\to\infty} x_n, x_n \in T_n \text{ pour tout sauf a nombre fini de } n \in \mathbb{N}\}$$

$$T = \lim_{k\to\infty} T_k \Leftrightarrow T = \overline{\lim_{k\to\infty}} T_k = \underline{\lim_{k\to\infty}} T_k$$

□

Une division va générer des suites décroissantes $\{M_k\}$ qui convergent vers un compact $M := \bigcap_k M_k$. notons $M_k \to M$.

Lemme 2.1 *([23]) Soit $S \in \mathbb{R}^n$ un compact et soit $f : \mathbb{R}^n \to \mathbb{R}$ continue. Alors l'estimation de borne est cohérente si, pour toute suite décroissante de compacts M_k, on a*
(i) $M_k \to M$ compact, $M \cap S \neq \emptyset$;
(ii) Il existe une suite de compacts T_k telle que

$$M_k \supseteq T_k \supseteq M_k \cap S, T_k \to M \cap S,$$

(iii)

$$\min f(T_k) \leq \beta(M_k) \leq \min f(M_k \cap S);$$

(iv)

$$\alpha(M_k) \to \min f(M \cap S);$$

L'estimation de borne présente toujours un dilemme entre la convergence et l'efficacité. Un algorithm SE va converger plus vite si on peut estimer des bornes d'une façon plus précise. Or, cela devrait coûter plus cher ce qui peut rendre l'algorithme moins efficace. L'utilisation de T_k donne une certaine souplesse dans l'estimation des bornes. Surtout, elle permet de combiner la technique de plans de coupe, en particulier AE, avec la technique SE. Cette approche paraît très prometteuse. Le premier algorithme de ce type a été proposé par Horst et al. [24] pour la minimisation concave. Les résultats numériques ont montré la supériorité de cet algorithme par rapport à ceux qui sont purement AE ou SE. Un de ses avantages, c'est que T_k peut être construit à l'aide de la programmation linéaire et l'algorithme SE se réduit ainsi à la résolution d'une suite de programmes linéaires.

2.3 B&B pour PLM01

Cette section aborde un cas particulier de B&B pour le problème linéaire en variables mixtes 0-1 (PLM01).

Considérons le problème (PLM01) suivant :

$$(PLM01) \qquad \begin{cases} \min f(x,y) = c^T x + d^T y \\ s.c. \\ Ax + By \leq b, \\ x \in \{0,1\}^n, \\ y \in \mathbb{R}^p_+. \end{cases} \tag{2.16}$$

Dénotons $z = (x, y)$. On désigne par

- $S := \{z = (x, y) : Ax + By \leq b, x \in \{0, 1\}^n, y \in \mathbb{R}^p_+\}$ l'ensemble de point réalisable de (PLM01),
- R_0 le domaine relaxé de S :

$$R_0 = \{z = (x, y) : Ax + By \leq b, x \in [0, 1]^n, y \in \mathbb{R}^p_+\},$$

- $\mathcal{R}$ l'ensemble des R_i à diviser,
- β_k (resp. γ_k) la borne inférieure (resp. la borne supérieure) de la valeur optimale à l'itération k.
- z^k la meilleure solution connue à l'itération k (tant que l'on ne connaît pas encore d'un point réalisable, $\gamma_k = \infty$),
- et $\overline{z}$ la solution optimale du problème relaxé de (PLM01).

Principe : On détermine, à chaque itération k, une borne inférieure β_k et une borne supérieure γ_k de la valeur optimale du problème (PLM01) tel que $\gamma_k - \beta_k \to 0$. On s'arrête quand $\gamma_k - \beta_k \leq \epsilon$. Plus précisément :
- la borne inférieure β_k est calculée par

$$\beta_k = \min\{\beta(R_i) : R_i \in \mathcal{R}\} = \min_{R_i \in \mathcal{R}}\{f(z) : z \in R_i\}.$$

- la meilleure borne supérieure à itération k est $\gamma_k = \min\{f(z_i) : \forall z_i \text{ connu dans } S\}$.
- à chaque itération k on considère l'ensemble $\mathcal{R} = \{R_i \in \mathcal{R} : \beta(R_i) < \gamma_k\}$ et on supprime tous les $R_i \in \mathcal{R}$ pour lequel $\beta(R_i) \geq \gamma_k$ (car on est sûr que R_i ne contient pas une solution optimale si $\beta(R_i) > \gamma_k$, dans le cas où $\beta(R_i) = \gamma_k$ on ne peut pas déterminer sur R_i une solution meilleure que z^k).

Algorithme 2.1 ***(BB)***

Étape 0 (Initialisation)

0.1. Calculer β_0 et $\overline{z}$ en résolvant le problème ($PLM01'$) relaxé de (PLM01).
- *Si $\overline{z} \in S$ alors STOP, $\overline{z}$ est une solution optimale de (PLM01).*
- *Sinon, poser $\mathcal{R} := \{R_0\}$, $\gamma_0 = +\infty$, $k := 1$.*

0.2. Aller à la procédure de choix (Étape 1).

Étape 1 (Procédure de choix)

1.2. Choisir un sous ensemble $R_k \in \mathcal{R}$ tel que

$$\beta_k = \min\{\beta(R_i) : R_i \in \mathcal{R}\}.$$

1.2. Aller à la procédure de séparation (Étape 2).

Étape 2 (Procédure de séparation)

2.1. Soit r un indice pour lequel $\overline{x}^k_r$ est non binaire. Séparer R_k en deux sous ensembles :

$$R_{k1} = \{z \in R_k : x_r = 0\}; \qquad R_{k2} = \{z \in R_k : x_r = 1\}.$$

2.2. Aller à la procédure d'évaluation (Étape 3).

Étape 3 (Procédure d'évaluation)

3.1. Calculer β_{k1} et β_{k2} en résolvant les deux problèmes $(PLM01'_{k1})$ et $(PLM01'_{k2})$.

3.2. Si $\overline{z}^{k1} \in S$ ($\overline{z}^{k2} \in S$ resp.), et $f(\overline{z}^{k1}) < \gamma_k$ (resp. $f(\overline{z}^{k2}) < \gamma_k$) alors mettre à jour γ_k

$$\gamma_k := f(\overline{z}^{k1}); z^k := \overline{z}^{k1}; \qquad \big(\textit{resp. } \gamma_k := f(\overline{z}^{k2}); z^k := \overline{z}^{k2}\big).$$

Étape 4 (Test d'optimalité)

4.1. $\mathcal{R} := \mathcal{R} \cup \{R_{ki} : f(\overline{z}^{ki}) < \gamma_k, i = 1, 2\} \setminus \{R_k\}$.

4.2. Si $\mathcal{R} = \emptyset$ alors STOP, z^k est une solution optimale, Sinon $k := k + 1$ et aller à la procédure de choix (Étape 1).

Pour les procédures de choix d'un sous ensemble R_k à séparer, on peut aussi utiliser une autre procédure de choix dite "exploration en profondeur d'abord" de l'arborescence (backtracking) : sélectionner R_k le plus récemment créé. Cette procédure possède deux avantages et un inconvénient :

- Obtenir rapidement une solution réalisable,
- Limiter au minimum les transferts entre la mémoire centrale et la mémoire périphérique.
- Ne pas tenir compte de la fonction d'évaluation.

On peut donc faire une combinaison de deux procédures présentées : on utilise une stratégie de backtracking pour "descendre" dans l'arborescence, mais quand on a trouvé un sous ensemble stérile on choisit le sous ensemble non stérile d'évaluation maximum

Pour les procédures de séparation, on a certaines règles à appliquer pour déterminer l'indice r :

- choisir r pour lequel l'un des deux programmes $(PLM01'_{k1})$ et $(PLM01'_{k2})$ n'a pas de solution réalisable (fausse séparation) ;
- choisir r pour lequel $\overline{x}_r(PLM01'_k)$ est la plus fractionnaire possible (sa valeur soit la plus voisine possible de 0.5).

Chapitre 3

Méthode d'Approximation Extérieure (AE)

Résumé *Ce chapitre présente le principe de l'algorithme d'approximation extérieure.*

3.1 Principe de AE

Considérons un problème d'optimisation du type "minimiser une fonction concave sur un ensemble convexe" sous la forme :

$$(CP) \quad \min\{f(x) : x \in S\}, \tag{3.1}$$

où S est un sous-ensemble convexe de $\mathbb{R}^n$ et f une fonction concave définie sur S. Les méthodes de plans de coupe constituent des outils de base dans différentes branches d'optimisation. Un plan est utilisé pour couper une partie de façon que cela n'exclut pas des points optimaux du problème d'optimisation. Nous rappelons dans la suite les coupes (linéaires) qui n'enlèvent aucun point réalisable.

Définition 3.1.1
(i) Une inégalité $l(x) \leq 0$ est dite valide pour (CP) si l'inégalité $l(x) \leq 0$ est satisfaite pour tout $x \in S$.
(ii) Étant donné un point $x^\star \in S$. L'inégalité $l(x) \leq 0$ est dite "coupe séparant strictement $x^\star$ de S" si et seulement si :

$$\begin{cases} l(x) \leq 0, & \text{pour tout } x \in S\,; \\ l(x^\star) \geq 0. \end{cases} \tag{3.2}$$

L'idée de base de AE est de relaxer et remplacer (CP) par une suite de problèmes (CP_k) :

$$(CP_k) \quad \min\{f(x) : x \in D_k \supset S\}$$

(avec D_k est un certain ensemble contenant S) qui sont plus faciles à résoudre et dont les solutions convergent vers une solution optimale de (CP).

Le prototype de la méthode de AE appliqué au problème (CP) peut être exprimé comme suit :

Prototype AE

- **Etape 0** (Initialisation)
  1. Choisir $D_1 \supset S$;
  2. Poser $k \leftarrow 1$;
- **Etape 1** (Résoudre le problème relaxé)
  1. Résoudre le problème relaxé (CP_k) :

$$(CP_k) \quad \min\{f(x) : x \in D_k\}$$

     pour obtenir x^k ;
  2. Si $x^k \in S$, x^k résoud (CP). S'arrêter ;
  3. Sinon, passer à l'étape 2 ;
- **Etape 2** (Ajouter coupe)
  1. Construire une coupe de la forme (3.2) séparant strictement x^k de S ;
  2. Poser $D_{k+1} \leftarrow D_k \cap \{x : l_k(x) \leq 0\}$;
  3. Poser $k \leftarrow k + 1$;
  4. Retourner à l'étape 1 ;

Pour la réalisation d'un algorithme AE, deux questions importantes sont posées :
- Déterminer des problèmes (CP_k) qui sont faciles à résoudre ?
- Construire $l_k(x)$ pour que la suite $\{x_k\}$ converge vers une solution de (CP) ?

3.2 Convergence de AE

3.2.1 Approximation polyédrale

Soient D_1 un ensemble convexe polyédral et $l_k(x) := \langle a_k, x\rangle + b_k$ une fonction affine. Alors $H_k := \{x \in \mathbb{R}^n : l_k(x) = 0\}$ est un hyperplan séparant strictement x^k de S. Tout ensemble D_k construit ainsi, est polyédral et on obtient une suite décroissante de polyèdres qui approximent S. Par construction des $a_k \neq 0$, on peut supposer que $\|a_k\| = 1$ (en prenant $l_k(x)/\|a_k\|$).

Définition 3.2.1 *On dit qu'une suite d'hyperplans* $\{H_k\}$ *:*

$$H_k := \{x : l_k(x) = \langle a^k, x\rangle + b_k = 0\}$$

converge vers un hyperplan $H := \{x \in I\!R^n : l(x) = \langle a, x\rangle + b = 0\}$ *si*

$$a^k \to a, \quad b_k \to b \quad (k \to \infty).$$

Théorème 3.1 *Dans le processus d'Approximation Extérieure, soit* D_1 *un polyèdre compact et supposons que pour chaque sous-suite convergente* $\{x^q\}$ *de* $\{x^k\}$ *telle que* $x^q \to \overline{x} \in S$*, et* $H_q \to H$ *on a* H *sépare strictement* $\overline{x}$ *de* S*. Alors tout point d'accumulation de* x^k *appartient à* S *et résout ainsi (CP).*

Preuve : Soit $\overline{x}$ un point d'accumulation de x^k et $x^{k_p} \to \overline{x}$. Comme $\|a^k\| = 1$, on a donc $\{a^k\}$ est bornée. D'autre part

$$-\langle a^k, x\rangle \geq b_k \geq -\langle a^k, x^k\rangle \quad \forall x \in S.$$

Puisque D_1 est un compact, il existe une constante M telle que $\|x\| \leq M$ pour tout $x \in D_1$. Par suite,

$$|\langle a^k, x^k\rangle| \leq M \quad |\langle a^k, x\rangle| \leq M \quad \forall x \in S$$

ce qui implique

$$b_k \leq M,$$

i.e., $\{b_k\}$ est aussi bornée. En prenant une sous-suite, on peut donc supposer $a^{k_p} \to a \quad b_{k_q} \to b$, autrement dit, $\{H_k\} \to H$. Supposons que $\overline{x} \notin S$. Alors H sépare strictement $\overline{x}$ de S i.e., $l(\overline{x}) = \langle a, x\rangle + b > 0$. D'autre part, $l_{k_p}(x^{k_q+r}) \leq 0 \quad \forall r \geq 0$ et en faisant $r \to \infty$, on obtient $l_{k_q}(\overline{x}) \leq 0$. Lorsque $\{H_{k_q}\} \to H$, on a $l(\overline{x}) \leq 0$ ce qui est absurde. Donc $\overline{x} \in S$. Puisque $f(x^k) \leq f(x) \quad \forall x \in S$ i.e., $\overline{x}$ est une solution optimale de (CP). □

3.2.2 Généralisation

En dehors des coupes linéaires (déterminées par une fonction affine) bien connues en programmation convexe, les coupes continues (des coupes déterminées par des fonctions continues non nécessairement linéaire) sont aussi utilisées (cf., par exemple, R.Horst, N. V. Thoai et H. Tuy [26]). Nous avons :

Théorème 3.2 *(Horst-Thoai-Tuy [26]) Dans le processus d'Approximation extérieure, soit D1 un compact. Supposons que* l_k *est continue et pour chaque suite* $\{x_q\} \subset \{x_k\}$ *telle que* $x_q \to \overline{x}$ *il existe une sous-suite* $\{x_{ql}\}$ *de* $\{x_q\}$ *telle que*
1. $l_{ql}(x) \to l(x)\ \forall x$.
2. $l_{ql}(x^{ql}) \to l(\overline{x})$ *où* $l(\overline{x})$ *est tel que* $l(\overline{x}) = 0$ *implique* $x \in S$.
Alors tout point d'accumulation de x_k *appartient à* S *et resout (CP).*

Preuve : Soit x_q une suite convergeant vers $\overline{x}$ - un point d'accumulation de $\{x_k\}$ et $\{x_{ql}\}$ la sous-suite qui satisfait les hypothèses 1 et 2. Puisque $l_{ql}(x^{ql+r}) \leq 0 \; \forall r$, en faisant $r \to +\infty$ on a $l_{ql}(\overline{x}) \leq 0$. Par suite, $l(\overline{x}) \leq 0$. D'autre part, $l_{ql}(x^{ql}) > 0 \; \forall l$ et $l_{ql}(x^{ql}) \to l(\overline{x})$ implique $l(\overline{x}) \geq 0$. Donc $l(\overline{x}) = 0$ et en vertu de 2 $x \in S$. D'une façon analogue comme ci-dessus on peut conclure que x est une solution optimale de (CP). □

En particulier, si $l_k(.)$ sont des fonctions Lipchitz, on a :

Théorème 3.3 *(Tuy [17]) Supposons que $l_k(.)$ est continue et*
1. $|l_k(z) - l_k(x)| \leq L|z - x| \; \forall z, x \in D1$ (L étant une constante).
2. Pour chaque suite $\{x^q\} \subset \{x^k\}$ telle que

$$x_q \to \overline{x}, \; l_q(x^q) \to 0$$

on a $s \in S$. Alors tout point d'accumulation de x_k appartient à S et résout (CP).

Preuve : En vertu de 1), on a $l_q(x^q) \leq l_q(\overline{x}) + L|x^q - \overline{x}|$. Par construction, $l_q(x^j) \leq 0, \; \forall j > q$, ce qui entraîne $l_q(\overline{x}) \leq 0$. Donc $0 < l_q(x^q) \leq L|x^q - x|$. Quand $x_q \to \overline{x}$ on a $l_q(x^q) \to 0$ et selon 2) $\overline{x} \in S$.

□

3.3 Coupes pour la programmation linéaire en variables mixtes

Cette section présente quelques coupes populaires pour la programmation linéaire en variables mixtes.

3.3.1 Coupe mixte de Gomory

On considère ici un ensemble Q de points de $\mathbb{R}^n$ défini par des contraintes mixtes :

$$Q := \{(x, y) \in \mathbb{Z}^n_+ \times \mathbb{R}^p_+ : Ax + by = b\}.$$

Pour tout $u \in \mathbb{R}^m_+$, on note f_0 la partie fractionnaire de ub (i.e. $f_0 = ub - \lfloor ub \rfloor$) et f_i, $(i = 1, ..., n + p)$ p), la partie fractionnaire de $(uA)_i$. L'inégalité

$$\sum_{i:f_i \leq f_0} f_i x_i + \frac{f_0}{1 - f_0} \sum_{i:f_i > f_0} (1 - f_i) x_i + \sum_{i:(uB)_i \geq 0} (uB)_i y_i + \frac{f_0}{1 - f_0} \sum_{i:(uB)_i > 0} (uB)_i y_i \geq f_0, \quad (3.3)$$

appelée coupe mixte de Gomory, est valide [15].

Comme pour le cas des variables entières, Gomory a pu établir un algorithme de résolution exacte des problèmes mixtes basé sur la séparation de ces coupes dans les années 1960.

3.3.2 Coupe MIR

On considère la coupe MIR, acronyme de "mixed integer rounding cut". Considérons l'ensemble $X = \{(x, y) \in \mathbb{Z}^n_+ \times \mathbb{R}_+ : a^T x - y \leq b\}$ avec $a \in \mathbb{Z}^n$, $b \in \mathbb{R}$. On note $f_i = a_i - \lfloor a_i \rfloor$ et $f_0 = b - \lfloor b \rfloor$.

Lemme 3.1 *L'inégalité*

$$\sum_{i=1}^{n} (\lfloor a_i \rfloor + \frac{(f_i - f_0)^+}{1 - f_0}) x_i - \frac{1}{1 - f_0} y \leq \lfloor b \rfloor \tag{3.4}$$

est valide pour conv(X), où $v^+ = \max(0, v)$ pour $v \in \mathbb{R}$. L'inégalité 3.4 est appelée l'inégalité MIR.

Preuve : Voir [73]. □

Lemme 3.2 *Les inégalités MIR impliquent les coupes mixtes de Gomory quand elles sont appliquées à l'ensemble $X = \{(x, y^-, y^+) \in \mathbb{Z}^n_+ \times \mathbb{R}^2_+ : a^T x + y^+ - y^- = b\}$.*

Preuve : Le lemme 3.1 donne :

$$\sum_{i=1}^{n} (\lfloor a_i \rfloor + \frac{(f_i - f_0)^+}{1 - f_0}) x_i - \frac{1}{1 - f_0} y^- \leq \lfloor b \rfloor. \tag{3.5}$$

La soustraction de l'égalité originale $a^T x + y^+ - y^- = b$ et (3.5) donne la coupe mixte de Gomory. □

Nemhauser et Wolsey [76] étudient les inégalités MIR dans le cas plus général. Ils prouvent que les inégalités MIR fournissent une description complète pour n'importe quel polyèdre mixte 0-1. Marchand et Wolsey [72] proposent des algorithmes utilisant cette coupe pour résoudre la programmation linéaire en variables mixtes.

Deuxième partie

La Programmation Linéaire en variables mixtes 0-1 et ses applications aux réseaux de télécommunication et à l'ordonnancement

Chapitre 4

DCA et Méthodes globales basées sur DCA pour la programmation linéaire en variables mixtes 0-1

Résumé *Dans ce chapitre, nous proposons des méthodes globales basées sur DCA pour la programmation linéaire en variables mixtes 0-1 (PLM01). Particulièrement, nous reconsidérons la méthode de coupe introduite par Nguyen V.V et Le Thi H. A. (notée DCACUT) et nous la perfectionnons. Les expériences numériques comparatives entre les algorithmes proposés sont rapportées.*

Les méthodes considérés dans ce chapitre sont basées sur DCA. Avant d'entrer dans le détail les méthodes, la première section est consacrée à la description de DCA pour (PLM01) [54].

4.1 DCA pour la résolution de PLM01

4.1.1 Reformulation

Considérons la programmation linéaire en variables mixtes 0-1 sous la forme :

$$(PLM01) \quad \begin{cases} \min f(x,y) = c^T x + d^T y \\ s.c. \\ Ax + By \leq b, \\ x \in \{0,1\}^n, \\ y \in \mathbb{R}^p_+. \end{cases} \tag{4.1}$$

où A et B sont deux matrices de taille $(m \times n)$ et $(m \times p)$ (resp.), m - le nombre de contraintes, n - le nombre de variables binaires, p - le nombre de variables continues, $(c,d) \in \mathbb{R}^n \times \mathbb{R}^p$ -

les vecteurs coûts et $b \in \mathbb{R}^m$.

On définit les deux ensembles S (l'ensemble des solutions réalisables) et K (l'ensemble des solutions de la relaxation linéaire standard de l'ensemble réalisable) suivants :

$$S := \{(x, y) \in \{0, 1\}^n \times \mathbb{R}^p_+ \ : \ Ax + By \leq b\},$$

$$K := \{(x, y) \in [0, 1]^n \times \mathbb{R}^p_+ \ : \ Ax + By \leq b\}.$$

Considérons la fonction p définie par :

$$p(x, y) \equiv p(x) := \sum_{j=1}^{n} \min\{x_j, 1 - x_j\}.$$

Il est clair que :

- p est concave, et que $-p$ est convexe polyédrale sur $\mathbb{R}^n \times \mathbb{R}^p$,
- $z \in K$ est un point réalisable de (PLM01) si et seulement si $p(z) = 0$,
- p est non négative, finie sur K et on a :

$$\begin{aligned} S &:= \{(x, y) \in \{0, 1\}^n \times \mathbb{R}^p_+ : Ax + By \leq b\} \\ &\equiv \{(x, y) \in K : \ p(x) = 0\} \\ &\equiv \{(x, y) \in K : \ p(x) \leq 0\}. \end{aligned}$$

Par conséquent, $(PLM01)$ peut s'écrire sous la forme d'un programme non convexe en variables continues défini comme suit :

$$(NCP1) \quad \min\{c^T x + d^T y : (x, y) \in K, p(x) \leq 0\}. \tag{4.2}$$

Théorème 4.1 *Soit K un polyèdre non vide, borné de $\mathbb{R}^n$. Soit f une fonction concave finie sur K et soit p une function concave finie et non négative sur K. Alors il existe un nombre fini $t_0 \geq 0$ tel que les deux problèmes suivants sont équivalents avec $t \geq t_0$(dans le sens où ils ont le même ensemble de solutions optimales) :*

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

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

De plus, si l'ensemble de points extrêmes $V(K)$ de K est contenu dans $\{x \in K, p(x) \leq 0\}$, alors $t_0 = 0$, sinon $t_0 = \min\{\frac{f(x)-\alpha(0)}{\xi}\}$, où $\xi := \min\{p(x) : x \in V(K), p(x) > 0\} > 0$.

Preuve : Voir H.A. Le Thi, T. Pham Dinh et M. Le Dung [67]. □

Compte tenu du Théorème 4.1, il existe un nombre fini t_0 tel que $\forall t \geq t_0$, (NCP1) est équivalent au problème suivant :

$$(NCP2) \quad \min\{c^T x + d^T y + tp(x) : (x, y) \in K\}.$$

La différence entre les deux problèmes $(NCP1)$ et $(NCP2)$ est que dans le premier la non convexité apparaît aux contraintes alors que dans le deuxième, elle réside dans la fonction objectif. On peut reformuler $(NCP2)$ comme une programmation DC, ensuite utiliser DCA pour le résoudre.

4.1.2 DCA appliqué au problème pénalisé de PLM01

Le problème $(NCP2)$ peut se récrire comme

$$\min\{\chi_K(z) + c^T x + d^T y + tp(x) \ : \ z = (x, y) \in \mathbb{R}^n \times \mathbb{R}^p\}, \tag{4.3}$$

où

$$\chi_K(z) := \begin{cases} 0 & \text{si z} \in \text{K}, \\ +\infty & \text{sinon} \end{cases} \tag{4.4}$$

est la fonction indicatrice de K.

On note $g(z) := \chi_K(z)$ et

$$h(z) := -c^T x - d^T y + t(-p)(z) = -c^T x - d^T y + t\sum_{j=1}^{n} \max\{-x_j, x_j - 1\}. \tag{4.5}$$

Comme K est convexe, $g := \chi_K$ est convexe [93]. Aussi, h est convexe car p est concave. Dès lors, le problème $(NCP2)$ est équivalent au problème DC suivant :

$$\min\{g(z) - h(z) \ : \ z = (x, y) \in \mathbb{R}^n \times \mathbb{R}^p\}. \tag{4.6}$$

Selon le schéma général de DCA, on doit calculer les deux suites $\{u^k\}$ et $\{z^k\}$ telles que

$$u^k \in \partial h(z^k); \quad z^{k+1} \in \partial g^*(u^k).$$

Soit $z = (x, y)$ un point quelconque de $\mathbb{R}^n \times \mathbb{R}^p$, à partir de la définition (4.5) de la fonction h, un vecteur sous gradient $u = (\sigma, \varsigma) \in \partial h(x, y)$ est choisit par :

$$\sigma_i = -c_i + \begin{cases} t, & \text{si } x_i \geq 0.5; \\ -t, & \text{sinon} \end{cases} \quad \text{et} \quad \varsigma_j = -d_j \quad (i = 1, \ldots, n; j = 1, \ldots, p). \tag{4.7}$$

Ensuite, le calcul de z^{k+1} revient à la résolution du problème suivant :

$$\min\{g(z) - \langle z, u\rangle : z \in \mathbb{R}^n \times \mathbb{R}^p\}. \tag{4.8}$$

DCA appliqué au problème (4.6) peut se décrire comme suit :

Algorithme 4.1 *DCA appliqué au problème (4.6)*
Étape 1. *Soit $z^0 = (x^0, y^0) \in \mathbb{R}^n \times \mathbb{R}^p$ un point initial. Poser $k \leftarrow 0$;*
Étape 2. *Calculer $u^k = (\sigma^k, \varsigma^k) \in \partial h(x^k, y^k)$ via (4.7).*
Étape 3. *Calculer*

$$z^{k+1} = (x^{k+1}, y^{k+1}) \in \arg\min\{-\langle(\sigma^k, \varsigma^k), (x, y)\rangle : (x, y) \in K\}. \tag{4.9}$$

Étape 4.
Si le critère d'arrêt est vérifié, STOP ;
Sinon retourner à l'étape 2 ;

Grâce au théorème sur la convergence de DCA pour la programmation polyédrale (cf. [54, 59, 43]), on a directement les propriétés suivantes de l'Algorithme 4.1.

Théorème 4.2 *La convergence de l'Algorithme 4.1*
i) L'algorithme 4.1 génère une suite $\{z^k = (x^k, y^k)\}$ contenue dans l'ensemble de sommets $V(K)$ de K telle que la suite $\{g(x^k, y^k) - h(x^k, y^k)\}$ soit décroissante.
ii) Si à l'itération r le point (x^r, y^r) satisfait $x^r \in \{0, 1\}^n$ alors (x^k, y^k) satisfait aussi $x^k \in \{0, 1\}^n$ pour tout $k \geq r$.
iii) La suite $\{(x^k, y^k)\}$ converge vers une solution $(x^\star, y^\star) \in V(K)$ après un nombre fini d'itérations et $(x^\star, y^\star)$ est un point critique du problème (4.6). De plus, si $x_i^\star \neq 0.5 \quad \forall i = 1, \ldots, n$ alors $(x^\star, y^\star)$ est une solution locale du problème (4.6).

Preuve : Voir, par exemple, H.A. Le Thi *et al.* [43]. □

4.2 DCACUT amélioré

4.2.1 Une inégalité valide

Dans cette partie, on va déterminer une inégalité valide pour tous les points de S à partir d'une solution locale de la fonction pénalité p sur K.

Soit $z^\star \in K$. Notons :

$$I_0(z^\star) = \{j \in \{1, \ldots, n\} : x_j^\star \leq 1/2\ \} \quad , \quad I_1(z^\star) = \{1, \ldots, n\} \setminus I_0(z^\star).$$

et

$$l_{z^\star}(z) \equiv l_{z^\star}(x) = \sum_{i \in I_0(z^\star)} x_i + \sum_{i \in I_1(z^\star)} (1 - x_i).$$

Les deux lemmes suivants seront utilisés dans la suite.

Lemme 4.1 *[78] Étant donné $z^\star \in K$, on a :*
(i) $l_{z^\star}(x) \geq p(x)\ \forall x \in I\!R^n$.
(ii) $l_{z^\star}(x) = p(x)$ *si et seulement si*

$$(x, y) \in R(z^\star) \ := \ \{(x, y) \in K \ : \ x_i \leq 1/2, i \in I_0(x^\star);\ x_i \geq 1/2, i \in I_1(x^\star)\ \}.$$

Preuve : (i) Par définition de la fonction $l_{z^\star}$, on a

$$l_{z^\star}(x) = \sum_{i \in I_0(x^\star)} x_i + \sum_{i \in I_1(x^\star)} (1 - x_i) \geq$$

$$\geq \sum_{i \in I_0(x^\star)} \min\{x_i, 1 - x_i\} + \sum_{i \in I_1(x^\star)} \min\{x_i, 1 - x_i\}$$

$$= \sum_{i \in I_0(x^\star)} x_i + \sum_{i \in I_1(x^\star)} (1 - x_i) \equiv p(x).$$

(ii) Il est facile de constater que

$$l_{z^\star}(x) = p(x) \Leftrightarrow \begin{cases} x_i = \min\{x_i, 1 - x_i\} & \forall i \in I_0(x^\star) \\ 1 - x_i = \min\{x_i, 1 - x_i\} & \forall i \in I_1(x^\star) \end{cases}$$

$$\Leftrightarrow \begin{cases} x_i \leq 1/2 & \forall i \in I_0(x^\star) \\ x_i \geq 1/2 & \forall i \in I_1(x^\star) \end{cases} \quad \Leftrightarrow x \in R(z^\star).$$

□

Lemme 4.2 *[78] Soit $z^\star = (x^\star, y^\star)$ un minimum local de la fonction p sur K, alors l'inégalité*

$$l_{z^\star}(x) \geq l_{z^\star}(x^\star) \tag{4.10}$$

est valide pour tout $(x, y) \in K$.

Preuve : Comme $z^\star$ est un minimum local de la fonction p sur K, il existe un voisinage $U(z^\star) \in \mathbb{R}^n \times \mathbb{R}^p$ de $z^\star$ tel que

$$p(x^\star) \leq p(x),\ \forall (x,y) \in K \cap U(z^\star).$$

Comme $l_{z^\star}(x^\star) = p(x^\star)$ et $l_{z^\star}(x) \geq p(x)$ (Lemme 4.1), on a :

$$l_{z^\star}(x^\star) = p(x^\star) \leq p(x) \leq l_{z^\star}(x),\ \forall (x,y) \in K \cap U(u^\star).$$

Alors $z^\star = (x^\star, y^\star)$ est un minimum local de la fonction $l_{z^\star}(x)$ sur l'ensemble convexe K. Comme $l_{z^\star}(x)$ est affine et K un polyèdre convexe, $z^\star$ doit être un minimum global de la fonction $l_{z^\star}(x)$ sur K, i.e.,

$$l_{z^\star}(x^\star) \leq l_{z^\star}(x),\ \forall (x,y) \in K.$$

□

Théorème 4.3 *[78] Il existe un nombre fini $t_1 \geq 0$ tel que, pour tout $t > t_1$, si $z^\star = (x^\star, y^\star) \in V(K) \setminus S$ est un minimum local du problème (4.6) alors*

$$l_{z^\star}(x) \geq l_{z^\star}(x^\star),\ \forall (x,y) \in K. \tag{4.11}$$

Preuve : Soit $z^\star$ un minimum local de (4.6) alors il existe un voisinage de $z^\star$, $U(z^\star) \in \mathbb{R}^n \times \mathbb{R}^p$, tel que :

$$c^T x^\star + d^T y^\star + tp(x^\star) \leq c^T x + d^T y + tp(x),\ \forall (x,y) \in K \cap U(z^\star).$$

Puisque $l_{z^\star}(x^\star) = p(x^\star)$ et $l_{z^\star}(x) \geq p(x)$ (Lemme 4.1), on a :

$$c^T x^\star + d^T y^\star + tl_{z^\star}(x^\star) = c^T x^\star + d^T y^\star + tp(x^\star) \leq$$

$$\leq c^T x + d^T y + tp(x) \leq c^T x + d^T y + tl_{z^\star}(x),\ \forall (x,y) \in K \cap U(z^\star).$$

Autrement dit, le point $z^\star = (x^\star, y^\star)$ est un minimum local de la fonction linéaire $c^T x + c_2^T y + tl_{z^\star}(x)$ sur l'ensemble convexe K. Dès lors, $z^\star$ est une solution optimale globale du programme linéaire :

$$\min\{c^T x + d^T y + tl_{z^\star}(x)\ :\ z \in K\},$$

i.e.,

$$c^T x^\star + d^T y^\star + tl_{z^\star}(x^\star) \leq c^T x + d^T y + tl_{z^\star}(x),\ \forall (x,y) \in K.$$

De manière équivalente, on a :

$$t[l_{z^\star}(x^\star) - l_{z^\star}(x)] \leq c^T x + d^T y - (c^T x^\star + d^T y^\star),\ \forall (x,y) \in K. \tag{4.12}$$

Pour tout $w = (x', y') \in V(K) \setminus S$, on note par :

$$S(w) = \{(x, y) \in V(K) \text{ tel que : } l_w(x') - l_w(x) > 0\},$$

et

$$W = \{w \in V(K) \setminus S \text{ tel que : } S(w) \neq \emptyset\}.$$

Si $W = \emptyset$, i.e., $S(w) = \emptyset,\ \forall w \in V(K) \setminus S$ alors

$$l_w(x) \geq l_w(x'),\ \forall (x, y) \in V(K).$$

Cette condition implique que

$$l_w(x) \geq l_w(x'),\ \forall (x, y) \in K.$$

Alors le lemme est vérifié pour tout $t > t_1 := 0$.

Supposons que $W \neq \emptyset$. Notons

$$\alpha = \min_{w \in W} \min_{(x,y) \in S(w)} \{l_w(x') - l_w(x)\},$$

$$\beta = \max_{w=(x',y') \in W} \max_{(x,y) \in S(w)} \{c^T x + d^T y) - (c^T x' + d^T y')\}.$$

Comme $S(w)$ et W sont bornés et non vides, on a $0 < \alpha < +\infty$ et $\beta < +\infty$.

Choisissons $t_1 = \max\{0, \beta/\alpha\}$. Il suffit de montrer que $z^\star \notin W$ pour tout minimum $z^\star$ du problème (NCP2).

Si $\beta \leq 0$ alors $t_1 = 0$ et grâce à la condition (4.12) on a $t[l_{z^\star}(x^\star) - l_{z^\star}(x)] \leq 0$. Autrement dit,

$$l_{z^\star}(x^\star) - l_{z^\star}(x) \leq 0,\ \forall t > t_1 = 0,$$

i. e., le lemme est vérifié. Sinon, à partir de la condition (4.12), on a, $\forall (x, y) \in K$,

$$t[l_{z^\star}(x^\star) - l_{z^\star}(x)] \leq c^T x + d^T y - (c^T x^\star + d^T y^\star) \leq \beta.$$

Par conséquent,

$$l_{z^\star}(x^\star) - l_{z^\star}(x) \leq \frac{\beta}{t} < \frac{\beta}{t_1} = \alpha,$$

i.e. $z^\star \notin W$ pour tout $z^\star$. □

Soit $z^\star = (x^\star, y^\star)$ une solution du problème (4.6). Si $x_i^\star \neq 0.5 \quad \forall i = 1, \ldots, n$ (i.e. $z^\star$ est un minimum local de p), selon les théorèmes 4.2 et 4.3, on a l'inégalité valide (4.11). Dans le cas où il existe un i tel que $x_i^\star = 0.5$, Nguyen V.V. ([78]) a proposé de diviser le présent problème en deux. L'un correspond à $x_{i^\star} = 0$ et l'autre correspond à $x_{i^\star} = 1$. Cette procédure comme pseudo-Branch and Bound est très chère et peut cycler dans certains cas.

Pour éviter ce phénomène, on pourrait modifier légèrement cette procédure en considérant le problème linéaire suivant :

$$\xi = l_{z^\star}(\tilde{z}) = \min\{l_{z^\star}(z) : z \in K\}. \tag{4.13}$$

Si $\xi = l_{z^\star}(z^\star)$, l'inégalité (4.11) est encore valide. Si $\xi < l_{z^\star}(z^\star)$, en appliquant DCA au problème

$$(Q) \quad \min\{p(x) : (x, y) \in K\}, \tag{4.14}$$

à partir du point initial $\tilde{x}$, on obtient une solution $\hat{x}$ vérifiant $p(\hat{x}) \leq p(\tilde{x}) \leq l_{z^\star}(\tilde{z}) = \xi < l_{z^\star}(z^\star) = p(x^\star)$. Dans ce qui suit, un tel point $\tilde{x}$ est appelé un *point potentiel* par rapport à $p(x^\star)$. A partir de $\hat{x}$, on peut encore construire une inégalité valide selon le Lemme (4.2).

4.2.2 Construction d'une coupe à partir d'une solution non réalisable

Soit $z^\star$ une solution qui n'est pas un point réalisable de S tel que $l_{z^\star}(z) \geq l_{z^\star}(z^\star) \quad \forall z \in K$. Dans ce cas, il existe au moins un indice $j_0 \in \{1, \ldots, n\}$ tel que $x^\star_{j_0}$ soit non entier. On distingue les deux cas suivants :

Cas 1 : La valeur $l_{z^\star}(z^\star)$ n'est pas entière. Comme $l_{z^\star}(z)$ est entier pour tout $z \in S$, on a immédiatement :

$$\begin{cases} l_{z^\star}(z) \geq \rho := \lfloor l_{z^\star}(z^\star) \rfloor + 1, \quad \forall z \in S \\ l_{z^\star}(z^\star) \leq \rho. \end{cases} \tag{4.15}$$

Autrement dit, l'inégalité

$$l_{z^\star}(z) \geq \rho \tag{4.16}$$

est une coupe séparant strictement $z^\star$ de S.

Cas 2 : La valeur $l_{z^\star}(z^\star)$ est entière. Il est possible qu'il existe des points réalisables z' tels que $l_{z^\star}(z') = l_{z^\star}(z^\star)$. Si un tel point existe, on pourrait mettre à jour la meilleure solution de (PLM01) et également améliorer la borne supérieure de la valeur optimale. Sinon, pour tout $z \in S$, on a $l_{z^\star}(z) > l_{z^\star}(z^\star)$. C'est-à-dire que

$$l_{z^\star}(z) \geq l_{z^\star}(z^\star) + 1 \tag{4.17}$$

est une coupe séparant $z^\star$ de S.

On considère ci-dessous une procédure (appelée *Procédure P*) qui fournit soit une telle coupe, soit un tel point réalisable, soit un point potentiel.

On note $I_F(z^\star) := \{i \in I \ : \ x^\star_i \notin \{0, 1\}\}$ l'ensemble des indices binaires sur lesquels $x^\star$ est non entière. Soit $i_\star \in I_F(z^\star)$. Considérons les deux problèmes linéaires suivants :

$$\min\{l_{x^\star}(x) \ : \ (x, y) \in K; \ x^\star_{i_\star} = 0\} \quad (P_1),$$

et

$$\min\{l_{x^\star}(x) \ : \ (x, y) \in K; \ x^\star_{i_\star} = 1\} \quad (P_2).$$

Soient $u^1 = (x^1, y^1)$ et $u^2 = (x^2, y^2)$ les solutions de P_1 et P_2, respectivement. Notons $\eta_1 := l_{x^\star}(x^1), \quad \eta_2 := l_{x^\star}(x^2)$ et $\eta = \min\{\eta_1, \eta_2\}$.

Si $\eta > l_{x^\star}(x^\star)$ alors pour tout $z \in S$, on a $l_{x^\star}(x) > l_{x^\star}(x^\star)$ i.e., l'inégalité (4.17) est une coupe séparant $z^\star$ de S.

Si $\eta = l_{x^\star}(x^\star)$, alors il y a deux cas possibles : soit $l_{x^\star}(x^\star) = \eta_1 = \eta_2$ (cas 2.1), soit $l_{x^\star}(x^\star) = \min\{\eta_1, \eta_2\} < \max\{\eta_1, \eta_2\}$ (cas 2.1).

Cas 2.1 : On note $\overline{v} = (\overline{x}, \overline{y})$ la solution du problème (P_1) ou (P_2) qui n'appartient pas à $R(z^\star)$. Par le Lemme 4.1 on a :

$$p(x^\star) = l_{x^\star}(x^\star) = l_{x^\star}(\overline{x}) > p(\overline{x}).$$

A partir de $\overline{v}$ on peut trouver un point $\hat{z}$ tel que $p(\hat{x}) \leq p(\overline{x}) < p(x^\star)$.

Cas 2.2 : $p(x^\star) = \min\{\eta_1, \eta_2\} < \max\{\eta_1, \eta_2\}$.
Si $\eta_1 < \eta_2$ (de façon analogue si $\eta_2 < \eta_1$) on a $u^1 \in R(z^\star)$ ((ii), Lemme 4.1). Remplaçons $z^\star$ par u^1, K par $K \cap \{(x, y) \in K : x_{i_\star} = 0\}$ et $I_F(z^\star)$ par $I_F(z^\star) \setminus \{i_\star\}$ et répétons le même processus.

La procédure précédente peut se décrire comme suit :

Procédure P

Entrée : Un point non réalisable $z^\star$ tel que $l_{x^\star}(x) \geq l_{x^\star}(x^\star)$ et $l_{x^\star}(x^\star) \in \mathbb{Z}$.
Sortie : Soit une coupe, soit une solution réalisable, soit un point potentiel.

0. Poser $k \leftarrow 0$; $P_0 \leftarrow K$.
1. Poser $I_F \leftarrow I_F(x^\star)$; Poser $k \leftarrow k + 1$; Choisir $i_k \in I_F$.
2. Résoudre les deux sous-problèmes linéaires :

$$(P_{k1}) \quad \eta_1 = l_{x^\star}(u^1) = \min\{l_{x^\star}(x) \ : \ (x, y) \in P_k, \ x_{i_k} = 0\},$$

$$(P_{k2}) \quad \eta_2 = l_{x^\star}(u^2) = \min\{l_{x^\star}(x) \ : \ (x, y) \in P_k, \ x_{i_k} = 1\}.$$

Si P_{k1} (P_{k2}) n'a pas de solution, poser $\eta_1 := +\infty$ ($\eta_2 := +\infty$, resp.).
Poser $\eta \leftarrow \min\{\eta_1, \eta_2\}$.
Si $\eta = +\infty$ alors passer à l'étape 3. Sinon, passer à l'étape 4.
3. Aucun des deux problèmes a de solution($\eta = +\infty$).
Si $k = 1$ alors le domaine K ne contient pas de point réalisable. **STOP**.
Si $k > 1$ alors l'inégalité $l_{z^\star}(z) \geq l_{z^\star}(z^\star)$ est une coupe, **STOP**.
4. ($\eta < +\infty$).
Si u^1 ou u^2 est réalisable, **STOP**.
Sinon, passer à l'étape 5.
5.
Si $l_{z^\star}(x^\star) < \eta$ alors l'inégalité $l_{z^\star}(z) \geq l_{z^\star}(z^\star)$ est une coupe, **STOP**.
Sinon, passer à l'étape 6.

6. $l_{z^\star}(x^\star) = \eta$
Si $\eta_1 = \eta_2 = l_{z^\star}(z^\star)$ alors déterminer un point potentiel $\overline{z} \in \{u^1, u^2\}$ par rapport à $p(x^\star)$, **STOP**.
Sinon, passer à l'étape 7.
7.
Si $\eta_1 < \eta_2$; poser $P_{k+1} \leftarrow \{(x, y) \in P_k \ : \ x_{i_k} = 0\}$;
Sinon ($\eta_1 > \eta_2$), poser $P_{k+1} \leftarrow \{(x, y) \in P_k \ : \ x_{i_k} = 1\}$
8. Poser $I_F \leftarrow I_F \setminus \{i_k\}$;
Si $I_F = \emptyset$, **STOP**. Sinon, passer à étape 1.

Lemme 4.3 *Soit $z^\star$ un point non réalisable tel que $l_{z^\star}(x) \geq l_{z^\star}(x^\star)$ sur K et $l_{z^\star}(x^\star) \in \mathbb{Z}$. La procédure P s'arrête après un nombre fini d'itérations et on doit obtenir :*
(i) soit une coupe séparant $z^\star$ de S,
(ii) soit une solution réalisable,
(iii) soit un point potentiel par rapport à $p(x^\star)$,
(iv) soit une affirmation que le domaine K est vide ou pas.

Preuve : Évidente. □

4.3 Algorithme DCACUT amélioré

Nous présentons dans cette partie la version améliorée d'algorithme DCACUT introduit dans [78] que nous noterons également DCACUT.

4.3.1 Idée de base l'algorithme DCACUT

Soit α la valeur optimale du problème ($PLM01$). Nous allons construire les deux suites $\{\beta_k\}$ et $\{\gamma_k\}$, dont l'une est croissante et l'autre décroissante, respectivement. La première joue le rôle des bornes inférieures et la seconde - celui des bornes supérieures de α telles que :

$$\begin{cases} (i) & \beta_k \leq \alpha \leq \gamma_k \quad \forall k = 1, 2, \ldots \\ (ii) & \lim\limits_{k\to\infty} \beta_k = \lim\limits_{k\to\infty} \gamma_k = \alpha. \end{cases} \tag{4.18}$$

Ces deux suites sont construites simultanément. A chaque étape, on cherche à obtenir soit une solution réalisable, soit une coupe coupant une partie du domaine relaxé courant. Si une

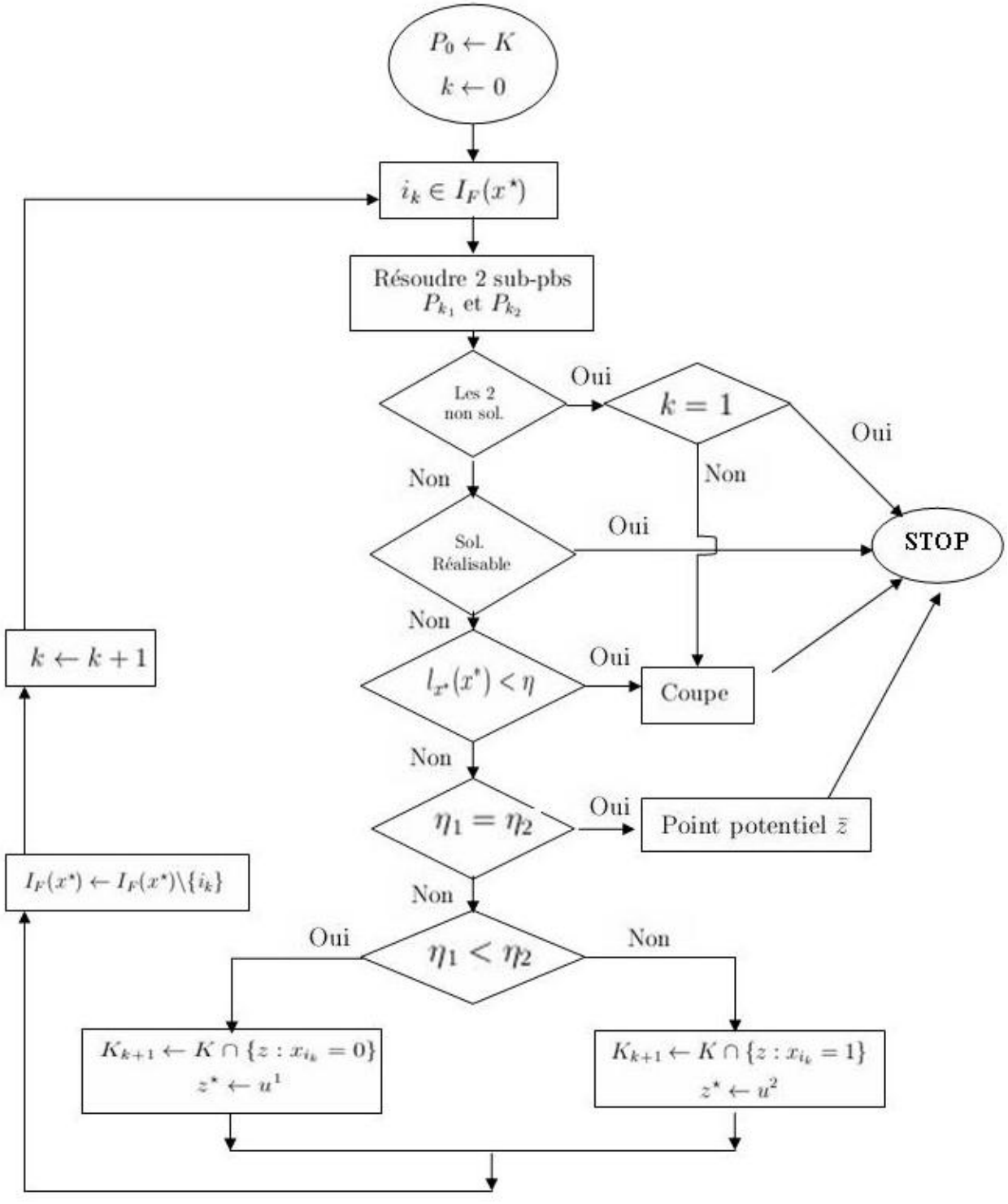

FIG. 4.1 – Procédure P

solution réalisable est trouvée, on met à jour la meilleure solution z^k et la borne supérieure γ_k. Sinon, on prend la coupe en considération pour réduire le domaine relaxé.

Posons $K_0 = K$. Soit K_k la relaxation et $z^k = (x^k, y^k)$ la meilleure solution connue à l'étape k.

Si aucune solution réalisable n'est connue, alors on pose $z^k = \emptyset$. Les deux suites sont construites de la manière suivante :

$$\beta_k = \min\{c_1^T x + c_2^T y \ : \ (x, y) \in K_k\}$$

et

$$\gamma_k = \begin{cases} +\infty & \text{si } z^k = \emptyset \\ c_1^T x^k + c_2^T y^k & \text{sinon.} \end{cases}$$

Remarque 4.1
(i) Par construction, la condition (i) de (4.18) est vérifiée.
(ii) Les propriétés de la convergence des deux suites $\{\beta_k\}$ et $\{\gamma_k\}$ (dans (ii) de (4.18)) dépendent de la façon dont on trouve les solutions réalisables et les coupes utilisées. Dans la suite, nous utilisons la coupe construite à partir de la solution de la fonction pénalité obtenue par DCA.

4.3.2 Description de l'algorithme DCACUT

4.3.2.1 Mise à jour de la borne supérieure et de la meilleure solution connue

Supposons qu'à l'étape k, $z^k = (x^k, y^k)$ soit la meilleure solution réalisable connue et soit γ_k la borne supérieure ($\gamma_k = c_1^T x^k + c_2^T y^k$). Par convention, si une solution réalisable n'est pas encore trouvée, on pose $z^k = \{\emptyset\}$ et $\gamma_k = +\infty$.

DCA appliqué au problème (4.6) fournit souvent des solutions réalisables. De plus, une solution réalisable du problème (NCP2) peut être trouvée de différentes façons, par exemple, grâce à la *Procédure-P*.

Lorsqu'une solution réalisable est trouvée, nous mettons à jour la meilleure solution réalisable et la borne supérieure.

4.3.2.2 Séparation d'une solution réalisable

Le but de la séparation d'une solution réalisable est d'éliminer les solutions qui sont déjà trouvées.

Soit z^k une solution réalisable. Après avoir mis à jour la meilleure solution réalisable et la borne supérieure, nous allons séparer cette solution du domaine réalisable dans le but d'éliminer les solutions et des minimums locaux qui sont déjà trouvés par DCA. Pour cela, on ajoute l'inégalité suivante aux contraintes :

$$h(z) \equiv h(x) := \sum_{j:x_j^k=0} x_j + \sum_{j:x_j^k=1} (1 - x_j) \geq 1. \tag{4.19}$$

Il est facile de vérifier que :

$$\begin{cases} h(z) \geq 1, \quad \forall z \in S \setminus \{z^k\}, \\ h(z^k) = 0 < 1. \end{cases} \tag{4.20}$$

4.3.2.3 Algorithme DCACUT

L'algorithme DCACUT pour résoudre le problème $(PLM01)$ peut être décrit comme suit :

Algorithme 4.2

Etape 0. (Initialisation)

+ Poser $k \leftarrow 0$ *;* $K_0 \leftarrow K$ *;* $z^0 \leftarrow \emptyset$

+ Poser $\beta_0 \leftarrow -\infty$ *(borne inf.),* $\gamma_0 \leftarrow +\infty$ *(borne sup.).*

Etape 1. (Résoudre le problème linéaire relaxé)

+ Résoudre le problème linéaire

$$\beta_k := \min\{c^T x + d^T y \ : \ (x, y) \in K_k\}$$

pour obtenir la borne inf. β_k *et la solution optimale* z_{LP}^k.

+ Si $z_{LP}^k \in S$ *alors* $z^k \leftarrow z_{LP}^k$. *STOP.*

Etape 2. (Appliquer DCA)

+ Appliquer DCA au problème $\min\{c^T x + d^T y + tp(x) \ : \ (x, y) \in K_k\}$ *pour obtenir sa solution* z_{DCA}^k *;*

+ Passer à l'étape 3 ;

Etape 3.

+ Si $z_{DCA}^k \in S$*, poser* $z \leftarrow z_{DCA}^k$ *et passer à l'Etape 6 ;*

+ Si $z_{DCA}^k \notin S$

- *Si* $x_{DCA\text{-}i}^k \neq 0.5 \quad \forall i$*, alors vérifier si* $l_{z_{DCA}^k}(z_{DCA}^k) \in \mathbb{Z}$.
  - · *Si* $l_{z_{DCA}^k}(z_{DCA}^k) \notin \mathbb{Z}$*, créer un coupe via (4.15) ;*
  - · *Sinon, appeler la Procédure P ;*
    - ◇ *Si la Procédure P fournit une solution réalisable* $z^\star$*, passer à l'étape 5 ;*
    - ◇ *Si la Procédure P fournit une coupe, passer à l'étape 7 ;*
    - ◇ *Si la Procédure P fournit un point potentiel, passer à l'étape 4 ;*
    - ◇ *Sinon, STOP ;*

- *Si* $\exists i, \quad x^k_{DCA\text{-}i} = 0.5$ *: résoudre le problème linéaire suivant :*

$$\xi = l_{z^k_{DCA}}(\overline{z}) = \min\{l_{z^k_{DCA}}(z) : z \in K_k\}.$$

  · *Si* $\xi < l_{x^k_{DCA}}(x^k_{DCA})$*, passer à l'étape 4.*
  · *Si* $\xi = l_{x^k_{DCA}}(x^k_{DCA})$*, passer à l'étape 7.*

Etape 4. (Appliquer DCA à partir d'un point potentiel)
+ *Appliquer DCA au problème* $\min\{p(x) : z \in K_k\}$ *pour obtenir* z^k_{DCA} *;*
+ *Passer à l'étape 3 ;*

Etape 5. (Mettre à jour la borne sup. et la meilleure solution connue)
+ *Si* $c^T x^k + d^T y^k < \gamma_k$*, mettre à jour* $z_{Opt} \leftarrow z^k$; $\gamma_k \leftarrow c^T x^k + d^T y^k$;
+ *passer à l'étape 6 ;*

Etape 6. (Séparer la solution réalisable)
+ *Déterminer la contrainte de séparation* $\phi(z) \geq \nu$ *via (4.19) ;*
+ *Poser* $K_{k+1} \leftarrow K_k \cap \{z : \phi(z) \geq \nu\}$ *et passer à l'étape 8 ;*

Etape 7. (Ajouter la coupe)
+ *Ajouter la coupe aux contraintes et passer à l'étape 8 ;*

Etape 8. (Test d'arrêt)
+ *Si* $\gamma_k - \beta_k \leq \epsilon$*, STOP ; Sinon, poser* $k \leftarrow k + 1$; *et passer à l'étape 1.*

La convergence de l'algorithme DCACUT est exprimée par le théorème ci-dessous :

Théorème 4.4 *[78] Soit K un polyèdre convexe borné. L'algorithme DCACUT converge vers une solution optimale après un nombre fini d'itérations.*

Preuve : Supposons au contraire, que l'algorithme génère une suite de coupes $\{l_k(x) \geq \eta_k\}_{k=1}^{\infty}$ de la forme (4.15) dont chacune sépare respectivement le point minimum $v^k \in K_k, \ (k = 1, 2, \ldots)$. On a donc :

$$\begin{cases} l_k(x^{k+q}) \geq \eta_k, \ \forall q = 1, 2 \ldots \\ l_k(x^k) < \eta_k. \end{cases}$$

D'autre part, à partir de la définition de la fonction $l_k, \ (k = 1, 2, \ldots)$, il est facile de voir que le nombre de coupes introduites est fini ($\leq 2^n$). Par conséquent, il existe une sous-suite de coupes, appelée $\{l_{k_s}(x) \geq \eta_{k_s}\}_{s=1}^{\infty}$, telle que toutes les fonctions l_{k_s} ont la même forme (notée par $\hat{l}$). Dès lors, pour tout $s \geq 1$, l'inégalité $\hat{l}(x) \geq \eta_{k_s}$ est valide pour tout point $v^{k_s+q}, \ \forall q \geq 1$, mais violée par v^{k_s}. Par conséquent,

$$\hat{l}(x^{k_1}) < \eta_{k_1} \leq \hat{l}(x^{k_2}) < \eta_{k_2} \leq \ldots < \eta_{k_s} \leq \hat{l}(x^{k_{s+1}}) < \ldots.$$

Alors la suite $\{\eta_{k_s}\}$ est strictement croissante. Comme η_k est entier, on a donc :

$$\lim_{t \to \infty} \eta_{s_t} = \lim_{t \to \infty} \hat{l}(x^{s_t}) = +\infty.$$

Ce qui est une contradiction puisque K_0 est borné. □

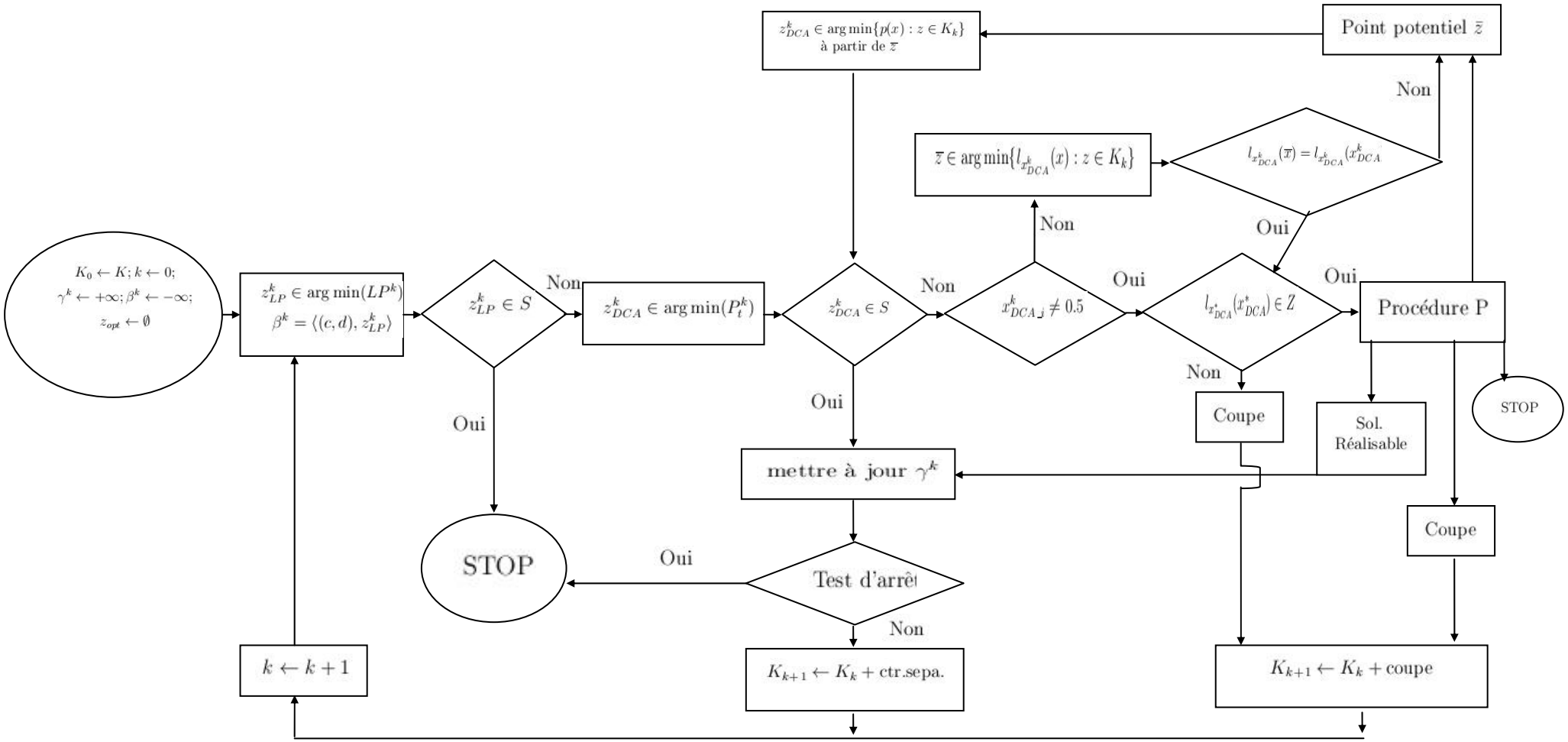

FIG. 4.2 – Schéma de DCACUT

4.4 B&B combinée avec DCA

Dans le schéma B&B (présenté dans 2.3), le problème de trouver une bonne borne supérieure joue un rôle important pour son efficacité. Plusieurs travaux dans la littérature ont montré que la valeur obtenue par DCA est souvent très proche/coïncide avec la valeur optimale (notamment quand on lance DCA à partir d'un bon point initial). Nous combinons B&B et DCA dans le but de trouver un bon point initial pour DCA et de prouver/trouver la globalité d'une solution.

Rappelons que PLM01 est transformé en une programmation DC. Nous appliquons DCA au (NCP2) (revoir la section 4.1.2) pour déterminer la borne supérieure de (PLM01).

En utilisant les même notations que dans la section 2.3, l'algorithme B&B combiné avec DCA peut se décrire comme suit :

Algorithme 4.3 ***(BBDCA)***

Etape 0 (Initialisation)

0.1. Calculer β_0 et $\overline{z}$ en résolvant le problème $(PLM01')$ relaxé de (PLM01).
- *Si $\overline{z} \in S$ alors STOP, $\overline{z}$ est une solution optimale de (PLM01).*
- *Sinon, poser $\mathcal{R} := \{R_0\}$ et $k := 0$.*

0.2. Appliquer DCA au problème (NCP2) à partir de $\overline{z}$ pour obtenir z^k_{DCA}.
- *Si $z^k_{DCA} \in S$,alors $\gamma_k := f(z^k_{DCA})$.*
- *Sinon, $\gamma_k := +\infty$.*

0.3. Poser $k := 1$ et aller à la procédure de choix (Etape 1).

Etape 1 (Procédure de choix)

1.2. Choisir un sous ensemble $R_k \in \mathcal{R}$ tel que

$$\beta_k = \min\{\beta(R_i) : R_i \in \mathcal{R}\}.$$

1.2. Aller à la procédure de séparation (Etape 2).

Etape 2 (Procédure de séparation)

2.1. Soit r un indice pour lequel $\overline{x}^k_r$ est non binaire. Séparer R_k en deux sous ensembles :

$$R_{k1} = \{z \in R_k : x_r = 0\}; \qquad R_{k2} = \{z \in R_k : x_r = 1\}.$$

2.2. Aller à la procédure d'évaluation (Etape 3).

Etape 3 (Procédure d'évaluation)

3.1. Calculer β_{k1} et β_{k2} en résolvant les deux problèmes $(PLM01'_{k1})$ et $(PLM01'_{k2})$.

3.2. Appliquer DCA au (NCP2) à partir de $\overline{z}^{k1}$ et $\overline{z}^{k2}$ pour obtenir z^{k1}_{DCA} et z^{k2}_{DCA}.

3.3. Si $z^{k1}_{DCA} \in S$ (resp. $z^{k2}_{DCA} \in S$) et $f(z^{k1}_{DCA}) < \gamma_k$ (resp. $f(z^{k2}_{DCA}) < \gamma_k$) alors mettre à jour γ_k

$$\gamma_k := f(z^{k1}_{DCA}); z^k := z^{k1}_{DCA}; \qquad (\textit{resp. } \gamma_k := f(z^{k2}_{DCA}); z^k := z^{k2}_{DCA}).$$

3.4. Aller à l'étape 4.

Etape 4 (Test d'optimalité)

4.1. $\mathcal{R} := \mathcal{R} \cup \{R_{ki} : f(\overline{z}^{ki}) < \gamma_k, i = 1, 2\} \setminus \{R_k\}$.

4.2. Si $\mathcal{R} = \emptyset$ *alors STOP,* z^k *est une solution optimale, Sinon* $k := k + 1$ *et aller à la procédure de choix (Etape 1).*

Notons que, dans l'algorithme 4.3, l'étape 3.2 n'est pas toujours effectuée à chaque itération. Un faible nombre de redémarrages de DCA à partir des meilleures solutions permettent d'atteindre des solutions globales et alors on épargne le temps de calcul.

4.5 Branch and Cut combinée avec DCA

La méthode de "Branch-and-cut" consiste en une combinaison d'une méthode de coupe avec un algorithme de B&B. Elle peut être considérée comme un algorithme de B&B où on essaye d'obtenir des relaxations plus serrées (à chaque noeud de l'arbre de recherche). Nous combinons B&B et DCACUT. Cette combinaison a un double intérêt : d'une part, DCA fournit une bonne borne supérieure, et d'autre part, une coupe peut être construite à partir de la solution du problème pénalisé donnée par DCA.

Le schéma de Branch-and-Cut combiné avec DCA peut se décrire comme suit :

Algorithme 4.4 ***(BCDCAaP)***

Étape 0 (Initialisation)

0.1. Calculer β_0 *et* $\overline{z}$ *en résolvant le problème* $(PLM01')$ *relaxé de (PLM01).*

- *Si* $\overline{z} \in S$ *alors STOP,* $\overline{z}$ *est une solution optimale de (PLM01).*
- *Sinon, poser* $\mathcal{R} := \{R_0\}$.

0.2. Poser $k := 0$ *et appliquer DCA au problème (NCP2) à partir de* $\overline{z}$ *pour obtenir* z^k_{DCA}.

- *Si* $z^k_{DCA} \in S$*,alors* $\gamma_k := f(z^k_{DCA})$.
- *Sinon,* $\gamma_k := +\infty$.

0.3. (Construire la coupe et l'ajouter aux contraintes)

+ *Si* $z^k_{DCA} \in S$*, poser* $z^k \leftarrow z^k_{DCA}$ *et ajouter la contrainte de séparation* $\phi(z) \geq \nu$ *via (4.19) ;*
+ *Si* $z^k_{DCA} \notin S$
  - *Si* $x^k_{DCA\text{-}i} \neq 0.5 \quad \forall i$*, alors vérifier si* $l_{z^k_{DCA}}(z^k_{DCA}) \in \mathbb{Z}$.
    Si $l_{z^k_{DCA}}(z^k_{DCA}) \notin \mathbb{Z}$*, créer un coupe via (4.15) ; Sinon, appeler la Procédure P ;*
  - *Si* $\exists i, \quad x^k_{DCA\text{-}i} = 0.5$*, résoudre le problème linéaire suivant :*

$$\xi = l_{z^k_{DCA}}(\overline{z}) = \min\{l_{z^k_{DCA}}(z) : z \in K_k\}.$$

Si $\xi = l_{x^k_{DCA}}(x^k_{DCA})$*, ajouter la coupe via (4.15).*

0.4. Poser $k := 1$ *et aller à la procédure de choix (Étape 1).*

Étape 1 (Procédure de choix)

1.2. Choisir un sous ensemble $R_k \in \mathcal{R}$ tel que

$$\beta_k = \min\{\beta(R_i) : R_i \in \mathcal{R}\}.$$

1.2. Aller à la procédure de séparation (Étape 2).

Étape 2 (Procédure de séparation)

2.1. Soit r un indice pour lequel $\overline{x}_r^k$ est non binaire. Séparer R_k en deux sous ensembles :

$$R_{k1} = \{z \in R_k : x_r = 0\}; \qquad R_{k2} = \{z \in R_k : x_r = 1\}.$$

2.2. Aller à la procédure d'évaluation (Étape 3).

Étape 3 (Procédure d'évaluation)

3.1. Calculer β_{k1} et β_{k2} en résolvant les deux problèmes $(PLM01'_{k1})$ et $(PLM01'_{k2})$.

3.2. Appliquer DCA au (NCP2) à partir de $\overline{z}^{k1}$ et $\overline{z}^{k2}$ pour obtenir z_{DCA}^{k1} et z_{DCA}^{k2}.

3.3. (Construire la coupe et l'ajouter aux contraintes). Pour $j = 1, 2$,

+ Si $z_{DCA}^{k_j} \in S$, poser $z^k \leftarrow z_{DCA}^{k_j}$ et ajouter la contrainte de séparation $\phi(z) \geq \nu$ via (4.19) ;

+ Si $z_{DCA}^{k_j} \notin S$

- Si $x_{DCA\text{-}i}^{k_j} \neq 0.5 \quad \forall i$, alors vérifier si $l_{z_{DCA}^{k_j}}(z_{DCA}^{k_j}) \in \mathbb{Z}$.

Si $l_{z_{DCA}^{k_j}}(z_{DCA}^{k_j}) \notin \mathbb{Z}$, créer un coupe via (4.15) ; Sinon, appeler la Procédure P ;

- Si $\exists i, \quad x_{DCA\text{-}i}^{k_j} = 0.5$, résoudre le problème linéaire suivant :

$$\xi = l_{z_{DCA}^{k_j}}(\overline{z}) = \min\{l_{z_{DCA}^{k_j}}(z) : z \in K_k\}.$$

Si $\xi = l_{x_{DCA}^{k_j}}(x_{DCA}^{k_j})$, ajouter la coupe via (4.15).

3.4. Si $z_{DCA}^{k1} \in S$ (resp. $z_{DCA}^{k2} \in S$) et $f(z_{DCA}^{k1}) < \gamma_k$ (resp. $f(z_{DCA}^{k2}) < \gamma_k$) alors mettre à jour γ_k

$$\gamma_k := f(z_{DCA}^{k1}); z^k := z_{DCA}^{k1}; \qquad (\text{resp. } \gamma_k := f(z_{DCA}^{k2}); z^k := z_{DCA}^{k2}).$$

3.5. Aller à l'étape 4.

Étape 4 (Test d'optimalité)

4.1. $\mathcal{R} := \mathcal{R} \cup \{R_{ki} : f(\overline{z}^{ki}) < \gamma_k, i = 1, 2\} \setminus \{R_k\}$.

4.2. Si $\mathcal{R} = \emptyset$ alors STOP, z^k est une solution optimale, Sinon $k := k + 1$ et aller à la procédure de choix (Étape 1).

Dans les étapes 0.3 et 3.3 de l'algorithme BCDCAaP, parfois, la coupe est construite par la procédure P. Cependant, le temps de calcul de cette procédure peut être grand à cause de la résolution de plusieurs sous-problèmes générés. Nous avons une variante de cet algorithme dans laquelle on n'appelle pas la procédure P (noté BCDCAsP).

4.6 Expériences numériques

Dans la suite, nous présentons les expériences numériques des algorithmes proposés précédemment. Ils ont été exécutés sur un Intel Core 2CPU de 1.86 Ghz, 2GB RAM en C. Pour résoudre la programmation linéaire, nous avons utilisé le logiciel CPLEX 11.2. Les données que nous avons utilisées sont les Benchmark (prises de MIPLIB 3.0 [113]). Les algorithmes s'arrêtent si $gap = \frac{UB-LB}{LB} < 5\%$ (UB (resp. LB) est la borne supérieure (resp. la borne inférieure)) ou le temps de calcul $t > 1h$.

4.6.1 Comparaison entre les algorithmes de AE

Nous comparons DCACUT avec GMI et DCAGMI, où GMI est l'algorithme de AE utilisant la coupe mixte de Gomory et DCAGMI est l'algorithme combinant la coupe mixte de Gomory et la coupe de DCA. Dans la combinaison DCAGMI, après chaque 5 additions de la coupe de DCA, si la borne inférieure n'augmente pas une quantité de ϵ, on ajoute alternativement une coupe mixte de Gomory.

Le tableau 4.1 présente les résultats de la comparaison. Nous utilisons les notations suivantes :
- UB : la borne supérieure,
- LB : la borne inférieure,
- $time$: le temps de calcul (en secondes),
- $sepa$: le nombre de contraintes de séparation ajoutées,
- $rbUB$: le nombre de contraintes $cx + dy \leq UB$ ajoutées,
- $cutDCA$: le nombre de coupes DCA ajoutées,
- $cutGMI$: le nombre de coupes GMI ajoutées,
- $demi$: le nombre de fois qu'on rencontre la composante 1/2,
- $still$: le nombre de fois qu'on rencontre la composante 1/2 et qu'on génère régulièrement l'inégalité valide,
- NbP : le nombre de fois on appelle la Procédure P,
- Sub : le nombre de sous-problèmes à résoudre.

On observe que :
- Dans la plupart de cas, GMI ne donne pas la borne supérieure après une heure (la limite du temps d'exécution de l'algorithme).
- DCACUT est souvent meilleur pour fournir la borne supérieure ce qui est tout à fait logique compte tenu de l'efficacité de DCA.
- Pour obtenir la borne inférieure, DCAGMI n'est pas meilleur que GMI mais meilleur que DCACUT.
- Dans tous les cas, on rencontre toujours la composante 1/2 plusieurs fois. Autrement dit, c'est un cas fréquent. Cela confirme l'intérêt indiscutable de traitement de ce cas effectué dans notre travail.
- Le nombre de fois que la procédure P est utilisée est grand. Le nombre de sous-problèmes

Ins	Méthode	UB	LB	time	sépa	rbUB	cutDCA	cutGMI	demi	still	NbP	Sub
Mod010	DCACUT	6877.00	6532.08	1.562	1	1	0	0	0	0	1	2
	DCAGMI	6877.00	6532.08	1.562	1	1	0	0	0	0	1	2
	GMI	6548.00	6548.00	265.574	-	-	-	174	-	-	-	-
Mod008	DCACUT	308.00	291.04	10.61	4	6	21	0	8	8	12	64
	DCAGMI	309.00	291.29	2.515	3	3	7	1	3	3	2	10
	GMI	$+\infty$	291.36	>3600.000	-	-	-	1939	-	-	-	-
Pp08acuts	DCACUT	10920.00	5480.61	>3600.000	14	1	646	0	419	416	932	6802
	DCAGMI	9400.00	5606.14	>3600.000	60	12	539	146	320	319	926	5660
	GMI	$+\infty$	5503.36	>3600.00	-	-	-	4669	-	-	-	-
Pp08a	DCACUT	10180.00	2748.35	>3600.000	24	7	582	0	501	489	1401	10884
	DCAGMI	11070.00	3687.11	>3600.000	1	1	706	141	87	87	623	8314
	GMI	$+\infty$	3704.91	>3600.000	-	-	-	1908	-	-	-	-
P0201	DCACUT	7855.00	6875.00	>3600.000	3	1	451	0	363	179	286	2059
	DCAGMI	8045.00	7340.71	>3600.000	2	2	522	105	64	60	249	1945
	GMI	$+\infty$	7352.77	>3600.000	-	-	-	2383	-	-	-	-
Mas74	DCACUT	14372.87	10490.59	>3600.000	17	1	672	0	479	411	638	7381
	DCAGMI	12879.86	10504.71	>3600.000	8	5	587	144	426	346	745	9966
	GMI	$+\infty$	10482.79	>3600.000	-	-	-	3373	-	-	-	-

TAB. 4.1 – Résultats comparatifs entre les algorithmes de AE

qu'on doit résoudre est aussi grand. C'est pourquoi le temps de calcul est grand.

4.6.2 Comparaison entre les algorithmes de AE et les algorithmes du type de B&B

Utilisant les mêmes Benchmark problèmes, nous comparons les algorithmes de AE avec les algorithmes par séparation et évaluation (BB, BBDCA, BCDCAsP, BCDCAaP). Les résultats sont rapportés dans le tableau 4.2.

Nous observons que :
- Les deux premières instances sont résolues par presque tous les algorithmes (sauf GMI résolvant le mod008). Le temps de calcul des algorithmes du type B&B est plus petit que celui des algorithmes de AE.
- Pour les dernières instances, presque tous les algorithmes s'arrêtent à cause de dépassement de temps de calcul limité (une heure).
- Les bornes inférieures et les bornes supérieures obtenues par les algorithmes du type B&B sont meilleures que celles des algorithmes de AE.

Alors, nous pouvons conclure que, pour ces exemples, les algorithmes BB, BBDCA, BCDCAsP, BCDCAaP sont beaucoup plus efficaces que les algorithmes de AE. L'efficacité est dans les deux sens : la qualité de la solution et le temps de calcul. Les quatre algorithmes les plus efficaces sont examinés dans la suite.

4.6.3 Comparaison entre les algorithmes du type de séparation et évaluation

Nous testons quatre algorithmes BB, BBDCA, BCDVAsP et BCDCAaP pour les instances de Benchmark (MIPLIB 3.0). Les résultats sont présentés dans les tableaux 4.3.

Quelques notations supplémentaires sont utilisées :
- fIt : le nombre d'itérations au moment où DCA donne la première solution réalisable,
- $fVal$: la valeur de la fonction objectif en itération fIt,
- $ftime$: le temps de calcul quand l'algorithme fournit $fVal$,
- $rat = \frac{fVal-LB}{LB}$.

Nous trouvons que :
- pour pp08acuts et pp08a, BB ne fournit pas de solution après une heure de calcul.
- BCDCAsP et BCDCAaP fournissent la meilleure valeur de la fonction objectif.
- BCDCAsP, BCDCAaP sont efficaces pour mod010, mod008 et p0201.
- le temps de calcul pour lequel DCA (dans BBDCA, BCDCAsP, BCDCAaP) donne une

Instance		DCACUT	DCAGMI	GMI	BB	BBDCA	BCDCAsP	BCDCAaP
Mod010	UB	6877.00	6877.00	6548.00	6551.00	6557.00	6557.00	6861.00
	LB	6532.08	6532.08	6548.00	6533.17	6532.08	6532.08	6532.08
	time	1.56	1.56	265.57	510.42	1.25	1.66	0.80
Mod008	UB	308.00	309.00	$+\infty$	307.00	307.00	307.00	307.00
	LB	291.04	291.29	291.36	298.95	291.64	291.64	291.64
	time	10.61	2.52	>3600.00	13.00	0.39	0.84	1.34
Pp08acuts	UB	10620.00	9400.00	$+\infty$	$+\infty$	11640.00	7970.00	7970.00
	LB	5480.61	5606.14	5503.36	6444.61	6432.75	6269.38	6227.84
	time	>3600.00	>3600.00	>3600.00	>3600.00	>3600.00	>3600.00	>3600.00
Pp08a	UB	10180	11070	$+\infty$	$+\infty$	11640.00	9370.00	9780.00
	LB	2748.35	3687.11	3704.91	5254.58	5254.11	4968.99	4855.29
	time	>3600.00	>3600.00	>3600.00	>3600.00	>3600.00	>3600.00	>3600.00
P0201	UB	7855.00	8045.00	$+\infty$	7665.00	7925.00	7775.00	7715.00
	LB	6875.00	7340.71	7352.71	7545.00	7528.75	7519.00	7336.00
	time	>3600.00	>3600.00	>3600.00	745.69	979.97	500.94	335.33
Mas74	UB	14372.87	12879.86	$+\infty$	12469.19	12469.19	11985.18	11985.18
	LB	10490.59	10504.71	10482.80	10991.14	10990.59	10893.22	10845.52
	time	>3600.00	>3600.00	>3600.00	>3600.00	>3600.00	>3600.00	>3600.00

TAB. 4.2 – Résultats comparatifs entre les algorithmes de AE et les algorithmes du type B&B

solution réalisable est souvent petit (voir la colonne $ftime$). A ce moment là, la valeur de la fonction objectif est proche de la valeur optimale pour les instances mod010, mod 008 et p0201 (voir la colonne rat).
- Le gain du temps de calcul peut être divisé par 500 entre BB et les algorithmes globaux combinés avec DCA (Instance mod008). Cela confirme l'intérêt indiscutable de l'utilisation DCA dans les algorithmes globaux.
- BBDCA est souvent efficace.

Nous analysons dans le détail deux algorithmes BCDDCAsP et BCDCAaP via le tableau 4.4 suivant :

- le cas dans lequel on rencontre la composante 1/2 de la solution est fréquent (la colonne *demi*). Mais on peut encore construitre l'inégalité valide dans plusieurs cas (la colonne *still*).
- BCDCAaP doit résoudre beaucoup de sous problème linéaire pour quelques instances (voir la colonne Sub). Cela peut être la raison pour laquelle le temps de calcul d'algorithme est grand.
- le nombre de coupes générées est grand.

4.7 Conclusion

Dans ce chapitre, nous avons proposé une amélioration de DCACUT. Nous avons également fait la comparaison entre les algorithmes combinés pour la résolution de PLM01. Ce sont les algorithmes du type "approximation extérieure" (DCACUT, DCAGMI, GMI) et les algorithmes du type "séparation et évaluation" (BB, BBDCA, BCDCAsP et BCDCAaP). Nous concluons que :
- Pour DCACUT, nous avons bien traité le cas où la composante 1/2 apparaît régulièrement.
- Les algorithmes du type "séparation et évaluation" sont souvent beaucoup plus efficaces que les algorithmes du type "approximation extérieure".
- L'algorithme de Branch-and-Bound combiné avec DCA (BBDCA) fournit souvent un bon résultat.-
- Les algorithmes de Branch-and-Cut (BCDCAsP, BCDCAaP) sont efficaces dans certains cas. Cependant, ils ne sont pas bons pour autres cas.

Ins	Méthode	UB	LB	gap	itBB	Sépa	NbUB	time	fIt	fVal	rat	ftime	NbDCA
Mod010	BB	6551.00	6533.167	0.27	3008	1	1	510.422					
	BBDCA	6557.00	6532.08	0.38	3	1	1	1.250	5	6557.000	0.38	1.250	5
	BCDCAsP	6557.00	6532.08	0.38	3	1	1	1.656	5	6557.000	0.38	1.250	5
	BCDCAaP	6861.00	6532.08	4.79	1	1	1	0.796					1
Mod008	BB	307.00	298.95	2.62	610	1	1	13.000					
	BBDCA	307.00	291.64	5.02	13	1	1	0.390	3	307.000	5.02	0.078	3
	BCDCAsP	307.00	291.64	5.02	13	1	1	0.844	3	307.000	5.02	0.110	3
	BCDCAaP	307.00	291.64	5.02	12	1	1	1.344	3	307.000	5.02	0.109	3
PP08acuts	BB	$+\infty$	6444.61	-	98241	0	0	$> 1h$					
	BBDCA	11640.00	6432.75	44.73	89407	1	1	$> 1h$	3265	11640.000	44.73	115.672	3265
	BCDCAsP	7970.00	6269.38	21.33	23172	11	11	$> 1h$	-	-	-	-	488
	BCDCAaP	7970.00	6227.84	21.86	16452	6	6	$> 1h$	-	-	-	-	314
Pp08a	BB	$+\infty$	5254.58	-	>100000	0	0	> 1h					
	BBDCA	11640.00	5254.11	54.86	>100000	1	1	> 1h	57145	11640	54.86	1090.469	57145
	BCDCAsP	9370.00	4968.99	46.97	32272	5	5	$> 1h$	-	-	-	-	5007
	BCDCAaP	9780.00	4855.29	50.35	18722	13	13	$> 1h$	-	-	-	-	2231
P0201	BB	7665.00	7545.00	1.56	22890	1	1	745.687					
	BBDCA	7925.00	7528.75	5.00	19413	1	1	979.969	20766	7925.000	5.00	658.062	20766
	BCDCAsP	7775.00	7519.00	3.29	3467	4	3	500.937	3431	8095.000	7.12	250.359	3431
	BCDCAaP	7715.00	7336.00	4.90	1508	3	3	335.328	-	-	-	-	97
Mas74	BB	12469.19	10991.14	11.85	86641	5	5	$> 1h$					
	BBDCA	12469.19	10990.59	11.85	85945	4	4	$> 1h$	1	14372.871	23.53	0.062	4
	BCDCAsP	11985.18	10893.22	9.11	20429	12	11	$> 1h$	1	14372.871	24.21	0.047	1
	BCDCAaP	11985.18	10845.52	9.50	9308	12	11	$> 1h$	1	14372.871	24.54	0.062	1
Stein45	BB	30.00	28.59	4.72	41513	44	44	1281.95					
	BBDCA	30.00	28.48	5.00	28786	29	29	776.813	8	42.000	32.18	0.250	36
	BCDCAsP	30.00	28.45	5.16	27324	22	2	2282.297	0	-	-	-	2
	BCDCAaP	30.00	28.45	5.16	22052	29	2	3126.594	0	-	-	-	2

TAB. 4.3 – Résultats comparatifs entre les algorithmes BB, BBDCA, BCDCAsP et BCDCAaP pour les Benchmarks

Ins	Méthode	UB	LB	time	nonP	demi	still	Cut	NbP	Sub
Mod010	BCDCAsP	6557.000	6532.083	1.656	4	0	0	0	-	-
	BCDCAaP	6861.000	6532.083	0.796	0	0	0	0	1	4
Mod008	BCDCAsP	307.000	291.643	0.844	5	1	1	19	-	-
	BCDCAaP	307.000	291.643	1.344	0	1	1	21	4	28
PP08acuts	BCDCAsP	7970.000	6269.376	$>$ 1h	5460	5518	2793	40818	-	-
	BCDCAaP	7970.000	6227.842	$>$ 1h	0	4529	2419	32867	6225	36460
Pp08a	BCDCAsP	9370.000	4968.988	$>$ 1h	5887	11617	5412	56439	-	-
	BCDCAaP	9370.000	4855.298	$>$ 1h	0	6878	2878	37292	6590	49007
P0201	BCDCAsP	7775.000	7519.000	500.937	1560	3905	2530	5121	-	-
	BCDCAaP	7715.000	7336.000	300.328	0	1886	1018	2978	539	1968
Mas74	BCDCAsP	11985.177	10893.219	$>$ 1h	11453	973	884	29020	-	-
	BCDCAaP	11985.177	10845.526	$>$ 1h	0	560	451	18308	4159	49266
Stein45	BCDCAsP	30.000	28.452	2282.297	23862	32929	30077	18169	-	-
	BCDCAaP	30.000	28.452	3126.594	0	23704	19501	33537	16457	46300

TAB. 4.4 – BnCDCAsP et BnCDCAaP pour les Benchmarks

Chapitre 5

Optimisation inter-couches dans les réseaux de télécommunication

Résumé *La conception inter-couches dans les réseaux sans fil devient de plus en plus importante. Dans ce chapitre, nous présentons l'optimisation inter-couches pour les réseaux TDMA (Time Division Multiple Access). Plus particulièrement, nous considérons deux problèmes dont l'un est de maximiser la durée de vie du réseau et l'autre est de minimiser la consommation d'énergie. Tous les deux peuvent être reformulés comme un problème linéaire en variables mixtes 0-1 (PLM01). Nous utilisons une approche basée sur la programmation DC et DCA et sa combinaison avec une autre méthode globale pour résoudre ces problèmes.*

5.1 Introduction

De nos jours, les réseaux sans fil sont devenus les moyens essentiels de communication pour fournir la transmission de données fiable entre les utilisateurs. Dans de tels réseaux, lorsque les noeuds qui sont sources du trafic sont éloignés des noeuds de destination, on utilise une approche multi-hop : des noeuds sont utilisés comme des relais. Dans un réseau à multi-hop, en raison de l'interférence entre des liaisons, la transmission va probablement mener aux collisions, aboutissant à la perte de ressources. Donc, le Contrôle d'Accès au Médium (Medium Access control - MAC) et la conception inter-couches (cross-layer design) sont particulièrement importants. Dans ce travail, nous adoptons la division de temps communiqué en multiplex (Time Division Multiplexing - TDM) pour allouer le droit de transmission aux liaisons sans fil.

Récemment, l'optimisation inter-couches avec différents objectifs de conception a reçu beaucoup d'attention du monde de la recherche ([12, 101, 70, 71, 5, 32, 30, 9, 3, 6, 88]). [12] calcule l'ordonnancement du lien et le contrôle de puissance pour réduire la puissance pour les réseaux ad-hoc tandis que [101] présente l'ordonnancement du lien et le schéma de contrôle de puissance pour réseaux TDMA-basés. Ensuite, les PLM01s proposés sont résolus par les méthodes heuristiques. Les problèmes d'ordonnancement TDMA dans lesquels on maximise

le taux de transmission moyen ou minimise l'interférence entre les liens sont considérés dans ([70]). Néanmoins, aucune approche efficace n'est proposée. On peut également maximiser la durée de vie du réseau. Ceci fait l'objet des travaux [71, 5, 32].

Dans ce chapitre, nous considérons deux objectifs de conception : maximiser la durée de vie et minimiser la consommation d'énergie du réseau. Les deux problèmes sont formulés sous la forme d'un problème PLM01. Donc, nous allons utiliser l'approche basée sur la programmation DC et DCA pour résoudre ces deux problèmes.

Le reste du chapitre est organisé de façon suivante : la deuxième section présente la description du système. La troisième section est consacrée au problème de la durée de vie tandis que la quatrième section concerne le problème de la consommation d'énergie.

5.2 Description du système

Nous considérons un réseau à multi-hop avec $\mathcal{N}$ noeuds et une destination finale $\hat{n}$. Dénotons $\mathcal{L}$ les liens unidirectionnels ; des liens bidirectionnels peuvent être représentés par deux liens unidirectionnels.

Chaque noeud $n \in \mathcal{N}$ génère le trafic à un taux r_n. Ici r_n est un nombre entier qui doit être plus grand ou égal à la valeur minimale r_n^{min} :

$$r_n \geq r_n^{\min},\ n \in \mathcal{N}. \tag{5.1}$$

Dans la transmission TDMA, le temps est divisé en trames de longueur fixe et chaque trame est de nouveau divisée en J créneaux temporels. L'assignation de ressource dans une trame est la même que dans d'autres trames. Dans chaque trame, un noeud peut transmettre et/ou recevoir à un ou plusieurs créneaux pour son trafic. Un canal est représenté par 2- tuples d'éléments (j, l), $j \in \mathcal{J}$, $l \in \mathcal{L}$ où $\mathcal{J} = \{1, 2, \ldots, J\}$. L'assignation de ressource est dénotée par (s_j^l, P_j^l) où $s_j^l = 1$ quand le lien l est actif au créneau j, sinon $s_j^l = 0$, et $P_j^l > 0$ dénote la puissance de la transmission du lien l au créneau j si $s_j^l = 1$, $P_j^l = 0$ sinon.

Si un noeud transmet à un créneau, ses gammes de puissance de transmission seront comprises dans l'intervalle $[0, P_{max}]$. Donc, on a :

$$0 \leq P_j^l \leq P_{max} s_j^l,\ \forall l \in \mathcal{L},\ j = 1, \ldots, J. \tag{5.2}$$

À chaque noeud, la différence de son trafic sortant et son trafic entrant devrait être le trafic qu'il génère lui même, c'est-à-dire :

$$\sum_{l \in \mathcal{O}(n)} \sum_{j=1}^{J} s_j^l - \sum_{l \in \mathcal{I}(n)} \sum_{j=1}^{J} s_j^l = r_n, \quad n \in \mathcal{N}, \tag{5.3}$$

où $\mathcal{O}(n)$ et $\mathcal{I}(n)$ sont ensembles des liens sortants et des liens entrants au noeud n, respectivement.

De plus, la conservation de flux à la destination commune pour tous les noeuds sources est :

$$\sum_{l\in\mathcal{I}(\hat{n})}\sum_{j=1}^{J} s_j^l = \sum_{n\in\mathcal{N}} r_n. \tag{5.4}$$

Dans le réseau à multi-hop, en général, tous les liens ne peuvent pas être programmés concurremment à cause de l'interférence directe ou indirecte. De plus, puisque chaque noeud ne peut pas transmettre et recevoir simultanément alors ses liens sortants et ses liens entrants ne peuvent pas être actifs en même temps. Dans un réseau "unicast", un émetteur ne peut pas transmettre à plus d'un récepteur. Donc, les liens sortants du noeud ne peuvent pas être actifs simultanément. De plus, on ne permet pas deux transmissions avec un récepteur commun (d'être en état de marche simultanément) puisqu'une collision corrompra la réception de paquet. Pour cela, on a :

$$\sum_{l\in\mathcal{O}(n)} s_j^l + \sum_{l\in\mathcal{I}(n)} s_j^l \leq 1,\ \forall n \in \{\mathcal{N}\cup\hat{n}\},\ j{=}1,\ldots,J. \tag{5.5}$$

Pour le problème d'interférence parmi les noeuds ou/et les liens, les deux modèles utilisés sont souvent le modèle de "contention-based"([32, 112, 6]) et le modèle de SINR-based (Signal to Interference plus Noise Ratio)([71, 101]). Dans ce travail, nous considérons le dernier. Spécifiquement, si le lien $l \in \mathcal{L}$ est actif dans le créneau j (c'est-à-dire, $s_j^l = 1$), l'inégalité suivante devrait être vérifiée afin de garantir la qualité de transmission du lien :

$$\mathrm{SINR}_j^l = \frac{P_j^l h_{ll}}{\sum_{k\neq l} P_j^k h_{kl} + \eta_l} \geq \gamma^{\mathrm{th}},$$

où $SINR_j^l$ est le SINR pour le lien l au créneau j ; h_{kl} est le gain de chemin de l'émetteur du lien k au récepteur du lien l, η_l est la puissance de bruit au récepteur du lien l et γ_{th} est le seuil SINR exigé pour la transmission précise de l'information.

Cette contrainte peut s'écrire ainsi :

$$h_{ll}P_j^l \geq \gamma^{\mathrm{th}}\sum_{k\neq l} P_j^k h_{kl} + \gamma^{\mathrm{th}}\eta_l + D(s_j^l - 1), \quad \forall l \in \mathcal{L},\ j = 1,\ldots,J, \tag{5.6}$$

où D est une constante très grande.

Supposons que tous les noeuds sans fil ne soient pas mobiles et/ou la topologie du réseau est statique ou change assez lentement. Dans ce cas, le besoin de la mise en oeuvre distribuée n'est pas nécessaire.

5.3 Maximisation de la durée de vie du réseau

5.3.1 Formulation mathématique

La durée de vie de réseau est un critère important pour un réseau sans fil à multi-hop. Dans cette section, nous proposons une structure d'optimisation incluant l'assignation de puissance, la planification de liaison, le routage et le contrôle de taux pour maximiser la durée de vie du réseau.

Soit T_n la durée de vie du noeud n, $n \in \mathcal{N}$. La durée de vie du réseau T est définie comme la durée de temps maximale quand tous les noeuds fonctionnent bien. Ainsi nous avons $T = \min_{n\in\mathcal{N}} T_n$.

L'énergie totale consommée pendant la durée de vie du réseau pour chaque noeud doit être plus petite que l'énergie disponible :

$$\Big[\sum_{l\in\mathcal{O}(n)}\sum_{j=1}^{J} P_j^l + \sum_{l\in\mathcal{I}(n)}\sum_{j=1}^{J}\varepsilon_l s_j^l\Big]\cdot T \le E_n,\ n\in\mathcal{N}, \tag{5.7}$$

où E_n est l'approvisionnement d'énergie initial au noeud n ; ϵ_l est l'énergie requise pour recevoir une unité du trafic via le lien l.

Le problème de maximisation de la durée satisfaisant les contraintes (5.1)-(5.7) peut s'écrire comme suit :

$$\max_{P_j^l,\, s_j^l,\, T} \quad T \tag{5.8a}$$

tel que

$$\Big[\sum_{l\in\mathcal{O}(n)}\sum_{j=1}^{J} P_j^l + \sum_{l\in\mathcal{I}(n)}\sum_{j=1}^{J}\varepsilon_l s_j^l\Big]\cdot T \le E_n,\ n\in\mathcal{N}, \tag{5.8b}$$

$$r_n \ge r_n^{\min},\ n\in\mathcal{N}, \tag{5.8c}$$

$$0 \le P_j^l \le P_{max} s_j^l,\ \forall l\in\mathcal{L},\ j=1,\ldots,J, \tag{5.8d}$$

$$\sum_{l\in\mathcal{O}(n)}\sum_{j=1}^{J} s_j^l - \sum_{l\in\mathcal{I}(n)}\sum_{j=1}^{J} s_j^l = r_n,\ n\in\mathcal{N}, \tag{5.8e}$$

$$\sum_{l\in\mathcal{I}(\hat{n})}\sum_{j=1}^{J} s_j^l = \sum_{n\in\mathcal{N}} r_n, \tag{5.8f}$$

$$\sum_{l\in\mathcal{O}(n)} s_j^l + \sum_{l\in\mathcal{I}(n)} s_j^l \le 1,\ \forall n\in\{\mathcal{N}\cup\hat{n}\},\ j{=}1,\ldots,J, \tag{5.8g}$$

$$h_{ll}P_j^l \ge \gamma^{\text{th}}\sum_{k\ne l} P_j^k h_{kl} + \gamma^{\text{th}}\eta_l + D(s_j^l - 1), \quad \forall l\in\mathcal{L},\ j=1,\ldots,J, \tag{5.8h}$$

$$s_j^l \in \{0,1\}, \quad \forall l\in\mathcal{L},\ j=1,\ldots,J. \tag{5.8i}$$

Par le changement de variable $q = \frac{1}{T}$, le problème devient le problème (MLT) suivant :

$$\min_{P_j^l,\, s_j^l,\, q} \quad q \tag{5.9a}$$

tel que

$$\sum_{l\in\mathcal{O}(n)} \sum_{j=1}^{J} P_j^l + \sum_{l\in\mathcal{I}(n)} \sum_{j=1}^{J} \varepsilon_l s_j^l \leq q.E_n,\ n \in \mathcal{N} \tag{5.9b}$$

Les contraintes (5.8c)–(5.8i). (5.9c)

Le problème (MLT) est une programmation linéaire en variables mixtes 0-1. Quand le nombre de noeuds, de liens, de créneaux temporels sont grands, la taille du problème est extrêmement grande. C'est un problème difficile. La plupart des travaux dans la littérature sont basées sur des méthodes heuristiques qui ne sont pas vraiment efficaces [101, 8, 12]. Nous utilisons dans ce travail DCA pour résoudre le problème linéaire en variables mixtes 0-1. La section prochaine présente les expériences numériques.

5.3.2 Expériences numériques

Les algorithmes ont été codés en C et exécutés sur un ordinateur Pentium 4.3GHz, 1GB RAM. La version CPLEX 9.1 a été utilisée afin de résoudre la programmation linéaire. La puissance maximale P_{max} est égale à 5. Toutes les puissances de bruit η_l sont égales à $-20dB$. Le seuil SINR $\gamma_{th} = 10dB$. La consommation d'énergie pour recevoir la donnée ϵ_l est supposée insignifiante [1]. Les gains pour chaque lien sont calculés en utilisant le "path loss model" comme $h_{ij} = \frac{1}{10}[\frac{1}{d}]$ pour $i \neq j$ et $h_{ii} = [\frac{1}{d}]$ où d est la distance euclidienne. Le facteur de $\frac{1}{10}$ peut être regardé comme le gain de propagation dans un système de CDMA.

Tout d'abord, nous testons sur le réseau généré aléatoirement. Le tableau 5.1 présente les résultats des algorithmes BB, BBDCA, BCDCAsP, BCDCAaP. Les notations suivantes sont utilisées :
- $VarB$: le nombre de variables binaires,
- $VarL$: le nombre de variables continues,
- Ctr : le nombre de contraintes,
- UB : la borne supérieure,
- LB : la borne inférieure,
- $gap = \frac{UB-LB}{UB}100\%$,
- $itBB$: le nombre d'itérations,
- $Sepa$: le nombre de contraintes de séparation ajoutées,
- $NbUB$: le nombre de contraintes $cx + dy \leq UB$ ajoutées,
- $time$: le temps de calcul (en secondes),
- fIt :le nombre d'itérations au moment où DCA donne la première solution réalisable,

1. à taux de trafic haut, l'énergie de transmission est dominante

- $fVal$: la valeur de la fonction objectif obtenue quand DCA donne la première solution réalisable,
- $rat = \frac{fVal-LB}{LB}100\%$,
- $ftime$: le temps de calcul quand on a $fVal$,
- $NbDCA$: le nombre de lancements de DCA.

Les algorithmes s'arrêtent si soit $gap < 5\%$, soit le temps de calcul $time > 1$ heure, soit $itBB > 100000$ itérations.

Nous observons que :
- BB ne trouve aucune solution réalisable dans certains problèmes,
- Comme d'habitude, DCA fournit une solution réalisable très rapidement,
- BCDCAsP, BCDCAaP ne sont pas efficaces pour ce problème,
- BBDCA ne donne pas la solution optimale dans un temps considérable.

Même si les résultats ne sont pas encore satisfaisants pour la configuration de réseau aléatoirement, BBDCA est le meilleur parmi les quatre algorithmes ci-dessus. Dans la suite, nous testons quelques configurations spéciales comme "rhombus", "string", "diamond"... Nous utilisons BBDCA et l'arrêtons quand il fournit une solution réalisable. Nous comparons BBDCA avec CPLEX pour PLM01. Les résultats numériques sont présentés dans le tableau 5.2.

Les notations suivantes sont utilisées :
- N : le nombre de noeuds dans le réseau,
- L : le nombre de liens,
- J : le nombre de créneaux temporels,
- $VarC$: le nombre de variables continues,
- $VarB$: le nombre de variables binaires,
- Con : le nombre de contraintes,
- $Value$: la valeur de la fonction objectif obtenue,
- CPU : le temps de calcul en secondes,
- $iter$: le nombre d'itérations de BBDCA.

Nous constatons que :
- BBDCA fournit une solution réalisable après un petit nombre d'itérations.
- BBDCA fournit une solution optimale pour les 6/10 jeux de données.
- Sur les 4/10 jeux de données restants, la solution obtenue est quasi-optimale.
- Dans tous les cas, le temps de calcul de BBDCA est beaucoup plus petit que celui de CPLEX. Le gain du temps va jusqu'à 1505.6 fois (Ins.4). Donc, BBDCA est très rapide.

Ins	VarB	VarL	Ctr	Méthode	UB	LB	gap	itBB	Sépa	NbUB	time	flt	fVal	rat	ftime	NbDCA
Tel1	72	76	202	BB	0.002857	0.002828	1.02	291	1	1	8.938					
				BBDCA	0.002857	0.002828	1.02	290	1	1	8.172	1	0.002857	1.02	0.032	1
				BCDCAsP	0.002857	0.002828	1.02	291	274	1	81.312	1	0.002857	1.02	0.031	1
				BCDCAaP	0.002857	0.002828	1.02	291	274	1	82.359	1	0.002857	1.02	0.031	1
Tel2	80	85	213	BB	0.018473	0.005714	69.07	>100000	2	2	3301.969					
				BBDCA	0.017223	0.005714	66.82	>100000	1	1	3317.234	1	0.017223	66.82	0.047	1
				BCDCAsP	0.013485	0.005032	62.68	26299	5	3	$> 1h$	1	0.017223	70.78	0.047	1
				BCDCAaP	0.013485	0.004709	65.08	22821	5	3	$> 1h$	1	0.017223	272.66	0.047	1
Tel3	90	96	256	BB	$+\infty$	0.009500	-	>100000	0	0	3395.234					
				BBDCA	0.016971	0.009500	44.02	>100000	1	1	3491.750	1	0.016971	44.02	0.032	1
				BCDCAsP	0.016971	0.007000	58.75	5402	367	2	$> 1h$	1	0.016971	58.75	0.031	1
				BCDCAaP	0.016971	0.006002	64.63	5179	381	2	$> 1h$	1	0.016971	64.63	0.047	1
Tel4	100	106	276	BB	0.013553	0.009428	20.21	34697	264	264	$> 1h$					
				BBDCA	0.013553	0.009428	20.21	34757	262	262	$> 1h$	1	0.013553	20.21	0.047	262
				BCDCAsP	0.013553	0.005657	58.26	1732	1130	2	$> 1h$	1	0.013553	58.26	0.047	1
				BCDCAaP	0.013553	0.005657	58.26	1390	1020	2	$> 1h$	1	0.013553	58.26	0.047	1
Tel5	100	105	263	BB	0.013417	0.010944	18.43	91669	8	8	$> 1h$					
				BBDCA	0.013417	0.009945	25.88	84094	11	1	$> 1h$	1	0.013417	25.88	0.047	11
				BCDCAsP	0.013417	0.008944	33.34	2956	839	1	$> 1h$	1	0.013417	33.34	0.046	1
				BCDCAaP	0.013417	0.008944	33.34	2673	736	1	$> 1h$	1	0.013417	33.34	0.047	1
Tel6	140	147	369	BB	$+\infty$	0.005657	-	70747	0	0	$> 1h$					
				BBDCA	0.012000	0.005657	52.77	68288	1	1	$> 1h$	1	0.012000	52.77	0.062	1
				BCDCAsP	0.012000	0.004000	66.67	10907	9	1	$> 1h$	1	0.012000	66.67	0.063	1
				BCDCAaP	0.012000	0.004000	66.67	10822	9	1	$> 1h$	1	0.012000	66.67	0.078	1

TAB. 5.1 – Résultats comparatifs entre les algorithmes pour le problème TDMA

N^o	Donnée						BBDCA			CPLEX	
	N	L	T	VarC	VarB	Con	Value	CPU	iter	Value	CPU
01	4	6	12	73	72	199	**0.014142**	0.156	04	**0.014142**	0.01
02	5	8	10	81	80	219	0.016900	0.078	01	0.012071	4865.47
03	6	9	10	91	90	251	**0.016971**	0.468	10	**0.016971**	167.45
04	6	10	10	101	100	271	**0.013553**	0.062	01	**0.013553**	903.38
05	6	10	10	101	100	271	**0.008944**	1.328	25	**0.008944**	0.01
06	7	10	10	101	100	283	0.017657	0.218	04	0.012000	4519.38
07	5	10	10	101	100	259	**0.013417**	0.546	09	**0.013417**	56.00
08	5	8	15	121	120	324	0.020241	0.390	06	0.016730	9621.15
09	6	13	10	131	130	331	**0.012000**	0.609	09	**0.012000**	850.52
10	7	14	10	141	140	363	0.017657	0.140	02	0.012000	834.22

TAB. 5.2 – Résultats utilisant BBDCA pour quelques réseaux spéciaux

5.4 Minimisation de la consommation d'énergie

5.4.1 Formulation mathématique

Soient ϵ_l, ε_l l'énergie requise pour transmettre, recevoir une unité du trafic du lien l, respectivement. La consommation d'énergie au noeud n peut s'écrire :

$$\mathcal{E}_n = \sum_{l\in\mathcal{O}(n)} \sum_{j=1}^{J} P_j^l + \sum_{l\in\mathcal{O}(n)} \sum_{j=1}^{J} \epsilon_l s_j^l + \sum_{l\in\mathcal{I}(n)} \sum_{j=1}^{J} \varepsilon_l s_j^l. \tag{5.10}$$

Notons que ϵ_l, ε_l incluent l'énergie consommée par les blocs de traitement des signaux aux extrémités de lien.

La minimisation de la consommation d'énergie peut être formulée comme suit :

$$(ME) \min_{r_n, P_j^l, s_j^l} \quad \sum_{n\in\mathcal{N}} \mathcal{E}_n \tag{5.11a}$$

tel que :

$$r_n \geq r_n^{\min},\ n \in \mathcal{N}, \tag{5.11b}$$

$$0 \leq P_j^l \leq P_{max} s_j^l,\ \forall l \in \mathcal{L},\ j = 1, \ldots, J, \tag{5.11c}$$

$$\sum_{l\in\mathcal{O}(n)} \sum_{j=1}^{J} s_j^l - \sum_{l\in\mathcal{I}(n)} \sum_{j=1}^{J} s_j^l = r_n,\ n \in \mathcal{N}, \tag{5.11d}$$

$$\sum_{l\in\mathcal{I}(\hat{n})} \sum_{j=1}^{J} s_j^l = \sum_{n\in\mathcal{N}} r_n, \tag{5.11e}$$

$$\sum_{l\in\mathcal{O}(n)} s_j^l + \sum_{l\in\mathcal{I}(n)} s_j^l \leq 1,\ \forall n \in \{\mathcal{N} \cup \hat{n}\},\ j{=}1, \ldots, J, \tag{5.11f}$$

$$h_{ll} P_j^l \geq \gamma^{\text{th}} \sum_{k\neq l} P_j^k h_{kl} + \gamma^{\text{th}} \eta_l + D(s_j^l - 1), \quad \forall l \in \mathcal{L},\ j = 1, \ldots, J, \tag{5.11g}$$

$$s_j^l \in \{0, 1\}, \quad \forall l \in \mathcal{L},\ j = 1, \ldots, J. \tag{5.11h}$$

Le problème ci-dessus est PLM01. On peut utiliser DCA pour le résoudre. Mais afin de garantir une solution réalisable, nous combinons DCA avec le schéma de Branch-and-Bound : la borne inférieure est calculée par la résolution de la relaxation linéaire du problème. A chaque itération, on lance DCA à partir de la solution optimale du problème relaxé. On arrête l'algorithme combiné dès l'obtention d'une solution réalisable.

Notre algorithme peut être décrit comme suit :

Algorithme 5.1 ***(DCA avec le point initial de B&B)***
(Initialisation)

Poser $R_0 := [0, 1]^{LJ}$, $k := 0$.

Résoudre le problème relaxé pour obtenir une solution optimale (P^0, s^0) *et la valeur optimale* $\beta(R_0)$.

Si (P^0, s^0) *est réalisable pour (ME)* ***alors*** *STOP* ***sinon*** *appliquer DCA à partir du point initial* (P^0, s^0) *pour obtenir* $(\overline{P}, \overline{s})$.

Si $(\overline{P}, \overline{s})$ *est réalisable pour (ME),* ***alors*** *STOP* ***sinon*** *poser* $\Re = \{R_0\}$ *et passer à l'étape d'itération.*

(Itération)

Tant que *(stop = false)* ***faire***

– *Poser* $k := k + 1$ *et sélectionner un rectangle* R_k.

– *Choisir* $j*$ *l'index à séparer. Diviser* R_k *en deux rectangles* R_{k_0} *and* R_{k_1} *tel que :*

$$R_{k_i} = \{s \in R_k : s_{j^*} = i, i = 0, 1\}.$$

– *Pour chaque* $i = 0, 1$, *résoudre le problème relaxé correspondant pour obtenir la solution optimale* (P^{k_i}, s^{k_i}) *et la valeur optimale* $\beta(R_{k_i})$.
– *Lancer DCA à partir de* (P^{k_i}, s^{k_i}) *pour obtenir* $(\overline{P^{k_i}}, \overline{s^{k_i}})$).
– ***Si*** $(\overline{P^{k_i}}, \overline{s^{k_i}})$ *est réalisable,* ***alors*** *STOP* ***sinon*** *:*

$$\Re \leftarrow \Re \cup \{R_{k_i}; i = 0, 1\} \setminus R_k$$

Fin tant que

5.4.2 Expériences numériques

L'algorithme 5.1 a été codé en C et exécuté sur un ordinateur Pentium 4.3GHz, 1GB RAM. La version CPLEX 9.1 a été utilisée afin de résoudre la programmation linéaire. Un réseau de petite taille avec 6 noeuds et 10 liens a été testé. Les coordonnées des noeuds sont indiquées dans le tableau 5.3. La puissance maximale P_{max} est égale à 5. Toutes les puissances de bruit η_l sont fixées à $-20dB$. Le seuil SINR $\gamma_{th} = 10dB$. Les énergies consommées pour transmettre et recevoir les données ϵ_l, ε_l sont supposées égales à 0.25. Les gains pour chaque lien sont calculés en utilisant le "path loss model" comme $h_{ij} = \frac{1}{10}[\frac{1}{d}]$ pour $i \neq j$ et $h_{ii} = [\frac{1}{d}]$ où d est la distance euclidienne. Le facteur de $\frac{1}{10}$ peut être regardé comme le gain de propagation dans un système de CDMA. Nous testons ce réseau avec les différents nombres de créneaux temporels $J = 10,\ 15,\ 20,\ 25,\ 30$.

Dans le tableau 5.4, nous présentons les résultats de l'algorithme 5.1. La valeur optimale est calculée par le logiciel CPLEX 9.1. Les notations suivantes sont utilisées :

- J : le nombre de créneaux temporels,
- $VarC$: le nombre de variables continues,
- $VarB$: le nombre de variables binaires,
- Con : le nombre de contraintes,
- $iter$: le nombre d'itérations,
- $Value$: la valeur de la fonction objectif obtenue par notre algorithme,
- $OptVal$: la valeur optimale,
- $Gap = \frac{Value - Optval}{Value} 100\%$.

Noeud	N1	N2	N3	N4	N5	Sink node
Coordonnée	(-20,20)	(0,0)	(0,40)	(40,40)	(40,0)	(80,25)

TAB. 5.3 – Coordonnées de noeuds

En visualisant les résultats par la figure 5.1 et en voyant la colonne *Gap* dans le tableau 5.4, on trouve que la solution fournie par notre algorithme est proche de la valeur optimale. De plus, il faut noter que tous les résultats obtenus par notre algorithme sont très rapides (quelques mini-secondes de calcul). Donc, notre approche est clairement intéressante.

J	VarC	VarB	Con	iter	Value	OptVal	Gap(%)
10	100	100	266	9	49.5	41.57855	16.0
15	150	150	396	72	71.5	60.56757	15.3
20	200	200	526	205	93.5	79.55660	14.9
25	250	250	656	770	115.5	98.54560	14.7
30	300	300	786	369	126.5	108.53460	14.2

TAB. 5.4 – Résultats de l'Algorithme 5.1

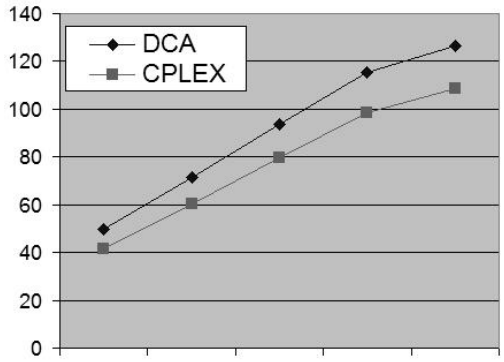

FIG. 5.1 – Résultats comparatifs entre la valeur obtenue par DCA et la valeur optimale

5.5 Conclusion

Dans ce chapitre, nous avons étudié deux problèmes d'optimisation inter-couches dans le réseau sans fil TDMA : maximiser la durée de vie du réseau et minimiser la consommation d'énergie du réseau. Nous avons prouvé que les problèmes d'optimisation proposés peuvent être formulés comme PLM01. Utilisant la technique de reformulation et ensuite la programmation DC et DCA, nous avons obtenu des résultats numériques prometteurs. Notons que plusieurs formulations du problème surgissant dans les réseaux TDMA peuvent être formulées sous la forme des PLM01s, notre approche semble attrayante et doit être étudiée dans le détail.

Chapitre 6

Ordonnancement

Résumé *Ce chapitre est consacré à deux problèmes d'ordonnancement qui sont formulés sous la forme de PLM01. Ils sont résolus par l'approche basée sur la programmation DC et DCA qui est présentée dans le chapitre 4. Les expériences numériques prouvent que cette approche est efficace.*

6.1 Introduction au problème d'ordonnancement

La théorie de l'ordonnancement est une branche de la recherche opérationnelle qui s'intéresse au calcul de dates d'exécution optimales de tâches. De façon plus précise, un problème d'ordonnancement consiste à organiser dans le temps, la réalisation de tâches, compte tenu de contraintes temporelles (délais, contraintes d'enchaînement, etc.) et de contraintes portant sur la disponibilité des ressources requises. Un ordonnancement est défini par le planning d'exécution des tâches et d'allocation des ressources et vise à satisfaire un ou plusieurs objectifs.

Dans ce chapitre, on utilise les notations suivantes :
- temps d'exécution (processing time) : si la tâche i est exécutée totalement sur la machine M_j alors son temps d'exécution est noté p_{ij} . Lorsque les machines sont identiques ou bien, on note le temps d'exécution de la tâche i par p_i.
- date d'échéance (due date) : la date à laquelle la tâche i doit être idéalement terminée est notée d_i.
- date de disponibilité (ou date de début au plus tôt) (release date) : une tâche i ne peut être ordonnancée sur la machine M_j qu'à partir de l'instant r_{ij} . Si les dates de disponibilité des tâches sont égales pour toutes les machines, alors nous notons r_i la date de disponibilité de i. Si ces dates sont égales à 0, alors les tâches sont toujours disponibles.
- la date de fin C_i (completion time) :
- la date de début S_i (starting time) :
- l'avance (earliness) de la tâche i : $E_i = \max\{0, d_i - C_i\}$.
- le retard (tardiness) de la tâche i : $T_i = \max\{0, C_i - d_i\}$.
- la pénalité d'avance α_i,

- la pénalité de retard β_i.

Dans ce chapitre, nous considérons deux problèmes d'ordonnancement différents. Tous les deux sont PLM01. Nous utilisons les algorithmes basés sur DCA (présentés dans le chapitre 4 de la thèse) pour résoudre ces problèmes.

6.2 Problème d'ordonnancement d'avance-retard avec variables indexées sur le temps

6.2.1 Description du problème

La formulation avec variables indexées sur le temps est très connue pour formuler le problème d'ordonnancement d'avance - retard comme un problème linéaire en nombres entiers [100]. Il est basé sur la discrétisation du temps : le temps est divisé en périodes de temps. La période t commence à l'instant t et finit à l'instant $t+1$. Pour chaque tâche i et chaque période t, la variable binaire x_{it} indique si la tâche i est accomplie en période t ou pas. L'avantage principal de cette formulation est que la borne inférieure fournie par la relaxation continue est meilleure que les bornes obtenues par les relaxations des autres formulations [100, 11].

Plusieurs travaux ont été effectués sur cette formulation. La plupart utilisent des méthodes heuristiques ou la recherche locale pour trouver la borne supérieure. Dans le problème à une seule machine, plusieurs auteurs ont étudié un cas particulier dans lequel chaque tâche a la même date d'échéance (common due date). Pour ce cas particulier, Van den Akker, Hoogeveen and Velde [108] ont résolu des instances de 125 tâches (en supposant que la date d'échéance commune soit très grande). Récemment, Sourd a traité des instances de 1000 tâches [97]. Cependant, dans le cas général, on ne peut que résoudre des instances de 50 tâches [97, 99, 110].

A notre connaissance, dans la literature, on n'a pas encore considéré une date limite pour chaque tâche dans la formulation avec variables indexées sur le temps. Un tel problème se rencontre en pratique quand il y a une limite pour le temps d'exécution pour chaque tâche. De plus, comme indiqué dans les papiers récents de T'kindt et de Billaut [104], la plupart des travaux existants traitent le problème à une seule machine. Dans ce travail, nous considérons le problème d'ordonnancement d'avance - retard sur des machines parallèles avec une date limite pour chaque tâche.

Soit $I = \{1, 2, \ldots, n\}$ l'ensemble des tâches indépendantes de n qui sont ordonnancées sur m machines parallèles identiques sans préemption. Pour chaque tâche i, on note $\alpha_i, \beta_i, p_i, r_i, d_i, \tilde{d}_i$ la pénalité d'avance, la pénalité de retard, le temps d'exécution, la date de disponibilité, la date d'échéance et la date-limite respectivement. Dénotons par c_{it} le coût d'exécution de la tâche i si elle est accomplie au temps t. L'horizon de l'ordonnancement est noté $T = \min\big(\max(\max_{i=1}^{n} r_i; \max_{i=1}^{n} d_i) + \sum_{i=1}^{n} p_i; \max_{i=1}^{n} \tilde{d}_i\big)$. On a une formulation avec va-

riables indexées sur le temps comme suit :

$$
\begin{aligned}
(TI)\quad & \min \sum_{i=1}^{n} \sum_{t=r_i+p_i}^{\tilde{d}_i} c_{it} x_{it} && \text{(6.1a)}\\
& \text{tel que} \\
& \sum_{t=r_i+p_i}^{\tilde{d}_i} x_{it} = 1, \quad \forall i \in [1,n], && \text{(6.1b)}\\
& \sum_{i=1}^{n} \sum_{s=t}^{t+p_i-1} x_{is} \leq m, \quad \forall t \in [1,T], && \text{(6.1c)}\\
& x_{it} = 0, \quad \forall i \in [1,n], \quad t \in [1, r_i + p_i) \cup (\tilde{d}_i, T], && \text{(6.1d)}\\
& x_{it} \in \{0,1\}, \quad \forall i \in [1,n], \quad t \in [0,T], && \text{(6.1e)}
\end{aligned}
$$

où la variable binaire x_{it} correspondant à la tâche i et la période de temps t indique si la tâche i accomplit au temps t ($x_{it} = 1$) ou pas ($x_{it} = 0$). La fonction objectif (6.1a) représente la somme de tous les coûts d'exécution. Les contraintes (6.1b) indiquent que chaque tâche doit être exécutée exactement une fois, et les contraintes de capacité (6.1c) indiquent que tout au plus m tâches peuvent être traitées à chaque période t. Les contraintes (6.1d) spécifient que chaque tâche i ne peut pas être exécutée à une certaine période t.

La formulation avec variables indexées sur le temps est très simple, et elle peut être employée pour modéliser de nombreux types de problèmes d'ordonnancement. Des fonctions objectifs différentes peuvent être modélisées par les choix appropriés des coefficients de coût. Pour le problème d'ordonnancement d'avance-retard, on peut choisir $c_{it} = \max\{\alpha_i(d_i - t); \beta_i(t - d_i)\}$.

Un des avantages de cette formulation est que la relaxation continue donne une borne inférieure qui est meilleure que les bornes obtenues par des programmes entiers mixtes, mais son inconvénient est sa taille. Le nombre de variables binaires nT devient très grand quand on a beaucoup de tâches ou bien le temps d'exécution est grand.

6.2.2 Expériences numériques

Nous utilisons DCA pour résoudre (TI) (les algorithmes globaux ne sont pas applicables à ce problème de très grande taille). Il a été exécuté sur un Intel Core 2CPU de 1.86GHz, 2GB RAM. A partir de r_i, p_i , d_i repris des données de Sourd et Kedad-Sidhoume [98], nous avons généré aléatoirement $\tilde{d}_i \in [\max(r_i + p_i; d_i), 1.1(\max(r_i) + \sum p_i)]$.

Afin d'évaluer la qualité de DCA, nous comparons DCA au logiciel CPLEX 11.2. Nous exécutons DCA dans un premier temps et ensuite nous fixons le temps limite qui est égal au temps requis par DCA pour exécuter CPLEX. Les programmes linéaires dans chaque itération de DCA et le problème linéaire relaxé sont également résolus par CPLEX 11.2.

Pour $n = 40, 60, 80$, nous faisons varier m de 1 à 3. Pour chaque n et m, 10 jeux de données sont utilisés. Les 5 premiers jeux de données ont des faibles temps d'exécution (processing time p_i) et les 5 derniers jeux de données ont des grands temps d'exécution.

Nous donnons les résultats dans les trois tableaux ci-dessous correspondant à $m = 1$, à $m = 2$ et à $m = 3$ respectivement. Quelques notations utilisées dans le tableau des résultats :
◇ Var : le nombre de variables binaires,
◇ Ctr : le nombre de contraintes,
◇ Val_{DCA} : la borne supérieure obtenue par DCA,
◇ it : le nombre d'itérations de DCA,
◇ $time$: le temps du calcul,
◇ Val_{CPLEX} : la borne supérieure obtenue par CPLEX après $time$ secondes,
◇ LB : la borne inférieure obtenue par la relaxation linéaire,
◇ $GAP_1 = \frac{(Val_{DCA}-LB)}{Val_{DCA}}100\%$,
◇ $GAP_2 = \frac{(Val_{CPLEX}-LB)}{Val_{CPLEX}}100\%$,
◇ NA : CPLEX n'a pas encore fourni un solution de nombre entier après $time_{DCA}$ secondes.

A travers des tableaux de résultats, nous observons que :
- Pour tous les jeux de données, DCA fournit toujours des solutions réalisables après un petit nombre d'itérations.
- DCA est rapide.
- $GAP_1 s$ sont petits. Cela signifie que soit les bornes supérieures obtenues par DCA sont proches des valeurs optimales, soit ces bornes supérieures peuvent être les valeurs optimales elles-mêmes. Parmi 90 jeux de données, 46 ont $GAP_1 < 5\%$.
- DCA fournit souvent une valeur de la fonction objectif meilleure que celle fournie par CPLEX (voir les deux colonnes Val_{DCA} et Val_{CPLEX}). La valeur moyenne est également meilleure. D'ailleurs, CPLEX n'a pas encore trouvé la solution réalisable pour 10/90 cas. Visuellement, les diagrammes de la fig. 6.1 montre que DCA est meilleur que CPLEX.

6.3 Problème d'ordonnancement d'avance-retard avec une date d'échéance commune

6.3.1 Description du problème

Dans quelques modèles, on considère non seulement l'ordonnancement de tâches mais on détermine aussi leur date d'échéance. En général, l'objectif du problème est défini par une combinaison linéaire des facteurs : la pénalité d'avance, la pénalité de retard et la pénalité sur la date d'échéance. On cherche un compromis pour avoir une petite date d'échéance tandis que la pénalité d'avance - retard est la plus petite possible.

Dans cette section, on considère un problème ayant n tâches sur une seule machine. Chaque tâche i est décrite par le temps d'exécution p_i, la pénalité d'avance α_i, la pénalité de retard

Ins. $n = 40$	Var	Ctr	Val_{DCA}	*it*	*time*	Val_{CPLEX}	*LB*	GAP_1(%)	GAP_2 (%)
40.1	17840	8998	**43320**	4	0.750	**51710**	41174.520	4.95	20.37
40.2	17640	9328	**53320**	3	0.578	**84499**	52685.000	1.19	37.65
40.3	17400	10366	46310	4	0.609	45462	43785.885	5.45	3.69
40.4	16040	10008	**39647**	4	0.593	**41083**	37558.949	5.27	8.58
40.5	18760	9233	66052	5	1.203	56802	52473.135	20,56	7.62
40.6	49960	27681	138229	5	12.297	138130	136074.190	1.56	1.49
40.7	54320	29374	**127837**	5	7.188	**181543**	120406.267	5.81	33.68
40.8	49880	26261	92233	5	10.016	92200	85615.029	7.18	7.14
40.9	53920	28695	**127057**	5	12.812	**131828**	110880.352	12.73	15.89
40.10	45560	26478	**154296**	3	5.422	**203692**	149695.486	2.98	26.51
Average			**88830.1**		5.1468	**102694.9**	83034,901	6.52	19.14
Ins. $n = 60$	Var	Ctr	Val_{DCA}	*it*	*time*	Val_{CPLEX}	*LB*	GAP_1(%)	GAP_2 (%)
60.1	42060	23980	**103391**	5	2.735	**142289**	99643.300	3.62	29.97
60.2	34680	18604	146876	4	1.297	119853	79851.000	45.63	33.38
60.3	40860	23232	**107608**	5	2.343	**192369**	100576.782	6.53	47.72
60.4	38220	22109	**85586**	4	1.765	**87561**	77452.772	9.50	11.54
60.5	41580	24313	101757	7	4.375	98281	90485.537	11.08	7.93
60.6	114900	61415	**305625**	5	37.829	**314041**	286025.176	6.41	8.92
60.7	107220	57703	**276135**	5	27.157	**320524**	242412.339	12.21	24.37
60.8	109440	62014	425790	9	46.360	395662	393970.541	7.47	0.43
60.9	99900	55908	235358	5	31.141	217826	216147.432	8.16	0.77
60.10	109620	61662	**228329**	5	26.500	**247768**	219715.290	3.77	11.32
Average			**201645.5**		18.150	**213617.4**	180628.017	10.42	15.44
Ins. $n = 80$	Var	Ctr	Val_{DCA}	*it*	*time*	Val_{CPLEX}	*LB*	GAP_1(%)	GAP_2 (%)
80.1	69440	34632	**151173**	4	7.297	**168084**	142212.646	5.93	15.39
80.2	63520	34656	**114351**	4	5.610	**117651**	106465.567	6.90	9.51
80.3	71040	42531	**197566**	5	4.266	**205009**	178696.958	9.55	12.83
80.4	69280	36684	**158556**	4	5.859	**174975**	148885.270	6.10	14.91
80.5	70080	38219	140620	4	5.781	139064	134850.896	4.10	3.03
80.6	179200	90880	436606	4	88.75	424506	404193.573	7.42	4.78
80.7	179600	104472	**371542**	6	66.178	**377821**	346234.718	6.81	8.36
80.8	202720	127666	**498690**	6	88.438	**537149**	479278.132	3.89	10.77
80.9	186880	103221	472060	5	92.094	468070	456885.464	3.21	2.39
80.10	195840	110698	**410939**	7	124.781	**444147**	366955.883	10.70	17.38
Average			**295210.3**		48.905	**305647.6**	276465,911	6.35	9.55

TAB. 6.1 – Comparaison entre DCA et CPLEX avec $m = 1$

$Ins.\ n = 40$	Var	Ctr	Val_{DCA}	it	$time$	Val_{CPLEX}	LB	GAP_1(%)	GAP_2 (%)
40.1	17840	8998	**15558**	5	0.688	**15573**	15485.500	0.47	0.56
40.2	17640	9328	**40777**	3	0.390	**56197**	24458.800	40.02	56.48
40.3	17400	10366	**20962**	2	0.391	**20962**	**20962.000**	0.00	0.00
40.4	16040	10008	**10774**	4	0.391	**14968**	9703.958	9.93	35.19
40.5	18760	9233	**20635**	3	0.547	**21448**	20331.220	1.47	5.21
40.6	49960	27681	62267	4	5.594	61931	61603.429	1.07	0.53
40.7	54320	29374	47113	4	4.047	45646	45610.556	3.19	0.08
40.8	49880	26261	28297	3	3.359	27209	26973.167	4.68	0.87
40.9	53920	28695	**48362**	4	3.937	**106144**	44184.667	8.64	58.37
40.10	45560	26478	**711386**	3	2.015	**106541**	51365.920	27.79	51.79
Average			**36588.3**		2.136	**47661.9**	32067.920	9.73	20.90
$Ins.\ n = 60$	Var	Ctr	Val_{DCA}	it	$time$	Val_{CPLEX}	LB	GAP_1(%)	GAP_2 (%)
60.1	42060	23980	58193	2	1.313	48210	41181.066	29.23	14.58
60.2	34680	18604	**32682**	4	1.109	**42708**	29000.546	11.26	32.10
60.3	40860	23232	**42980**	3	1.109	**106721**	29148.500	32.18	72.69
60.4	38220	22109	**42702**	3	0.907	**69280**	35206.188	17.55	49.18
60.5	41580	24313	**44850**	3	1.203	**48083**	42437.618	5.38	11.74
60.6	114900	61415	166901	4	11.782	148260	143213.800	14.19	3.40
60.7	107220	57703	**118189**	4	12.719	**118476**	114229.902	3.35	3.58
60.8	109440	62014	**235263**	3	5.750	NA	192238.617	18.29	NA
60.9	99900	55908	**76133**	3	5.313	NA	69745.571	8.39	NA
60.10	109620	61662	94210	4	13.140	94208	93931.810	0.30	0.29
Average			**75088.4**		5.410	**84493.3**	66043.680	14.18	23.45
$Ins.\ n = 80$	Var	Ctr	Val_{DCA}	it	$time$	Val_{CPLEX}	LB	GAP_1(%)	GAP_2 (%)
80.1	69440	34632	**63244**	5	4.156	**63543**	61884.476	2.15	2.61
80.2	63520	34656	**45346**	5	5.406	**45515**	40859.412	9.89	10.23
80.3	71040	42531	87370	5	5.140	86964	79537.072	8.97	8.54
80.4	69280	36684	**60761**	3	2.641	**88915**	58694.052	3.40	33.99
80.5	70080	38219	**62905**	3	2.672	**82472**	51179.486	18.64	37.94
80.6	179200	90880	**201140**	5	26.681	**206425**	194438.654	3.33	5.81
80.7	179600	104472	140571	5	26.250	140366	137070.218	2.49	2.35
80.8	202720	127666	**277727**	3	13.171	NA	238847.972	13.99	NA
80.9	186880	103221	**281009**	2	15.575	NA	192435.624	31.52	NA
80.10	195840	110698	**183856**	4	27.750	**189340**	177302.436	3.56	6.36
Average			**105649.1**		12.587	**112942.5**	100120.700	6.55	13.48

TAB. 6.2 – Comparaison entre DCA et CPLEX avec $m = 2$

$Ins.\ n = 40$	Var	Ctr	Val_{DCA}	it	$time$	Val_{CPLEX}	LB	GAP_1(%)	GAP_2 (%)
40.1	17840	8998	**11620**	3	0.375	**16433**	11424.400	1.68	30.48
40.2	17640	9328	**25558**	2	0.328	**26110**	21653.000	15.28	17.07
40.3	17400	10366	**19112**	2	0.328	**19290**	18533.000	3.03	3.92
40.4	16040	10008	**7011**	2	0.281	**7011**	**7011.000**	0.00	0.00
40.5	18760	9233	**16087**	2	0.375	**24556**	14356.000	10.76	41.54
40.6	49960	27681	**47505**	3	1.985	**60117**	47360.000	0.31	21.22
40.7	54320	29374	36744	2	3.812	36169	36012.000	1.99	0.43
40.8	49880	26261	20488	3	3.453	20474	20336.500	0.74	0.67
40.9	53920	28695	**35566**	2	2.218	**35566**	**35566.000**	0.00	0.00
40.10	45560	26478	**39393**	2	1.610	**82527**	35893.071	8.88	56.51
Average			**25908.4**		1.477	**32825.3**	24814.500	4.27	17.18
$Ins.\ n = 60$	var	Ctr	Val_{DCA}	it	$time$	Val_{CPLEX}	LB	GAP_1(%)	GAP_2 (%)
60.1	42060	23980	**35028**	2	0.937	**48131**	33831.000	3.42	29.71
60.2	34680	18604	**23927**	2	0.703	**24280**	22015.667	7.99	9.33
60.3	40860	23232	**20095**	2	0.938	NA	19489.500	3.01	NA
60.4	38220	22109	**31956**	3	0.828	NA	31819.000	0.43	NA
60.5	41580	24313	**54859**	4	1.609	**54859**	36387.136	33.67	33.67
60.6	114900	61415	**127663**	5	6.546	**167944**	125582.667	1.63	25.22
60.7	107220	57703	98859	4	6.719	98661	94465.500	4.44	4.25
60.8	109440	62014	**165403**	3	5.156	**208397**	165206.750	0.12	20.72
60.9	99900	55908	**53307**	3	4.484	**82853**	52855.200	0.85	36.21
60.10	109620	61662	**82501**	2	5.000	NA	82501.000	0.00	NA
Average			**79863.71**		3.736	**97875**	75763.420	7.45	22.73
$Ins.\ n = 80$	Var	Ctr	Val_{DCA}	it	$time$	Val_{CPLEX}	LB	GAP_1(%)	GAP_2 (%)
80.1	69440	34632	**51322**	3	1.968	NA	50627.375	1.35	NA
80.2	63520	34656	**33228**	3	1.578	NA	32177.167	3.16	NA
80.3	71040	42531	81255	3	2.109	74933	67514.000	16.91	9.90
80.4	69280	36684	**53244**	2	1.640	NA	51148.500	3.94	NA
80.5	70080	38219	**37242**	3	2.297	**58271**	37105.393	0.37	36.32
80.6	179200	90880	166973	4	17.234	165230	164133.774	1.70	0.66
80.7	179600	104472	**108115**	3	12.093	**166474**	106583.680	1.42	35.98
80.8	202720	127666	215892	3	14.078	215892	215809.000	0.04	0.04
80.9	186880	103221	**152880**	2	11.532	**152979**	152766.500	0.07	0.14
80.10	195840	110698	**159153**	3	11.375	**203835**	157810.358	0.84	22.58
Average			**131644.3**		10.103	**148230.6**	128817.500	3.05	15.09

TAB. 6.3 – Comparaison entre DCA et CPLEX avec $m = 3$

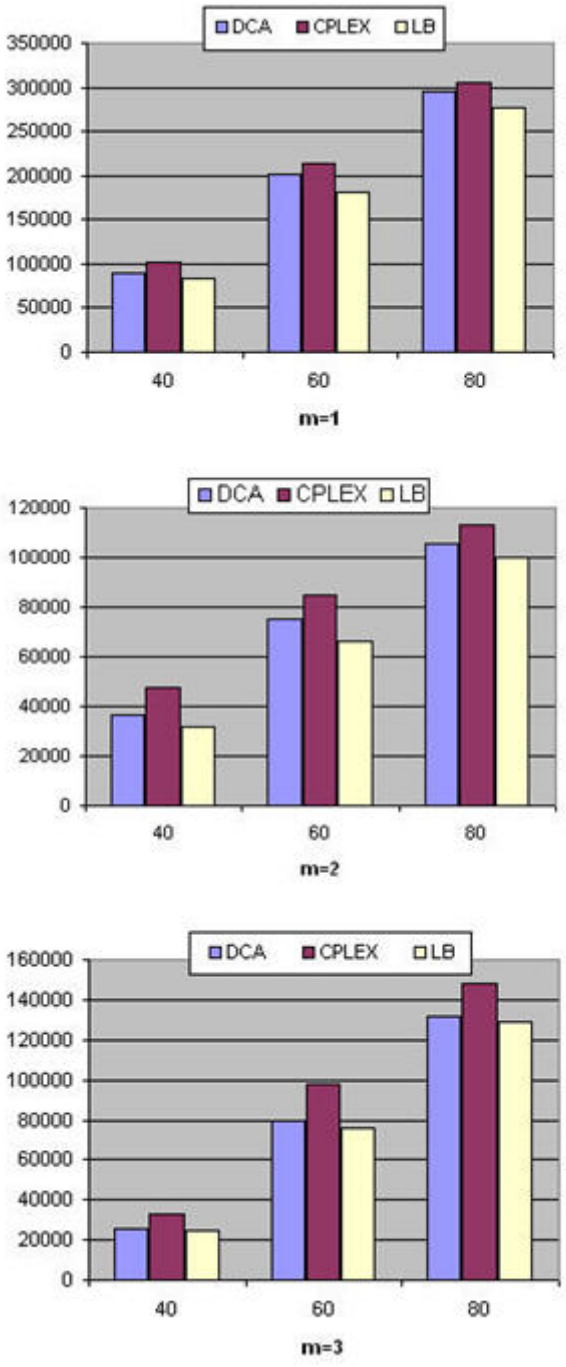

FIG. 6.1 – Résultats comparatifs entre DCA et CPLEX

β_i, la date d'échéance commune d et la pénalité sur la date d'échéance γ. Selon la notation classique à trois champs introduite par Graham et al. [16], le problème considéré est dénoté $(1|d_i = d|\sum(\alpha_i E_i + \beta_i T_i + \gamma d))$. Il est NP-difficile.

Soit R_k la pénalité d'avance-retard de la tâche ordonnancée en position k. La variable binaire x_{ik} est égale à 1 si et seulement si la tâche i est ordonnancée en position k. La variable t_k est le temps de début de la tâche qui est ordonnancée en position k. Nous avons le modèle suivant :

$$
\begin{aligned}
(JIT1)\quad & \min(\sum_{k=1}^{n} R_k + n\gamma d) && \text{(6.2a)}\\
& \text{tel que} \\
& t_{k+1} \geq t_k + \sum_{i=1}^{n} x_{ik}p_i, \forall k \in [1,n], && \text{(6.2b)}\\
& \sum_{i=1}^{n} x_{ik} = 1, \forall k \in [1,n], && \text{(6.2c)}\\
& \sum_{k=1}^{n} x_{ik} = 1, \forall i \in [1,n], && \text{(6.2d)}\\
& R_k \geq \sum_{i=1}^{n}(\alpha_i x_{ik} d) - \sum_{i=1}^{n}(\alpha_i x_{ik} p_i) - \sum_{i=1}^{n}(\alpha_i x_{ik} t_k), && \text{(6.2e)}\\
& R_k \geq \sum_{i=1}^{n}(\beta_i x_{ik} p_i) + \sum_{i=1}^{n}(\beta_i x_{ik} t_k) - \sum_{i=1}^{n}(\beta_i x_{ik} d), && \text{(6.2f)}\\
& P \geq d \geq 0; \quad t_k, R_k \geq 0, \forall k \in [1,n]; \quad x_{it} \in \{0,1\}, \forall i,t \in [1,n] && \text{(6.2g)}
\end{aligned}
$$

où $P = \sum_{i=1}^{n} p_i$.

Les contraintes (6.2b) prouvent que la tâche programmée en position $(k+1)$ devrait être commencée après la date de fin de la tâche programmée en position k. Les contraintes (6.2c) indiquent qu'une tâche peut être affectée à une et seulement une tâche ; et inversement, les contraintes (6.2d) garantissent qu'une tâche peut être affectée à une et seulement une position.

Le modèle ci-dessus est non-linéaire parce que les contraintes (6.2e)(6.2f) sont quadratiques. On va les linéariser en ajoutant quelques variables complémentaires. Remplaçant $x_{ik}t_k$ et $x_{ik}d$ par w_{ik} et Z_{ik}, les contraintes (6.2e)(6.2f) sont transformées en contraintes linéaires (6.3b)-(6.3j) :

$$(JIT2) \quad \min(\sum_{k=1}^{n} R_k + n\gamma d) \tag{6.3a}$$

tel que :

les contraintes (6.2b)(6.2c)(6.2d) et

$$R_k \geq \sum_{i=1}^{n}(\alpha_i Z_{ik}) - \sum_{i=1}^{n}(\alpha_i x_{ik} p_i) - \sum_{i=1}^{n}(\alpha_i w_{ik}), \forall k \in [1,n], \tag{6.3b}$$

$$R_k \geq \sum_{i=1}^{n}(\beta_i x_{ik} p_i) + \sum_{i=1}^{n}(\beta_i w_{ik}) - \sum_{i=1}^{n}(\beta_i Z_{ik}), \forall k \in [1,n], \tag{6.3c}$$

$$Z_{ik} \leq P x_{ik}, \forall i,k \in [1,n], \tag{6.3d}$$

$$Z_{ik} \leq d, \forall i,k \in [1,n], \tag{6.3e}$$

$$Z_{ik} \geq d - P(1-x_{ik}), \forall i,k \in [1,n], \tag{6.3f}$$

$$w_{ik} \leq P x_{ik}, \forall i,k \in [1,n], \tag{6.3g}$$

$$w_{ik} \leq t_k, \forall i,k \in [1,n], \tag{6.3h}$$

$$w_{ik} \geq t_k - P(1-x_{ik}), \forall i,k \in [1,n], \tag{6.3i}$$

$$d \leq P, \tag{6.3j}$$

$$d, t_k, R_k, Z_{ik}, w_{ik} \geq 0; \quad x_{ik} \in \{0,1\} \quad \forall i,k \in [1,n]. \tag{6.3k}$$

Le problème (JIT2) est PLM01. La section suivante présente les expériences numériques de nos algorithmes.

6.3.2 Expériences numériques

Les algorithmes ont été exécutés sur un Intel Core 2CPU de 1.86 Ghz, 2GB RAM en C. Pour résoudre la programmation linéaire, nous avons utilisé le logiciel CPLEX 11.2. Les instances sont prises dans OR-Library [114].

Le tableau 6.3.2 présente les résultats pour BB, BBDCA, BCDCAsP, BCDCAaP. Les notations suivantes sont utilisées :

- UB : la borne supérieure,
- LB : la borne inférieure,
- $gap = \frac{UB-LB}{UB}100\%$,
- $itBB$: le nombre d'itérations,
- $Sepa$: le nombre de contraintes de séparation ajoutées,
- $NbUB$: le nombre de contraintes $cx + dy \leq UB$ ajoutées,
- $time$: le temps de calcul (en secondes),
- fIt :le nombre d'itérations au moment où DCA donne la première solution réalisable,
- $fVal$: la valeur de la fonction objectif obtenue quand DCA donne la première solution réalisable,

- $rat = \frac{fVal-LB}{LB}100\%$,
- $ftime$: le temps de calcul quand on a $fVal$,
- $NbDCA$: le nombre de lancements de DCA,
- $nonP$: le nombre de fois qu'on rencontre la procédure P mais non-entrée,
- $demi$: le nombre de fois qu'on rencontre la composante 1/2,
- $still$: le nombre de fois qu'on rencontre la composante 1/2 et qu'on génère régulièrement l'inégalité valide,
- $cutDCA$: le nombre de coupes DCA ajoutées,
- NbP : le nombre de fois qu'on appelle la Procédure P,
- Sub : le nombre de sous-problèmes à résoudre.

Nous observons que :
- DCA fournit toujours une solution réalisable à la première itération avec un faible temps de calcul (voir les colonnes de fIt et $ftime$). DCA fournit la solution globale pour 1 sur 6 instances.
- BBDCA est le meilleur des quatre algorithmes pour 4 sur 6 instances. Cela aussi prouve l'efficacité de BBDCA par rapport à BB.
- BCDCAaP est le meilleur pour seulement une instance. Pour les autres instances, BCDCAsP et BCDCAaP demandent beaucoup de temps pour résoudre les sous-problèmes générés par la Procédure P.

6.4 Conclusion

Nous avons présenté l'application de notre approche sur deux problèmes d'ordonnancement sous la forme de PLM01. Le premier est le problème très connu d'ordonnancement d'avance-retard avec variables indexées sur le temps. Un nouveau modèle concernant la date limite pour chaque tâche, qui est plus proche de la réalité, a été considéré. Nous avons utilisé DCA pour la résolution. Les résultats ont montré l'efficacité de DCA. Le second est le problème d'ordonnancement d'avance-retard avec une date d'échéance commune qui est reformulé après sous la forme de PLM01. Les résultats montrent que BBDCA est prometteur.

Ins	Méthode	UB	LB	gap	itBB	Sépa	NbUB	time	fIt	fVal	rat	ftime	NbDCA	nonP	demi	still	Cut	NbP	Sub
Sche1	BB	210	207.205	1.33	26	1	1	2.094											
	BBDCA	210	201.762	3.92	25	1	1	1.781	1	210	3.92	0.031	1						
	BCDCAsP	210	201.762	3.92	24	1	1	2.078	1	210	3.92	0.031	1	10	1	1	15		
	BCDCAaP	210	206.221	1.80	21	1	1	1.031	1	210	1.80	0.031	1	0	0	0	21	7	34
Sche2	BB	260	247.146	4.94	140	2	2	6.844											
	BBDCA	260	247.146	4.94	139	2	2	2.828	1	294	15.94	0.031	2						
	BCDCAsP	260	247.146	4.94	140	2	2	7.094	1	294	15.94	0.032	2	60	13	13	125		
	BCDCAaP	260	247.146	4.94	119	3	3	8.828	1	294	15.94	0.031	1	0	1	1	131	34	204
Sche3	BB	342	324.875	5.00	724	4	4	37.469											
	BBDCA	342	324.875	5.00	724	3	3	21.359	1	392	17.12	0.031	3						
	BCDCAsP	342	324.875	5.00	726	3	3	51.187	1	392	17.12	0.047	3	251	29	29	741		
	BCDCAaP	342	325.001	4.97	668	3	3	66.453	1	392	17.09	0.031	1	0	9	9	786	136	910
Sche4	BB	428	406.586	5.00	3502	7	7	143.938											
	BBDCA	428	406.586	5.00	3502	6	6	121.562	1	504	19.33	0.047	6						
	BCDCAsP	428	406.579	5.00	3523	5	5	351.953	1	504	19.33	0.047	5	1237	212	209	3496		
	BCDCAaP	428	406.579	5.00	3062	4	3	412.469	1	504	19.33	0.047	1	0	120	113	3540	501	3778
Sche5	BB	528	501.553	5.00	16578	6	6	701.656											
	BBDCA	528	501.553	5.00	16576	5	5	707.719	1	630	24.33	0.078	5						
	BCDCAsP	528	501.553	5.00	16604	6	6	2386.735	1	630	24.33	0.063	5	5176	891	805	17562		
	BCDCAaP	528	501.553	5.00	14674	6	6	2789.610	1	630	24.33	0.078	1	0	456	404	16995	2125	17836
Sche6	BB	642	586.435	8.65	62482	9	9	$> 1h$											
	BBDCA	642	583.609	9.09	60684	7	7	$> 1h$	1	770	24.07	0.078	7						
	BCDCAsP	770	457.438	40.46	15068	1	1	$> 1h$	1	770	40.46	0.094	1	7561	585	526	21750		
	BCDCAaP	658	409.187	37.81	8299	3	3	$> 1h$	1	770	46.86	0.078	1	0	247	223	15466	3280	38746

TAB. 6.4 – Résultats comparatifs entre les algorithmes pour le problème d'ordonnancement

Troisième partie

La programmation DC en variables mixtes 0-1 et ses applications aux problèmes d'affectation des tâches aux véhicules aériens non pilotés et des tournées de véhicules dans une chaîne d'approvisionnement

Algorithme général

Nous étudions dans cette partie une classe des problèmes, plus large que celle étudié dans la deuxième partie, qui est souvent rencontrée dans les applications : la programmation DC en variables mixtes 0-1. Il a été prouvé très récemment dans [64] que la technique de pénalité exacte est également valable pour cette classe des problèmes. Exploitant ce nouveau résultat, nous développons des schémas DCA pour résoudre ces problèmes et les mettons en oeuvre dans deux applications.

Considérons le problème en variables mixtes 0-1 suivant :

$$(DC01) \quad \min\{f(x,y) = g(x,y) - h(x,y) : z(x,y) \leq 0, x \in [0,1]^n, y \in \mathbb{R}^k\}, \tag{6.4}$$

où $g(x,y)$, $h(x,y)$ sont convexes sur $S := \{(x,y) : z(x,y) \leq 0, x \in [0,1]^n, y \in \mathbb{R}^k\}$ et $z(x,y)$ est linéaire.

Ce problème joue un rôle important et est très populaire dans la classe du problème d'optimisation non convexe. Cependant, il est NP- difficile. Trouver une approche efficace pour résoudre ce problème est une grande question et DCA peut être une bonne réponse.

On définit K (l'ensemble des solutions de la relaxation standard de l'ensemble réalisable S) suivant :

$$K := \{(x,y) \in [0,1]^n \times \mathbb{R}^k \ : \ z(x,y) \leq 0\}.$$

Considérons la fonction p définie soit par :

$$p(x,y) \equiv p(x) := \sum_{j=1}^{n} \min\{x_j, 1 - x_j\}, \tag{6.5}$$

soit par :

$$p(x,y) \equiv p(x) := \sum_{j=1}^{n} x_j(1 - x_j). \tag{6.6}$$

Il est clair que :

- p est concave sur $\mathbb{R}^n \times \mathbb{R}^k$,

- $(x, y) \in K$ est un point réalisable si et seulement si $p(x, y) = 0$,
- p est non négative, finie sur K et on a :

$$\begin{aligned} S &:= \{(x, y) \in \{0, 1\}^n \times \mathbb{R}^k_+ : g(x, y) \leq 0\} \\ &\equiv \{(x, y) \in K : \ p(x) = 0\} \\ &\equiv \{(x, y) \in K : \ p(x) \leq 0\}. \end{aligned}$$

Par conséquent, $(DC01)$ peut s'écrire sous la forme d'un programme non convexe en variables continues défini comme suit :

$$(DC01a) \quad \min\{f(x, y) : (x, y) \in K, p(x) \leq 0\}. \tag{6.7}$$

Théorème 6.1 *Soit K polyèdre non vide et borné de $\mathbb{R}^n$. Soient f une fonction DC sur K et p une function concave finie et non négative sur K. Alors il existe un nombre fini $t_0 \geq 0$ tel que avec $t \geq t_0$ les deux problèmes suivants sont équivalents :*

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

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

Preuve : Voir H.A. Le Thi, T. Pham Dinh et N.V. Huynh [64]. □

Compte tenu du Théorème 6.1, il existe un nombre fini t_0 tel que $\forall t \geq t_0$, $(DC01a)$ est équivalent au problème suivant :

$$(DC01b) \quad \min\{c^T x + d^T y + tp(x) : (x, y) \in K\}.$$

La différence entre les deux problèmes $(DC01a)$ et $(DC01b)$ est que dans le premier la non convexité apparaît dans les contraintes alors que dans le deuxième, elle réside dans la fonction objectif. On peut reformuler $(DC01b)$ comme une programmation DC, ensuite utiliser DCA pour le résoudre.

$(DC01b)$ peut se récrire comme

$$\min\{\chi_K(x, y) + g(x, y) - h(x, y) + tp(x) \ : \ (x, y) \in \mathbb{R}^n \times \mathbb{R}^k\}.$$

où

$$\chi_K(x, y) := \begin{cases} 0 & \text{si (x,y)} \in \text{K} \\ +\infty & \text{sinon.} \end{cases} \tag{6.8}$$

est la fonction indicatrice de K.

On note $G(x,y) := \chi_K(x,y) + g(x,y)$ et $H(x,y) := h(x,y) + t(-p)(x,y)$. Comme K est convexe, χ_K est convexe [93]. Donc, G est convexe.
Aussi, H est convexe car h est convexe et p est concave. Dès lors, le problème $(DC01a)$ ou bien $(DC01b)$ est équivalent au problème DC suivant :

$$\min\{G(x,y) - H(x,y) \ : \ (x,y) \in \mathbb{R}^n \times \mathbb{R}^k\}. \tag{6.9}$$

Maintenant, on peut résoudre (6.9) par DCA.

Chapitre 7

Problème d'affectation des tâches aux véhicules aériens non - pilotés

Résumé *Ce chapitre s'intéresse au problème d'affectation des tâches aux véhicules aériens non-pilotés fonctionnant dans les environnements incertains dont l'objectif est de maximiser les scores de cible. Le modèle considéré est sous la forme d'un problème non linéaire en nombres entiers. Nous proposons une approche basée sur la programmation DC et DCA pour résoudre ce problème. Les expériences numériques prouvent l'efficacité de notre approche.*

7.1 Introduction

L'utilisation des véhicules aériens non-pilotés, qui, en anglais, sont appelés unmanned aerial vehicles (UAV) pour les différentes missions militaires attire de plus en plus l'attention depuis ces dernières années. Les avantages sont que, d'une part, on ne met pas la vie humaine en danger et d'autre part, on peut surmonter le manque de pilotes humains ou bien réduire considérablement des coûts. Autrement dit, les UAVs offrent une occasion de nouveaux paradigmes opérationnels. Cependant, afin de réaliser ces avantages, les UAVs doivent avoir un haut niveau d'autonomie et être capables de travailler coopérativement. Dans ce contexte, plusieurs algorithmes qui traitent le problème dans lequel les multi-UAVs effectuent coopérativement des tâches sont développés. Le but est d'affecter des tâches spécifiques et des trajectoires de vol à chaque UAV et de maximiser l'efficacité d'exécution.

Plusieurs recherches ont été effectuées récemment dans ce domaine [4, 35, 74, 91, 92, 95, 96]. Dans [4, 91], l'affectation de tâche a été formulée sous la forme d'une programmation linéaire en variables mixtes 0-1 (PLM01). Dans cette approche, le problème est résolu comme un problème d'optimisation déterministe avec des paramètres connus. Cependant, ce PLM01 n'a pas de résolution efficace. D'ailleurs, les opérations militaires sont en général dynamiques et incertaines en raison de la limitation de détection de l'UAV et des stratégies antagonistes. Ainsi, la replanification est nécessaire à chaque fois que l'information est mise à jour. Les

méthodes heuristiques et ad-hoc ont été considérées pendant la replanification dans [35]. De plus, l'incertitude est considérée via des paramètres d'optimisation, et des techniques de gestion des risques dans les finances sont utilisées [92, 96]. Dans [92], un problème de programmation non linéaire en nombres entiers est formulé dans lequel une mesure du risque par valeur-à-risque conditionnelle est considérée comme la contrainte. Dans [96], une approche robuste utilisant la formulation de Soyster sur l'espérance des scores de cible est étudiée. Ces approches sont basées sur la résolution des problèmes d'optimisation combinatoires difficiles. Une approche alternative traitant les incertitudes se propose de formuler un problème de contrôle optimal stochastique en employant la méthode de MPC (Model Predictive Control)[10, 95].

Dans notre travail, nous nous sommes intéressés à des modèles d'affectation de tâche où nous cherchons à affecter un ensemble de m UAVs à un ensemble de n tâches d'une manière optimale. La mission est de maximiser les scores de cible en satisfaisant des contraintes de capacité des UAV et des tâches. Le modèle que nous considérons est sous la forme d'une programmation non linéaire en nombres entiers pour laquelle les méthodes classiques de solution de PLM01 ne peuvent pas être employées. Nous proposons une approche efficace basée sur la programmation DC et DCA pour le résoudre.

Le reste du chapitre est organisé de façon suivante. La section 7.2 présente la description du problème. La section 7.3 est consacrée à la présentation des algorithmes. Les résultats sont rapportés dans la section 7.4. La section 7.5 termine le chapitre avec quelques conclusions et perspectives.

7.2 Description du problème

Soient V et T les ensembles de m UAVs et de n cibles, respectivement. Chaque cible j a un score associé basé sur la probabilité de succès de tâche r_j et un poids mesurant l'importance de la cible w_j. La probabilité que la tâche soit effectuée avec succès pour cette cible dépend du y_j, le nombre d'UAVs qui sont affectés à la cible j, de la façon suivante :

$$1 - (1 - r_j)^{y_j}. \tag{7.1}$$

Un score de cible est calculé par le produit de la probabilité de succès et de son poids :

$$g_j(y_j) = w_j(1 - (1 - r_j)^{y_j}), \tag{7.2}$$

et l'efficacité de groupe d'UAVs est simplement la somme de tous les scores de cible individuells : $\sum_{j \in T} g_j(y_j)$. Alors le but est de maximiser l'efficacité de groupe d'UAVs.

Le modèle mathématique de l'UTAP peut s'écrire comme suit :

$$(UTAP) \quad \max \sum_{j\in T} w_j\Big(1-(1-r_j)^{\sum\limits_{i\in V} a_{ij}x_{ij}}\Big) \tag{7.3a}$$

tel que :

$$\sum_{j\in T} x_{ij} = 1, \quad i \in V \tag{7.3b}$$

$$x_{ij} \in \{0,1\}, \quad i \in V, j \in T. \tag{7.3c}$$

Les contraintes (7.3b) obligent que chaque UAV soit utilisé pour une seule cible.

7.3 Méthode de résolution

Le problème $(UTAP)$ peut se réécrire comme suit :

$$(UTAP1) \quad \min f(x) := \sum_{j=1}^{n} w_j\Big((1-r_j)^{\sum\limits_{i=1}^{m} a_{ij}x_{ij}}\Big) \tag{7.4a}$$

tel que :

$$x \in K, \tag{7.4b}$$

$$x_{ij} \in \{0,1\}, \quad i \in \{1,\cdots,m\}; j \in \{1,\cdots,n\}, \tag{7.4c}$$

où $K := \big\{x \in \mathbb{R}^{m.n} : \sum\limits_{j=1}^{n} x_{ij} = 1, \quad i \in \{1,\cdots,m\}\big\}$.

Par la technique de pénalité exacte, ce problème est reformulé comme un problème d'optimisation continue de la forme :

$$\min\big\{F(x) := \sum_{j=1}^{n} w_j(1-r_j)^{\sum\limits_{i=1}^{m} a_{ij}x_{ij}} + tp(x) : x \in K'\big\}, \tag{7.5}$$

où $K' := K \cap \big\{[0,1]^{m.n}\big\}$ et p est définie par (6.5).

Pour appliquer DCA, nous avons besoin d'une décomposition DC de F. Il y a plusieurs manières de décomposer la fonction F et nous souhaitons choisir une décomposition DC de telle sorte que le sous-problème convexe (dans le schéma DCA) puisse être facilement résolu.

Dans ce travail, nous utilisons la décomposition DC suivante :

$$F(x) = g(x) - h(x) := \frac{\lambda}{2}\|x\|^2 - \left(\frac{\lambda}{2}\|x\|^2 - F(x)\right) \tag{7.6}$$

où λ est un nombre positif tel que la fonction

$$h(x) := \frac{\lambda}{2}\|x\|^2 - F(x) = \frac{\lambda}{2}\|x\|^2 - f(x) - tp(x)$$

soit convexe. Puisque $-p$ est déjà fonction convexe, h sera convexe si la fonction $\frac{\lambda}{2}\|x\|^2 - f(x)$ est convexe, i.e. si sa matrice hessienne est semi-définie positive (notons que f et $\frac{\lambda}{2}\|x\|^2 - f(x)$) sont deux fois-différentiable).

Pour la simplicité d'écriture, nous représentons $x = (x_{11}, x_{12}, ..., x_{1n}, ..., x_{m1}, x_{m2}, ..., x_{mn})$ comme $x = (x_1, x_2, ..., x_{m.n})$. Autrement dit, un élément x_{ij}, pour $i = 1, ..., m; j = 1, ..., n$, est présenté comme x_l avec $l = n.(i-1) + j$.

Nous avons $\nabla^2 h(x) = \lambda I - \nabla^2 f(x)$ et la matrice hessienne de f est calculée par

$$\big(\nabla^2 f(x)\big)_{ll'} = \frac{\partial f^2}{\partial x_l \partial x_{l'}} = \begin{cases} 0, & \text{si } j \neq j'; \\ a_{ij}a_{i'j'}w_j(1-r_j)^{\sum\limits_{i=1}^{m} a_{ij}x_{ij}}(\log(1-r_j))^2, & \text{sinon,} \end{cases} \tag{7.7}$$

avec $l = (i-1).n + j$ et $l' = (i'-1).n + j'$ pour $i, i' \in \{1, ..., m\}; j, j' \in \{1, ..., n\}$.

La norme de $\nabla^2 f(x)$ peut être calculée :

$$\|\nabla^2 f(x)\|_\infty = \max_{u=1..m,\ v=1..n} \big\{a_{uv}(\sum_{i=1}^{m} a_{iv})w_v(1-r_v)^{\sum\limits_{i=1}^{m} a_{iv}x_{iv}}(\log(1-r_v))^2\big\}.$$

Comme $\sum\limits_{i=1}^{m} a_{iv}x_{iv} \geq 0$ et $(1-r_v) \leq 1$, on a

$$\|\nabla^2 f(x)\|_\infty \leq \max_{u=1..m,\ v=1..n} \big\{a_{uv}(\sum_{i=1}^{m} a_{iv})w_v(\log(1-r_v))^2\big\}. \tag{7.8}$$

Donc, la matrice $\nabla^2 h$ est semi-définie positive quand

$$\lambda \geq \max_{u=1..m,\ v=1..n} \big\{a_{uv}(\sum_{i=1}^{m} a_{iv})w_v(\log(1-r_v))^2\big\}. \tag{7.9}$$

Le problème (7.5) peut être maintenant écrit sous la forme standard de la programmation DC :

$$\min\{G(x) - h(x) : x \in \mathbb{R}^{m.n}\}, \tag{7.10}$$

où $G(x) := \chi_{K'}(x) + \frac{\lambda}{2}\|x\|^2$ est clairement convexe.

Selon le schéma générique, DCA appliqué au dernier problème (7.10) consiste à calculer deux suites $\{y^k\}$ et $\{x^k\}$ telles que :

$$y^k \in \partial h(x^k); \quad x^{k+1} \in \arg\min\big\{G(x) - \langle x, y^k\rangle : x \in \mathbb{R}^{m.n}\big\}.$$

Par la définition de h, y^k peut être calculé comme suit :

$$y_l^k = \begin{cases} \lambda x_l^k + t - w_j \log(1-r_j)a_{ij}(1-r_j)^{\sum\limits_{i=1}^{m} a_{ij}x_{ij}^k}, & \text{si } x_l \geq 0.5; \\ \lambda x_l^k - t - w_j \log(1-r_j)a_{ij}(1-r_j)^{\sum\limits_{i=1}^{m} a_{ij}x_{ij}^k}, & \text{si } x_l < 0.5 \end{cases} \tag{7.11}$$

avec $l = i.(n-1) + j$ pour $i = 1, ..., m; j = 1, ..., n$.
x^{k+1} est en fait une solution optimale du problème quadratique convexe suivant :

$$\min\left\{\frac{\lambda}{2}\|x\|^2 - \langle y^k, x\rangle : x \in K'\right\}.$$

Finalement, DCA appliqué à (7.10) peut être décrit comme suit :

Algorithme 7.1 ***(Algorithme DCA-UTAP)***
Etape 1. Choisir un point initial x^0 et un $\epsilon > 0$ assez petit. Poser $k = 0$.
Etape 2. Calculer $y^k \in \partial h(x)$ via (7.11).
Etape 3. Résoudre le problème quadratique convexe (7.12) pour obtenir x^{k+1} :

$$\min\left\{\frac{\lambda}{2}\|x\|^2 - \langle y^k, x\rangle : x \in K'\right\}. \tag{7.12}$$

Etape 4. ***Si*** *($\|x^{k+1} - x^k\| \leq \epsilon(\|x^k\| + 1)$* ***Alors*** *STOP, x^k est la solution calculée,*
Sinon poser $k := k + 1$ et passer à l'étape 2.

Pour résoudre globalement ($UTAP1$), nous combinons DCA avec le schéma de B&B classiques. Les bornes inférieures sont calculées en résolvant le problème relaxé tandis que les bornes supérieures sont calculées en appliquant **DCA-UTAP** à (7.10). Au début, le point initial de **DCA-UTAP** est la solution optimale du problème relaxé. En outre, quand un point réalisable est trouvé tout en calculant les bornes inférieures, nous relançons **DCA-UTAP** à partir de ce point.

Borne inférieure
Nous calculons une borne inférieure en résolvant le problème relaxé du ($UTAP1$) :

$$\min\{f(x) = \sum_{j=1}^{n} w_j(1 - r_j)^{\sum_{i=1}^{m} a_{ij}x_{ij}} : (x, y) \in K'\}. \tag{7.13}$$

Le dernier problème est convexe. Donc, nous pouvons appliquer DCA à nouveau pour le résoudre.

Comme DCA appliqué au (7.10), la décomposition DC suivante est employée pour (7.13) :

$$f(x) = \sum_{j=1}^{n} w_j(1 - r_j)^{\sum_{i=1}^{m} a_{ij}x_{ij}} = \overline{g}(x) - \overline{h}(x), \tag{7.14}$$

où

$$\overline{g}(x) := \frac{\lambda}{2}\|x\|^2; \quad \overline{h}(x) := \frac{\lambda}{2}\|x\|^2 - \sum_{j=1}^{n} w_j(1 - r_j)^{\sum_{i=1}^{m} a_{ij}x_{ij}}. \tag{7.15}$$

Ici, λ prend la même valeur que dans la formule (7.9). DCA appliqué au (7.13) n'est rien d'autre que **DCA-UTAP** avec la modification suivante dans l'étape du calcul de y^k :

$$y_l^k = \lambda x_l^k - w_j \log(1 - r_j) a_{ij} (1 - r_j)^{\sum_{i=1}^{m} a_{ij} x_{ij}^k} \tag{7.16}$$

avec $l = i.(n-1) + j$ pour $i = 1, ..., m, j = 1, ..., n$.

Algorithme 7.2 ***Algorithme DCABB***
Poser $R_0 := [0,1]^{m.n}$ *et soit* ϵ *un nombre positif suffisamment petit. Poser restart := true ;*
Résoudre le problème relaxé convexe (7.13) pour obtenir une solution x^{R_0} *et la première borne inférieure* $\beta_0 := \beta(R_0)$ *;*
Résoudre (7.10) par DCA à partir du point x^{R_0} *pour obtenir* $x_t^{R_0}$ *;*
Si $x_t^{R_0}$ *est réalisable pour (UTAP)* **alors**
poser $\gamma_0 := f(x_t^{R_0})$, $x^0 := x_t^{R_0}$, *restart := false* **sinon** $\gamma := +\infty$ *;*
Fin si
Si $(\gamma_0 - \beta_0) \leq \epsilon|\gamma_0|)$ **alors** *STOP,* x^0 *est une* ϵ*-solution optimale de (UTAP1)* **sinon** *poser* $\Re \leftarrow \{R_0\}, k \leftarrow 0$ *;*
Fin si
Tant que *TRUE* **faire**
Choisir un rectangle R_k *tel que* $\beta_k = \beta(R_k) = \min\{\beta(R) : R \in \Re\}$.
Choisir $j^* \in \{1, \ldots, m.n\}$ *l'index tel que* $x_{j^*}^{R_k} \notin \{0,1\}$. *Diviser* R_k *en deux sous-rectangles* R_{k_0} *et* R_{k_1} *via l'index* j^* *:*

$$R_{k_i} = \{x \in R_k : x_{j^*} = i; i = 0, 1\}. \tag{7.17}$$

Résoudre les sous-problèmes (P_{k_i}) *pour obtenir* $\beta(R_{k_i})$ *et* $(x^{R_{k_i}})$ *:*

$$(P_{k_i}) \quad \beta(R_{k_i}) = \min\{f(x) : x \in K, x \in R_{k_i}\}. \tag{7.18}$$

Si *soit* $x^{R_{k_i}}$ *est réalisable pour (UTAP1), soit* restart = true **alors**
- appliquer DCA au (7.10) à partir de $x^{R_{k_i}}$ *pour obtenir* $x_t^{R_{k_i}}$ *;*
- **Si** $x_t^{R_{k_i}}$ *est réalisable pour (UTAP1)* **alors**
mettre à jour γ_k *et la meilleure solution réalisable* x^k *;*
poser restart := false.
Fin si
Fin si
Poser $\Re \leftarrow \Re \cup \{R_{k_i} : \beta(R_{k_i}) < \gamma_k - \epsilon, i = 0, 1\} \setminus R_k$.
Si $\Re = \emptyset$ **alors** *STOP,* x^k *est une* ϵ*-solution optimale* **sinon** *poser* $k \leftarrow k + 1$.
Fin tant que

7.4 Expériences numériques

Les algorithmes ont été exécutés sur un Intel Core 2CPU de 1.86 Ghz, 2GB RAM. Pour résoudre la programmation quadratique convexe, nous avons utilisé le logiciel CPLEX 11.2.

Nous avons testé 30 jeux de données avec 3 tailles différentes : $(m = 10, n = 10)$, $(m = 20, n = 10)$ et $(m = 30, n = 10)$ (10 jeux de données par chaque taille). Les poids w_j sont générés aléatoirement dans $[0, 10]$. Les probabilités r_j et la matrice A sont également générées aléatoirement.

Les résultats comparatifs entre trois algorithmes **DCA-UTAP**, **DCABB** et **BB** sont rapportés dans les Tableaux 7.1, 7.2 et 7.3. Les notations suivantes sont employées dans ces tableaux :

- Pb : le problème,
- Bin : le nombre de variables binaires,
- Ctr : le nombre de contraintes,
- OPT : la valeur de la fonction objectif obtenue par chaque algorithme,
- LB : la meilleure borne inférieure (donnée par **BB**),
- It : le nombre d'itérations de chaque algorithme,
- $Time$: le temps de calcul en seconde de chaque algorithme,
- $GAP = \frac{OPT-LB}{OPT} 100\%$,
- Nb : le nombre de lancements de DCA dans **DCABB** pour obtenir l'OPT.

Commentaires : A partir des résultats dans les tableaux suivants, nous constatons que :

◇ **DCA-UTAP** fournit une bonne solution approximative en peu de temps. Pour la taille $(m = 10, n = 10)$ (variables binaires de 100) le *Gap* moyen (entre la valeur optimale et le meilleure borne inférieure) est de $6,12\%$ tandis que le temps de calcul moyen est de $0,625$ secondes. Quand $(m = 20, n = 10)$ (resp. $(m = 30, n = 10)$) le gap moyen est de $11,23$ (resp. $10,46$) et le temps de calcul moyen est de $1,88$ secondes (resp. de $2,85$). Donc, l'utilisation **DCA-UTAP** est intéressante pour le problème d'UTAP, en particulier pour le problème de grande taille. En effet, avec $(m = 30, n = 10)$**BB** ne fonctionne pas (l'algorithme ne peut pas fournir une solution réalisable après une heure) et **DCABB** améliore légèrement la solution obtenue par**DCA-UTAP** tandis que le temps consommé augmente nettement (plus d'une heure pour 7/10 jeux de données).

◇ **DCABB** est intéressant pour les problèmes de taille moyenne. Le Tableau 7.1 montre que, en général, après quelques lancements de **DCA-UTAP** l'algorithme donne une solution optimale. Mais, comme dans n'importe quel algorithme de B&B, nous devons continuer la procédure B&B pour prouver la globalité de la solution en améliorant les bornes inférieures. Ici l'intérêt d'utilisation de schéma de B&B est double : pour trouver un bon point initial de DCA, et, pour prouver la globalité de la solution obtenue par DCA. Grâce à DCA, **DCABB** est beaucoup plus efficace que **BB**.

◇ **BB** ne fonctionne pas pour les problèmes de plus de 100 variables binaires.

7.5 Conclusion

Nous avons développé une approche efficace basée sur la programmation DC et DCA pour le problème d'affectation des tâches aux UAVs dans un environnement incertain qui est

Data			DCA				DCABB					BB			
Pb	Bin	Ctr	It	OPT	Time	Gap	It	Nb	OPT	Time	Gap	It	OPT	Time	Gap
1	100	10	25	33.125266	0.953	11.40	3	5	29.911818	10.562	1.98	59	29.425858	170.890	0.26
2	100	10	3	43.600538	0.125	6.71	5	9	41.502606	5.922	2.03	1276	40.790448	1919.812	0.29
3	100	10	19	31.875111	0.875	6.33	1	1	29.927311	3.000	0.23	158	29.927309	776.766	0.17
4	100	10	13	28.845089	0.641	12.07	187	130	26.106369	723.906	2.99	1147	NA	>1h	NA
5	100	10	9	34.970767	0.500	0.15	1	1	34.964411	0.641	0.13	11	34.964398	8.937	0.06
6	100	10	3	30.422789	0.110	7.95	794	5	28.286803	2896.813	1.03	794	28.286803	2887.265	1.03
7	100	10	8	39.284260	0.375	10.75	40	2	35.454166	48.422	1.15	40	35.454166	47.859	1.15
8	100	10	9	29.738714	0.516	1.03	18	2	29.528116	23.437	0.33	18	29.528116	23.110	0.33
9	100	10	21	19.188294	1.313	0.46	12	2	19.188199	40.063	0.46	12	19.188199	38.578	0.46
10	100	10	21	16.315438	0.844	4.35	299	2	15.761578	816.407	2.08	299	15.761578	797.063	2.08

TAB. 7.1 – Résultats comparatifs avec $m = 10$ et $n = 10$

Data			DCA			DCABB				BB		
Pb	Bin	Ctr	It	OPT	Time	Gap	It	Nb	OPT	Time	Gap	It	OPT	Time	Gap
1	200	20	7	25.800585	0.812	8.16	35	69	23.765625	675.485	0.30	218	NA	>1h	NA
2	200	20	26	23.642896	2.985	10.33	85	27	22.784809	>1h	6.95	85	NA	>1h	NA
3	200	20	15	26.760725	1.734	14.99	7	12	22.883666	136.656	0.60	168	NA	>1h	NA
4	200	20	12	25.687555	1.391	10.11	56	7	28.328693	>1h	22.69	58	NA	>1h	NA
5	200	20	3	21.372439	0.375	14.87	33	5	24.216463	>1h	22.09	40	NA	>1h	NA
6	200	20	47	28.852983	5.485	13.97	52	NA	NA	>1h	NA	59	NA	>1h	NA
7	200	20	6	24.942689	0.718	9.71	94	187	23.102439	655.064	2.59	507	NA	>1h	NA
8	200	20	24	21.801630	3.105	11.49	64	NA	NA	>1h	NA	275	NA	>1h	NA
9	200	20	4	14.539893	0.813	12.57	36	1	14.060431	>1h	10.61	37	NA	>1h	NA
10	200	20	12	25.004022	1.391	6.09	878	7	25.418418	>1h	8.25	878	NA	>1h	NA

TAB. 7.2 – Résultats comparatifs avec $m = 20$ et $n = 10$

Data			DCA				DCABB					BB			
Pb	Bin	Ctr	It	OPT	Time	Gap	It	Nb	OPT	Time	Gap	It	OPT	Time	Gap
1	300	30	8	16.681124	2.484	14.08	236	471	14.766721	2167.078	3.04	319	NA	>1h	NA
2	300	30	3	15.189069	0.750	9.95	398	NA	NA	>1h	NA	486	NA	>1h	NA
3	300	30	4	15.970833	0.984	7.44	96	35	16.023174	>1h	7.74	112	NA	>1h	NA
4	300	30	26	16.599515	6.343	14.02	630	967	15.400154	>1h	7.91	765	NA	>1h	NA
5	300	30	15	19.453271	3.658	9.41	47	93	18.481463	1619.641	4.87	73	NA	>1h	NA
6	300	30	14	20.639400	3.365	11.36	42	45	20.195123	>1h	10.38	57	NA	>1h	NA
7	300	30	17	26.161186	4.680	6.75	9	17	25.394145	235.125	4.09	139	NA	>1h	NA
8	300	30	4	18.779954	0.969	8.35	197	18	19.256186	>1h	11.89	202	NA	>1h	NA
9	300	30	7	22.471048	1.734	10.77	77	1	25.752417	>1h	28.55	77	NA	>1h	NA
10	300	30	13	20.846978	3.584	12.50	121	13	19.914118	>1h	9.17	126	NA	>1h	NA

TAB. 7.3 – Résultats comparatifs avec $m = 30$ et $n = 10$

un problème difficile. Le problème combinatoire original est reformulé sous la forme d'un problème continu par la technique de pénalité exacte.

Le schéma DCA proposé consiste à résoudre un programme quadratique convexe à chaque itération. Bien que DCA soit une approche continue, elle donne des solutions en nombres entiers. Les résultats numériques prouvent l'efficacité et la robustesse du DCA : il donne une bonne solution optimale approximative en peu de temps. Donc, DCA est recommandé pour ce problème, particulièrement pour les instances de grande taille. En outre, pour les instances de taille moyenne, nous suggérons DCA-B&B puisqu'il peut trouver une solution optimale dans un temps modéré.

Ce travail suggère qu'il soit intéressant d'étudier DCA pour d'autres modèles non linéaires pour le problème d'affectation de tâches.

Chapitre 8

Problème des tournées de véhicules dans une chaîne d'approvisionnement

Résumé *Dans ce chapitre, nous étudions le problème des tournées de véhicules dans une chaîne d'approvisionnement. Il s'agit d'affecter des véhicules à partir d'un dépôt central aux points de vente et déterminer ses cycles pour minimiser le coût de transport, le coût de stockage, le coût de traitement... C'est un problème important et très difficile à cause de la forme complexe de la fonction objectif, en plus des variables binaires. Nous proposons un schémas de DCA pour sa résolution numérique.*

8.1 Introduction

Le problème des tournées de véhicules et stockage (en anglais, inventory routing problem-IRP) consiste à établir un plan de distribution cyclique d'un produit à partir d'un dépôt central à un ensemble de points de vente. Chaque point de vente a un taux de demande. L'objectif est de minimiser les coûts opératoires, de transport, de manutention et de stockage. Deux décisions doivent être prises. La première concerne la fréquence avec laquelle chaque point de vente devrait être servi (la durée de cycle). La deuxième est de déterminer la route qui va être utilisée. Les deux décisions ne peuvent pas être indépendantes. C'est la raison pour laquelle on a besoin de l'intégration de la gestion des stocks et le routage de véhicule. Federgruen et Zipkin [14] sont probablement les pionniers étudiant ce problème. Ils ont modélisé le problème comme un problème non linéaire en variables mixtes 0-1 et proposé une méthode d'approximation pour la résolution. Chien et al. [7] ont formulé le problème comme une programmation entière. Puis elle a été résolue par la méthode duale lagrangienne. Récemment, Aghezzaf et al.[1, 111] ont étudié un modèle non convexe. Pour sa résolution, ils ont proposé la méthode de la génération de colonnes [1]. Ils ont également essayé de traiter ce problème en résolvant plusieurs problèmes linéaires en variables mixtes 0-1 [111]. Clairement, une telle procédure est coûteuse.

Dans ce chapitre, nous considérons un IRP dans lequel il y a un seul véhicule. Ce problème est formulé sous la forme d'une programmation non convexe en variables mixtes 0-1 qui est très difficile à résoudre. Plus particulièrement, la non convexité se manifeste dans la fonction objectif. Notre travail est de chercher une décomposition DC de cette fonction. Ensuite, nous appliquons DCA pour le résoudre. Ce chapitre est organisé comme suit. La section 8.2 est consacrée à la description du problème. La section 8.3 présente la formulation DC de la fonction objectif. La section 8.4 est dédiée à la détermination d'une borne inférieure. Les expériences numériques sont présentées dans la section 8.5. Nous terminons ce chapitre par la conclusion.

8.2 Description du problème

Considérons un véhicule servant n points de vente à partir d'un dépôt central r. Soit K la capacité du véhicule. Il transporte à la vitesse moyenne ν (km/h). Chaque point de vente a un taux de demande d_i (i.e. la demande pour unité de temps). La durée de transport à partir du point de vente $i \in S^+ = S \cup \{r\}$ au point $j \in S^+$ est dénotée t_{ij}. Les variables utilisées sont :

- x_{ij} : la variable binaire $x_{ij} = 1$ si le point de vente $j \in S^+$ est immédiatement servi après la sortie du point de vente $i \in S^+$, et $x_{ij} = 0$ sinon,
- Q_{ij} : la quantité de produits restante dans le véhicule quand il voyage de $i \in S^+$ à $j \in S^+$,
- q_j : la quantité délivrée à $j \in S$,
- T : la durée de cycle.

On a quatre composantes de coût (le taux de coût) suivants :

- le coût opératoire ϕ,
- le coût de transport : $\frac{1}{T}\sum_{i\in S^+}\sum_{j\in S^+}(\delta\nu t_{ij}x_{ij})$, où δ est le coût de voyage par kilomètre.
- le coût de manutention $\frac{\sum_{i\in S}\varphi_i}{T}$, où φ_i est le coût de livraison pour chaque point i,
- le coût de stockage $\sum_{i\in S}\frac{1}{2}\eta_i q_i$, où η est le coût de stockage (euros/tonne.h), la variable q_i est la quantité de produit livré pour le point i (supposons que le stockage moyen soit $\frac{q_i}{2}$).

Le problème (IRP) peut se réécrire comme suit :

$$(IRP) \qquad \min \phi + \frac{1}{T}\Big(\sum_{i\in S^+}\sum_{j\in S^+}(\delta\nu t_{ij}x_{ij}) + \sum_{i\in S}\varphi_i\Big) + \sum_{i\in S}\frac{1}{2}\eta_i q_i \tag{8.1}$$

tel que :

$$\sum_{i\in S^+} x_{ij} = 1, \quad \forall j \in S, \tag{8.2}$$

$$\sum_{i\in S^+} x_{ij} - \sum_{k\in S^+} x_{jk} = 0, \quad \forall j \in S^+, \tag{8.3}$$

$$\sum_{i\in S^+}\sum_{j\in S^+} t_{ij}x_{ij} - T \le 0, \tag{8.4}$$

$$\sum_{i\in S^+} Q_{ij} - \sum_{k\in S^+} Q_{jk} = q_j, \quad \forall j \in S, \tag{8.5}$$

$$Q_{ij} \leq K x_{ij}, \quad \forall i, j \in S^+, \tag{8.6}$$

$$q_j \geq d_j T, \quad \forall j \in S, \tag{8.7}$$

$$\sum_{j\in S} q_j \leq K, \tag{8.8}$$

$$x_{ij} \in \{0,1\}, \quad Q_{ij} \geq 0, \quad q_j \geq 0, \quad T \geq 0, \quad i, j \in S^+. \tag{8.9}$$

Les contraintes (8.2) indiquent que chaque point de vente est servi une et seulement une fois. Les contraintes (8.3) assurent que le véhicule doit partir immédiatement après avoir servi un point de vente. La contrainte (8.4) montre que la durée de cycle est égale ou supérieure à la durée totale de transport. Les contraintes (8.5) assurent que la quantité transportée par le véhicule ne dépasse pas sa capacité maximale. L'équilibre entre la charge et la livraison est indiqué dans les contraintes (8.6). Les contraintes (8.7) montrent que la quantité livrée au point de vente est supérieure ou égale à sa demande.

Notons que la durée du cycle T est bornée dans un intervalle $[T_{min}, T_{max}]$. T_{min} peut être calculé en résolvant le problème du voyageurs de commerce tandis que T_{max} est déterminé à partir des contraintes (8.7)-(8.8) :

$$T_{max} = \frac{K}{\sum_{i\in S} d_i}.$$

La fonction objectif est non-convexe. Dans la suite, nous la transformons en une fonction DC et utilisons DCA pour sa résolution.

8.3 Méthode de résolution

Puisque ϕ est une constante, (IRP) se réécrit comme suit :

$$(IRP1) \quad \min f(z) := \frac{1}{T}\Big(\sum_{i\in S^+}\sum_{j\in S^+} (\delta\nu t_{ij} x_{ij}) + \sum_{i\in S} \varphi_i\Big) + \sum_{i\in S} \frac{1}{2}\eta_i q_i \tag{8.10}$$

tel que :

$$\sum_{i\in S^+} x_{ij} = 1, \quad \forall j \in S, \tag{8.11}$$

$$\sum_{i\in S^+} x_{ij} - \sum_{k\in S^+} x_{jk} = 0, \quad \forall j \in S^+, \tag{8.12}$$

$$\sum_{i\in S^+}\sum_{j\in S^+} t_{ij} x_{ij} - T \leq 0, \tag{8.13}$$

$$\sum_{i\in S^+} Q_{ij} - \sum_{k\in S^+} Q_{jk} = q_j, \quad \forall j \in S, \tag{8.14}$$

$$Q_{ij} \leq K x_{ij}, \quad \forall i,j \in S^+, \tag{8.15}$$

$$q_j \geq d_j T, \quad \forall j \in S, \tag{8.16}$$

$$\sum_{j\in S} q_j \leq K, \tag{8.17}$$

$$x_{ij} \in \{0,1\}, \quad Q_{ij} \geq 0, \quad q_j \geq 0, \quad T \geq 0, \quad i,j \in S^+, \tag{8.18}$$

où la vecteur de variable z (écrite dans la fonction objectif) est $z = (x, Q, q, T)$.

Tout d'abord, on cherche une décomposition DC de f. La fonction f est décomposée comme suit :

$$f(z) := g(z) - h(z) = \frac{\lambda}{2}\|z\|^2 - \Big(\frac{\lambda}{2}\|z\|^2 - f(z)\Big), \tag{8.19}$$

où λ est un nombre positif tel que la fonction

$$h(z) := \frac{\lambda}{2}\|z\|^2 - f(z) \tag{8.20}$$

est convexe i.e. sa matrice hessienne est semi-définie positive.

Pour la simplicité d'écriture, nous représentons

$$z = (x_{00}, x_{01}, ..., x_{0n}, ..., x_{n0}, x_{n1}, ..., x_{nn}, Q_{00}, Q_{01}, ..., Q_{0n}, ..., Q_{n0}, Q_{n1}, ..., Q_{nn}, q_1, q_2, ..., q_n, T)$$

comme $z = (z_1, z_2, ..., z_{2(n+1)^2+n+1})$.

Nous avons $\nabla^2 h(z) = \lambda I - \nabla^2 f(z)$ où $\big(\nabla^2 f(z)\big)_{ll'}$ est calculée par :

$$\frac{\partial f^2}{\partial z_l \partial x_{l'}} = \begin{cases} -\frac{\delta\nu t_{ij}}{T^2}, & \text{si } \begin{cases} (l = 1, ..., (n+1)^2 & \text{et } l' \neq 2(n+1)^2 + n + 1) \text{ ou} \\ (l \neq 2(n+1)^2 + n + 1 & \text{et } l' = 1, ..., (n+1)^2 \end{cases} \\ \frac{2}{T^3}\big(\sum\limits_{i,j\in S^+} \delta\nu t_{ij} x_{ij} + \sum\limits_{i\in S} \varphi_i\big), & \text{si } l = l' = 2(n+1)^2 = n+1\,; \\ 0, & \text{sinon.} \end{cases} \tag{8.21}$$

La norme de $\nabla^2 f(z)$ peut être calculée par :

$$\|\nabla^2 f(z)\|_\infty = \sum_{i,j\in S^+} \frac{\delta\nu t_{ij}}{T^2} + \frac{2}{T^3}\big(\sum_{i,j\in S^+} \delta\nu t_{ij} x_{ij} + \sum_{i\in S} \varphi_i\big).$$

$$\|\nabla^2 f(z)\|_\infty \leq \sum_{i,j\in S^+} \frac{\delta\nu t_{ij}}{T^2} + \frac{2}{T^3}\big(\delta\nu T + \sum_{i\in S} \varphi_i\big) \leq \frac{\delta\nu}{T_{min}^2}\big(\sum_{i,j\in S^+} t_{ij} + 2\big) + \frac{2\sum\limits_{i\in S}\varphi_i}{T_{min}^3}. \tag{8.22}$$

Ici T_{min} est la borne inférieure de T. Clairement, la durée d'un cycle du problème du voyageur de commerce (noté T_{TSP}) est la borne inférieure de T. Cependant, le calcul de T_{TSP} est aussi difficile. Donc, on peut calculer T_{min} plus facilement par la borne inférieure de T_{TSP} en cherchant l'arbre couvrant de poids minimal (l'algorithme de Prim, par exemple).

Pour que $\nabla^2 h$ soit semi-définie positive, il faut que λ satisfasse la condition suivante :

$$\lambda \geq \Big(\frac{\delta\nu}{T_{min}^2}\big(\sum_{i,j\in S^+} t_{ij} + 2\big) + \frac{2\sum\limits_{i\in S}\varphi_i}{T_{min}^3}\Big). \tag{8.23}$$

Le problème (IRP1) peut s'écrire comme suit :

$$\begin{aligned}
(IRP2)\quad & \min\big(g(z) - h(z)\big) && (8.24a)\\
& \text{tel que :} \\
& z \in D, && (8.24b)\\
& p(z) \leq 0, && (8.24c)\\
& x \in [0,1]^{(n+1)^2}, && (8.24d)
\end{aligned}$$

où D est l'ensemble de contraintes (8.11)-(8.17), et $p(z) \equiv p(x) = \sum\limits_{i,j\in S^+} x_{ij}(1 - x_{ij})$.

Par la technique de pénalité exacte, ce problème est reformulé comme un problème d'optimisation continue de la forme :

$$\min\Big\{F(z) := \big(\chi_{D'}(z) + g(z)\big) - \big(h(z) - \xi p(z)\big)\Big\}, \tag{8.25}$$

où ξ est une constante suffisamment grande et $D' := D \cap \big\{x : x \in [0,1]^{(n+1)^2}\big\}$.

Clairement, $G(z) := \chi_{D'}(z) + g(z)$ et $H(z) := h(z) - \xi p(z)$ sont convexes. Donc on peut utiliser DCA pour résoudre (8.25).

Selon le schéma générique, DCA appliqué au problème (8.25) consiste à calculer les deux suites $\{u^k\}$ et $\{z^k\}$ telles que :

$$u^k \in \partial H(z^k); \quad z^{k+1} \in \arg\min\Big\{G(z) - \langle z, u^k\rangle : z \in \mathbb{R}^{2(n+1)^2+n+1}\Big\}.$$

On réécrit

$$H(z) = \frac{\lambda}{2}\|z\|^2 - \frac{1}{T}\Big(\sum_{i\in S^+}\sum_{j\in S^+}(\delta\nu t_{ij}x_{ij}) + \sum_{i\in S}\varphi_i\Big) - \sum_{i\in S}\frac{1}{2}\eta_i q_i + \xi\sum_{i\in S^+}\sum_{j\in S^+} x_{ij}(x_{ij} - 1). \tag{8.26}$$

Donc,

$$\begin{cases}
\frac{\partial H(z)}{\partial x_{ij}} = \lambda x_{ij} - \frac{\delta\nu t_{ij}}{T} + 2\xi x_{ij} - \xi, \\
\frac{\partial H(z)}{\partial Q_{ij}} = \lambda Q_{ij}, \\
\frac{\partial H(z)}{\partial q_i} = \lambda q_i - \frac{\eta_i}{2}, \\
\frac{\partial H(z)}{\partial T} = \lambda T + \frac{1}{T^2}\Big(\sum\limits_{i\in S^+}\sum\limits_{j\in S^+}(\delta\nu t_{ij}x_{ij}) + \sum\limits_{i\in S}\varphi_i\Big).
\end{cases}$$

La suite u_l^k est calculée par

$$u_l^k = \begin{cases} \lambda z_l^k - \frac{\delta \nu t_{ij}}{T} + 2\xi z_l^k - \xi, & l = 1, ..., (n+1)^2\,; \\ \lambda z_l^k, & l = (n+1)^2 + 1, ..., 2(n+1)^2\,; \\ \lambda z_l^k - \frac{\eta_i}{2}, & l = 2(n+1)^2 + 1, ..., 2(n+1)^2 + n\,; \\ \lambda z_l^k + \frac{1}{(z_l^k)^2}\Big(\sum\limits_{i \in S^+} \sum\limits_{j \in S^+} (\delta \nu t_{ij} x_{ij}) + \sum\limits_{i \in S} \varphi_i \Big), & l = 2(n+1)^2 + n + 1. \end{cases} \tag{8.27}$$

Par ailleurs, z^{k+1} est une solution optimale du problème quadratique convexe suivant :

$$\min \left\{ \frac{\lambda}{2}\|z\|^2 - \langle u^k, z \rangle : z \in D' \right\}. \tag{8.28}$$

Procedure PI (Point initial pour DCA)
Pour choisir un point initial pour DCA, nous cherchons un cycle *Hamiltonien* qui passe par de tous les points de vente et le dépôt central. Les variables binaires du point initial sont égales aux valeurs données par le cycle *Hamiltonien*. Tant qu'aux variables continues, elles sont choisies aléatoirement. La procédure pour chercher un cycle *Hamiltonien* est la suivante :
– **Entrée** : Un dépôt central r et n points de vente.
– **Initialisation** : Le dépôt central r est choisi comme le premier point du cycle. Il est aussi marqué "point courant". Tous les points de vente sont marqués *non-visités*.
– **Tant qu**'il existe encore des points de vente *non-visité*, **faire**
  – Choisir i-le point de vente *non-visité* le plus proche du point courant.
  – Ajouter i au cycle. On le marque *visité* et i devient donc le point courant du cycle.
– **Sortie** : un cycle *Hamiltonien*.

Finalement, DCA appliqué au (IRP1) se décrit comme suit :
Algorithme DCA-IRP :
Etape 1 : Choisir un point initial z^0 via la procédure PI ci-dessus et $\epsilon > 0$. Poser $k = 0$;
Etape 2 : Calculer $u^k \in \partial h(x)$ via (8.27) ;
Etape 3 : Résoudre le problème quadratique convexe (8.28) pour obtenir z^{k+1} ;
Etape 4 : **Si** $(\|z^{k+1} - z^k\| \leq \epsilon(\|z^k\| + 1)$ **alors** stop, z^k est la solution obtenue,
sinon poser $k = k + 1$ et aller à l'Etape 2 ;

8.4 Calculer une borne inférieure

Lemme 8.1 [20] *Soient $p \geq 0$, $r \geq 0$. Si $u(x)$, $v(x)$ sont des fonctions convexes et finies sur $\mathbb{R}^n$ alors la fonction* $\max\{ru(x) + pv(x) - pr, su(x) + qv(x) - qs\}$ *est un minorant convexe de $u(x)v(x)$ sur $\{x \in \mathbb{R}^n | p \leq u(x) \leq q, r \leq v(x) \leq s\}$.*

Preuve : cf. [20]. □

Considérons la fonction objectif de $(IRP1)$. On a

$$f(z) \geq \sum_{i\in S^+}\sum_{j\in S^+} \delta\nu t_{ij} z_{ij} + \sum_{i\in S} \frac{\varphi_i}{T} + \sum_{i\in S} \frac{1}{2}\eta_i q_i,$$

où z_{ij} est minorant convexe de la fonction $f_{ij} := x_{ij}\frac{1}{T}$. Il est facile de vérifier que les fonctions x_{ij} et $\frac{1}{T}$ sont convexes. De plus, nous avons : $0 \leq x_{ij} \leq 1$ et $\frac{1}{T_{max}} \leq \frac{1}{T} \leq \frac{1}{T_{min}}$.

A partir du Lemme 8.1, un minorant convexe de la fonction $f_{ij} = x_{ij}\frac{1}{T}$ peut être défini comme suit :

$$z_{ij} = \max\{\frac{x_{ij}}{T_{max}}; \frac{x_{ij}}{T_{min}} + \frac{1}{T} - \frac{1}{T_{min}}\}. \tag{8.29}$$

Soit α la valeur optimale du problem original. Dénotons α_1 la valeur optimale du problème suivant :

$$(Relax1) \qquad \alpha_1 = \min \sum_{i\in S^+}\sum_{j\in S^+} (\delta\nu t_{ij} z_{ij}) + \frac{\sum_{i\in S}\varphi_i}{T} + \sum_{i\in S} \frac{1}{2}\eta_i q_i \tag{8.30a}$$

tel que :

$$z_{ij} \geq \frac{x_{ij}}{T_{max}}, \quad \forall i,j \in S^+, \tag{8.30b}$$

$$z_{ij} \geq \frac{x_{ij}}{T_{min}} + \frac{1}{T} - \frac{1}{T_{min}}, \quad \forall i,j \in S^+, \tag{8.30c}$$

$$\text{les contraintes (8.2)-(8.8)}, \tag{8.30d}$$

$$x_{ij} \in \{0,1\}, Q_{ij} \geq 0, q_j \geq 0, T \geq 0, z_{ij} \geq 0, \quad i,j \in S^+. \tag{8.30e}$$

α_1 est donc une borne inférieure de α.

Dans le problème $(Relax1)$, nous relaxons la contrainte $x_{ij} \in \{0,1\}$ par $x_{ij} \in [0,1]$ et remplaçons le terme $\frac{1}{T}$ par son minorant affine. Puisque $\frac{1}{T}$ est convexe sur l'intervalle $[T_{min}, T_{max}]$, un minorant affine de $\frac{1}{T}$ peut être choisi par sa tangente au point $T_0 \in [T_{min}, T_{max}]$. La tangente est déterminée comme suit : $-\frac{1}{T_0^2}T + \frac{2}{T_0}$.

Alors, nous obtenons un nouveau problème relaxé :

$$(Relax2) \qquad \alpha_2 = \min \sum_{i,j\in S^+} (\delta\nu t_{ij} z_{ij}) + \sum_{i\in S} \varphi_i(\frac{2}{T_0} - \frac{T}{T_0^2}) + \sum_{i\in S} \frac{1}{2}\eta_i q_i \tag{8.31a}$$

tel que :

$$z_{ij} \geq \frac{x_{ij}}{T_{max}}, \quad \forall i,j \in S^+, \tag{8.31b}$$

$$z_{ij} \geq \frac{x_{ij}}{T_{min}} - \frac{T}{T_0^2} + \frac{2}{T_0} - \frac{1}{T_{min}}, \quad \forall i,j \in S^+, \tag{8.31c}$$

$$\text{les contraintes (8.30d)-(8.30e)} . \tag{8.31d}$$

Le problème $(Relax2)$ n'est rien d'autre qu'un problème linéaire que nous pouvons facilement résoudre. Sa valeur optimale α_2 donne une borne inférieure de $(IRP1)$ $(\alpha_2 \leq \alpha_1 \leq \alpha)$.

8.5 Expériences numériques

Dans notre implementation, nous choisissons $T_0 = \frac{T_{min}+T_{max}}{2}$. Notre algorithme a été codé en C et les tests ont été réalisés sur un ordinateur Intel CPU 1.73Ghz, 2GB RAM. Pour résoudre un problème linéaire ou quadratique, CPLEX version 9.1 a été utilisé.

Les données de tests sont générées de façon suivante. Les points de vente sont générés aléatoirement dans une zone de $200 \times 200km$. Le dépôt central est placé au centre de la zone considérée. Nous utilisons la distance Euclidienne comme la distance entre deux points de vente. Pour tous points de vente, nous avons le même coût de manutention ($\phi = 10$ euro) et le même coût de stockage ($\eta = 0.1$ euro). La vitesse moyenne des véhicules est de 50 km/h. Le coût de transport est de 1 euro au kilomètre. Nous avons testé $DCA - IRP$ avec $n = 10, K = 100$; $n = 20, K = 200$ et $n = 30, K = 300$. Dans les Tableaux 8.1, 8.2 et 8.3, nous présentons les résultats obtenus. Les notations suivantes sont utilisées :
- $VarBin$: le nombre de variables binaires,
- $VarLin$: le nombre de variables continues,
- $Contr$: le nombre de contraintes,
- UB : la valeur de la fonction objectif obtenue par $DCA - IRP$,
- CPU : le temps CPU de l'algorithme,
- LB : la borne inférieure obtenue par le problème relaxé ($Relax2$),
- $Gap = \frac{UB-LB}{UB}100\%$.

Instance	VarBin	VarLin	Contr	UB	CPU	Iter	LB	Gap
1	121	132	164	88.866	0.453	24	76.939	13.42
2	121	132	164	90.893	23.063	966	78.891	13.20
3	121	132	164	89.889	0.094	8	77.279	14.03
4	121	132	164	88.221	6.094	444	76.438	13.36
5	121	132	164	87.420	0.234	18	77.153	11.74
6	121	132	164	92.846	0.531	39	84.700	08.77
7	121	132	164	91.686	1.219	94	78.604	14.27
8	121	132	164	88.972	0.078	6	80.769	09.22
9	121	132	164	84.799	1.859	103	71.111	16.14
10	121	132	164	88.821	0.156	9	74.441	16.19
Moyenne	121	132	164		3.378			13.03

TAB. 8.1 – Résultats avec $n = 10$ et $K = 100$.

Nous observons que :

- $DCA - IRP$ fournit toujours une solution entière bien qu'il travaille sur le domaine continu.
- Notre algorithme est relativement rapide. Avec $n = 10$ (121 variables binaires), le temps de calcul de notre algorithme est inférieure à une seconde pour 6/10 des instances. Avec $n = 30$ (961 variables binaires), le temps de calcul moyen est de 40 secondes.
- $DCA - IRP$ fournit une solution avec un Gap assez petit (entre 10% et 13%). Cela prouve la

Instance	VarBin	VarLin	Contr	UB	CPU	Iter	LB	Gap
1	441	462	564	113.506	2.156	35	99.539	12.31
2	441	462	564	121.532	2.797	44	108.393	10.81
3	441	462	564	116.688	2.032	33	104.176	10.72
4	441	462	564	124.376	4.547	55	110.054	11.52
5	441	462	564	119.869	2.843	34	105.058	12.36
6	441	462	564	109.476	4.703	49	97.913	10.56
7	441	462	564	124.611	4.187	28	113.702	08.75
8	441	462	564	121.762	5.406	88	110.508	09.24
9	441	462	564	117.138	5.062	79	102.551	12.45
10	441	462	564	110.562	2.141	31	99.677	09.85
Moyenne	441	462	564		3.587			10.86

TAB. 8.2 – Résultats avec $n = 20$ et $K = 200$.

Instance	VarBin	VarLin	Contr	UB	CPU	Iter	LB	Gap
1	961	992	1084	135.613	38.047	222	116.722	13.93
2	961	992	1084	135.038	44.672	240	119.570	11.45
3	961	992	1084	136.940	69.859	358	123.642	09.71
4	961	992	1084	137.389	23.985	127	117.343	14.59
5	961	992	1084	142.434	31.313	169	124.221	12.79
6	961	992	1084	143.933	28.578	133	126.003	12.46
7	961	992	1084	132.314	19.125	124	114.258	13.65
8	961	992	1084	139.348	60.560	420	120.501	13.52
9	961	992	1084	136.612	36.750	225	121.671	10.94
10	961	992	1084	139.485	31.734	179	120.269	13.78
Moyenne	961	992	1084		38.462			12.68

TAB. 8.3 – Résultats avec $n = 30$ et $K = 300$.

bonne qualité de solution donnée par notre algorithme.

8.6 Conclusion

Nous avons considéré le problème des tournées de véhicules dans une chaîne d'approvisionnement sous la forme d'une programmation non convexe en variables mixtes 0-1. De nos jours, trouver une méthode efficace pour le résoudre reste un défi. Nous avons proposé un schéma DCA pour résoudre ce problème. Les résultats numériques préliminaires montrent que DCA est prometteur. C'est intéressant de combiner DCA avec un algorithme global comme Branch-and-bound pour résoudre globalement ce problème difficile. Ce travail est en cours d'étude.

Conclusion

Nous avons développé dans cette thèse les deux approches locales et globales basées sur la programmation DC et DCA pour l'optimisation combinatoire en variables mixtes 0-1 et leurs applications à la résolution de nombreux problèmes en planification opérationnelle.

Nos contributions principales portent sur les points suivants :

- L'amélioration de l'algorithme d'approximation extérieure basée sur DCA (DCACUT) introduit dans [78] pour la programmation linéaire en variables mixtes 0-1.
- Les combinaisons des algorithmes globaux et DCA, et l'étude numérique comparative de ces approches DCACUT, DCAGMI (combinaison de DCACUT et la coupe mixte de Gomory), BBDCA (B&B combiné avec DCA) et BCDCA (Branch-and-Cut combiné avec DCA) pour la programmation linéaire en variables mixtes 0-1.
- L'utilisation de DCA pour la résolution de la programmation DC en variables mixtes 0-1 en utilisant la pénalité exacte.
- La mise en oeuvre des algorithmes développés pour la résolution des problèmes de grande taille en planification opérationnelle. Plus précisement :
  - DCA et les méthodes globales combinées avec DCA (DCACUT, BBDCA, BCDCA) pour le problème de maximisation de la durée de vie d'un réseau de télécommunication sans fil.
  - DCA et un schéma combiné B&B et DCA pour le problème de minimisation d'énergie d'un réseau de télécommunication sans fil.
  - DCA pour le problème d'ordonnancement d'avance-retard avec les variables indexées sur le temps. C'est un problème de très grande dimension auquel les algorithmes de type global ne sont pas applicables alors que DCA est très rapide, même pour les exemples où le nombre de variables binaires est de l'ordre de 200000.
  - DCA et les méthodes globales combinées avec DCA (DCACUT, BBDCA, BCDCA) pour le problème d'ordonnancement d'avance-retard avec une date d'échéance commune.
  - La modélisation DC et le développement d'un schéma DCA et d'un algorithme combiné BBDCA pour la résolution du problème de l'affectation de tâche des véhicules aériens non pilotés.
  - La modélisation DC et le développement d'un schéma de DCA pour la résolution du problème des tournées de véhicules dans une chaîne d'approvisionnement.

Plusieurs questions issues de nos travaux :

– Nos résultats numériques montrent que DCACUT est très efficace pour certains types de problèmes (mod008, mod010 de Benchmark) mais moins performante que les méthodes de type B&B pour

certaines autres. Le gain de DCACUT par rapport à la méthode de coupe de Gomory peut aller jusqu'à 170 fois au niveau du temps de calcul. Pour pouvoir exploiter l'efficacité de DCACUT il est important de savoir quels sont des structures spéciales de la programmation linéaire en variables mixtes 0-1 qui pourraient être résolues efficacement par DCACUT ?
Cette question mérite une étude théorique et algorithmique plus profonde avec un nombre important de tests numériques.

- Bien que la combinaison B&B et DCA ait prouvé sa meilleure efficacité à travers de nombreux résultats numériques, la question de trouver des bonnes bornes inférieures dans le schémas B&B est toujours pertinente. En effet, dans nos travaux, cette efficacité est due à la performance de DCA dans la recherche d'une bonne solution réalisable (ce qui donne une bonne borne supérieure). Cependant, faute du temps, nous n'avons pas pu étudier une borne inférieure autre que la relaxation linéaire classique pour la programmation linéaire en variables mixtes 0-1.
- Pour les deux problèmes de la programmation DC en variables mixtes 0-1 (l'affectation des tâches aux véhicules aériens non piloté et des tournées de véhicules dans une chaîne d'approvisionnement) il est intéressant de chercher d'autres décompositions DC pour répondre à la question cruciale lors d'utilisation de DCA : quelle est la meilleure décomposition DC de la fonction objectif ?
- Pour résoudre globalement le problème des tournées de véhicules dans une chaîne d'approvisionnement nous devrions étudier les procédures de calculs des bornes inférieures de la fonction objectif et ainsi développer des schémas combinés B&B et DCA.

Pour terminer, nous insistons encore une fois le rôle crucial de DCA dans nos travaux et son efficacité indiscutable à travers de résultats numériques. Les applications considérées sont importantes et difficiles à traiter. Et DCA est indispensable dans le développement de nos méthodes de résolution efficaces. Ce travail contribuerait, une fois de plus, à la popularisation de DCA dans le monde de recherche et de l'industrie. DCA et les algorithmes combinés avec DCA seront sans doute de plus en plus utilisés par des chercheurs et des praticiens.

Bibliographie

[1] Aghezzaf E.H., Raa B., Landeghem H.V., *Modelling inventory routing problems in supply chains of high consumption products,* European Journal of Operational Research Vol. **169** pp. 1048–1063, 2006.

[2] A. Auslender *Optimisation Méthodes Numériques* Paris : Masson, 1976.

[3] R. Bhatia, M. Kodialam, *On power efficient communication over multi-hop wireless networks : Joint routing, scheduling and power control,* in Proc. IEEE INFOCOM'04, pp. 1457-1466, Hong Kong, Mar. 2004.

[4] P. R. Chandler, M. Pachter, S. R. Rasmussen, and C. Schumacher, *Multiple Task Assignment for a UAV Team.* AIAA Guidance, Navigation, and Control Confer- ence, Monterey, CA, Aug. (2002)

[5] J.-H. Chang, and L. Tassiulas, *Energy conserving routing in wireless ad-hoc networks,* in Proc. IEEE INFOCOM-00, pp. 21-31, Tel-Aviv, Israel, Mar. 2000

[6] L. Chen, S. H. Low, M. Chiang, J. C. Doyle, *Cross-layer congestion control, routing and scheduling design in ad hoc wireless networks,* in Proc. IEEE INFOCOM'06, pp. 1-13, Barcelona, Spain, Apr. 2006.

[7] Chien T.W., Balakrishnan A., Wong R.T., *An integrated inventory allocation and vehicle routing problem*, Trasportation Science, Vol. **23**, pp.67–76, 1989

[8] C. W. Commander, and P. M. Pardalos, *A combinatorial algorithm for the TDMA message scheduling problem,* Computational Optimization and Applications, 2006.

[9] R. L. Cruz, A. V. Santhanam, *Optimal routing, link scheduling and power control in multihop wireless networks,*, Proc. IEEE INFOCOM'03, pp. 702-711, San Francisco, USA, Mar. 2003.

[10] J. B. Cruz, Jr., G. Chen, D. Li, and X. Wang, *Particle Swarm Optimization for Resource Allocation in UAV Cooperative Control*, AIAA Guidance, Navigation, and Control Conference, Providence, RI, Aug. 2004

[11] M.E. Dyer, L.A. Wolsey. *Formulating the single machine sequencing problem with release dates as a mixed integer problem*, Discrete Applied Mathematics, Vol. **26**, pp.255–270, 1990

[12] T. ElBatt, A. Ephremides, *Joint scheduling and power control for wireless ad hoc networks,* in Proc. IEEE INFOCOM-02, pp. 976-984, New York, USA, June 2002.

[13] Falk, J.E. and Soland, R.M., *An algorithm for separable nonconvex programming problems*, Management Science, Vol. **15**, pp. 550–569, 1969

[14] Federgruen A., Zipkin P., *A combined vehicle routing and inventory allocation problem*, Operations Research, Vol. **32**, pp. 1019–1037, 1984

[15] R. E. Gomory, *An Algorithm for the Mixed Integer Problem*, RM-2597, The Rand Corporation, 1960.

[16] R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnooy Kan, *Optimization and approximation in deterministic sequencing and scheduling : a survey.* Annals of Discrete Mathematics, Vol. 5 pp. 287–326, 1979

[17] Hoang Tuy, G*lobal Minimization of a Difference of Two Convex Functions. Mathematics Programming Study*, Vol. 30 pp. 150-182, 1987. Acta Mathematica Vietnamica, Vol. 8 , pp. 3-34, 1983

[18] Hoang Tuy *Global Optimization : Deterministic Approaches* 2nd revised edition, Springer-Verlag, Berlin, 1993.

[19] Hoang Tuy *DC Optimisation : Theory, Methods and Algorithms, Handbook of Global Optimisation* Horst and Pardalos eds, Klaver Academic Publishers, pp. 149–216, 1995.

[20] Hoang Tuy *Convex Analysis and Global Optimization*, Kluwer Academic Publishers, 1998.

[21] Hoang Tuy, T.V. Thieu and Ng.Q. Thai, *A conical algorithm for globally minimizing a concave function over a convex set*, Math. Operations Research, Vol. **10**, pp. 498–514, 1985.

[22] Horst, R., *Deterministic global optimization with partition sets whose feasiblity is not known. Application to concave minimization, reverse convex constraints, DC programming and Lipschitzian optimization,* Journal of Optimization Theory and Application, Vol. **58**, pp. 11–37, 1988.

[23] Horst, R., *A general class of branch-and-bound methods in global optimization with some new approaches for concave minimization*, Journal of Optimization Theory and Application, Vol. **51** (2), pp. 271–291, 1986.

[24] Horst, R., Nguyen Van T. and Benson, H., *Concave minimization via conical partitions and polyhedral outer approximation*, Mathematical Programming, Vol. **50**, pp. 259–274, 1991.

[25] Horst, R., T. Nguyen Van, De Vries J., *On finding new vertices and redundant constraints in cutting plane algorithms for global optimization*, Operations Research Letters, Vol. **7**(2), pp. 85–90, 1988.

[26] R. Horst, Nguyen. V. T, and Hoang Tuy, *On a outer approximation concept in global optimisation*, Optimization, Vol. 20 pp. 255-264. 1989

[27] Horst, R. and Hoang Tuy, *Global optimization (Deterministic Approaches)*, Third edition, Springer, Berlin 1996.

[28] Horst R. and Hoang Tuy, *On the convergence of global methods in multiextremal optimization*, Journal of Optimization Theory and Application, **54** (2), pp 253–271, 1987.

[29] Horst R., P.M. Pardalos, Nguyen V.T., *Introduction to global optimization*, Kluwer Academic Publishers, 1995.

[30] H. Jiang, W. Zhuang, X. Shen, *Cross-layer design for resource allocation in 3G wireless networks and beyond,* IEEE Communications Magazine, Vol. **43**, no. 12, pp. 120-126, 2005.

[31] Kalantari B. and Rosen, J.B., *Algorithm for global minimization of linearly constrained concave quadratic functions*, Mathematics of Operations Research, Vol. **12**, pp. 544–561, 1987.

[32] S.-J. Kim, X. Wang, and M. Madihian, *Distributed joint routing and medium access control for lifetime maximization of wireless sensor networks,* IEEE Trans. Wireless Commun., Vol. **6**, no. 7, pp. 2669-2677, Jul. 2007.

[33] P.J Laurent *Approximation et optimisation* Paris : Hermann, 1972.

[34] Le Dung Muu, Thai Quynh Phong and Pham Dinh Tao, *Decomposition methods for solving a class of nonconvex programming problems dealing with bilinear and quadratic functions*, Computational Optimization and Application, Vol. **4**, pp. 203–216, 1995.

[35] J. Le Ny, and E. Feron, *An Approximation Algorithm for the Curvature- Constrained Travelling Salesman Problem*, 43rd Annual Allerton Conference on Communications, Control and Computing, Monticello, IL, Sept. 2005

[36] 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.

[37] 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.

[38] H.A Le Thi *An efficient algorithm for globally minimizing a quadratic function under convex quadratic constraints* Mathematical Programming, Ser. A, Vol. **87**(3), pp. 401–426, 2000.

[39] 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**, pp. 375–397, 2003.

[40] 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.

[41] 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.

[42] 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, pp. 703–714, 2007.

[43] H.A. Le Thi, T.P. Nguyen, T. Pham Dinh *A Continuous DC Programming Approach to the Strategic Supply Chain Design Problem from Qualified Partner Set.* European Journal of Operational Research. Vol. **183**(3), pp. 1000-1012, 2007.

[44] Le Thi H. A., Ndiaye B. M., Pham Dinh T., *Solving a multimodal transport problem by DC*, Proceedings of RIVF'08, IEEE International conference on Research, Innovation and Vision for the future in Computing and Communications Technologies, Ho Chi Minh city, July 13-17, pp. 49-56 2008

[45] Le Thi H. A., Ndiaye B. M., Pham Dinh T., *Models and Simulations for Container Allocation on trains in a rapid transhipment shunting yard by DC Programming and DCA*, Third International Conference on Computational Management Science, special invited session, 2006

[46] Le Thi H. A., Ndiaye B. M., Pham Dinh T., *Programmation DC et DCA pour Le Dispatching des Véhicules Guidés Automatisés dans un Terminal à Conteneurs Automatisé*, Session spéciale invitée, Conférence conjointe FRANCORO V / ROADEF pp. 241-342, Grenoble 20-23 février 2007

[47] H. A. Le Thi, Nguyen.V. V., T. Pham Dinh, *A Combined DCA and New Cutting Plane Techniques for Globally Solving Mixed Linear Programming*, SIAM Conference on Optimization, Stockholm (Sweden) special invited session, 2005

[48] 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**, pp. 253–285, 1997.

[49] 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**, pp. 171–206, 1998.

[50] 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, pp. 301–339, 2000.

[51] 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, pp. 260-277, 2000.

[52] 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, pp. 231–276, 2001.

[53] 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, pp. 300–317, 2001.

[54] H.A. Le Thi, T. Pham Dinh *A Continuous Approach for Globally Solving Linearly Constrained Quadratic Zero - One Programming Problems*. Optimization, Vol. **50**, pp. 93–120, 2001.

[55] 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**, pp. 204–233, 2002.

[56] 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.

[57] 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, pp. 77–116, 2003.

[58] 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, pp. 276–296, 2003.

[59] H.A. Le Thi, 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**, pp. 23–46, 2005.

[60] 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.

[61] 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**, pp. 1073–1078, 2002.

[62] 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**, pp. 407–423, 2003.

[63] 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**, pp. 545–577, 2003.

[64] H.A Le Thi, T. Pham Dinh and N. Huynh Van *Exact penalty techniques in DC Programming.* Research Report INSA Rouen 2005.

[65] 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**, pp.117–128, 1996.

[66] 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, pp. 9–28, 1998.

[67] H.A Le Thi, T. Pham Dinh, M. Le Dung, *Exact penalty in DC Programming.* Vietnam Journal of Mathematics Vol. **27** (2), pp. 169–178, 1999.

[68] 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, pp. 503-531, 2003.

[69] 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**, pp ; 257–270, 2002.

[70] R. Madan, S. Cui, S. Lall, and A. Goldsmith, *Mixed integer-linear programming for link scheduling in interference-limited networks*, Proc. of 1st workshop on Resource Allocation in Wireless Networks, Italy, Apr. 2005.

[71] R. Madan, S. Cui, S. Lall, and A. Goldsmith, *Cross-layer design for lifetime maximization in interference-limited wireless sensor networks,* IEEE Trans. Wireless Commun., Vol. **5**, no. 11, pp. 3142-3152, Nov. 2006.

[72] H. Marchand, *A polyhedral study of the mixed knapsack set and its use to solve mixed integer programs*, PhD thesis, univérsité Catholique de Louvain, Belgium 1998.

[73] H. Marchand, A. Martin, R. Weismantel, L. Wolsey, *Cutting planes in integer and mixed integer programming*, Technical report of project TMR-DONET nr. ER FMRX-CT98-0202.

[74] R. M. Murray, *Recent Research in Cooperative Control of Multivehicle Systems.* Journal of Dynamic Systems, Measurement, and Control, Vol. 129, No. 5, pp. 571–583, 2007.

[75] B. M. Ndiaye, T. Pham Dinh, H. A. Le Thi, *DC programming and DCA for SSCRP*, in Modelling, Computation and Optimization in Information Systems and Management Sciences, Communications in Computer and Information Science CCIS Volume 14, Springer, pp. 21-30, 2008

[76] G.L. Nemhauser, L.A. Wolsey, *A recursive procedure to generate all cuts for 0-1 mixed integer programs*, Mathematical Programming, vol. 46, pp. 379-390, 1990.

[77] Nguyen Van T. and Hoang Tuy, *Convergent algorithms for minimizing a concave function*, Mathematic Operations Research, Vol. **5**, pp. 556–566, 1980.

[78] V.V. Nguyen, *Méthodes exactes pour l'optimisation DC polyédrale en variables mixtes 0-1 basées sur DCA et nouvelles coupes.* Thèse de Doctoral, INSA de Rouen, 2006.

[79] 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.

[80] 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.

[81] 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.

[82] T. Pham Dinh *Duality in DC (difference of convex functions) optimization. Subgradient methods* Trends in Mathematical Optimization, International Series of Numer Math., Vol. **84**, pp. 277–293, 1988.

[83] 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, pp. 379–384, 1994.

[84] 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**, pp. 263–276, 1995.

[85] 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**, pp. 207–216, 1996.

[86] 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, pp. 289–355, 1997.

[87] 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, pp.476–505, 1998.

[88] K. T. Phan, H. Jiang, C. Tellambura, S. A. Vorobyov, R. Fan, *Joint medium access control, routing and energy distribution in multi-hop wireless networks,* under revision, IEEE Trans. Wireless Commun.

[89] Phillips, A.T. and Rosen, J.B., *A parallel algorithm for constrained concave quadratic global minimization*, Mathematical Programming, Vol. **42**, pp. 412–448, 1988

[90] S. Ramanathan, and E.L. Lloyd, *"Scheduling algorithms for multi-hop radio network,"* IEEE ACM Trans. Networking, Vol. **1** (2), pp. 166- 177, 1993.

[91] A. Richards, J. Bellingham, M. Tillerson, and J. P. How, *Coordination and control of multiple UAVs.* AIAA Guidance, Navigation, and Control Conference, Monterey, CA, Aug. 2002

[92] A. Richards, and J. P. How, *Aircraft Trajectory PlanningWith Collision Avoidance Using Mixed Integer Linear Progrmming.* American Control Conference, Anchor- age, AK, May 2002

[93] R.T. Rockafellar, *Convex Analysis*, Princeton University Press, Princeton, 1970.

[94] Rosen, J.B. and Pardalos, P.M., *Global minimization of large scale constrained quadratic problem by separable programming*, Mathematical Programming, Vol. **34** (2), pp. 163-174, 1986.

[95] A. Salman, I. Ahmad, and S. Al-Madani, *Particle Swarm Optimization for Task Assignment Problem*, Microprocessors and Microsystems, Vol. 26, No. 8 pp. 363–371, 2002

[96] C. Schumacher, P. Chandler, M. Pachter, and L. Pachter, *Constrained Optimization for UAV Task Assignmnet.* AIAA Guidance, Navigation, and Control Confer- ence, Providence, RI, Aug. 2004

[97] Sourd F., *New exact algorithms for one machine earliness tardiness scheduling*, INFORMS Journal of Computing, Vol. 21, pp.167-175, 2009

[98] Sourd F., S. Kedad-Sidhoum. Set of instances for Earliness-tardiness scheduling with distinct due dates, http ://www-poleia.lip6.fr/≫sourd/project/et/

[99] Sourd F., S. Kedad-Sidhoum. *A faster branch-and-bound algorithm for the earliness tardiness scheduling problem*, Journal of Scheduling, Vol. 11, pp.49–58, 2008

[100] Sousa J.P., L.A. Wolsey. *A time-indexed formulation of non-preemptive single machine scheduling problems.* Mathematical Programming, Vol. **54**, pp.353–367, 1992

[101] J. Tang, G. Xue, C. Chandler, and W. Zhang, *Link scheduling with power control for throughput enhancement in multihop wireless networks,* IEEE Trans. Vehicular Tech., Vol. **55**, no. 3, pp. 733-742, May 2006.

[102] Q.P. Thai, *Analyse Numérique des Méthodes d'Optimisation Globale. Codes et Simulations numériques. Applications,* Thèse de doctorat, Université de Rouen, 1994.

[103] Q. P. Thai, Le Thi H. A. and Pham D. T., *On the global solution of linearly constrained indefinite quadratic minimization problems by decomposition branch and bound method.* RAIRO, Recherche Opérationnelle, Vol. **30**(1), pp. 31–49, 1996.

[104] V. T'kindt, J.C.Billaut, *Multicriteria Scheduling :Theory, models and Algorithms*, second ed., springer Verlag, 2006.

[105] J.F Toland *Duality in nonconvex optimization* Journal of Mathematical Analysis and Applications, Vol.66, pp. 399-415, 1978.

[106] 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, pp. 260–277, 1985.

[107] J.B.H Urruty *Conditions nécessaires et suffisantes d'optimalité globale en optimisation de différences de deux fonctions convexes* CRAS, Vol. 309, Série I, pp. 459–462, 1989.

[108] Van den Akker, M., H. Hoogeveen, Velde S. Van de. *Combining column generation and lagrangean relaxation to solve single machine common due date problem*, INFORMS Journal on Computing, Vol. 14, pp.35-51, 2002

[109] A. F . Veinott, T*he supporting hyperplane method for unimodal programming*, Operations Research, Vol. 15, pp. 147-152, 1967.

[110] Yau, H., Pan Y., Shi L., *New solution approaches to the general single machine earliness tardiness problem*, Automation Science and Engineering, IEEE Transactions, Vol. 5(2, pp.349-360), 2008

[111] Y. Zhong, E.H. Aghezzaf, *Analysis and solution developement of the single vehicle inventory routing problem*, in Proc. MCO08, pp. 369-38, Metz, France, September 2008.

[112] J. Zhu, S. Chen, B. Bensaou, and K.-L. Hung, *Tradeoff between lifetime and rate allocation in wireless sensor networks : A cross-layer approach*, in Proc. IEEE INFOCOM-07, pp. 267-275, Alaska, USA, May 2007.

[113] Benchmark instance for mixte integer problem. http ://miplib.zib.de/miplib3/miplib3.html

[114] OR Library. http ://people.brunel.ac.uk/ mastjjb/jeb/orlib/schinfo.html

Résumé :
Cette thèse développe les deux approches locales et globales basées sur la programmation DC et DCA pour l'optimisation combinatoire en variables mixtes 0-1 et leurs applications à la résolution de nombreux problèmes en planification opérationnelle.
Plus particulièrement, cette thèse adresse à:
- L'amélioration de l'algorithme d'approximation extérieure basée sur DCA (appelé DCACUT) introduit par Nguyen V.V. et Le Thi pour la programmation linéaire en variables mixtes 0-1,
- Les combinaisons des algorithmes globaux et DCA et l'étude numérique comparative de ces approches pour la programmation linéaire en variables mixtes 0-1,
- L'utilisation de DCA à la résolution de la programmation DC en variables mixtes 0-1 en utilisant la pénalité exacte,
- La mise en oeuvre des algorithmes développés à la résolution des problèmes de grande taille en planification opérationnelle comme les problèmes dans le réseau de télécommunication sans fils, les problèmes d'ordonnancement ainsi que le problème d'affectation de tâches des véhicules aériens non pilotés ou bien le problème des tournées de véhicules dans une chaîne d'approvisionnement.

Mots-clés : Programmation DC, DCA, Programmation en variables mixtes 0-1, Branch-and-Bound, Méthode d'approximation extérieure.

Abstract :
This thesis develops two local and global approaches based on DC programming and DCA for mixed 0-1 combinatorial optimization and their applications to many problems in operational planning.
More particularly, this thesis consists of:
- The improvement of the outer approximation algorithm based on DCA (called DCACUT) introduced by Nguyen V.V and Le Thi for mixed 0-1 linear programming,
- The combinations of global algorithms and DCA and the comparative numerical study of these approaches for mixed 0-1 linear programming,
- The use of DCA for solving mixed 0-1 programming via an exact penalty technique,
- The implementation of the algorithms developed for solving large scale problems in operational planning: two problems in wireless telecommunication network, two scheduling problems, an UAV task assignment problem and an inventory routing problem in supply chains.

Keywords: DC programming, DCA, Mixed 0-1 programming, Branch-and-Bound, Outer approximation method.