

Programmation DC et DCA pour l'optimisation non convexe/optimisation globale en variables mixtes entières : Codes et Applications

Viet Nga Pham

▶ **To cite this version:**

Viet Nga Pham. Programmation DC et DCA pour l'optimisation non convexe/optimisation globale en variables mixtes entières : Codes et Applications. Mathématiques générales [math.GM]. INSA de Rouen, 2013. Français. NNT : 2013ISAM0005 . tel-00833570

HAL Id: tel-00833570

https://theses.hal.science/tel-00833570

Submitted on 13 Jun 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE DE DOCTORAT

présentée par

PHAM Viet Nga

en vue de l'obtention du grade de

DOCTEUR DE L'INSTITUT NATIONAL
DES SCIENCES APPLIQUÉES DE ROUEN

(arrêté ministériel du 7 août 2006)

Spécialité: **Mathématiques Appliquées**

Programmation DC et DCA pour l'optimisation non convexe/optimisation globale en variables mixtes entières. Codes et Applications.

Soutenue le 18 avril 2013 devant le jury composé de

M.	Arnaud LALLOUET	*Professeur* *Université de Caen*	Président
M.	Pierre MARECHAL	*Professeur* *Université de Paul Sabatier, Toulouse 3*	Rapporteur
M.	Christian PRINS	*Professeur* *Université de Technologie de Troyes*	Rapporteur
M.	Pascal DAMEL	*Maître de Conférences, HDR* *Université de Lorraine*	Examinateur
M.	Viet Hung NGUYEN	*Maître de Conférences* *Université Pierre et Marie Curie, Paris 6*	Examinateur
M^{me}	Hoai An LE THI	*Professeur* *Université de Lorraine*	Directeur de thèse
M.	Tao PHAM DINH	*Professeur* *INSA de Rouen*	Directeur de thèse

Thèse préparée au sein du Laboratoire de Mathématiques
de l'Institut National des Sciences Appliquées de Rouen (LMI), France

Remerciements

Tout d'abord, je voudrais exprimer ma profonde reconnaissance à Madame Le Thi Hoai An, Professeur à l'Université de Lorraine, et à Monsieur Pham Dinh Tao, Professeur à l'Institut National des Sciences Appliquées (INSA) de Rouen, mes directeurs de thèse. Je les remercie de m'avoir donnée des conseils, de m'avoir encouragée et soutenue tout au long de ce travail avec patience et disponibilité.
Je salue aussi leur gentillesse, leur droiture et leur rigueur.

Je tiens ensuite à remercier vivement Monsieur Pierre Maréchal, Professeur à l'Université de Paul Sabatier, Toulouse 3, et Monsieur Christian Prins, Professeur à l'Université de Technologie de Troyes, qui ont accepté d'être rapporteurs de cette thèse. Je les remercie pour l'honneur qu'ils m'ont fait en participant au jury.

Mes plus sincères remerciements vont également à Monsieur Pascal Damel, Maître de Conférences à l'Université de Lorraine, Monsieur Arnaud Lallouet, Professeur à l'Université de Caen, et Monsieur Nguyen Viet Hung, Maître de Conférences à l'Université Pierre et Marie Curie, Paris 6, qui m'ont fait le plaisir d'accepter d'être membres du jury.

Cette thèse a été réalisée au sein du Laboratoire de Mathématiques de l'INSA de Rouen, France, où j'ai rencontré des personnes sympathiques et très gentilles. Merci à Brigitte Diarra, Christian Goût, Omar Jaumat, Carole Le Guyader, Bruno Portier, Anastasia Zakharova,... pour les encouragements et les soutiens chaleureux. Je pense aussi à mes amis doctorants au LMI : Benoit, Florentina, Imed, Lamia, Mamadou, Duc Manh, Anh Son, Solène, Yi-Shuai,.... Merci pour les bons moments partagés avec vous.

J'aimerais remercier très sincèrement Monsieur Bernard Malécot et son épouse du généreux accueil qu'ils m'ont réservée sous leur toit dès mes premiers jours en France.

Je remercie tous les amis que j'ai rencontrés à Rouen : Viet Dung, Thu Hanh, Kim Hang, Truc Ngan, Thanh Phuong, Khanh Son, Huu Thai, Huyen Trang, Van Trien,.... Merci pour les moments agréables que vous avez partagé avec moi.

Mes remerciements s'adressent également au Gouvernement Vietnamien qui a financé mes études pendant 3 ans. Je n'oublie pas de remercier toute l'équipe du personnel de l'Université d'Agriculture de Hanoi pour son soutien. Je remercie particulièrement Monsieur Le Duc Vinh pour sa compétence, sa gentillesse et ses encouragements à toutes épreuves.

Un grand merci à mon mari, Thanh Hai. Merci de ton amour, de ta confiance, de ton soutien comme toujours et de tout ce que tu m'apportes. Un grand merci va aussi à mon fils, Gia Hung, qui a supporté avec courage d'être loin de moi pendant de longues années.

Et enfin, je n'oublie jamais le soutien inconditionnel, l'encouragement et l'aide de tous les membres de ma famille tout au long de ce parcours : mes parents, mes beaux parents, ma sœur, mes belles sœurs. Il est impossible pour moi de trouver les mots pour leur exprimer mes reconnaissances. Ce fruit du travail leur est dédié.

Résumé

Basés sur les outils théoriques et algorithmiques de la programmation DC et DCA, les travaux de recherche dans cette thèse portent sur les approches locales et globales pour l'optimisation non convexe et l'optimisation globale en variables mixtes entières. La thèse comporte 5 chapitres. Le premier chapitre présente les fondements de la programmation DC et DCA, et des méthodes de Séparation et Evaluation (B&B) (utilisant la technique de relaxation DC pour le calcul des bornes inférieures de la valeur optimale) pour l'optimisation globale. Y figure aussi des résultats concernant la pénalisation exacte pour la programmation en variables mixtes entières. Le deuxième chapitre est consacré au développement d'une méthode DCA pour la résolution d'une classe NP-difficile des programmes non convexes non linéaires en variables mixtes entières. Ces problèmes d'optimisation non convexe sont tout d'abord reformulés comme des programmes DC via les techniques de pénalisation en programmation DC de manière que les programmes DC résultants soient efficacement résolus par DCA et B&B bien adaptés. Comme première application en optimisation financière, nous avons modélisé le problème de gestion de portefeuille sous le coût de transaction concave et appliqué DCA et B&B à sa résolution. Dans le chapitre suivant nous étudions la modélisation du problème de minimisation du coût de transaction non convexe discontinu en gestion de portefeuille sous deux formes : la première est un programme DC obtenu en approximant la fonction objectif du problème original par une fonction DC polyédrale et la deuxième est un programme DC mixte 0-1 équivalent. Et nous présentons DCA, B&B, et l'algorithme combiné DCA-B&B pour leur résolution. Le chapitre 4 étudie la résolution exacte du problème multi-objectif en variables mixtes binaires et présente deux applications concrètes de la méthode proposée. Nous nous intéressons dans le dernier chapitre à ces deux problématiques challenging : le problème de moindres carrés linéaires en variables entières bornées et celui de factorisation en matrices non négatives (Nonnegative Matrix Factorization (NMF)). La méthode NMF est particulièrement importante de par ses nombreuses et diverses applications tandis que les applications importantes du premier se trouvent en télécommunication. Les simulations numériques montrent la robustesse, rapidité (donc scalabilité), performance et la globalité de DCA par rapport aux méthodes existantes.

Mot clés : Programmation DC et DCA, Séparation et Evaluation, Technique de pénalisation exacte, Programmation en variables mixtes entières, Programmation multi-objectif, Gestion de portefeuille, Moindres carrés linéaires en variables entières, Factorisation en matrices non négatives (NMF).

Publications et Conférences

PHAM Viet Nga

Article avec comité de lecture

- Viet-Nga PHAM, Hoai An LE THI, Tao PHAM DINH, *Solving Nurse Rostering Problems by a Multiobjective Programming Approach,* In N.-T. Nguyen, K. Hoang, and P. Jędrzejowicz editors, Proceedings of ICCCI 2012, Computational Collective Intelligence. Technologies and Applications, Lecture Notes in Computer Science, Part I, LNAI 7653, pp. 544–552, Springer-Verlag Berlin Heidelberg 2012.

- Viet-Nga PHAM, Hoai An LE THI, Tao PHAM DINH, *A DC programming framework for portfolio selection by minimizing the transaction costs,* In N.T. Nguyen, T. Van Do, and H.A. Le Thi editors, Proceedings of ICCSAMA 2013, Advanced Computational Methods for Knowledge Engineering, Studies in Computational Intelligence, SCI 479, pp. 31–40, Springer International Publishing Switzerland 2013.

- Viet-Nga PHAM, Hoai An LE THI, Tao PHAM DINH, *Solving a facility location problem by a multiobjective programming approach,* will be appeared in Proceedings of MIM 2013, IFAC Conference on Manufacturing Modelling, Management and Control, June 19-21, 2013, Saint Petersburg, Russia.

- Hoai An LE THI, Viet-Nga PHAM, Tao PHAM DINH, Yi-Shuai NIU, *DC Programming Approach for a class of nonconvex nonlinear mixed-integer programming problems,* submitted.

Communications aux colloques nationaux/internationaux

- Viet-Nga PHAM, Hoai An LE THI, Tao PHAM DINH, Yi-Shuai NIU, *DC Programming Approaches for Discrete Portfolio Optimization under concave transaction costs,* ICOTA 8 - The 8th International Conference on Optimization : Techniques and Applications, 10-13 December 2010, Shanghai, China.

- Viet-Nga PHAM, Hoai An LE THI, Tao PHAM DINH, *Approche basée sur la Programmation DC et DCA pour le problème de moindre carrées linéaires en variables entières bornées,* ROADEF 2011- Recherche Opérationelle et l'Aide à la Décision, 02-04 Mars 2011, Saint-Etienne, France.

- Viet-Nga PHAM, Hoai An LE THI, Tao PHAM DINH, *A new efficient approach for solving mixed 0-1 linear multiobjective programming. Application to shift scheduling problems,* EURO 2012- 25th European Conference on Operational Research, 8-11 July 2012, Vilnius, Lithuania.

Table des matières

1 Méthodologie **5**
- 1.1 Introduction à la programmation DC et DCA . . . 5
  - 1.1.1 Éléments d'analyse convexe . . . 7
  - 1.1.2 Programmation DC . . . 9
  - 1.1.3 DCA (DC Algorithms) . . . 11
- 1.2 Méthode par Séparation et Evaluation (SE) . . . 15
  - 1.2.1 Méthode de résolution et convergence . . . 15
  - 1.2.2 Séparation et Evaluation avec des ensembles non réalisables . . . 18
- 1.3 Techniques de pénalisation . . . 25
  - 1.3.1 Pénalisation extérieure . . . 26
  - 1.3.2 Pénalisation intérieure . . . 27
  - 1.3.3 Pénalisation exacte en programmation DC . . . 27
  - 1.3.4 Technique de pénalisation en programmation avec des variables mixtes entières . . . 31
- 1.4 Conclusion . . . 33

2 Programmation en variables mixtes entières **35**
- 2.1 Introduction . . . 35
- 2.2 DC reformulation of (NIP) and DCA for solving the penalized problem . . . 38
- 2.3 A Branch-and-Bound algorithm using DC relaxation techniques . . . 40
  - 2.3.1 DC relaxation technique . . . 40
  - 2.3.2 Branching procedure . . . 43
  - 2.3.3 Bounding procedure . . . 44
  - 2.3.4 A B&B algorithm for solving (NIP) . . . 44
- 2.4 Application to Discrete Portfolio Optimization under Concave Transaction Costs . . . 45
  - 2.4.1 Related works . . . 45
  - 2.4.2 Mean-variance model under concave transaction costs . . . 46
  - 2.4.3 Computational experiments . . . 47
- 2.5 Conclusion . . . 48

3 Gestion de portefeuille : Minimisation du coût de transaction non convexe **53**
- 3.1 Introduction . . . 53
- 3.2 Problem description and mathematical formulation . . . 55
- 3.3 DC programming and DCA for solving (3.2) . . . 56
  - 3.3.1 DC approximation problem . . . 56
  - 3.3.2 DCA for solving (P_{dc}) . . . 57
  - 3.3.3 A hybrid algorithm Branch-and-Bound-DCA for solving (P) . . . 58

3.4 Solving (P) by a zero-one approach 62
3.4.1 A mixed zero-one formulation 63
3.4.2 DC programming and DCA for solving ($Q01$) 63
3.4.3 A combined DCA-Branch and Bound algorithm for solving ($Q01$) . . 65
3.5 Computational results 68
3.6 Conclusion 70

4 Programmation linéaire multi-objectif en variables mixtes zéro-un 71
4.1 Introduction 71
4.2 Methodological approach 74
4.3 Applications 75
4.3.1 Nurse rostering problems 75
4.3.2 Facility location 84
4.4 Conclusion 89

5 Programmation DC et DCA pour la résolution du problème de moindres carrés linéaires en variables entières bornées/ factorisation en matrices non négatives 91
5.1 Problème de moindres carrés linéaires en variables entières bornées 91
5.1.1 Introduction 91
5.1.2 Programmation DC et DCA pour la résolution de (BILS) 92
5.2 Problème de factorisation en matrices non négatives (NMF) 94
5.2.1 Introduction 94
5.2.2 Décomposition DC de la fonction de coût 96
5.2.3 Programmation DC et DCA pour la résolution de NMF 97
5.3 Conclusion 98

Bibliographie 101

Introduction générale

Cette thèse représente une contribution de la Programmation DC et DCA pour l'optimisation non convexe ainsi que pour l'optimisation globale en variables mixtes entières.

La programmation DC (Difference of Convex functions) et DCA, qui constituent l'épine dorsale de la programmation non convexe, sont introduits en 1985 par Pham Dinh Tao et intensivement développés par Le Thi Hoai An et Pham Dinh Tao depuis 1994 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. Leur popularité réside dans leur versatilité, flexibilité, rapidité, robustesse et performance comparées à des méthodes existantes, leur adaptation aux structures des problèmes traités et leur capacité de résoudre des problèmes industriels de grande dimension.

Un programme DC est de la forme

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

où $g, h : \mathbb{R}^n \longrightarrow \mathbb{R} \cup \{+\infty\}$ sont convexes semi-continues inférieurement et propres. La dualité DC associe au programme (P_{dc}), dit programme DC primal, son dual, qui est aussi un programme DC

$$\alpha = \inf\{h^*(y) - g^*(y) \mid y \in \mathbb{R}^n\} \qquad (D_{dc})$$

Basé sur les conditions d'optimalité locale et la dualité DC, DCA consiste en la construction de deux suites $\{x^k\}$ et $\{y^k\}$, candidats à être solutions optimales de (P_{dc}) et (D_{dc}) respectivement, de telle manière que les suites $\{g(x^k)-h(x^k)\}$ and $\{h^*(y^k)-g^*(y^k)\}$ soient décroissantes et $\{x^k\}$ (resp. $\{y^k\}$) converge vers une solution primal réalisable $\widetilde{x}$ (resp. une solution dual réalisable $\widetilde{y}$) vérifiant les conditions d'optimalité locale et

$$\widetilde{x} \in \partial g^*(\widetilde{y}); \quad \widetilde{y} \in \partial h(\widetilde{x}).$$

Les problèmes initiaux étudiés dans cette thèse ne sont pas de la forme d'un programme DC, car soit ils contiennent des variables mixtes entières, soit la fonction objectif est discontinue. Grâce à la technique de pénalisation en proprammation DC, nous pouvons les reformuler en programmes DC et puis appliquons DCA pour la résolution. Afin d'évaluer la qualité des solutions fournies par DCA, la combinaison de DCA et une technique de globalisation de Séparation et Evaluation (en anglais Branch-and-Bound (B&B)), utilisant la relaxation DC pour le calcul des bornes inférieures de la valeur optimale, a été proposée. Il est clair que la technique de pénalisation joue un rôle crucial dans la reformulation d'un programme non convexe en variables mixtes entières (MINLP). Dans cette thèse, nous nous sommes intéressés aux problématiques suivantes comme des applications concrètes de nos travaux de recherche :

1. Minimisation d'une classe NP-difficile de programmes DC en variables mixtes entières.

Les MINLPs, particulièrement difficiles mais importants en applications, constituent toujours un challenge d'actualité. Ils appartiennent à la fois à l'optimisation continue et l'optimisation combinatoire. Parmi les méthodes existantes, on peut citer méthode de décomposition de Benders généralisée, méthode par Séparation et Evaluation, Approximation Extérieure, méthode de coupe, et des méthodes hybrides, etc. Ces méthodes exactes déterministes avec garantie théorique de globalité sont trop coûteuses en temps de calcul pour être scalables (i.e., capables de traiter les très grandes dimensions). Les méthodes heuristiques bien adaptées aux structures spécifiques des problèmes traités pourraient être utiles dans la mesure où elles seraient susceptibles de fournir de bonnes solutions avec un CPU très faible. Le développement des méthodes (déterministes) locales scalables est le principal objectif de nos travaux de recherche.

2. Minimisation du coût de transaction non convexe discontinu du problème de gestion de portefeuille sous des contraintes linéaires et une contrainte quadratique convexe.

   Le problème de gestion de portefeuille avec coût de transaction a reçu l'attention de nombreux chercheurs. Néanmoins, à cause de la non-convexité de la fonction coût de transaction dans le cas général, ces problèmes sont difficiles à traiter. La plupart des méthodes développées pour ces problèmes sont soit heuristiques, soit basées sur la méthode par Séparation et Evaluation (B&B) très coûteuse en CPU. Nos approches proposées dans ce travail sont prometteuses et moins chères par rapport à SE.

3. Minimisation du multi-objectif linéaires en variables mixtes binaires.

   Beaucoup de problèmes d'optimisation rencontrés en réalité sont de nature multi-objectif car plusieurs critères d'évaluation souvent contradictoires sont à considérer simultanément. De nombreux de problèmes réels peuvent être exprimés sous la forme de multi-objectif linéaires en variables mixtes 0-1 comme en planification, affectation, localisation, network flow, etc. Les algorithmes utilisés pour résoudre ces programmes non convexes sont composés de méthodes exactes (soit à l'aide d'une méthode de scalarisation, soit basées sur la méthode B&B, soit par évaluation directe du vecteur objectif) et de méthodes heuristiques/métaheuristiques. La méthode proposée dans cette thèse utilise une autre technique de scalarisation permettant de formuler un programme mathématique équivalent. De plus, cette technique introduit une fonction de pénalisation qui est utilisée pour transformer un problème de minimisation d'une fonction DC sur l'ensemble des solutions du problème linéaire multi-objectif en variables mixtes 0-1.

4. Reformulation du problème de moindres carrés linéaires en variables entières bornées en programme DC et résolution par DCA.

5. Problème de factorisation en matrices non négatives : Approches basées sur la programmation DC et DCA.

Contributions de la thèse

Nos contributions dans ce travail de thèse portent sur :

- Développement de la technique de pénalisation exacte et inexacte en programmation en variables mixtes entières.
- Reformulation du MINLP sous la forme d'un programme DC et sa résolution par DCA.

- Approximation d'une fonction discontinue par une fonction DC. Reformulation d'un problème non convexe dont l'objectif est discontinu en un programme DC à l'aide des variables binaires et de la technique de pénalisation. Combinaison de DCA et B&B pour la résolution du problème original.
- Nouvelle approche pour la résolution exacte des problèmes linéaires multi-objectif en variables mixtes 0-1.
- Méthodes de résolution pour le problème de moindres carrés linéaires en variables entières bornées et celui de factorisation en matrices non négatives (Nonnegative Matrix Factorization (NMF)).

Organisation de la thèse

La thèse est composée de cinq chapitres.

1. Dans le premier chapitre, des outils théoriques et algorithmiques indispensables à nos travaux de recherche sont présentés. Il s'agit de la Programmation DC et DCA, et la méthode par Séparation et Evaluation (SE) (Branch-and-Bound (B&B) en anglais) ainsi que des résultats plus récents sur la technique de pénalisation exacte en programmation DC, en particulier ceux relatifs à la programmation non convexe en variables mixtes entières.
2. Le chapitre 2 est consacré à l'étude de la programmation DC et DCA, et des techniques B&B pour la résolution d'une classe des programmes non convexes non linéaires en variables mixtes entières dans lesquels la fonction objectif est DC avec la deuxième composante séparable. Applications au problème de gestion de portefeuille sous le coût de transaction concave.
3. Dans le chapitre 3, nous établissons d'abord la modélisation du problème de minimisation du coût de transaction non convexe discontinu en gestion de portefeuille sous deux formes : la première est un programme DC obtenu en approximant la fonction objectif du problème original par une fonction DC polyédrale et la deuxième est un programme DC mixte 0-1 équivalent. Et nous proposons les versions de DCA, B&B, leur combiné DCA-B&B adaptées aux structures spécifiques de ces deux problèmes pour leur résolution.
4. Le problème multi-objectifs linéaires en variables mixtes binaires est réformulé et résolu dans le chapitre 4.
5. Last but not least, le problème de moindres carrés linéaires en variables entières bornées et celui de factorisation en matrices non négatives (Nonnegative Matrix Factorization (NMF)) font l'objet du dernier chapitre dans lequel nous développons la programmation DC et DCA pour leur résolution.

Nos algorithmes développés ont été implémentés à l'aide de MATLAB, C, AMPL, et le logiciel utilisé pour résoudre les sous-programmes convexes est CPLEX.

Chapitre 1

Méthodologie

Nous présentons dans ce chapitre les fondements de la programmation DC et DCA (DC Algorithms), les techniques de Séparation et Evaluation, ainsi que des résultats concernant la pénalisation exacte pour la programmation en variables mixtes entières.

1.1 Introduction à la programmation DC et DCA

L'optimisation offre un cadre de modélisation et d'algorithmique très riche pour tous les domaines de sciences appliquées. On peut distinguer deux branches de l'optimisation déterministe : la programmation convexe et la programmation non convexe.

Théoriquement on peut résoudre tout programme convexe, mais encore faut-il bien étudier la formulation du programme convexe en question - la reformulation constitue d'ailleurs un thème de recherche d'actualité - et qu'il soit bien adapté aux structures spécifiques des problèmes traités, pour proposer des variantes performantes peu coûteuses et donc capables d'atteindre des dimensions réelles très importantes.

Lorsque la convexité de l'objectif et des contraintes n'est pas vérifiée, on est en face d'un problème d'optimisation non convexe. L'absence de cette double convexité rend la résolution d'un programme non convexe difficile voire impossible en l'état actuel des choses. Contrairement à la programmation convexe, les solutions optimales locales et globales sont à distinguer dans un programme non convexe. L'analyse et l'optimisation convexes modernes se voient ainsi contraintes à une extension logique et naturelle à la non convexité et la non différentiabilité. Les méthodes numériques conventionnelles de l'optimisation convexe ne fournissent que des solutions locales bien souvent éloignés des solutions globales si les structures spécifiques des programmes non convexes ne sont pas exploitées dans leur construction.

Durant ces deux dernières décennies, la recherche en optimisation non convexe a largement bénéficié des efforts des chercheurs et s'est enrichie de nouvelles avancées considérables. Il y a deux approches différentes mais complémentaires en programmation non convexe :

1. Approches globales continues dont les nouveaux outils algorithmiques sont inspirés par 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 depuis plus d'un quart de siècle, à travers les travaux de Hoang Tuy (reconnu comme le pionnier), R. Horst, P. Pardalos, Pham Dinh Tao, Le Thi Hoai An, N. V. Thoai... L'inconvénient majeur des méthodes globales est leur coût exhorbitant en temps de calcul qui les empêche de traiter, en pratique, des problèmes d'optimisation non convexe réels de très grande taille. Le calcul d'une solution globale d'un programme non convexe reste la quête du Saint Graal pour les optimiseurs. Et tout le monde convient qu'il faille développer les approches locales performantes et économiques capables de résoudre ces problèmes à très grande dimension. On rejoint ainsi la communauté de l'Optimisation Non Linéaire avec des outils théoriques et algorithmiques spécifiques : la Programmation DC et DCA. Dans ce contexte, les problèmes d'optimisation combinatoire sont reformulés grâce à des techniques de pénalité exacte en Programmation DC et traités par DCA.

2. Approches locales et globales d'analyse convexe qui sont basées sur l'analyse et l'optimisation convexe. Ici la programmation DC (Difference of Convex functions) et DCA (DC Algorithms) jouent le rôle central car la plupart des problèmes d'optimisation non convexe sont formulés/reformulés sous la forme des programmes DC. La programmation DC et DCA constituent l'épine dorsale de la programmation non convexe et de l'optimisation globale. Ils sont introduits par Pham Dinh Tao en 1985 à l'état préliminaire ([121–123]) et développés intensivement à travers de nombreux travaux communs de Le Thi Hoai An et Pham Dinh Tao depuis 1994 ([85–92,124–127]) 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. Ces outils théoriques et algorithmiques constituent une extension de l'analyse et l'optimisation convexes, assez large pour couvrir la quasi-totalité des problèmes d'optimisation non convexe de la vie courante mais pas trop pour pouvoir exploiter l'arsenal puissant de ces derniers.

En particulier, la programmation DC et DCA ont été utilisés, avec beaucoup de succès, par de nombreux chercheurs et praticiens de différents domaines en sciences appliquées des différents Laboratoires dans le monde (Princeton, Stanford, MIT, Berkeley, Carnegie Mellon, Cornell, Imperial College, Institut fur Allgemeine Mechanik (IAM, RWTH-Aachen), California, Mannheim, Heidelberg, Wisconsin, Iowa, Minnesota, Florida, North Carolina at Chapel Hill, Tokyo Institute of Technology, Fribourg, Hanoi Institute of Mathematics, Coimbra, Vienna, Copenhague, Louvain, Pukyong, Namur, Microsoft, Google, Yahoo, Nasa, ...) pour modéliser et résoudre leurs programmes non convexes issus de différents domaines, en particulier : Transport-Logistique, Télécommunication, Bioinformatique, Finance, Data Mining and Machine Learning, Cryptologie, Mécanique, Traitement d'Image, Robotique & Vision par Ordinateur, Pétrochimie, Contrôle Optimal, Problèmes Inverses, Problèmes Mal-Posés, Programmation Multiobjectif, Multilevel Programming, Variational Inequalty Problems (VIP), Mathematical Programs with Equilibrium Constraints (MPEC), etc.

La popularité de la programmation DC et DCA réside dans leur versatilité, flexibilité, robustesse, rapidité et performance comparées à des méthodes existantes, leur adaptation aux structures des problèmes traités et leur capacité à résoudre des problèmes industriels de grande dimension. Pour une étude plus détaillée de DCA nous renvoyons le lecteur à la page web `http://lita.sciences.univ-metz.fr/~lethi/` [84].

1.1.1 Éléments d'analyse convexe

Notations et propriétés

Ce paragraphe est consacré à un rapide rappel d'analyse convexe. Pour plus de détails en analyse convexe, on pourra se reporter, par exemple aux ouvrages de R.T. Rockafellar [136], de J.B. Hiriart-Urruty et al. [58]. Dans la suite, $\mathbb{R}^n$ est l'espace euclidien muni du produit scalaire usuel noté $\langle .,. \rangle$ et de la norme euclidienne associée $\|x\|_2 = \sqrt{\langle x, x \rangle}$. On note $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$ muni d'une structure algébrique déduite de celle de $\mathbb{R}$ avec la convention $+\infty - (+\infty) = +\infty$ ([136]).

Définition 1.1 (Fonction indicatrice d'un ensemble) *Soit C un sous-ensemble de $\mathbb{R}^n$. La fonction indicatrice de C, notée χ_C, est définie ($\forall x \in \mathbb{R}^n$) par*

$$\chi_C(x) := \begin{cases} 0 & si\ x \in C \\ +\infty & si\ x \notin C \end{cases}$$

Définition 1.2 (Ensemble convexe) *Un sous-ensemble C de $\mathbb{R}^n$ est dit convexe si*

$$(1-\lambda)x^1 + \lambda x^2 \in C, \quad \forall x^1, x^2 \in C \text{ et } \forall \lambda \in [0,1]$$

Définition 1.3 (Polyèdre convexe) *Un sous-ensemble C de $\mathbb{R}^n$ est dit polyèdre convexe s'il est l'intersection d'un nombre fini de demi-espaces de $\mathbb{R}^n$,*

$$C = \bigcap_{i=1}^{m} \left\{ x \in \mathbb{R}^n \mid \langle a^i, x \rangle - b^i \leq 0, a^i \in \mathbb{R}^n, b^i \in \mathbb{R} \right\}.$$

Définition 1.4 (Fonction convexe) *Une fonction f définie sur un sous ensemble C de $\mathbb{R}^n$ est dite convexe si C est convexe et*

$$f((1-\lambda)x^1 + \lambda x^2) \leq (1-\lambda)f(x^1) + \lambda f(x^2), \quad \forall x^1, x^2 \in S, \forall \lambda \in [0,1]. \tag{1.1}$$

f est dite strictement convexe si l'inégalité (1.1) est stricte dès que $x^1 \neq x^2$ et $\lambda \in]0,1[$.

L'ensemble des fonctions convexes sur un ensemble convexe C est un cône convexe que l'on note $\mathrm{Conv}(C)$.

Le domaine effectif d'une fonction convexe f définie sur C, noté $\mathrm{dom} f$, est défini par

$$\mathrm{dom} f = \{x \in C \mid f(x) < +\infty\}.$$

La fonction convexe f est dite *propre* si elle ne prend jamais la valeur $-\infty$ et $\mathrm{dom} f \neq \emptyset$.

Définition 1.5 (Sous-gradient et sous-différentiel) *Soit f une fonction convexe sur $\mathbb{R}^n$ et $x^0 \in \mathrm{dom} f$. On appelle sous-gradient de f au point x^0 tout vecteur $y \in \mathbb{R}^n$ vérifiant*

$$f(x) \geq f(x^0) + \langle y, x - x^0 \rangle, \quad \forall x \in \mathbb{R}^n.$$

L'ensemble de tous les sous-gradients de f en x^0 est appelé sous-différentiel de f au point x^0. On le note $\partial f(x^0)$.
Si f est différentiable en x^0, le sous-différentiel de f en x^0 est réduit à un seul élément, le gradient de f en x^0, $\partial f(x^0) = \{\nabla f(x^0)\}$.
Le domaine du sous-différentiel de f, noté $\mathrm{dom}\ \partial f$, *est défini par*

$$\mathrm{dom}\ \partial f = \{x \in \mathbb{R}^n \mid \partial f(x) \neq \emptyset\}.$$

Pour une fonction convexe f sur $\mathbb{R}^n$ et $x \in \mathrm{dom} f$, $\partial f(x)$ est un *sous-ensemble convexe fermé* de $\mathbb{R}^n$.

Définition 1.6 (ϵ-sous-gradient et ϵ-sous-différentiel) *Soit ϵ un réel strictement positif, f une fonction convexe sur $\mathbb{R}^n$ et $x^0 \in \mathrm{dom} f$. Un vecteur $y \in \mathbb{R}^n$ est appelé un ϵ-sous-gradient de f au point x^0 si*

$$f(x) \geq (f(x^0) - \epsilon) + \langle y, x - x^0 \rangle, \quad \forall x \in \mathbb{R}^n.$$

L'ensemble de tous les ϵ-sous-gradients de f en x^0 est appelé ϵ-sous-différentiel de f au point x^0. On le note $\partial_\epsilon f(x^0)$.

Définition 1.7 (Fonction s.c.i.) *Soit $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ et $x^0 \in \mathbb{R}^n$. La fonction f est dite semi-continue inférieurement (s.c.i) en x^0 si et seulement si,*

$$\liminf_{x \to x^0} f(x) \geq f(x^0).$$

On note $\Gamma_0(\mathbb{R}^n)$ l'ensemble des fonctions convexes s.c.i. et propres sur $\mathbb{R}^n$.

Définition 1.8 (Fonction conjuguée) *Soit $f \in \Gamma_0(\mathbb{R}^n)$. La fonction conjuguée de f, notée f^*, est définie par*

$$y \in \mathbb{R}^n, \quad f^*(y) = \sup\{\langle x, y \rangle - f(x) \mid x \in \mathbb{R}^n\}.$$

Proposition 1.1 *Si $f \in \Gamma_0(\mathbb{R}^n)$ et $x \in \mathbb{R}^n$ alors*

$$y \in \partial f(x) \Longleftrightarrow f(x) + f^*(y) = \langle x, y \rangle \text{ et}$$

$$y \in \partial f(x) \Longleftrightarrow x \in \partial f^*(y).$$

Soit $f \in \Gamma_0(\mathbb{R}^n)$. Alors f est convexe polyédrale si et seulement si $\mathrm{dom} f$ est un polyèdre convexe et

$$f(x) = \sup\{\langle a^i, x \rangle - b^i, \ i = 1, \ldots, m\} + \chi_{\mathrm{dom} f}(x).$$

Proposition 1.2 *Si $f \in \Gamma_0(\mathbb{R}^n)$ est convexe polyédrale,*

$$f(x) = \sup\{\langle a^i, x \rangle - b^i, \ i = 1, \ldots, m\} + \chi_{\mathrm{dom} f}(x), \quad a^i \in \mathbb{R}^n, b^i \in \mathbb{R}, \forall i = 1, \ldots, m,$$

alors

- $\diamond$ *$\partial f(x)$ est un polyèdre convexe non vide pour tout $x \in \mathrm{dom} f$.*
- $\diamond$ *f^* l'est aussi et $\mathrm{dom}\, \partial f = \mathrm{dom} f$. De plus, si f est finie partout alors*

$$\mathrm{dom} f^* = co\{a^i, i = 1, \ldots, m\},$$

$$f^*(y) = \min \left\{ \sum_{i=1}^m \lambda_i b^i \mid y = \sum_{i=1}^m \lambda_i a^i, \sum_{i=1}^m \lambda_i = 1, \lambda_i \geq 0, \forall i = 1, \ldots, m \right\}.$$

Fonctions DC

Définition 1.9 (Fonction DC) *Soit C un sous-ensemble convexe non vide de $\mathbb{R}^n$ et $f : C \longrightarrow \overline{\mathbb{R}}$ une fonction. La fonction f est dite DC sur C si elle peut s'écrire comme la différence de deux fonctions g, h convexes sur C,*

$$f(x) = g(x) - h(x),\ \forall x \in C.$$

On dit que $g - h$ est une décomposition DC de f, et que g, h sont des composantes DC de f.

Si f est une fonction DC sur un ensemble convexe C et si f admet une décomposition DC comme $f = g - h$ alors pour toute fonction ϕ convexe finie sur C, $(g + \phi) - (h + \phi)$ fournit aussi une décomposition DC de f. Ainsi, toute fonction DC admet une infinité de décompositions DC.

L'ensemble des fonctions DC sur C est noté $\mathcal{DC}(C)$. L'espace vectoriel engendré par $\mathrm{Conv}(C)$ est exactement $\mathcal{DC}(C)$,

$$\mathcal{DC}(C) = \mathrm{Conv}(C) - \mathrm{Conv}(C).$$

L'espace vectoriel $\mathcal{DC}(C)$ est assez vaste pour contenir la plupart des fonctions objectifs de la vie courante et est stable par rapport à des opérations usuelles en optimisation.

Pour plus de détails sur les fonctions DC, le lecteur peut consulter [61,84,85,92,93,126, 127,149].

1.1.2 Programmation DC

Programme DC

Définition 1.10 (Programme DC) *On appelle programme DC tout problème d'optimisation de la forme*

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

où $g, h \in \Gamma_0(\mathbb{R}^n)$.

(P_{dc}) est un problème d'optimisation sans contrainte. Un problème d'optimisation avec une contrainte convexe (i.e., un convexe fermé non vide C) de la forme

$$\inf\{f(x) = g(x) - h(x) \mid x \in C\}$$

est équivalent à (P_{dc}) via l'addition de la fonction indicatrice de C à la première composante de f.

Dualité DC

Considérons le programme DC

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

où $g, h \in \Gamma_0(\mathbb{R}^n)$.

Grâce à la convention $(+\infty) - (+\infty) = (+\infty)$, le problème dual de (P_{dc}) est donc

$$\alpha = \inf\{h^*(y) - g^*(y) \mid y \in \mathbb{R}^n\} \qquad (D_{dc})$$

où $g^*, h^* \in \Gamma_0(\mathbb{R}^n)$ sont des fonctions conjuguées de g et de h respectivement. (D_{dc}) est aussi un programme DC. De plus, (P_{dc}) et (D_{dc}) ont la même valeur optimale et on peut observer la parfaite symétrie entre ces deux problèmes : le dual de (D_{dc}) est exactement (P_{dc}).

Les résultats suivants donnent quelques propriétés concernant les solutions de (P_{dc}) et (D_{dc}).

Théorème 1.1 *Soient $g, h \in \Gamma_0(\mathbb{R}^n)$.*

(i) x^0 est une solution globale de (P_{dc}) si et seulement si

$$\alpha = (g-h)(x^0) \leq (h^* - g^*)(y), \ \forall y \in \mathbb{R}^n.$$

(ii) y^0 est une solution globale de (D_{dc}) si et seulement si

$$\alpha = (h^* - g^*)(y^0) \leq (g-h)(x), \ \forall x \in \mathbb{R}^n.$$

Théorème 1.2 *Soient $g, h \in \Gamma_0(\mathbb{R}^n)$.*

(i) $\inf\{g(x) - h(x) \mid x \in \mathrm{dom} g\} = \inf\{h^*(y) - g^*(y) \mid y \in \mathrm{dom} h^*\}$

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

Si x^0 est un minimum de $g - h$ alors chaque $y^0 \in \partial h(x^0)$ est un minimum de $h^* - g^*$.

Ce dernier théorème montre que la résolution de l'un des deux problèmes (P_{dc}) et (D_{dc}) implique celle de l'autre.

Conditions d'optimalité en programmation DC

Il est bien connu en optimisation convexe que $x^0 \in \mathbb{R}^n$ minimise une fonction convexe $f \in \Gamma_0(\mathbb{R}^n)$ si et seulement si : $0 \in \partial f(x^0)$.

En programmation DC, la condition nécessaire et suffisante d'optimalité globale est formulée à l'aide des ϵ-sous-différentiels de g et h.

Théorème 1.3 (Condition d'optimalité globale) *Soient $g, h \in \Gamma_0(\mathbb{R}^n)$ et $x^0 \in \mathbb{R}^n$. x^0 est un minimum global de $g - h$ si et seulement si*

$$\partial_\epsilon h(x^0) \subset \partial_\epsilon g(x^0), \quad \forall \epsilon > 0. \tag{1.2}$$

Définition 1.11 (Programme DC polyédral) *Le programme DC (P_{dc}) est dit polyédral si l'une de ses composantes g ou h est une fonction convexe polyédrale.*

Définition 1.12 (Minimum local) *Soient $g, h \in \Gamma_0(\mathbb{R}^n)$. Un point $x^* \in \mathrm{dom} g \cap \mathrm{dom} h$ est un minimum local de $g - h$ s'il existe un voisinage $V(x^*)$ de x^* tel que*

$$(g-h)(x^*) \leq (g-h)(x), \ \forall x \in V(x^*).$$

Définition 1.13 (Point critique) *Un point $x^* \in \mathbb{R}^n$ est dit point critique ou point KKT généralisé de $g - h$ si $\partial g(x^*) \cap \partial h(x^*)$ est non vide.*

On note

$$\mathcal{P}_l = \{x \in \mathbb{R}^n \mid \partial h(x) \subset \partial g(x)\}, \quad \mathcal{D}_l = \{y \in \mathbb{R}^n \mid \partial g^*(y) \subset \partial h^*(y)\}$$

et note $\mathcal{P}$ (resp. $\mathcal{D}$) l'ensemble des solutions globales du problème (P_{dc}) (resp. (D_{dc})).

Théorème 1.4 (Condition nécessaire d'optimalité locale) *Si x^* est un minimum local de $g-h$ alors $x^* \in \mathcal{P}_l$. Cette condition est suffisante si h est polyédrale. De plus si f est localement convexe en x^*, en particulier si h est polyédrale et différentiable en x^*, alors x^* est une solution locale.*

Théorème 1.5 (Condition suffisante d'optimalité locale) *Si $x^* \in \mathrm{dom}g \cap \mathrm{dom}h$ admet un voisinage V tel que*

$$\partial h(x) \cap \partial g(x^*) \neq \emptyset, \ \forall x \in V \cap \mathrm{dom}g$$

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

Théorème 1.6

1. $\partial h(x^*) \subset \partial g(x^*), \ \forall x^* \in \mathcal{P}$ *et* $\partial g^*(y^\bullet) \subset \partial h^*(y^\bullet), \ \forall y^\bullet \in \mathcal{D}$.
2. (Transport de minima globaux)

$$\bigcup_{x^* \in \mathcal{P}} \partial h(x^*) \subseteq \mathcal{D} \subset \mathrm{dom}h^*.$$

La première inclusion devient égalité si g^* est sous-différentiable dans $\mathcal{D}$ (en particulier si $\mathcal{D} \subset \mathrm{ir}(\mathrm{dom}g^*)$ ou si g^* est sous-différentiable dans $\mathrm{dom}h^*$*). Dans ce cas, $\mathcal{D} \subset \mathrm{dom}\partial g^* \cap \mathrm{dom}\partial h^*$.*
Par dualité,

$$\bigcup_{y^\bullet \in \mathcal{D}} \partial g^*(y^\bullet) \subseteq \mathcal{P} \subset \mathrm{dom}g.$$

La première inclusion devient égalité si h est sous-différentiable dans $\mathcal{P}$ (en particulier si $\mathcal{P} \subset \mathrm{ir}(\mathrm{dom}h)$ ou si h est sous-différentiable dans $\mathrm{dom}g$*). Dans ce cas, $\mathcal{P} \subset \mathrm{dom}\partial g \cap \mathrm{dom}\partial h$.*

3. (Transport de minima locaux) *Soit $x^* \in \mathrm{dom}\partial h$ un minimum local de $g-h$ et $y^\bullet \in \partial h(x^*)$. Supposons que $V(x^*)$ est un voisinage de x^* vérifiant $(g-h)(x) \geq (g-h)(x^*), \ \forall x \in V(x^*) \cap \mathrm{dom}g$. Si*

$$x^* \in \mathrm{int}(\mathrm{dom}h), \quad y^\bullet \in \mathrm{int}(\mathrm{dom}g^*) \text{ et } \partial g^*(y^\bullet) \subset V(x^*)$$

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

1.1.3 DCA (DC Algorithms)

Basé sur la condition d'optimalité locale et la dualité DC, DCA consiste en la construction des deux suites $\{x^k\}$ et $\{y^k\}$, candidates à être solutions optimales des programmes DC primal et dual respectivement, de telle manière que les suites $\{g(x^k)-h(x^k)\}$ et $\{h^*(y^k)-g^*(y^k)\}$ soient décroissantes et $\{x^k\}$ (resp. $\{y^k\}$) converge vers une solution primal réalisable $\widetilde{x}$ (resp. une solution dual réalisable $\widetilde{y}$) vérifiant les conditions d'optimalité locale et

$$\widetilde{x} \in \partial g^*(\widetilde{y}), \quad \widetilde{y} \in \partial h(\widetilde{x}).$$

Les deux suites $\{x^k\}$ et $\{y^k\}$ sont déterminées de façon que
$\diamond$ x^{k+1} $(k \geq 0)$ est une solution du problème convexe :

$$\min\left\{g(x) - [h(x^k) + \langle x - x^k, y^k\rangle] \mid x \in \mathbb{R}^n\right\} \qquad (P_k)$$

$\diamond$ y^{k+1} $(k \geq 0)$ est une solution du problème convexe :

$$\min \left\{ h^*(y) - [g^*(y^k) + \langle y - y^k, x^{k+1} \rangle] \mid y \in \mathbb{R}^n \right\} \qquad (D_k)$$

L'interprétation de DCA est simple : à chaque itération k, la seconde composante du problème primal (P_{dc}) (resp. problème dual (D_{dc})) est remplacée par sa minorante affine $h_k(x) := h(x^k) + \langle x - x^k, y^k \rangle$ au voisinage de x^k, définie par un sous-gradient y^k de h en x^k (resp. $g_k^*(y) := g^*(y^k) + \langle y - y^k, x^{k+1} \rangle$ au voisinage de y^k, définie par un sous-gradient x^{k+1} de g^* en y^k). L'ensemble des solutions du problème convexe obtenu (P_k) (resp. (D_k)) n'est rien d'autre que $\partial g^*(y^k)$ (resp. $\partial h(x^{k+1})$),

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

Le processus est alors répété jusqu'à la convergence.

Algorithme DCA
$x^0 \in \mathrm{dom} g$, $k \longleftarrow 0$.
Répéter

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

Jusqu'à la convergence.

En résumé, on peut décrire l'algorithme DCA à l'aide du schéma suivant :

$$\begin{array}{ccc} x^k & \longrightarrow & y^k \in \partial h(x^k) \\ & \swarrow & \\ x^{k+1} \in \partial g^*(y^k) & \longrightarrow & y^{k+1} \in \partial h(x^{k+1}) \end{array}$$

Convergence de DCA

DCA n'est défini que si l'on peut construire les suites $\{x^k\}$ et $\{y^k\}$ à partir d'un point x^0 choisi à l'avance.

Lemme 1.1 (Existence des suites) *Les suites $\{x^k\}$ et $\{y^k\}$ sont bien définies si et seulement si*

$$\mathrm{dom}\partial g \subset \mathrm{dom}\partial h \text{ et } \mathrm{dom}\partial h^* \subset \mathrm{dom}\partial g^*$$

Supposons que $\{x^k\}$ et $\{y^k\}$ sont bien définis. Le théorème de convergence de DCA requiert que ces deux suites soient bornées.

Lemme 1.2 (Bornitude des suites) *Si $g - h$ est cœrcive (i.e. $\lim\limits_{\|x\| \to +\infty} (g-h)(x) = +\infty$) alors*

(i) la suite $\{x^k\}$ est bornée,
(ii) si de plus, $\{x^k\} \subset \mathrm{int}(\mathrm{dom} h)$ alors $\{y^k\}$ est aussi bornée.
Par dualité, si $h^* - g^*$ est cœrcive alors
(iii) la suite $\{y^k\}$ est bornée,
(iv) si de plus, $\{y^k\} \subset \mathrm{int}(\mathrm{dom} g^*)$ alors $\{x^k\}$ est aussi bornée.

La convergence de l'algorithme est assurée par les résultats suivants :

Théorème 1.7 (Convergence de DCA) *Supposons que les suites $\{x^k\}$ et $\{y^k\}$ sont bien définies. Alors DCA est une méthode de descente sans recherche linéaire mais avec une convergence globale et possède les caractéristiques suivantes :*

1. *Les suites $\{(g-h)(x^k)\}$ et $\{(h^*-g^*)(y^k)\}$ sont décroissantes et*
   - *$(g-h)(x^{k+1}) = (g-h)(x^k)$ si et seulement si*

$$\begin{cases} y^k \in \partial g(x^k) \cap \partial h(x^k), \\ y^k \in \partial g(x^{k+1}) \cap \partial h(x^{k+1}), \\ [\rho(g) + \rho(h)]\|x^{k+1} - x^k\| = 0. \end{cases}$$

   De plus, si g et h sont strictement convexes sur $\mathbb{R}^n$ alors $x^{k+1} = x^k$. Dans ce cas, DCA se termine à la $(k+1)^{ème}$ itération (convergence finie de DCA).
   - *$(h^*-g^*)(y^{k+1}) = (h^*-g^*)(y^k)$ si et seulement si*

$$\begin{cases} x^{k+1} \in \partial h^*(y^k) \cap \partial g^*(y^k), \\ x^{k+1} \in \partial h^*(y^{k+1}) \cap \partial g^*(y^{k+1}), \\ [\rho(h^*) + \rho(g^*)]\|y^{k+1} - y^k\| = 0. \end{cases}$$

   De plus, si h^* et g^* sont strictement convexes sur Y alors $y^{k+1} = y^k$. Dans ce cas, DCA se termine à la $(k+1)^{ème}$ itération (convergence finie de DCA).
2. *Si $[\rho(g) + \rho(h)] > 0$ (resp. $[\rho(h^*) + \rho(g^*)] > 0$) alors $\lim\limits_{k\to+\infty} \|x^{k+1} - x^k\| = 0$ (resp. $\lim\limits_{k\to+\infty} \|y^{k+1} - y^k\| = 0$).*
3. *Si la valeur optimale α du problème primal (P_{dc}) est finie alors*
   - *Les suites décroissantes $\{(g-h)(x^k)\}$ et $\{(h^*-g^*)(y^k)\}$ convergent vers la même limite $\beta \geq \alpha$.*
   - *Si de plus, les suites $\{x^k\}$ et $\{y^k\}$ sont bornées alors pour toute valeur d'adhérence $\widetilde{x}$ de $\{x^k\}$ il existe une valeur d'adhérence $\widetilde{y}$ de $\{y^k\}$ telle que*

$$\widetilde{y} \in \partial g(\widetilde{x}) \cap \partial h(\widetilde{x}) \text{ et } g(\widetilde{x}) - h(\widetilde{x}) = \beta.$$

   De même pour la suite $\{y^k\}$.
4. *DCA a une convergence linéaire pour un programme DC dans le cas général. Dans le cas polyédral, cette convergence est finie.*
5. *DCA appliqué aux programmes DC avec des données subanalytiques a la convergence de la suite entière $\{x^k\}$ (resp. $\{y^k\}$).*

Remarque 1.1

1. *Il est clair que DCA s'applique aux fonctions convexes g et h, et non à la fonction f elle-même. On voit ainsi comment le mécanisme de DCA fonctionne pour les programmes DC non différentiables (f est une fonction DC non différentiable). Et puisqu'une fonction DC admet une infinité de décompositions DC, il en aura autant de DCA. Le choix d'une décomposition DC appropriée est crucial car il conditionne les qualités essentielles (rapidité, robustesse, globalité des solutions calculées) du DCA résultant. Théoriquement, le problème de décomposition DC optimale reste*

à définir et à étudier. En pratique on cherche des décompositions DC bien adaptées à la structure spécifique du problème traité afin que les deux suites $\{x^k\}$ et $\{y^k\}$ soient obtenues à moindre coût en temps de calcul, si elles ne sont pas explicites. Ici, peut être plus qu'ailleurs, les techniques de reformulation sont omniprésentes. Il est important de noter qu'avec des décompositions DC appropriées, DCA permet de retrouver, comme cas particuliers, la plupart des méthodes standard en programmation convexe/non convexe. D'autre part un programme convexe est un programme DC pour lequel DCA converge vers une solution (locale qui est aussi globale) : de cette manière DCA permet de construire une infinité d'algorithmes pour la programmation convexe, qui pourraient être plus performants que les méthodes existantes.

2. *Il est important de noter qu'avec des décompositions DC appropriées, DCA permet de retrouver, comme cas particuliers, la plupart des méthodes standards en programmation convexe/non convexe.*

   A notre connaissance, DCA est actuellement parmi les rares algorithmes de la programmation non convexe, capables de traiter des problèmes de très grande taille et reste le seul algorithme performant pour des programmes non convexes non différentiables.

Programmation DC polyédrale

Considérons le problème (P_{dc}) dont l'une des composantes g, h est convexe polyédrale. On peut supposer que la fonction h est polyédrale et est donnée par $h(x) = \sup\{\langle a^i, x\rangle - b^i,\ i = 1, \ldots, m\} + \chi_C(x)$ où C est un polyèdre convexe non vide de $\mathbb{R}^n$ (si dans (P_{dc}) la fonction g est polyédrale et pas h, on va considérer le problème dual (D_{dc}) car g^* est polyédrale).

Supposons que α est fini. Ce qui implique $\mathrm{dom}g \subset \mathrm{dom}h = C$. Dès lors, au lieu de (P_{dc}) on peut considérer le problème équivalent $(\widetilde{P}_{dc})$:

$$\alpha = \inf\{g(x) - \widetilde{h}(x) \mid x \in \mathbb{R}^n\} \qquad (\widetilde{P}_{dc})$$

où $\widetilde{h}(x) = \sup\{\langle a^i, x\rangle - b^i,\ i \in I\}$ avec $I = \{1, \ldots, m\}$.

On a donc

$$\alpha = \inf_{i\in I} \inf_{x\in\mathbb{R}^n} \{g(x) - (\langle a^i, x\rangle - b^i)\}$$

Pour chaque $i \in I$, on note (P^i) le problème convexe suivant

$$\alpha_i = \inf\{g(x) - (\langle a^i, x\rangle - b^i) \mid x \in \mathbb{R}^n\} \qquad (P^i)$$

dont l'ensemble des solutions est $\partial g^*(a^i)$.

Le problème dual $(\widetilde{D}_{dc})$ de $(\widetilde{P}_{dc})$ est :

$$\alpha = \inf\{\widetilde{h}^*(y) - g^*(y) \mid y \in co\{a^i, i \in I\}\} \qquad (\widetilde{D}_{dc})$$

On note également

$$J(\alpha) = \{i \in I \mid \alpha_i = \alpha\} \text{ et } I(x) = \{i \in I \mid \langle a^i, x\rangle - b^i = \widetilde{h}(x)\}$$

Théorème 1.8

(i) $x^* \in \mathcal{P}$ si et seulement si $I(x^*) \subset J(\alpha)$ et $x^* \in \bigcap_{i\in I(x^*)} \partial g^*(a^i)$.

(ii) $\mathcal{P} = \bigcup_{i \in J(\alpha)} \partial g^*(a^i)$. *Si* $\{a^i, i \in I\} \subset \text{dom } \partial g^*$ *alors* $\mathcal{P} \neq \emptyset$.
(iii) $\widetilde{h}(x) = \max\{\langle x, y\rangle - \widetilde{h}^*(y) \mid y \in co\{a^i, i \in I\}\} = \max\{\langle a^i, x\rangle - \widetilde{h}^*(a^i) \mid i \in I\}$.
(iv) $J(\alpha) = \{i \in I \mid a^i \in \widetilde{D}, \widetilde{h}^*(a^i) = b^i\}$; $\widetilde{D} \supset \{a^i, i \in J(\alpha)\}$.

Ainsi, résoudre le programme DC polyédrale $(\widetilde{P}_{dc})$ ramène à résoudre m programmes convexes (P^i). Afin de générer l'ensemble $\mathcal{P}$, on peut déterminer d'abord $J(\alpha)$ puis appliquer le théorème 1.8. En pratique, ceci peut être utilisé si m est relativement petit. Dans le cas où m est grand, DCA sera appliqué pour résoudre localement $(\widetilde{P}_{dc})$. On rappelle ci-dessous la description de DCA concernant le programme DC polyédral :

- $\diamond$ $x^0 \in \mathbb{R}^n$ choisi à l'avance.
- $\diamond$ Itération k, $x^k \longrightarrow y^k \in \partial\widetilde{h}(x^k) = co\{a^i, i \in I(x^k)\}$ et $x^{k+1} \in \partial g^*(y^k)$.
  En prenant $y^k = a^i, i \in I(x^k)$, le calcul de x^{k+1} est réduit à résoudre le programme convexe suivant :
  $$\min\{g(x) - \langle y^k, x\rangle \mid x \in \mathbb{R}^n\}.$$
  Noter que si $y^k = a^i$ avec $i \in J(\alpha)$, alors $x^{k+1} \in \mathcal{P}$ (selon théorème 1.8).

Les suites obtenues $\{x^k\}$ et $\{y^k\}$ sont discrètes (c-à-d elles ont seulement un nombre fini d'éléments différents). La convergence de DCA est donc finie dans ce cas.

1.2 Méthode par Séparation et Evaluation (SE)

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

$$\min\{f(x) \mid x \in S \subset \mathbb{R}^n\}. \qquad (P)$$

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

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

L'idée de base de la méthode par Séparation et Evaluation (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 ce sous-ensemble en vue d'éliminer les parties non prometteuses et de sélectionner un sous-ensemble que l'on va diviser par la suite.

Définition 1.14 *Soit M un compact de $\mathbb{R}^n$ et I un ensemble fini d'indices. Un ensemble $\{M_i : i \in I\}$ des sous-ensembles compacts est appelé une 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.

1.2.1 Méthode de résolution et convergence

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

Prototype 1 ([59, 130])

- **Initialisation**

  1. Choisir un compact $M_0 \supset S$, un ensemble fini d'indices I_0, une partition $\mathcal{M}_0 = \{M_{0,i} : i \in I_0\}$ de M_0 satisfaisant $M_{0,i} \cap S \neq \emptyset, \forall i \in I_0$.
  2. Pour chaque $i \in I_0$, déterminer $S_{0,i} \subset M_{0,i} \cap S,\ S_{0,i} \neq \emptyset$ et
  $$\gamma_{0,i} = \gamma(M_{0,i}) := \min f(S_{0,i}), \quad x^{0,i} \in \arg\min f(S_{0,i}).$$
  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
  $$\gamma_0 = \min_{i \in I_0} \gamma_{0,i}, \tag{1.3}$$
  $$x^0 \in \arg\min\{f(x^{0,i}), i \in I_0\}, \tag{1.4}$$
  $$\beta_0 = \min_{i \in I_0} \beta_{0,i}. \tag{1.5}$$

- **Itération** $k = 0, 1, \ldots$

  k.1 Supprimer tout $M_{k,i} \in \mathcal{M}_k$ vérifiant soit
  $$\beta_{k,i} \geq \gamma_k,$$
  soit sur lequel $\min f(S)$ ne peut pas avoir lieu. Soit $\mathcal{R}_k$ la collection des éléments restants dans $\mathcal{M}_k$.
  Si $\mathcal{R}_k = \emptyset$ alors s'arrêter : x^k est une solution globale de (P).

  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 $M_{k+1,i} \cap S \neq \emptyset, \forall i \in J_{k+1}$.

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

  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 tel que $\mathcal{M}_{k+1} = \{M_{k+1,i} : i \in I_{k+1}\}$ est la partition courante.
  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
  $$\gamma_{k+1} = \min_{i \in I_{k+1}} \gamma_{k+1,i} \tag{1.6}$$
  $$x^{k+1} \in \arg\min\{f(x^{k+1,i}), i \in I_{k+1}\} \tag{1.7}$$
  $$\beta_{k+1} = \min_{i \in I_{k+1}} \beta_{k+1,i} \tag{1.8}$$
  et aller à l'itération $k+1$.

Conditions de la convergence

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

$$x^k \in S, \quad k = 0, 1, \ldots \tag{1.9}$$

$$\gamma_k \geq \gamma_{k+1} \geq \min f(S) \geq \beta_{k+1} \geq \beta_k \tag{1.10}$$

$$f(x^k) \geq f(x^{k+1}), \quad k = 0, 1, \ldots \tag{1.11}$$

Définition 1.15 *Une estimation de borne est dite « cohérente » (consistent) si, pour une suite décroissante des sous-ensembles quelconque $M_{k_q,i_{k_q}}$ générée par la phase de séparation, i.e.,*

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

on a

$$\lim_{q\to\infty}(\gamma_{k_q,i_{k_q}} - \beta_{k_q,i_{k_q}}) = 0. \tag{1.12}$$

Puisque $\beta_{k_q,i_{k_q}} \leq \gamma_{k_q} \leq \gamma_{k_q,i_{k_q}}$, la condition (1.12) peut s'écrire

$$\lim_{q\to\infty}(\gamma_{k_q} - \beta_{k_q,i_{k_q}}) = 0. \tag{1.13}$$

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 1.16 *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

$$M_{k,i_k} \in \arg\min\{\beta(M) : M \in \mathcal{R}_k\}. \tag{1.14}$$

Théorème 1.9 [59] *Supposons que S est fermé, $\min f(S)$ existe et que dans le prototype, l'opération d'estimation de borne est cohérente.*

(i) Si la sélection est complète alors

$$\gamma := \lim_{k\to\infty} \alpha_k = \lim_{k\to\infty} f(x^k) = \min f(S). \tag{1.15}$$

(ii) Si la sélection est borne-améliorante alors

$$\beta := \lim_{k\to\infty} \beta_k = \min f(S). \tag{1.16}$$

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

1.2.2 Séparation et Evaluation avec des ensembles non réalisables

Définition 1.17 *Un ensemble M vérifiant $M \cap S = \emptyset$ est appelé « non réalisable ». Un ensemble M vérifiant $M \cap S \neq \emptyset$ est appelé « réalisable ». Un ensemble M est dit « incertain » lorsque nous ne savons pas si M est réalisable ou non.*

Evidemment, un ensemble sera éliminé si on sait qu'il est non réalisable. Lorsque les ensembles incertains sont admis, on va demander pour que

$$\begin{aligned} -\infty < \beta(M) \leq \min f(M \cap S), &\qquad \text{si } M \text{ est réalisable} \\ -\infty < \beta(M) \leq \min f(M), &\qquad \text{si } M \text{ est incertain.} \end{aligned}$$

En général, $S_M \subset M \cap S$ peut être vide et il est possible que la borne $\alpha(M)$ égale l'infini. La variante du prototype ci-dessous sera appliquée lorsqu'on ne peut pas décider définitivement si $M \cap S \neq \emptyset$ a lieu pour tous les ensembles de la partition donnée. Noter que dans ce cas, les bornes supérieures ne sont pas toujours disponibles. Pour la clarté, on va décrire cette variante en détail. Remarquons que, par convention, le minimum sur un ensemble vide prend la valeur infinie.

Prototype 2 ($[60, 130]$)

- **Initialisation**

  1. Choisir un compact M_0 tel que $S \subset M_0$, un ensemble fini d'indices I_0, une partition $\mathcal{M}_0 = \{M_{0,i} : i \in I_0\}$ de M_0.
  2. Pour chaque $i \in I_0$, déterminer $S_{0,i} \subset M_{0,i} \cap S,\ S_{0,i} \neq \emptyset$ et

  $$\alpha_{0,i} = \alpha(M_{0,i}) := \min f(S_{0,i}), \quad x^{0,i} \in \arg\min f(S_{0,i}).$$

  Si $S_{0,i}$ n'est pas disponible (par des efforts raisonnables), on pose $S_{0,i} = \emptyset$.
  3. Pour chaque $i \in I_0$, déterminer $\beta_{0,i} = \beta(M_{0,i})$ vérifiant

  $$\begin{aligned} \beta(M_{0,i}) \leq \min f(S \cap M_{0,i}), &\qquad \text{si } M_{0,i} \text{ est réalisable} \\ \beta(M_{0,i}) \leq \min f(M_{0,i}), &\qquad \text{si } M_{0,i} \text{ est incertain.} \end{aligned}$$

  4. Calculer

  $$\alpha_0 = \min_{i \in I_0} \alpha_{0,i} \tag{1.17}$$
  $$x^0 \in \arg\min\{f(x^{0,i}) : i \in I_0\} \tag{1.18}$$
  $$\beta_0 = \min_{i \in I_0} \beta_{0,i} \tag{1.19}$$

- **Itération** $k = 0, 1, \ldots$

  k.1 Supprimer tout $M_{k,i} \in \mathcal{M}_k$ vérifiant soit

  $$\beta_{k,i} \geq \alpha_k$$

  soit sur lequel $\min f(S)$ ne peut pas avoir lieu. Soit $\mathcal{R}_k$ la collection des éléments restants de $\mathcal{M}_k$.
  Si $\mathcal{R}_k = \emptyset$ alors s'arrêter : x^k est une solution globale.

k.2 Sélectionner $M_{k,i_k} \in \mathcal{R}_k$, choisir un ensemble fini d'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}. Appliquer des règles pour éliminer les sous-ensembles non réalisables.

k.3 Pour chaque $i \in J_{k+1}$, déterminer $S_{k+1,i} \subset M_{k+1,i} \cap S,\ S_{k+1,i} \neq \emptyset$ et

$$\alpha_{k+1,i} = \alpha(M_{k+1,i}) := \min f(S_{k+1,i}), \quad x^{k+1,i} \in \arg\min f(S_{k+1,i}).$$

Si $S_{k+1,i}$ n'est pas disponible, on pose $S_{k+1,i} = \emptyset$.

k.4 Pour chaque $i \in J_{k+1}$, 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 d'indices tel que $\mathcal{M}_{k+1} = \{M_{k+1,i} : i \in I_{k+1}\}$ est la partition courante.

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

k.6 Calculer

$$\alpha_{k+1} = \min_{i \in I_{k+1}} \alpha_{k+1,i} \tag{1.20}$$

$$\beta_{k+1} = \min_{i \in I_{k+1}} \beta_{k+1,i} \tag{1.21}$$

Si $\alpha_{k+1} < \infty$ alors, on prend $x^{k+1} \in S$ tel que $f(x^{k+1}) = \alpha_{k+1}$.

Aller à l'itération $k+1$.

Définition 1.18 *Une procédure de division est dite « exhaustive » si pour chaque suite décroissante $\{M_{k_q}\}$ d'ensembles générés par cette procédure, la suite des diamètres $d(M_{k_q})$ associés à M_{k_q} vérifie*

$$\lim_{q \to \infty} d(M_{k_q}) = 0. \tag{1.22}$$

Evidemment, pour une suite décroissante d'ensembles générés par une division exhaustive, on a

$$\lim_{q \to \infty} M_{k_q} = \bigcap_q M_{k_q} = \{\overline{x}\}, \quad \overline{x} \in \mathbb{R}^n. \tag{1.23}$$

Définition 1.19 *L'opération d'estimation de borne inférieure est appelée « fortement cohérente » (strongly consistent) si quelque soit la suite décroissante $\{M_{k_q}\}$ d'ensembles générés par une division exhaustive satisfaisant*

$$M_{k_q} \to \{\overline{x}\}, \quad q \to \infty,$$

il existe une sous-suite $\{M_{k'_q}\}$ de $\{M_{k_q}\}$ pour laquelle

$$\beta(M_{k'_q}) \to f(\overline{x}), \quad q \to \infty. \tag{1.24}$$

Définition 1.20 *L'élimination-par-non réalisabilité est appelée « certaine à la limite » si pour n'importe quelle suite décroissante $\{M_{k_q}\}$ d'ensembles générés par une division exhaustive vérifiant*

$$M_{k_q} \to \{\overline{x}\}, \quad q \to \infty,$$

on a

$$\overline{x} \in S. \tag{1.25}$$

On désigne Y^α l'ensemble des points d'accumulation de la suite $\{y^k\}$ des points correspondants à β_k. Soit $X^* = \arg\min f(S)$ l'ensemble des solutions optimales de (P).

Théorème 1.10 [60] *Supposons que le Prototype 2 vérifie les conditions suivantes :*
(i) la division est exhaustive ;
(ii) la sélection est borne-améliorante ;
(iii) la borne inférieure est fortement cohérente ;
(iv) l'élimination est certaine à la limite.
Alors on a

$$\beta := \lim_k \beta_k = \min f(S) \tag{1.26}$$

et

$$Y^\alpha \subset X^*. \tag{1.27}$$

Réalisation

Bien entendu, la réalisation d'un algorithme SE dépend du choix des opérations suivantes à chaque itération :

- Diviser M_{k,i_k}.
- Sélectionner M_{k,i_k}.
- Estimer les bornes inférieures $\beta_{k,i}$.

Stratégie de division

Les éléments de la partition $\mathcal{M}_k$ de 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 par SE.

Subdivision simpliciale

M_0 et tous les éléments de subdivision sont des n-simplexes. Ce n'est pas difficile de construire le premier simplexe M_0 contenant S. Soit $M = \text{conv}\,\{v^0, v^1, \ldots, v^n\}$ un simplexe avec des sommets $v^0, v^1, \ldots, v^n$. Alors un point $s \in M$ peut être représenté par

$$s = \sum_{i=0}^{n} \lambda_i v^i, \quad \lambda_i \geq 0, \quad \sum_{i=0}^{n} \lambda_i = 1.$$

Supposons que $s \neq v^i, \forall i = 0, 1, \ldots, n$. Posons $J = \{j : \lambda_j > 0\}$. En remplaçant un sommet v^j, $j \in J$ par s on obtient un simplexe

$$M_j = \text{conv}\,\{v^0, \ldots, v^{j-1}, s, v^{j+1}, \ldots, v^n\}$$

et ainsi on peut construire une subdivision, appelée *radiale*, de M. Très souvent, on choisit s comme le milieu de la plus longue arête de M et M est divisé en deux simplexes. Dans ce cas, on a une bissection de M. Il est démontré que la bissection est exhaustive. Pourtant, on constate que les procédures exhaustives de division (en particulier bissection) ne sont pas très efficaces ; la convergence de la méthode est assez lente. On suggère alors d'utiliser comme s un point ω obtenu dans la procédure d'estimation de borne (par exemple ω est le point correspondant à $\beta(M)$). On va appeler cette procédure ω-subdivision. Le problème est qu'on ne peut plus assurer que la procédure de division soit exhaustive.

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 contenant l'ensemble convexe S peut être déterminé en résolvant $2n$ problèmes convexes

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

Les processus de subdivision rectangulaire jouent un rôle important dans des méthodes par Séparation et Evaluation. L'approche de Phillips et Rosen [129] (voir également Kalantari et Rosen [67]) emploie la subdivision exhaustive, i.e., toute suite décroissante de rectangles générés par l'algorithme tendra vers un point. Une bissection rectangulaire adaptative prétendue proposée par Muu et al. [115] semble être plus efficace parce que l'exhaustivité n'est pas nécessaire pour la convergence. Dans Horst et Tuy [63], un concept de la subdivision rectangulaire normale (normal rectangular subdivision - NRS) a été présenté pour la classe des problèmes concaves séparables de minimisation. Intuitivement, la variante des algorithmes rectangulaires en utilisant ω-subdivision et subdivision adaptative devrait converger plus rapidement que les algorithmes qui emploient la subdivision exhaustive, parce qu'ils tiennent compte des conditions du sous-problème relaxé courant.

Nous détaillerons maintenant la subdivision rectangulaire normale aussi bien que sa construction quand on utilise dans un algorithme SE pour résoudre une classe des problèmes importants dans la programmation DC.

Considérons le problème DC suivant

$$\min\{f(x) = g(x) - h(x) \mid x \in B\} \tag{1.28}$$

où $g, h \in \Gamma_0(\mathbb{R}^n)$ $B = \prod_{i=1}^{n} [l_i, L_i]$ est un rectangle dans $\mathbb{R}^n$ et h est une fonction convexe séparable, $h(x) = \sum_{i=1}^{n} h_i(x_i)$.

Une méthode standard pour l'estimation de borne dans un schéma de SE est d'utiliser une minorante convexe de la fonction objectif. La meilleure minorante convexe de f sur B est son enveloppe convexe que nous notons $\text{conv}_B(f)$. Cependant, dans le cas général, au lieu de calculer $\text{conv}_B(f)$, nous construisons souvent $\text{conv}_B(-h)$ et utilisons $(g+\text{conv}_B(-h))$ comme la minorante convexe pour l'estimation de borne.

Comme la fonction h est séparable, l'enveloppe convexe $\text{conv}_B(-h)$ de $(-h)$ sur le rectangle B est simplement la somme des fonctions affines ϕ_{B_i} qui coïncident avec $-h_i$ aux sommets du segment $B_i = [l_i, L_i]$ ([67, 129, 137])

$$\phi_B(x) = \sum_{i=1}^{n} \phi_{B_i}(x_i) \tag{1.29}$$

Par conséquent, la solution du programme convexe

$$\min\{g(x) + \phi_B(x) \mid x \in B\} \tag{1.30}$$

fournit un point x^B tel que

$$g(x^B) + \phi_B(x^B) \leq \min\{f(x) \mid x \in B\} \leq f(x^B) \tag{1.31}$$

Subdivision Rectangulaire Normale (NRS)

Nous rappelons maintenant le concept d'une subdivision rectangulaire normale comme présenté par Tuy (voir par exemple [63]).

Soit $B = \{x : l_i \leq x_i \leq L_i\}$ un rectangle et soit ϕ_B la minorante convexe de $(-h)$ définie ci-dessus sur B. Dénotons par x^B et $\beta(B)$ une solution optimale et la valeur optimale, respectivement, du programme convexe (1.30).

Considérons un processus de subdivision rectangulaire dans lequel un rectangle est subdivisé en sous-rectangles. Un tel processus génère une famille des rectangles qui peut être représentée par un arbre avec la racine $B^0 = B$. Un chemin infini dans cet arbre correspond à une séquence contractée de rectangles B^i, $i = 0, 1, \ldots$. Pour chaque i, posons $x^i = x^{B^i}$, $\phi_i(x) = \phi_{B^i}(x)$.

Définition 1.21 *Une séquence contractée B^i est appelée normale si*

$$\lim_{i\to\infty} | - h(x^i) - \phi_i(x^i)| = 0. \tag{1.32}$$

Un processus de subdivision rectangulaire est appelé normal si toute séquence contractée de rectangles qu'il génère est normale.

Nous discuterons dans ce qui suit quelques méthodes pour construire un processus de subdivision rectangulaire normale. Supposons maintenant qu'un processus de NRS a été défini. En utilisant ce processus en même temps que la procédure d'estimation de borne développée plus haut nous pouvons construire un algorithme de SE pour résoudre (1.28).

Algorithme ([130])

- **Initialisation**
  Prendre $B^0 \leftarrow B$. Calculer ϕ_{B^0} et résoudre le programme convexe

  $$\min\{g(x) + \phi_{B^0}(x) \mid \ x \in B^0\}$$

  pour obtenir une solution optimale x^{B^0} et la valeur optimale $\beta(B^0)$.
  Poser $\mathcal{B}_0 = \{B^0\}, \beta_0 = \beta(B^0), \alpha_0 = f(x^{B^0})$ et $x^* = x^{B^0}$.
- **Itération** $k = 0, 1, \ldots$
  k.1 Supprimer tout $S \in \mathcal{B}_k$ tel que $\beta(S) \geq \alpha_k$. Soit $\mathcal{R}_k$ l'ensemble de rectangles restants. Si $\mathcal{R}_k = \emptyset$ alors STOP, x^* est une solution optimale globale.
  k.2 Sinon, sélectionner $B^k \in \mathcal{R}_k$ tel que

  $$\beta_k := \beta(B^k) = \min\{\beta(R) \mid R \in \mathcal{R}_k\}$$

  et subdiviser B^k en B^{k_1} et B^{k_2} selon le processus de subdivision rectangulaire normale choisi.
  k.3 Pour chaque ensemble B^{k_1}, B^{k_2}, calculer $\phi_{B^{k_i}}$ et résoudre

  $$\min\{g(x) + \phi_{B^{k_i}}(x) \mid x \in B^{k_i}\}$$

  pour obtenir $x^{B^{k_i}}$ et $\beta(B^{k_i})$.
  k.4 Mettre à jour la meilleure solution réalisable x^* et α_{k+1}.
  k.5 Poser $\mathcal{B}_{k+1} := (\mathcal{B}_k \setminus B^k) \cup \{B^{k_1}, B^{k_2}\}$ et aller à la prochaine itération.

Théorème 1.11 [130] *(i) Si l'algorithme se termine à l'itération k alors x^* est une solution optimale globale du problème (1.28).*
(ii) Si l'algorithme est infini alors il génère une séquence bornée $\{x^k\}$ dont tout point d'accumulation est une solution optimale globale de (1.28), et

$$\alpha_k \searrow f_*, \quad \beta_k \nearrow f_*.$$

Construction d'une NRS

Les auteurs de [63] ont présenté plusieurs façons de construire un processus de NRS. Supposons qu'un rectangle $B^k = \{x \mid l_i^k \leq x_i \leq L_i^k\}$ est sélectionné en étape $k.2$. La règle suivante de bissection de B^k a été utilisée dans la littérature.

(i) **Bissection exhaustive**

En général, i_k est choisi comme l'indice du plus long côté de B^k, i.e., i_k vérifie

$$(L_{i_k} - l_{i_k})^2 = \max\{(L_i - l_i)^2, i = 1, \ldots, n\}.$$

Soit $\delta = \dfrac{1}{2}(L_{i_k} + l_{i_k})$. Alors R_k est divisé en deux sous-rectangles :

$$B^{k_1} = \{x \in B^k \mid x_{i_k} \leq \delta\}, \quad B^{k_2} = \{x \in B^k \mid x_{i_k} \geq \delta\}.$$

En particulier, pour une fonction quadratique concave séparable, l'indice i_k peut être choisi tel que

$$\lambda_{i_k}(L_{i_k} - l_{i_k})^2 = \max\{\lambda_i(L_i - l_i)^2, i = 1, \ldots, n\}.$$

Il a été montré que dans les deux cas, toute séquence contractée de rectangles tend vers un point.

(ii) **ω-bissection**

Par description de l'algorithme, pour B^k choisi, $\beta(B^k) < f(x^k)$, alors

$$-h(x^k) - \phi_k(x^k) > 0.$$

Choisir un indice i_k satisfaisant

$$i_k \in \arg\max_i\{-h_i(x_i^k) - \phi_{i_k}(x_i^k)\}$$

et subdivisons B^k en sous-rectangles

$$B^{k_1} = \{x \in B^k \mid x_{i_k} \leq x_{i_k}^k\}, \quad B^{k_2} = \{x \in B^k \mid x_{i_k} \geq x_{i_k}^k\}$$

(iii) **Bissection adaptative**

Pour chaque B^k sélectionné, deux points sont considérés. L'un est x^k, l'autre est v^k tel que

$$v_i^k \in \arg\min_i\{-h_i(l_i^k), -h_i(L_i^k)\}.$$

Autrement dit

$$v^k \in \arg\min_{B^k} \phi_{B^k}(x).$$

On appelle ces points les points de bissection.

Choisir un indice i_k tel que

$$|(v^k - x^k)_{i_k}| = \max_i |(v^k - x^k)_i|$$

et prendre $\delta = \frac{1}{2}(x^k_{i_k} + v^k_{i_k})$.

Diviser R_k en deux sous-rectangles

$$B^{k_1} = \{x \in B^k \mid x_{i_k} \leq \delta\}, \quad B^{k_2} = \{x \in B^k \mid x_{i_k} \geq \delta\}.$$

Subdivision conique

Supposons que ω est un point intérieur de S. Soit M_0 un n-simplexe tel que $S \subseteq M_0$ et M_0 possède $(n+1)$ faces $F_{0,i}, i = 1, 2, \ldots, n+1$, de dimension $(n-1)$ qui sont des $(n-1)$-simplexes. Les cônes polyédraux (appelés cônes) $C_{0,i}$, centrés au ω et engendrés par $F_{0,i}$, constituent une division de $\mathbb{R}^n$. Ensuite, un cône C est défini par un $(n-1)$-simplexe F dans $F_{0,i}$. Evidemment, une division de F en simplexes $\{F_j : j \in J\}$ va créer une division de cône C en cônes $\{C_j : j \in J\}$ correspondant aux F_j. Particulièrement, la bissection de cônes est entraînée par la bissection de simplexes. Si la procédure de division de simplexes est exhaustive alors la procédure de division de cônes sera aussi exhaustive dans le sens que chaque suite décroissante de cônes $\{C_k\}$ (générée par cette procédure) va tendre vers un rayon sortant de ω.

Subdivision prismatique

Une extension de cône est la notation de prismes dans $\mathbb{R}^n \times \mathbb{R}$ quand le sommet ω est considéré comme un point fictif à l'infini. Chaque simplexe ou rectangle M définit un prisme $T = \{(x,t) : x \in M\}$. Les procédures exhaustives de division de simplexes et de rectangles engendrent les procédures de division de prismes qui sont exhaustives dans le sens que chaque suite décroissante de prismes $\{T_k\}$ (générée par cette procédure) va tendre vers une droite verticale. Les prismes sont très utiles pour construire des algorithmes SE quand des fonctions DC interviennent dans la formulation du problème.

Règle de sélection

Naturellement, on peut choisir à chaque itération

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

qui satisfait (1.14), i.e., cette sélection améliore les bornes. Pourtant, il y a bien d'autres règles qui n'utilisent pas explicitement cette propriété. Par exemple :

(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

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

(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

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

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 telle manière que la borne $\beta(M_k) = \min f(T_k)$ soit estimée par des efforts raisonnables.

Définition 1.22 *Soit $\{T_k\}$ une suite de sous ensembles de $\mathbb{R}^n$. Alors*

- $\overline{\lim_{k\to\infty}} T_k := \{x \in \mathbb{R}^n \mid x = \lim_{j\to\infty} x_{n_j}, x_{n_j} \in T_{n_j}\}$,
- $\underline{\lim_{k\to\infty}} T_k := \{x \in \mathbb{R}^n \mid x = \lim_{n\to\infty} x_n, x_n \in T_n$ *pour tout sauf un nombre fini de* $n \in \mathbb{N}\}$,
- $T = \lim_{k\to\infty} T_k \Longleftrightarrow T = \overline{\lim_{k\to\infty}} T_k = \underline{\lim_{k\to\infty}} T_k$.

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 1.3 [60] *Soit $S \in \mathbb{R}^n$ un compact et soit $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ une fonction 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 algorithme de SE va converger plus vite si on peut estimer les 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 coupe, en particulier approximation extérieure (AE), avec la technique par SE. Cette approche paraît très prometteuse. Le premier algorithme de ce type a été proposé par les auteurs de [62] 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.

1.3 Techniques de pénalisation

En optimisation non convexe, on affronte très souvent des problèmes dont une ou plusieurs contraintes sont complexes et difficiles à prendre en compte directement. L'approche par pénalisation est parfois utilisée, du point de vue théorique, pour transformer un tel problème soit en un problème ou une suite de problèmes d'optimisation sans contraintes, soit en un problème dont les propriétés sont mieux comprises ou plus simples à mettre en évidence. Si la pénalisation est bien choisie, dans ces cas, les propriétés recherchées du problème original peuvent être obtenues directement à partir des propriétés trouvées du problème pénalisé ; dans ce cas on parle de *pénalisation exacte*. Sinon, on parle de *pénalisation inexacte* ([50]).

Du point de vue numérique, la pénalisation permet d'utiliser des algorithmes d'optimisation mieux adaptés à la structure du problème pénalisé afin d'obtenir des solutions admissibles. Cette approche est de ce fait très souvent utilisée. Elle permet de trouver une solution de qualité suffisante rapidement sans avoir à entrer dans l'algorithmique sophistiqué de l'optimisation avec contrainte. Pourtant, cette technique ne passe pas partout car la recherche d'une fonction de pénalisation appropriée à un problème n'est pas toujours évidente.

Les différentes techniques de pénalisation relèvent souvent du principe suivant. Considérons le problème d'optimisation (P) avec contraintes :

$$(P) \quad \begin{cases} \min \ f(x) \\ \text{s.t. } g(x) \leq 0, \\ \quad\ h(x) = 0, \\ \quad\ x \in C \subseteq \mathbb{R}^n, \end{cases}$$

où C est un sous-ensemble de $\mathbb{R}^n$ défini par des contraintes qui peuvent être facilement incorporées dans l'optimisation (par exemple contraintes d'égalité linéaire).

Une fonction $p : \mathbb{R}^n \longrightarrow \mathbb{R}$ est appelée *fonction de pénalisation* pour (P) si elle vérifie :

- $p(x) = 0$ si $g(x) \leq 0, h(x) = 0$ et
- $p(x) > 0$ si $g(x) \not\leq 0$ ou $h(x) \neq 0$.

Le programme de pénalisation est donc

$$(P_t) \quad \begin{cases} \min \ f_t(x) = f(x) + tp(x) \\ \text{s.t. } x \in C. \end{cases}$$

Ici, t est un scalaire strictement positif, appelé *facteur de pénalisation*. Le but du terme additionnel est de pénaliser la violation des contraintes (on parle alors de *pénalisation extérieure*) ou l'abord de la frontière du domaine admissible (on parle dans ce cas de *pénalisation intérieure*). La question qui se pose immédiatement est de savoir si en résolvant (P_t) on résout (P). Autrement dit, on cherche à savoir quand les ensembles de solutions de (P) et (P_t) coïncident. Cela va dépendre du choix de la fonction p et du paramètre $t > 0$.

Proposition 1.3 (Monotonie en pénalisation) [50] *Supposons que, pour tout t considéré, (P_t) ait au moins une solution, notée $\overline{x}_t$. Alors, lorsque $t > 0$ croît,*

(i) $p(\overline{x}_t)$ décroît,

(ii) $f(\overline{x}_t)$ croît,

(iii) $f_t(\overline{x}_t)$ croît (si $p(.) \geq 0$).

1.3.1 Pénalisation extérieure

On parle de *pénalisation extérieure* lorsque la fonction de pénalisation p vérifie les propriétés suivantes :

$$\begin{aligned} &(i) \quad p \text{ est s.c.i. sur } \mathbb{R}^n, \\ &(ii) \quad p(x) \geq 0, \forall x \in \mathbb{R}^n, \\ &(iii) \quad p(x) = 0 \Longleftrightarrow x \in C. \end{aligned} \tag{1.33}$$

Le qualificatif « extérieur » vient de la propriété (iii), qui exprime que f_t ne modifie f qu'à l'extérieur de l'ensemble admissible.

Proposition 1.4 (Convergence de la pénalisation extérieure) [50] *Soient C un fermé, non vide de $\mathbb{R}^n$ et $p : \mathbb{R}^n \longrightarrow \mathbb{R}$ une fonction vérifiant (1.33). Supposons que f soit s.c.i et qu'il existe un $t_0 > 0$ tel que $f_{t_0}(x) \to +\infty$ quand $\|x\| \to +\infty$. Alors, on a*

(i) $\forall t \geq t_0$, (P_t) a au moins une solution $\overline{x}_t$,

(ii) la suite $\{\overline{x}_t\}_{t\uparrow\infty}$ est bornée,

(iii) tout point d'adhérence de la suite $\{\overline{x}_t\}_{t\uparrow\infty}$ est solution de (P).

1.3.2 Pénalisation intérieure

Dans certains problèmes, le fait que les itérés $\overline{x}_t$ générés par pénalisation extérieure ne soient pas admissibles peut être un inconvénient, par exemple, parce que f n'est pas défini à l'extérieur de C. On peut introduire des méthodes de pénalisation dans lesquelles les itérés $\overline{x}_t$ restent dans C. On parle alors de pénalisation intérieure. L'idée est d'utiliser un terme de pénalisation p qui tend vers l'infini lorsque x s'approche de la frontière de C.

Supposons que l'intérieur C° de C est non vide : $C^\circ \neq \emptyset$.

La fonction de pénalisation p considérée dans ce paragraphe satisfait les conditions suivantes :

$$\begin{array}{ll} (i) & p \text{ est continue sur } C^\circ, \\ (ii) & p(x) \geq 0, \forall x \in C^\circ, \\ (iii) & p(x) \to +\infty \text{ quand } x \to \partial C, x \in C^\circ. \end{array} \tag{1.34}$$

Le problème de pénalisation est alors

$$(P_t) \quad \begin{cases} \min f_t(x) = f(x) + tp(x) \\ \text{s.t. } x \in C^\circ. \end{cases}$$

La condition (iii) de (1.34) crée une « barrière » au bord de l'ensemble admissible, si bien que f_t porte parfois le nom de *fonction barrière*.

Proposition 1.5 (Convergence de la pénalisation intérieure) [50] *Supposons que f soit continu sur $\mathbb{R}^n$ et que l'ensemble admissible C non vide vérifie*

$$C = \overline{C^\circ}.$$

On suppose également que soit C est borné, soit $f(x) \to +\infty$ quand $\|x\| \to +\infty$. Alors, si p satisfait les conditions (1.34), on a :

(i) $\forall t \geq 0$, (P_t) a au moins une solution $\overline{x}_t$,

(ii) la suite $\{\overline{x}_t\}_{t\downarrow 0}$ est bornée,

(iii) tout point d'adhérence de la suite $\{\overline{x}_t\}_{t\downarrow 0}$ est solution de (P).

1.3.3 Pénalisation exacte en programmation DC

Dans ce paragraphe, nous extrayons des résultats de [94] sur la pénalisation exacte en programmation DC.

Tout d'abord nous présentons la technique de pénalisation exacte pour les programmes non convexes contenant la fonction objectif concave, les fonctions de contraintes convexes polyédrales bornées, et fonctions de contrainte concaves.

Soient K un polyèdre convexe borné non vide de $\mathbb{R}^n$, et f, p deux fonctions concaves finies sur K et continues relativement à K. Rappelons que la fonction f est dite continue relativement à K si sa restriction sur K est continue. Considérons le problème (P) suivant :

$$\alpha = \min\{f(x) \mid x \in K, p(x) = 0\}. \tag{P}$$

Ce problème peut être réécrit d'une façon équivalente sous la forme :

$$\alpha = \min\{f(x) \mid x \in C, p(x) \leq 0\} \tag{P}$$

où C est l'ensemble convexe défini par

$$C := \{x \in K, p(x) \geq 0\}. \tag{1.35}$$

Il est clair que l'ensemble C est convexe polyédral si et seulement si la fonction $(-p)$ est convexe polyédrale sur K.

Le problème de pénalisation (P_t) de (P) est défini par, pour $t > 0$,

$$\alpha(t) = \min\{f(x) + tp(x) \mid x \in K, p(x) \geq 0\}. \tag{P_t}$$

Soient $V(K)$ l'ensemble de sommets de K, et $\mathcal{P}$, $\mathcal{P}_t$ l'ensemble de solutions optimales de (P) et de (P_t) respectivement.

Théorème 1.12 *Soient K un polyèdre convexe borné non vide de $\mathbb{R}^n$, et f, p deux fonctions concaves finies sur K et continues relativement à K. Supposons que l'ensemble réalisable du problème (P) est non vide. Alors il existe $t_0 \geq 0$ tel que pour tout $t > t_0$, les problèmes (P) et (P_t) ont la même valeur optimale et $\mathcal{P} \equiv \mathcal{P}_t$. De plus, on peut prendre $t_0 = \dfrac{f(x^0) - \alpha(0)}{\mu}$ où $\mu = \min\{p(x) \mid x \in V(K),\ p(x) > 0\}$ et $x^0 \in K$ vérifiant $p(x^0) = 0$.*

Considérons ensuite le problème (P') ci-dessous :

$$\alpha' = \min\{f(x) \mid x \in K,\ p(x) \leq 0\}. \tag{P'}$$

Évidemment, (P') est équivalent au problème suivant :

$$\alpha' = \min\{f(x) \mid (x, y) \in K \times [0, \beta],\ p(x) + y = 0\} \tag{$\overline{P'}$}$$

où $\beta \geq \max\{-p(x) \mid x \in K\}$, dans le sens qui suit : *si x^* est une solution optimale de (P') alors $(x^*, -p(x^*))$ résout $(\overline{P'})$. Réciproquement, si (x^*, y^*) est une solution optimale de $(\overline{P'})$ alors x^* résout (P').*

Grâce au théorème 1.12, il existe un nombre $t_0 \geq 0$ tel que pour tout $t > t_0$, le problème $(\overline{P'})$ est équivalent au problème :

$$\min\{f(x) + t(p(x) + y) \mid (x, y) \in K \times [0, \beta], p(x) + y \geq 0\} \tag{$\overline{P'_t}$}$$

Dès lors, le corollaire suivant apparaît :

Corollaire 1.1 *En utilisant les même hypothèses que pour le théorème 1.12, les problèmes $(\overline{P'})$ et $(\overline{P'_t})$ sont alors équivalents.*

En appliquant ce corollaire, nous pouvons citer ci-dessous des programmes non convexes (dont les fonctions objectifs sont DC et les fonctions de contrainte sont aussi DC) fréquemment rencontrés en pratique que l'on peut avantageusement reformuler en programmes DC équivalents plus adaptés (dans le sens donné par Théorème 1.12).

(a) *Fonction objectif convexe polyédrale et fonctions de contrainte concaves*
Soient K un polyèdre convexe de $\mathbb{R}^n$ et $f \in \Gamma_0(\mathbb{R}^n)$ une fonction convexe polyédrale, i.e.,

$$f(x) := \max\{\langle x, a_i\rangle - b_i,\ i = 1, \ldots, r\} + \chi_D(x) = \overline{f}(x) + \chi_D(x),$$

tel que $D \cap K$ est borné non vide. Soit p une fonction concave finie sur $D \cap K$ et continue relativement à $D \cap K$.
Selon le corollaire 1.1, si l'ensemble $\{x \in K, p(x) \leq 0\}$ est non vide, il existe $t_0 \geq 0$ tel que pour tout $t > t_0$, le problème :

$$\min\{f(x) \mid x \in K, p(x) \leq 0\} \tag{1.36}$$

est équivalent au problème suivant :

$$\min\{\overline{f}(x) + t(p(x) + y) \mid (x, y) \in [D \cap K] \times [0, \beta], p(x) + y \geq 0\} \tag{1.37}$$

où $\beta \geq \max\{-p(x) \mid x \in D \cap K\}$.

(b) *Fonction objectif concave et fonctions de contrainte DC dont les premières composantes DC sont convexes polyédrales*
Soit f une fonction concave finie sur un polyèdre convexe K et p une fonction DC. Supposons que $p = g - h$ est une décomposition DC de p où g est une fonction convexe polyédrale définie par

$$g(x) := \max\{\langle x, a_i\rangle - b_i,\ i = 1, \ldots, r\} + \chi_D(x) = \overline{g}(x) + \chi_D(x),$$

telle que $D \cap K$ est borné non vide et h est convexe fini sur $D \cap K$.
Selon le corollaire 1.1, si l'ensemble réalisable du problème

$$\min\{f(x) \mid x \in K, g(x) - h(x) \leq 0\} \tag{1.38}$$

est non vide, il existe $t_0 \geq 0$ tel que pour tout $t > t_0$, ce problème est équivalent au problème suivant :

$$\begin{cases} \min\ f(x) + t(-h(x) + y + z) \\ \text{s.t. } (x, y, z) \in [D \cap K] \times [a, b] \times [0, \beta] \\ \qquad \overline{g}(x) - y \leq 0 \\ \qquad -h(x) + y + z \geq 0 \end{cases} \tag{1.39}$$

où

$$\begin{aligned} &a \leq \min\{g(x) \mid x \in D \cap K\}, \\ &b \geq \max\{g(x) \mid x \in D \cap K\}, \\ &\beta \geq \max\{h(x) - a \mid x \in D \cap K\}. \end{aligned}$$

(c) *Fonction objectif DC et fonctions de contrainte DC dont les premières composantes DC sont convexes polyédrales*
Soit K un polyèdre convexe non vide de $\mathbb{R}^n$, et $f = g_0 - h_0$, $f_1 = g_1 - h_1$ des fonctions DC polyédrales. g_0, g_1 sont des fonctions convexes polyédrales sur $\mathbb{R}^n$,

$$g_0(x) := \max\{\langle x, a_i^0\rangle - b_i^0,\ i = 1, \ldots, r\} + \chi_D(x) = \overline{g_0}(x) + \chi_D(x),$$

$$g_1(x) := \max\{\langle x, a_i^1\rangle - b_i^1,\ i = 1, \ldots, s\} + \chi_S(x) = \overline{g_1}(x) + \chi_S(x).$$

avec D, S étant polyèdres convexes non vides. h_0, h_1 sont des fonctions convexes finies sur $D \cap S \cap K$ et continues relativement à $D \cap S \cap K$. Supposons que $D \cap S \cap K$ est non vide borné. Considérons le problème :

$$\min\{g_0(x) - h_0(x) \mid x \in K,\ g_1(x) - h_1(x) \leq 0\}. \tag{1.40}$$

Basé sur le corollaire 1.1, il existe $t_0 \geq 0$ tel que pour tout $t > t_0$, le problème (1.40) équivaut au problème ci-dessous :

$$\begin{cases} \min\ \overline{g_0}(x) - h_0(x) + t(-h_1(x) + y + z) \\ \quad \text{s.t. } (x, y, z) \in [D \cap S \cap K] \times [c, d] \times [0, \beta] \\ \qquad \overline{g_1}(x) - y \leq 0 \\ \qquad -h_1(x) + y + z \geq 0 \end{cases} \tag{1.41}$$

où c, d, β sont des nombres satisfaisant

$$\begin{aligned} &c \leq \min\{\overline{g_1}(x) \mid x \in D \cap S \cap K\}, \\ &d \geq \max\{\overline{g_1}(x) \mid x \in D \cap S \cap K\}, \\ &\beta \geq \max\{h_1(x) - c \mid x \in D \cap S \cap K\}. \end{aligned}$$

Une application importante de cette partie est concernée à la programmation en variables mixtes 0-1 dont la fonction objectif est concave.

Considérons le programme en variables mixtes 0-1 suivant :

$$\min\{f(x, y) \mid (x, y) \in S,\ x \in \{0, 1\}^n\} \tag{1.42}$$

où f est concave sur un polyèdre convexe borné non vide $S \subset \mathbb{R}^{n+m}$.

On définit la fonction $p : \mathbb{R}^n \times \mathbb{R}^m \longrightarrow \mathbb{R}$ par :

- soit $p(x, y) = \sum_{i=1}^{n} x_i(1 - x_i)$;
- soit $p(x, y) = \sum_{i=1}^{n} \min\{x_i, 1 - x_i\}$.

Il est clair que la fonction p définie au-dessus est concave, finie sur $K = \{(x, y) \in S \mid x \in [0, 1]^n\}$ et continue relativement à K. De plus, p est non négative sur K et

$$\{(x, y) \in S \mid x \in \{0, 1\}^n\} \equiv \{(x, y) \in K \mid p(x, y) = 0\}.$$

Le problème (1.42) peut être récrit comme suit

$$\min\{f(x, y) \mid (x, y) \in K,\ p(x, y) = 0\} \tag{1.43}$$

Selon le théorème 1.12, pour t suffisamment grand, (1.42) est équivalent au problème DC suivant :

$$\min\{f(x, y) + tp(x, y) \mid (x, y) \in K\}. \tag{1.44}$$

1.3.4 Technique de pénalisation en programmation avec des variables mixtes entières

Dans ce paragraphe, nous présentons un nouveau résultat sur la technique de pénalisation appliquée au programme d'optimisation en variables mixtes entières.

Considérons le problème d'optimisation en variables mixtes entières ci-dessous :

$$\alpha = \min\{f(x,y) \mid (x,y) \in S,\ x \in [l,u],\ x \in \mathbb{Z}^n,\ y \in \mathbb{R}^m\} \qquad (MIP)$$

où S est un polyèdre convexe borné non vide de $\mathbb{R}^{n+m}$, la fonction objectif f est supposée concave finie sur $K = \{(x,y) \in S \mid x \in [l,u],\ y \in \mathbb{R}^m\}$ et continue relativement à K, $l = (l_1,\ldots,l_n), u = (u_1,\ldots,u_n)$, $l,u \in \mathbb{Z}^n$ et $l_i < u_i,\ \forall i = 1,\ldots,n$.

Nous étudions dans la suite la pénalisation pour (MIP).

Pénalisation exacte pour (MIP)

Premièrement, on considère la fonction p définie sur $\mathbb{R}^{n+m}$ comme suit :

$$p(x,y) = \sum_{i=1}^{n} |\sin(\pi x_i)| \qquad (1.45)$$

La fonction p vérifie des propriétés suivantes :
- $0 \le p(x,y) \le n,\ \forall (x,y) \in \mathbb{R}^{n+m}$;
- $p(x,y) \le 0 \Longleftrightarrow p(x,y) \equiv 0 \Longleftrightarrow x \in \mathbb{Z}^n$;
- p est finie sur K et continue relativement à K ;
- pour tout fermé de la forme $[a,a+1]$ où $a \in \mathbb{Z}^n$, la fonction p est concave sur $\{(x,y) \in S \mid x \in [a,a+1],\ y \in \mathbb{R}^m\}$.

De plus, l'ensemble $[l,u]$ peut être exprimé par réunion des fermés comme :

$$[l,u] = \bigcup [a,a+1],\ a = (a_1,\ldots,a_n) \text{ tel que } a_i \in [l_i, u_i - 1] \cap \mathbb{Z}.$$

Cette réunion finie se compose de $\prod_{i=1}^{n} m_i$ éléments où $m_i = u_i - l_i,\ \forall i$. Noter B_j ces éléments, $j \in J = \left\{1,\ldots,\prod_{i=1}^{n} m_i\right\}$, J l'ensemble d'indices et $K_j = \{(x,y) \in S \mid x \in B_j, y \in \mathbb{R}^m\}$. Chaque B_j de la forme $[a,a+1]$ est un polyèdre convexe borné non vide de $\mathbb{R}^n$.

Par conséquent, le problème (MIP) s'écrit comme suit :

$$\begin{aligned}
\alpha &= \min\{f(x,y) \mid (x,y) \in K,\ x \in \mathbb{Z}^n\} \\
&= \min\{f(x,y) \mid (x,y) \in K,\ p(x,y) = 0\} \\
&= \min\{f(x,y) \mid (x,y) \in \bigcup_{j \in J} K_j,\ p(x,y) = 0\} \\
&= \min_{j \in J} \{\alpha_j = \min\{f(x,y) \mid (x,y) \in K_j,\ p(x,y) = 0\}\}.
\end{aligned}$$

En appliquant le théorème 1.12, pour chaque problème (P_j) suivant,

$$\alpha_j = \min\{f(x,y) \mid x \in K_j,\ p(x,y) = 0\}, \qquad (P_j)$$

comme f est supposée concave finie sur K, il existe $t_j \ge 0$ tel que pour tout $t > t_j$, (P_j) est équivalent au problème (P_j^t) suivant :

$$\alpha_j^t = \min\{f(x,y) + tp(x,y) \mid (x,y) \in K_j\}, \qquad (P_j^t)$$

Notez bien que cette équivalence assure l'égalité $\alpha_j = \alpha_j^t$ et aussi l'égalité de deux ensembles des solutions optimales de (P_j) et (P_j^t).

Dès lors, si l'on choisit t tel que $t > \max\{t_j,\ j \in J\}$ alors le problème (MIP) équivaut au problème ci-dessous :

$$\min_{j\in J}\left\{\alpha_j^t = \min\{f(x,y) + tp(x,y) \mid (x,y) \in K_j\}\right\} \tag{1.46}$$

qui est exactement

$$\alpha^t = \min\{f(x,y) + tp(x,y) \mid (x,y) \in K\} \tag{$\overline{MIP}$}$$

dans le sens donné par le théorème 1.12. On obtient donc la pénalisation exacte pour (MIP) en utilisant la fonction de pénalisation $p(x,y) = \sum_{i=1}^{n} |\sin(\pi x_i)|$.

Il est aussi important de voir que, pour tout i, comme $|\sin(\pi x_i)|$ est une fonction DC avec, par exemple, une décomposition

$$|\sin(\pi x_i)| = \max\{\eta x_i^2 + \sin(\pi x_i), \eta x_i^2 - \sin(\pi x_i)\} - (\eta x_i^2)$$

où $\eta > \dfrac{\pi^2}{2}$ (car si $\eta > \dfrac{\pi^2}{2}$ alors $(\eta x_i^2 + \sin(\pi x_i))$ et $(\eta x_i^2 - \sin(\pi x_i))$ sont convexes sur $\mathbb{R}$). On en déduit que p est une fonction DC. Ce qui entraîne que $(\overline{MIP})$ est un programme DC.

Cette pénalisation reste encore exacte si la fonction objectif f de (MIP) est supposée convexe polyédrale. La démonstration est similaire à celle présentée dans la section 2 de [94].

Considérons ensuite la fonction $q : \mathbb{R}^{n+m} \longrightarrow \mathbb{R}$ définie par

$$q(x,y) = \sum_{i=1}^{n} \max_{a_j}\{(x_j - a_j)(1 + a_j - x_j) \mid a_j \in [l_j, u_j - 1] \cap \mathbb{Z}\}. \tag{1.47}$$

La fonction q vérifie des propriétés suivantes :

- $0 \le q(x,y) \le n,\ \forall (x,y) \in \mathbb{R}^{n+m}$;
- $q(x,y) \le 0 \Longleftrightarrow q(x,y) \equiv 0 \Longleftrightarrow x \in [l,u] \cap \mathbb{Z}^n$;
- q est une fonction DC dont la première composante est convexe polyédrale car $q = \phi - \psi$ où

$$\begin{aligned} \phi(x,y) &:= \sum_{j=1}^{n} \max_{a_j}\{(2a_j + 1)x_j - a_j(a_j + 1) \mid a_j \in [l_j, u_j - 1] \cap \mathbb{Z}\}; \\ \psi(x,y) &:= \frac{\eta}{2} x^T x \text{ avec } \eta = 2. \end{aligned} \tag{1.48}$$

En conséquence, le problème (MIP) peut être récrit équivalent à

$$\begin{aligned} \alpha &= \min\{f(x,y) \mid (x,y) \in K,\ x \in [l,u], x \in \mathbb{Z}^n\} \\ &= \min\{f(x,y) \mid (x,y) \in K,\ q(x,y) \le 0\} \\ &= \min\{f(x,y) \mid (x,y) \in K,\ \phi(x,y) - \psi(x,y) \le 0\} \end{aligned}$$

En appliquant le résultat présenté dans la section 1.3.3, on voit clairement que si la fonction objectif f est concave ou si f est DC avec la première composante convexe polyédrale, le problème (MIP) peut être transformé en un programme DC équivalent satisfaisant : la fonction objectif et la fonction de contrainte sont DC dont les premières composantes sont convexes polyédrales. Ceci montre que l'on obtient la pénalisation exacte pour (MIP) en utilisant q avec ses composantes DC présentées dans (1.48).

Pénalisation inexacte pour (MIP)

Considérons maintenant le cas plus général du problème (MIP) où la fonction objectif f ne satisfait pas des hypothèses mentionées dans la section 1.3.3 (par exemple, f est DC mais ses composantes DC ne sont pas polyédrales). Dans ce cas là, malgré le fait que l'on n'a pas de pénalisation exacte mais on peut toujours remplacer la contrainte intégrale par une contrainte DC et construire un problème de pénalisation de (MIP) à l'aide des fonctions de pénalisation p et q. De plus, une des fonctions q_1, q_2 ci-dessous peut être utile :

$$q_1(x,y) = \sum_{i=1}^{n} \sin^2(\pi x_i), \tag{1.49}$$

$$q_2(x,y) = \sum_{i=1}^{n} \min_{a_j}\{(x_j - a_j)^2 \mid a_j \in [l_j, u_j] \cap \mathbb{Z}\}. \tag{1.50}$$

car q_1, q_2 vérifient aussi des propriétés :

- $0 \leq q_1(x,y) \leq n,\ \forall (x,y) \in \mathbb{R}^{n+m}$;
- $0 \leq q_2(x,y) \leq (n/4),\ \forall (x,y) \in \mathbb{R}^{n+m}$;
- $q_1(x,y) \leq 0 \Longleftrightarrow q_1(x,y) \equiv 0 \Longleftrightarrow x \in \mathbb{Z}^n$;
- $q_2(x,y) \leq 0 \Longleftrightarrow q_2(x,y) \equiv 0 \Longleftrightarrow x \in [l,u] \cap \mathbb{Z}^n$;
- q_1 est une fonction DC avec la décomposition DC : $q_1 = \phi_1 - \psi_1$ où

$$\begin{aligned} \phi_1(x,y) &:= \frac{\eta}{2} x^T x \text{ avec } \eta \geq 2\pi^2; \\ \psi_1(x,y) &:= \sum_{j=1}^{n} \left(\frac{\eta}{2} x_j^2 - \sin^2(\pi x_j) \right). \end{aligned} \tag{1.51}$$

- q_2 est une fonction DC polyédrale avec la décomposition DC : $q_2 = \phi_2 - \psi_2$ où

$$\begin{aligned} \phi_2(x,y) &:= \frac{\eta}{2} x^T x \text{ avec } \eta = 2; \\ \psi_2(x,y) &:= \sum_{j=1}^{n} \max_{a_j}\{2a_j x_j - a_j^2 \mid a_j \in [l_j, u_j] \cap \mathbb{Z}\}. \end{aligned} \tag{1.52}$$

Dès lors, en utilisant une des fonctions p, q, q_1, q_2, on peut toujours transformer un problème contenant des variables mixtes entières qui est combinatoire en un programme continu avec une contrainte DC.

1.4 Conclusion

Nous avons présenté dans ce chapitre les fondements de la Programmation DC et DCA, nécessaires aux développements des travaux de recherche dans cette thèse concernant la classe NP-difficile des problèmes d'optimisation non convexes non linéaires en variables mixtes entières. Les techniques les plus populaires, que sont celles de Séparation et Evaluation (SE), ou Branch-and-Bound (B&B) en anglais, sont aussi décrites avec détails pour des utilisations effectives. Etant bien entendu que B&B, de par son coût exhorbitant en temps de calcul, n'est applicable que pour des dimensions relativement faibles, comme le montrent les simulations numériques dans les chapitres suivants. De nouveaux résultats utiles pour la pénalisation des contraintes intégrales (« entières ») sont abordés pour la reformulation

des programmes non convexes non linéaires mixtes entiers comme des programmes DC. Il est aussi à noter que l'exactitude de certaines fonctions de pénalisation est obtenue grâce à des résultats généraux de la pénalisation exacte en programmation DC.

Chapitre 2

Programmation en variables mixtes entières

Nous développons dans ce chapitre de nouvelles méthodes pour la résolution d'une classe NP-difficile des problèmes non convexes non linéaires en variables mixtes entières. Le problème considéré est d'abord reformulé comme un programme DC à l'aide de la technique de pénalisation en programmation DC. Ensuite DCA (l'Algorithme DC), une approche locale prometteuse pour la programmation non convexe, est développée pour résoudre efficacement le problème DC résultant. Une méthode par Séparation et Evaluation est également mise en place dans laquelle la technique de relaxation DC est utilisée pour les bornes inférieures. Les méthodes proposées sont appliquées à la résolution du problème de gestion de portefeuille discrète où le coût de transaction concave est considéré. Des résultats préliminaires présentés en comparant avec ceux fournis par deux solveurs LaGO et COUENNE montrent l'efficacité de nos méthodes.

2.1 Introduction

The general form of a mixed-integer nonlinear program (MINLP) is the following :

$$\begin{cases} \min \ f(x) \\ \text{s.t. } g_k(x) \leq 0, \ k = 1, \ldots, m, \\ \quad x \in X, \\ \quad x_j \in \mathbb{Z}, j = 1, \ldots, n_1, \end{cases} \qquad (MINLP)$$

where $f, g_k : \mathbb{R}^n \longrightarrow \mathbb{R}, k = 1, \ldots, m$, are arbitrary functions, $X \subset \mathbb{R}^n$ is a polyhedral convex set and $n_1 \leq n$.

The class of MINLPs contains as special cases mixed-integer linear programs (MILPs) and nonlinear programs (NLPs), thus MINLP problems are NP-hard (for a detailed discussion on the complexity of MINLPs we refer the reader to [57, 82]).

We distinguish MINLPs into two classes. If the functions $f, g_1, \ldots, g_m$ are all convex, the MINLP is called convex ; otherwise, it is called nonconvex. Although both kinds of MINLPs are NP-hard but solving nonconvex MINLPs is much more difficult than solving

convex ones. Let consider the continuous relaxation of the MINLP problem by relaxing the integrality condition of x :

$$\begin{cases} \min \ f(x) \\ \text{s.t. } g_k(x) \leq 0, \ k = 1, \ldots, m, \\ \quad x \in X. \end{cases} \tag{2.1}$$

In the convex case, (2.1) is itself convex while in contrast, this relaxation problem is still nonconvex in the second case and it is unlikely to be solved easily. Some exact methods for convex MINLPs can be listed, for example, generalized Benders' decomposition (see e.g., [48]), branch and bound (see e.g., [54]), outer-approximation (see e.g., [37]), LP/NLP based branch and bound (see e.g., [134]), cutting plane method (see e.g., [155]), branch and cut (see e.g., [148]) and the hybrid methods (see e.g., [4, 19]).

For general nonconvex MINLPs, most of the algorithms for finding its globally optimal solutions are based on branch and bound. We can summarize below specific algorithms. First of all we mention to spatial branch-and-bound method (applied to [99,144]). As in the branch-and-bound method for mixed-integer linear programs [83], in branching procedure, a subproblem can be pruned if either it is infeasible, or the best current upper bound cannot be improved at this node. The branch-and-reduce technique (as mentioned in [11,139]) is an improved technique of spatial branch-and-bound. This technique is usually performed using LP relaxations rather than other complex convex relaxation techniques. Another improved version of spatial branch-and-bound algorithm for global optimization of nonconvex nonlinear programs which was proposed in [7] is α-Branch-and-Bound (α-BB). This method is based on the construction of underestimators and is extended to the mixed-integer nonlinear problems in [5]. Several other exact approaches such as outer-approximation algorithms for separable nonconvex MINLPs in [71], Lagrangian based branch and cut algorithm for global optimization of non-convex MINLPs with decomposable structures [68] were presented. It is clear that an exact algorithm applied to MINLPs with the guarantee of the global optimality of solutions computed is very expensive, especially in large-scale problems. Besides these exact methods, some heuristic methods which design to find rapidly good feasible solutions but without guarantee for their optimality have been proposed. The heuristic developed in [101] combines branch and bound with sequential quadratic programming for solving MINLPs. Other heuristic methods for MINLPs have been presented in [32, 116].

In order to attack the mixed-integer nonlinear problems, some of available software packages have been developed (several solvers can solve nonconvex MINLPs to proven optimality, others can be used to find heuristic solutions) : BARON (due to [138, 139]), α-BB (introduced in [5]), COUENNE (developed by [10]), BONMIN (described in [19]), DICOPT (presented in [52, 73]), LaGO (described in [118]),... A survey and an annotated bibliography of nonconvex mixed-integer nonlinear programs can be found in [21].

In this chapter, we are just interested in a special case of nonconvex MINLPs in which the objective function f is a DC (Difference of convex functions) function. Precisely, let us consider the function f defined by $f(x) := g(x) - h(x)$ where $g(x) := x^T Q x$, $Q \in \mathbb{R}^{n \times n}$ is a positive semi-definite symmetric matrix, h is convex separable, say $h(x) = \sum_{i=1}^{n} h_i(x_i)$ with h_i being convex differentiable functions on $\mathbb{R}$. The nonconvex mixed-integer nonlinear

problem considered is the following :

$$\begin{cases} \min f(x) = g(x) - h(x), \\ \text{s.t. } Ax \leq b, \\ \quad l \leq x \leq u, \\ \quad x_j \in \mathbb{Z}, j = 1, \ldots, n_1, \end{cases} \tag{NIP}$$

where $l, u \in \mathbb{Z}^n, l_i \leq u_i, \forall i = 1, \ldots, n$, $A \in \mathbb{R}^{m \times n}$ is a nonnegative matrix, $b \in \mathbb{R}^m$, $n_1 \leq n$ and the feasible set of (NIP) is nonempty bounded set : $X = \{x \in \mathbb{R}^n \mid Ax \leq b,\ l \leq x \leq u,\ x_j \in \mathbb{Z}, j = 1, \ldots, n_1\} \neq \emptyset$. The separability of the function h does not intervene in the investigation of DC programming and DCA for (NIP). It is used only, in DC relaxation technique, for explicitly computing a convex minorant of the objective function f on the convex set $\{x \in \mathbb{R}^n \mid Ax \leq b,\ l \leq x \leq u\}$ (see section 2.3). The latter serves to lower bounding in Branch-and-Bound (B&B) scheme.

It is clear that if the symmetric matrix Q is indefinite then, using the decomposition $Q = (Q + \rho I) - \rho I$ with $\rho \geq -\lambda_1(Q)$, $\lambda_1(Q)$ being the smallest eigenvalue of Q, we can reformulate its corresponding problem in the form of (NIP) with Q replaced by $Q + \rho I$. It is worth noting that if the objective function f is twice continuously differentiable on a open convex set Ω containing X, then for every $\rho \geq \max\{\|\nabla^2 f(x)\| \mid x \in F\}$, where $F \subset \Omega$ is a bounded closed convex set such that its interior contains K. It is easy to see that such F exists and the function f is a DC function with the following DC components

$$g(x) = \left[\frac{\rho}{2}\|x\|^2 + f(x)\right], \qquad h(x) := \frac{\rho}{2}\|x\|^2,$$

that satisfies the separability of the function h.

The problem (NIP) is very difficult, due to the double nonconvexity of the objective function and of the feasible set. When h is a linear function, say $h(x) = q^T x = \sum_{j=1}^{n} q_j x_j$, (NIP) becomes a convex mixed-integer quadratic program which is still difficult even though the objective function is convex. In general case, it is a great challenge to develop deterministic methods for solving large-scale (NIP). The approach presented in this work is certainly a step in this direction.

In this study, we first use penalty techniques in DC programming to recast the nonconvex mixed-integer nonlinear program (NIP) into the DC programming (continuous) framework, and then develop suitable DCA (DC Algorithms) for its solutions. Although DCA is a local method, it is proved to be more robust and more efficient than related standard methods for nonconvex programs, especially in large-scale settings (see [84] and references therein for more details). With good initial points and appropriate DC decompositions, DCA converges very fast to fairly good solutions. A customized (B&B) scheme, using DC relaxation techniques for lower bounding, is also introduced for checking globality of solutions computed by DCA. Finally, computational experiments related to discrete portfolio optimization under concave transaction costs are reported, which show the inexpensiveness (and so the scalability), efficiency and globality of solutions computed by DCA. As for global algorithms, B&B provides always global solutions and is by far the best compared to the two standard solvers LaGO and COUENNE [1,3,10,118]). However, as expected, B&B is too expensive and cannot handle the large-scale setting.

The rest of the chapter is organized as follows. The next section 2.2 concerns the DC reformulation of our model in the DC programming framework and the description of

DCA for solving the resulting DC program. Section 2.3 is devoted to the Branch-and-Bound scheme with DC relaxation techniques for lower bounding. Numerical simulations related to the discrete portfolio optimization under concave transaction costs are presented in section 2.4. The chapter ends with some conclusions.

2.2 DC reformulation of (NIP) and DCA for solving the penalized problem

The problem (NIP) is rewritten as follows :

$$\alpha = \min \left\{ f(x) = x^T Qx - \sum_{i=1}^{n} h_i(x_i) \mid x \in B \cap K,\ x_j \in \mathbb{Z}, j = 1, \ldots, n_1 \right\}, \qquad (NIP)$$

where $B = [l, u] = \prod_{i=1}^{n} [l_i, u_i] \subset \mathbb{R}^n$, $K = \{x \in \mathbb{R}^n \mid Ax \leq b\}$. Under the assumptions mentioned above, the objective function f is a DC function with explicit DC decomposition.

For applying DCA, we will reformulate the problem (NIP) as a DC program by using penalty techniques in DC programming.

Consider now a continuous function $p : \mathbb{R}^n \longrightarrow \mathbb{R}$ verifying $p(x) \geq 0, \forall x \in B$ and $\{x \in B,\ p(x) \equiv 0\} \Longleftrightarrow \{x \in B,\ x_j \in \mathbb{Z}, j = 1, \ldots, n_1\}$. There are some ways to choose such a function p :

- **Case 1 :** $p(x) := p_1(x) = \sum_{j=1}^{n_1} \sin^2(\pi x_j)$ (see [47, 117]),
- **Case 2 :** $p(x) := p_2(x) = \sum_{j=1}^{n_1} \min_{a_j} \left\{ (x_j - a_j)^2 \mid a_j \in [l_j, u_j] \cap \mathbb{Z} \right\}$.

It is easy to see that each of function p defined above is DC, $p = \phi - \psi$, with the following DC components :

- For $p(x) := p_1(x) = \sum_{j=1}^{n_1} \sin^2(\pi x_j)$,

$$\phi(x) := \left[\frac{\eta}{2} \sum_{j=1}^{n_1} x_j^2 \right] \text{ with } \eta \geq 2\pi^2; \quad \psi(x) := \sum_{j=1}^{n_1} \left(\frac{\eta}{2} x_j^2 - \sin^2(\pi x_j) \right);$$

- For $p(x) := p_2(x) = \sum_{j=1}^{n_1} \min_{a_j} \left\{ (x_j - a_j)^2 \mid a_j \in [l_j, u_j] \cap \mathbb{Z} \right\}$,

$$\phi(x) := \left[\frac{\eta}{2} \sum_{j=1}^{n_1} x_j^2 \right] \text{ with } \eta = 2; \quad \psi(x) := \sum_{j=1}^{n_1} \max_{a_j} \{ 2a_j x_j - a_j^2 \mid a_j \in [l_j, u_j] \cap \mathbb{Z} \}.$$

This DC decomposition of $p(x) := p_2(x)$ shows that in this choice p is a DC polyhedral function.

Exact penalty related to p occurs if the penalty problem (NIP_t) of (NIP) and itself are equivalent in the sense that : there is $t_0 \geq 0$ such that for every $t > t_0$, (NIP_t) and (NIP) have the same optimal value and the same solution set. The question concerning

exact penalty related to p_1 and p_2 remains open. Let us mention the exact penalty of the function p_3 defined by

$$p_3(x) = \sum_{j=1}^{n_1} |\sin(\pi x_j)|\,, \tag{2.2}$$

whose proof relies on the general result concerning polyhedral DC programs with concave penalty functions (see Chapter 1).

In this work, we choose the DC function $p(x) := p_1(x) = \sum_{j=1}^{n_1} \sin^2(\pi x_j)$ with the above mentioned DC decomposition to construct a penalty function for integrality constraints. Then (NIP) takes the form

$$\alpha = \min\{f(x) \mid x \in B \cap K,\ p(x) \leq 0\} \tag{2.3}$$

With $t > 0$, we define :

$$F_t(x) = f(x) + tp(x) \tag{2.4}$$

and consider the penalized problem of (NIP) :

$$\alpha(t) = \min\{F_t(x) \mid x \in B \cap K\}. \tag{NIP_t}$$

Clearly, (NIP_t) is a DC program with the following DC decomposition for $F_t : F_t(x) = G(x) - H(x)$ with

$$\begin{aligned} G(x) &= \frac{1}{2}x^T(2Q + t\eta D)x, \\ H(x) &= \sum_{i=1}^{n} h_i(x_i) + \sum_{j=1}^{n_1}\left[\frac{t\eta}{2}x_j^2 - t\sin^2(\pi x_j)\right], \end{aligned}$$

where $D = (d_{i,j}) \in \mathbb{R}^{n\times n}$ is the diagonal matrix such that $d_{j,j} = 1, \forall j \leq n_1$, $d_{j,j} = 0, \forall j > n_1$.

According to Chapter 1, section 1.1, applying DCA to problem (NIP_t) consists of constructing two sequences $\{x^k\}$ and $\{y^k\}$,

$$y^k \in \partial H(x^k); \quad x^{k+1} \in \partial G^*(y^k).$$

More precisely, at each iteration $k = 0, 1, \ldots$

$$y^k \in \partial H(x^k) \Longleftrightarrow \begin{cases} y_j^k = h_j'(x_j^k) + t\eta x_j^k - \pi t \sin(2\pi x_j^k),\ \forall j \leq n_1, \\ y_j^k = h_j'(x_j^k),\ \forall j > n_1, \end{cases} \tag{2.5}$$

and

$$x^{k+1} \in \partial G^*(y^k) \Longleftrightarrow x^{k+1} \in \arg\min\{G(x) - \langle y^k, x\rangle \mid x \in B \cap K\} \tag{2.6}$$

(2.6) is a convex quadratic program which can be solved via CPLEX [2].

Relationship between solutions of (NIP_t) and (NIP) is given in the following theorem :

Theorem 2.1 *Let ϵ be a real number, $0 < \epsilon < \frac{1}{2}$. There exists $t > 0$ (which depends on ϵ), large enough, such that if $\widetilde{x}$ is a global solution of (NIP_t) and $\widetilde{x}$ is in a ϵ-neighborhood of a point $x^* \in B \cap K$ satisfying $x_j^* \in \mathbb{Z}, \forall j = 1, \ldots, n_1$, then x^* will be a global solution of (NIP).*

The proof of this theorem is similar to that in [47].

We describe now the DCA algorithm applied to (NIP_t).

Algorithm 1 (DCA applied on (NIP_t))

- ***Initialization :***
  Choose an initial point $x^0 \in \mathbb{R}^n$.
  Let θ_1 and θ_2 be sufficiently small positive numbers.
  Let t be a real positive number ; $k \longleftarrow 0$.
- ***Repeat :***
  $\diamond$ Calculate $y^k \in \partial H(x^k)$ via (2.5) ;
  $\diamond$ Solve the problem (2.6) to obtain x^{k+1} ;
  $\diamond$ $k \longleftarrow k+1$.
- ***Until :*** IF $\|x^k - x^{k+1}\| \leq \theta_1$ or $|F_t(x^k) - F_t(x^{k+1})| \leq \theta_2(1 + |F_t(x^k)|)$ then **STOP** : x^{k+1} is a computed solution of (NIP_t).
  If $x_j^{k+1} \in \mathbb{Z}, \forall j \leq n_1$, then $x^* = x^{k+1}$ is a computed solution of (NIP). Otherwise, from x^{k+1}, get a feasible solution $\widehat{x}$ of (NIP) satisfying $\widehat{x}_j \in \mathbb{Z}, \forall j \leq n_1$. Because the matrix A is nonnegative, we can choose $\widehat{x}_j = \lfloor x_j^{k+1} \rfloor, \forall j \leq n_1$, and $\widehat{x}_j = x_j^{k+1}, \forall j > n_1$.

There is no common rule for the choice of the starting point x^0 for Algorithm 1 but if we find a good one, Algorithm 1 will converge to a fairly good solution.

2.3 A Branch-and-Bound algorithm using DC relaxation techniques

In Algorithm 1 displayed above, a computed solution x^* of (NIP_t) could be not satisfied $x_j^* \in \mathbb{Z}, \forall j \leq n_1$, but we can find easily a feasible solution of (NIP) from x^*. In this section, we present a Branch-and-Bound method for globally solving the nonconvex mixed-integer nonlinear problem (NIP). In our proposed algorithm, the DC relaxation technique in DC programming serves to compute lower bounding while the largest distance bisection is used for branching procedure.

2.3.1 DC relaxation technique

Relaxation techniques play a crucial role in lower bounding process. These techniques are used in the B&B scheme, outer-approximation procedures, etc. Based on [128] (and references therein), we outline here the DC relaxation technique in DC programming.

Convex hull of nonconvex functions

Consider the nonconvex program

$$\alpha := \inf\{\phi(x) \mid x \in \mathbb{R}^n\}, \tag{2.7}$$

where $\phi : \mathbb{R}^n \longrightarrow \mathbb{R} \cup \{+\infty\}$ is proper and has an affine minorant on $\mathbb{R}^n$. The optimal way to convexify (2.7) passes by the convex hull of ϕ defined by (see [58, 136]) :

$$\text{co } \phi(x) := \inf\left\{\sum_i \lambda_i \phi(x_i) \mid \lambda_i \geq 0, \sum_i \lambda_i = 1, x \in \text{dom } \phi, x = \sum_i \lambda_i x_i\right\}, \tag{2.8}$$

where the infimum is taken over all representations of x as a convex combination of elements x_i, such that only finitely many coefficients λ_i are nonzero. The convex function co ϕ with

$$\text{dom co } \phi = \text{co dom } \phi, \tag{2.9}$$

is the greatest convex function majorized by ϕ. It leads to the convex programs with the same optimal value

$$\alpha := \inf\{\text{co } \phi(x) \mid x \in \mathbb{R}^n\} = \inf\{\overline{\text{co}}\ \phi(x) \mid x \in \mathbb{R}^n\}. \tag{2.10}$$

It is well known in [58] that :

(i) $\arg\min \phi \subset \arg\min \text{ co } \phi \subset \arg\min \overline{\text{co}}\ \phi$,

(ii) co $(\arg\min \phi) \subset \overline{\text{co}}\ (\arg\min \phi) \subset \arg\min \overline{\text{co}}\ \phi$,

(iii) $\overline{\text{co}}\ \phi = \phi^{**}$,

(iv) If, in addition, ϕ is lower semicontinuous and 1-coercive (the latter means $\lim\limits_{\|x\|\to+\infty} \frac{\phi(x)}{\|x\|} = +\infty$), then co $\phi = \overline{\text{co}}\ \phi = \phi^{**}$.

Finding the convex hull of a nonconvex function is in general very difficult, except for those of concave functions over bounded polyhedral convex sets (polytopes). One seeks instead some convex relaxations more tractable to compute lower bounds for the optimal value α, as DC relaxations presented below.

Convex hull of concave functions over bounded polydedral convex sets

Let S be a nonempty bounded polyhedral convex set whose vertex set is $V(S) := \{v^1, \ldots, v^m\}$. Then $S = \text{co } V(S)$. The vertices $v^1, \ldots, v^m$ are said affinely independent (see [58, 136]) if there are no real numbers $\lambda_i, i = 1, \ldots, m$ not all zero such that

$$\sum_{i=1}^{m} \lambda_i = 0 \text{ and } \sum_{i=1}^{m} \lambda_i v^i = 0. \tag{2.11}$$

In this case S is called an $(m-1)$-simplex and every $x \in S$ is uniquely expressible as convex combination of $v^1, \ldots, v^m$. If ϕ is a finite concave on S then the expression (2.8) for $\text{co}_S\ \phi$ becomes simpler and computable.

Theorem 2.2 *If ϕ is a finite concave on S, there hold*

(i) $co_S\ \phi$ is the polyhedral convex function on S defined by

$$co_S\ \phi(x) = \min\left\{\sum_{i=1}^{m} \lambda_i \phi(v^i) \mid \lambda_i \geq 0, \sum_{i=1}^{m} \lambda_i = 1, x = \sum_{i=1}^{m} \lambda_i v^i\right\}. \tag{2.12}$$

Moreover $co_S\ \phi$ and ϕ agree on $V(S)$.

(ii) If S is an $(m-1)$-simplex, then $co_S\ \phi$ is the affine function determined by

$$co_S\ \phi(x) = \sum_{i=1}^{m} \lambda_i \phi(v^i), \text{ with } \lambda_i \geq 0, \sum_{i=1}^{m} \lambda_i = 1, x = \sum_{i=1}^{m} \lambda_i v^i. \tag{2.13}$$

Convex hull of separable function

Let $\phi = (\phi_1, \ldots, \phi_m)$ be a separable function on $C = \prod_{i=1}^{m} C_i$ with $C_i \subset \text{dom } \phi_i \subset \mathbb{R}^{n_i}$, $i = 1, \ldots, m$, i.e.,

$$\phi(x) = \sum_{i=1}^{m} \phi_i(x_i), \ \forall x = (x_1, \ldots, x_m) \in C, \tag{2.14}$$

then $\text{co}_C\ \phi$ can be computed explicitly from the $\text{co}_{C_i}\ \phi_i$, $i = 1, \ldots, m$.

Proposition 2.1 *If for $i = 1, \ldots, m$, ϕ_i is minorized on C_i by an affine function, then for every $x = (x_1, \ldots, x_m) \in C$*

$$co_C\ \phi(x) \geq \sum_{i=1}^{m} co_{C_i}\ \phi_i(x_i) \geq \sum_{i=1}^{m} \overline{co}_{C_i}\ \phi_i(x_i) = \sum_{i=1}^{m} (\phi_i + \chi_{C_i})^{**}(x_i) = \overline{co}_C\ \phi(x). \tag{2.15}$$

Convex minorant of DC function over bounded closed convex sets

We are now in a position to build convex minorant for the objective DC function in DC programs with the explicit constraint C (a nonempty bounded closed convex set in $\mathbb{R}^n$) and $\varphi, \psi \in \Gamma_0(\mathbb{R}^n)$ such that $C \subset \text{dom } \varphi \subset \text{dom } \psi$:

$$\alpha = \inf\{\phi(x) := \varphi(x) - \psi(x) \mid x \in C\}. \tag{2.16}$$

Clearly there holds

$$\phi(x) \geq \text{co}_C\ (\varphi - \psi)(x) \geq \varphi(x) + \text{co}_C\ (-\psi)(x) \geq \varphi(x) + \overline{\text{co}}_C\ (-\psi)(x), \ \forall x \in C.$$

Therefore, the DC relaxation technique consists in replacing DC objective function ϕ with the convex function ϕ_{re} defined by

$$\text{either } \phi_{re} := \varphi + \text{co}_C\ (-\psi) \text{ or } \phi_{re} := \varphi + \overline{\text{co}}_C\ (-\psi).$$

According to the results displayed in 2.3.1 and 2.3.1, we propose the following computable convex minorants of ϕ on C :

(i) $\varphi + \text{co}_C\ (-\psi)$ if $V(C)$ is easy to compute, for example in case C is a bounded polyhedral convex set with known vertex set $V(C)$.

(ii) For the general case, $\text{co}_C\ (-\psi)$ will be replaced with
- $\overline{\text{co}}_L\ (-\psi)$ where L is a polytope containing C defined in 2.3.1, ($L_i := [a_i, b_i]$ quite often in practice), if ψ is separable, or
- $\text{co}_S\ (-\psi)$ with S being a simplex containing C.

DC relaxation in nonconvex mixed-integer nonlinear programming problem

We return now to our nonconvex mixed-integer nonlinear program (NIP) and show how to apply the results mentioned above for computing a convex minorant of f by DC relaxation techniques.

$$\alpha = \min\left\{ f(x) = x^T Q x - \sum_{i=1}^{n} h_i(x_i) \mid x \in B \cap K, \ x_j \in \mathbb{Z}, \forall j = 1, \ldots, n_1 \right\}.$$

Note that f is a DC function with a DC decomposition $f(x) = g(x) - h(x)$ where h is separable function,

$$h(x) = \sum_{i=1}^{n} h_i(x_i),$$

and $B = [l, u] = \prod_{i=1}^{n} [l_i, u_i]$. According to section 2.3.1, we will build a convex minorant on $B \cap K$ for f as follows :

$$\varphi(x) := g(x) + \overline{\text{co}}_B\ (-h)(x), \tag{2.17}$$

where

$$\overline{\text{co}}_B\ (-h)(x) = \sum_{i=1}^{n} \overline{\text{co}}_{[l_i,u_i]}\ (-h_i)(x_i), \tag{2.18}$$

(from Theorem 2.2 and Proposition 2.1) and the convex relaxation problem will be constructed :

$$\beta = \min\left\{\varphi(x) := g(x) + \overline{\text{co}}_B\ (-h)(x) \mid x \in B \cap K\right\}. \tag{P_{re}}$$

If in the B& B scheme, at a node $k = 0, 1, ...$, we have to solve a subproblem :

$$\alpha_k = \min\{f(x) = g(x) - h(x) \mid x \in B_k \cap K,\ x_j \in \mathbb{Z}, j = 1, \ldots, n_1\}, \tag{P^{B_k}}$$

where $B_k = [l^k, u^k] = \prod_{i=1}^{n} [l_i^k, u_i^k],\ l^k, u^k \in \mathbb{Z}^n,\ l_i^k \leq u_i^k,\ \forall i = 1, \ldots, n$, a convex underestimation of f on $B_k \cap K$ is chosen as :

$$\varphi_k(x) := g(x) + \overline{\text{co}}_{B_k}\ (-h)(x), \tag{2.19}$$

with

$$\overline{\text{co}}_{B_k}\ (-h)(x) = \sum_{i=1}^{n} \overline{\text{co}}_{[l_i^k,u_i^k]}\ (-h_i)(x_i), \tag{2.20}$$

$$\overline{co}_{[l_i^k,u_i^k]}\ (-h_i)(x_i) = \begin{cases} \left(\dfrac{-h_i(u_i^k) + h_i(l_i^k)}{u_i^k - l_i^k}\right)(x_i - l_i^k) - h_i(l_i^k) & \text{if } l_i^k < u_i^k \\ -h_i(l_i^k) & \text{if } l_i^k = u_i^k. \end{cases} \tag{2.21}$$

And a convex relaxation problem of (NIP) considered at this node is of the form :

$$\beta(B_k) = \min\left\{\varphi_k(x) := g(x) + \overline{\text{co}}_{B_k}\ (-h)(x) \mid x \in B_k \cap K\right\}. \tag{$P_{re}^{B_k}$}$$

2.3.2 Branching procedure

Suppose that, at a node in the B&B scheme, the considered subproblem is (P^{B_k}) :

$$\alpha_k = \min\left\{f(x) = g(x) - h(x) \mid x \in B_k \cap K,\ x_j \in \mathbb{Z}, j = 1, \ldots, n_1\right\} \tag{2.22}$$

where $B_k = [l^k, u^k]$, $l^k, u^k \in \mathbb{Z}^n$, $l_i^k \leq u_i^k,\ \forall i = 1, \ldots, n$.

Suppose that $\overline{x} \in B_k \cap K$ is chosen to start branching procedure. Let the index $j^* \in \{1, \ldots, n_1\}$ be defined by

$$(-h_{j^*})(\overline{x}_{j^*}) - \overline{\text{co}}_{[l_{j^*}^k,u_{j^*}^k]}\ (-h_{j^*})(\overline{x}_{j^*}) = \max_{i=1,\ldots,n_1} \{(-h_i)(\overline{x}_i) - \overline{\text{co}}_{[l_i^k,u_i^k]}\ (-h_i)(\overline{x}_i)\}. \tag{2.23}$$

Let

$$B_{k0} = \{x \in B_k \mid x_{j*} \leq \lfloor \overline{x}_{j*} \rfloor\}, \quad B_{k1} = \{x \in B_k \mid x_{j*} \geq 1 + \lfloor \overline{x}_{j*} \rfloor\},$$

then

$$\{x \in B_k \cap K \mid x_j \in \mathbb{Z}, \forall j \leq n_1\} = \\ = \{x \in B_{k0} \cap K \mid x_j \in \mathbb{Z}, \forall j \leq n_1\} \cup \{x \in B_{k1} \cap K \mid x_j \in \mathbb{Z}, \forall j \leq n_1\}.$$

Therefore, two subproblems are constructed :

$$\alpha_{k0} = \min\{f(x) = g(x) - h(x) \mid x \in B_{k0} \cap K,\ x_j \in \mathbb{Z}, j = 1, \ldots, n_1\}, \qquad (P^{B_{k0}})$$
$$\alpha_{k1} = \min\{f(x) = g(x) - h(x) \mid x \in B_{k1} \cap K,\ x_j \in \mathbb{Z}, j = 1, \ldots, n_1\}. \qquad (P^{B_{k1}})$$

2.3.3 Bounding procedure

To calculate a lower bound for α at a node k, we solve the convex relaxation problem $(P_{re}^{B_k})$ of (P^{B_k}), which is a convex quadratic program, by using CPLEX solver :

$$\beta(B_k) = \min\left\{\varphi_k(x) := g(x) + \overline{\text{co}}_{B_k}\ (-h)(x) \mid x \in B_k \cap K\right\}. \qquad (P_{re}^{B_k})$$

Let x^{B_k} be the computed solution. The value of f at the feasible point $\overline{x}^{B_k} := \lfloor x^{B_k} \rfloor$, that means $\overline{x}_j^{B_k} = \lfloor x_j^{B_k} \rfloor, \forall j \leq n_1$, and $\overline{x}_j^{B_k} = x_j^{B_k}, \forall j > n_1$, provides an upper bound for α.

2.3.4 A B&B algorithm for solving (NIP)

Our customized B&B algorithm is described below :

Algorithm 2 (BB applied on (NIP))

- ***Initialization :***
  Let $B_0 = B$. The algorithm starts by solving the DC relaxation problem (P_{re}) of (NIP) to get a solution x^{B_0}.
  If $x_j^{B_0} \in \mathbb{Z}, \forall j \leq n_1$ then $x^* := x^{B_0}$ solves (NIP) and **the algorithm stops.**
  Otherwise, find a feasible point $\overline{x}^{B_0}$ of (NIP) from x^{B_0}.
  ◇ first lower bound : $\beta_0 = \beta(B_0) = \varphi(x^{B_0})$;
  ◇ first upper bound : $\gamma = f(\overline{x}^{B_0})$.
  Set $x^* := \overline{x}^{B_0}$, $\mathcal{B} \longleftarrow \{B_0\}$, $k \longleftarrow 0$.
- ***While* TRUE *do***
  ◇ Delete from $\mathcal{B}$ all the subbox B_j whose lower bound $\beta(B_j)$ is greater or equal to the current upper bound γ.
  ◇ If $(\mathcal{B} = \emptyset)$ or $(\gamma = \min\{\beta(B_i) :\ B_i \in \mathcal{B}\})$ then **STOP**, x^* is an optimal solution of (NIP).
  ◇ Choose B_k in $\mathcal{B}$ such that $\beta_k = \beta(B_k) = \min\{\beta(B_i) :\ B_i \in \mathcal{B}\}$.
  ◇ Bisect $B_k \cap K$ into two subsets $B_{k0} \cap K$ and $B_{k1} \cap K$ by branching procedure from x^{B_k}, an optimal solution of the DC relaxation problem $(P_{re}^{B_k})$ of (P^{B_k}).
  Two new subproblems are obtained :

$$\alpha_{k0} = \min\{f(x) = g(x) - h(x) \mid x \in B_{k0} \cap K,\ x_j \in \mathbb{Z}, j = 1, \ldots, n_1\}, \qquad (P^{B_{k0}})$$
$$\alpha_{k1} = \min\{f(x) = g(x) - h(x) \mid x \in B_{k1} \cap K,\ x_j \in \mathbb{Z}, j = 1, \ldots, n_1\}. \qquad (P^{B_{k1}})$$

  For each new subproblem, calculate a lower bound and an upper bound by bounding procedure.

$\diamond$ Update the upper bound γ and the best current solution known so far x^*.
Set $\mathcal{B} \longleftarrow \mathcal{B} \cup \{B_{k0}, B_{k1}\} \backslash B_k$.
End While.

The correctness and the convergence of the algorithm are stated in the following result whose proof is fairly standard from the branching procedure and the bounding one (see [88, 91, 131, 132]). Its finiteness is due to the integral bisection used in the algorithm.

Theorem 2.3 *Algorithm 2 terminates after finitely many iterations and yields an optimal solution of Problem (NIP).*

2.4 Application to Discrete Portfolio Optimization under Concave Transaction Costs

2.4.1 Related works

Before applying our proposed approaches to discrete portfolio optimization which is a modified mean-risk model of Markowitz's portfolio selection problem, we mention below some related works on portfolio selection as well as some approaches to tackle discrete features of the portfolio selection problem in the literature.

Mean-risk models were proposed in the early fifties in order to provide a practical solution for the portfolio selection problem by Markowitz [112]. In his pioneering work, Markowitz proposed a famous mean-risk model where variance was considered as a measure of risk and formulated the portfolio optimization problem as a quadratic program. Nearly forty years later, instead of variance which is L_2 measure in Markowitz' model, Konno and Yamazaki [81] introduced L_1 risk (absolute deviation) ; therefore the portfolio optimization problem can be reduced to a linear programming problem. Based on the mean absolute deviation model of Konno et al., Mansini and Speranza [110] studied the portfolio selection problem with minimum transaction lots and proposed three heuristic solution algorithms. Their problem with integer variables is proved NP-complete. Later on, Kellerer et al. [70] developed the modelling of this problem with fixed costs into account. They suggested also different heuristic procedures to specifically deal with transaction costs.

The problem of selecting a portfolio with considering transaction costs has been received a very attention of many researchers. Moreover, in the real trade practice, certain discrete features as minimum transaction lots, buy-in threshold, cardinality are also appeared. As a consequence, the portfolio selection model needs to be modified into a combinatorial programming problem, specially an integer program in which these factors are introduced. In order to tackle discrete features in the portfolio selection problem, some methodologies have been proposed, among which included : a branch-and-cut method for cardinality constrained mean-variance model of Bienstock [16], heuristic algorithms for mean-absolute deviation formulation with minimum transaction lots of Mansini et Speranza [110], heuristic algorithms for mean-semi absolute deviation model of Kellerer et al. [70], a branch and bound algorithm using CPLEX solver of Konno et al. [78–80], a convergent Lagrangian and contour-domain cut method by Li et al. [102], genetic algorithms for portfolio selection problems with minimum transaction lots by Lin and Liu [105], a distributed computation algorithm for solving portfolio problems with integer variables by Li and Tsai [103], ect.

In this application, we consider a modified mean-risk model based on Markowitz's portfolio selection problem taking into account concave transaction costs and a budget

constraint with integer portfolio allocations to assets. These factors make the portfolio selection problem more difficult to solve because it is a nonseparable, nonconvex, nonlinear integer programming problem.

2.4.2 Mean-variance model under concave transaction costs

Consider a market with n assets and a total allocation budget in dollars b. Let R_j be the random variable representing the rate of return per lot of the jth asset without transaction cost, $E(R_j) = r_j$ and $\sigma_{ij} = cov(R_i, R_j), i, j = 1, \ldots, n$.

Let x_j be the number of lots invested into the jth asset and $x = (x_1, \ldots, x_n)$ be the portfolio decision. We assume that no shortselling is allowed for any asset, thus $x \in \mathbb{N}^n$. If the current price in dollars of a lot of the jth asset is $p_j, (j = 1, \ldots, n)$, then the random return in dollars from holding securities is $R(x) = \sum_{j=1}^{n} R_j p_j x_j$. The mean and variance of $R(x)$ are :

$$E[R(x)] = E\left[\sum_{j=1}^{n} R_j p_j x_j\right] = \sum_{j=1}^{n} r_j p_j x_j,$$

$$Var[R(x)] = Var\left[\sum_{j=1}^{n} R_j p_j x_j\right] = \sum_{i=1}^{n}\sum_{j=1}^{n} p_i p_j \sigma_{ij} x_i x_j = x^T Q x,$$

where $Q = (p_i p_j \sigma_{ij})_{n\times n}$ is a positive semi-definite matrix.

$Var[R(x)]$ represents the variance of the portfolio decision.

The budget constraint of the investor is $\sum_{j=1}^{n} p_j x_j \leq b$.

Suppose that the balance $\left(b - \sum_{j=1}^{n} p_j x_j\right)$ is deposited to the riskless account (e.g., a bank account) whose interest rate is r (in %). Let $c(x) = \sum_{j=1}^{n} c_j(x_j)$ be the transaction cost (in dollars) associated with the portfolio decision $x = (x_1, \ldots, x_n)$. Each c_j is assumed a non-decreasing differentiable concave function and $c_j(0) = 0$; $c_j(x_j) > 0$ if $x_j > 0$, $\forall j$.

The total expected return (in dollars) of the portfolio decision x is

$$\begin{aligned} T(x) &= \sum_{j=1}^{n} r_j p_j x_j + r\left(b - \sum_{j=1}^{n} p_j x_j\right) - \sum_{j=1}^{n} c_j(x_j) \\ &= rb + \sum_{j=1}^{n} \left[(r_j - r) p_j x_j - c_j(x_j)\right]. \end{aligned}$$

The mean-variance model discussed in this study is formulated as follows :

$$\begin{cases} \min\ x^T Q x - r_a T(x) \\ \text{s.t. } \sum_{j=1}^{n} p_j x_j \leq b, \\ \quad 0 \leq x_j \leq \overline{u}_j,\ \forall j = 1, \ldots, n, \\ \quad x_j \in \mathbb{Z},\ \forall j = 1, \ldots, n. \end{cases}$$

where $r_a \geq 0$ is the risk-aversion parameter, $\overline{u}_j$ is the maximum of lots allowed for buying of the jth asset.

This problem reduces to :

$$\begin{cases} \alpha = \min \left\{ f(x) = x^T Q x - \sum_{j=1}^{n} [(r_j - r) p_j x_j - c_j(x_j)] r_a \right\} \\ \text{s.t. } \sum_{j=1}^{n} p_j x_j \leq b, \\ \quad 0 \leq x_j \leq \overline{u}_j, \ \forall j = 1, \ldots, n, \\ \quad x_j \in \mathbb{Z}, \ \forall j = 1, \ldots, n. \end{cases} \tag{P}$$

Set $B = [0, \overline{u}] = \prod_{j=1}^{n} [0, \overline{u}_j] \subset \mathbb{R}^n$ and $K = \left\{ x \in \mathbb{R}^n \mid \sum_{j=1}^{n} p_j x_j \leq b \right\}$.

Note that Q is positive semi-definite matrix and c_j are concave functions for all j. Thus (P) has the form of (NIP) with $h_j(x_j) = [(r_j - r) p_j x_j - c_j(x_j)] r_a$. h_j are differentiable convex functions and $A = (p_1, \ldots, p_n) \in \mathbb{R}^{1 \times n}$ is a nonnegative matrix. The DCA algorithm presented in section 2.2 and the B&B algorithm using DC relaxation in section 2.3 will be applied for solving (P).

2.4.3 Computational experiments

We present in this section some computational experiments on several sets of test problems of (P). The two algorithms were coded by C and have been implemented on a PC equipped with Window XP Intel(R) Core(TM) 2 Duo CPU 2.26 GHz, 1.92 GB RAM. For solving convex quadratic subproblems, we used CPLEX solver version 12.1. To evaluate the performance of solutions computed by DCA and by B&B, we compared them with numerical results provided by LaGO and COUENNE, two standard solvers for solving nonconvex nonlinear mixed integer problems (see [1, 3]).

The test data are randomly generated and uniformly distributed in MATLAB as $p_i \in [100, 1000]$, the random rate of return per lot of the ith asset is in $[0.0, 0.05]$. The risk aversion parameter is chosen as $r_a = 5$, $\eta = 20$, and the interest rate of the riskless asset r is set at 1%. The maximum purchasing lot of each stock $\overline{u}_j$ is set to be 20 for each asset. The initial budget $b = 0.4 \sum_{j=1}^{n} p_j \overline{u}_j$. The concave transaction cost function associated is chosen by $c(x) = \sum_{j=1}^{n} c_j(x_j) = \sum_{j=1}^{n} \gamma_j p_j \ln(1 + x_j)$ where γ_j is parameter randomly chosen from $[0.001, 0.004]$. We let $\theta_1 = 10^{-6}$ and $\theta_2 = 10^{-6}$ in DCA algorithm (Algorithm 1).

We specify the following options in our codes :

– DCA is applied to the penalty DC program.
– DC relaxation techniques have been used for lower bounds computation in B&B scheme.

In the following numerical results, we declare the notations :

◇ Prob 20. : problem (P) considering $n = 20$;
◇ # : number of iterations (when the algorithms DCA, B&B, also LaGO or COUENNE stopped) ;
◇ bestLB : best lower bound obtained by B&B ;

- ◇ bestUB : best upper bound obtained by each solver (included B&B, LaGO, COUENNE) ;
- ◇ ValDCA : value of the objective function obtained by DCA ;
- ◇ For three algorithms B&B, LaGO and COUENNE, the relative gap is calculated by

$$\text{gap}(\%) = \frac{100 * (\text{bestUB} - \text{bestLB})}{\max\{|\text{bestUB}|, |\text{bestLB}|\}}$$

To calculate the relative gap for DCA, "bestUB" will be replaced by "ValDCA".

In all test problems we limit the CPU time to 2 hours for three algorithms B&B, LaGO and COUENNE, and limit the number of iterations in B&B scheme (Algorithm 2) to 100000. The computational results are presented in four tables 2.1, 2.2, 2.3, 2.4 with $n = 20, 50, 100, 200$ respectively.

We tested 10 problems for each value of n and reported the average of the quantities (gap, number of iterations, and computation time) of B&B algorithm, DCA, LaGO and COUENNE. In most instances, when comparing DCA with the global optimization B&B method, we observed that, with a suitable parameter $t > 0$ and a good initial point, DCA required a small average number of iterations (respectivly equal to 24.2, 48.3, 46.6, and 61.9), and quite often provided good solutions (average gap taking the values 0.776, 6.523, 8.729, 4.401 respectively) in a very small average computation time (which got the corresponding values $0.29s$, $0.998s$, $2.76s$, $22.97s$) against more than 1 hour, even 2 hours for B&B. In some instances ($n = 20$), global optimal solutions are also found by DCA in less than 0.4 second. Of course, B&B provided better solutions than DCA : Its average gap took the values $0, 1.621, 7.499$ and 3.290 respectively. But its average number of iterations increases too much (with corresponding values 583.8, 86175, 56290.1, and 8755.7) as well as its average computation time (whose values were 14.24s, 4148.83s, 7200.03s and 7200.42s respectively) prevent B&B from handling large-scale (P). Overall, DCA converges very fast to good feasible solutions.

In comparison of our B&B algorithm with LaGO and COUENNE, LaGo can provide good solutions only for $n = 20$ but is much more expensive. For other values of n, the results obtained by LaGO and COUENNE after 2 hours were very bad.

2.5 Conclusion

In this chapter, we investigated a DC programming approach for solving a NP-hard class of nonconvex mixed-integer nonlinear programs and applied it for solving a discrete portfolio optimization problem under concave transaction costs. The problem was first recast as a DC program by using penalty techniques in DC programming and then an appropriate DCA (Algorithm 1) was developed for tackling the penalized DC program. To evaluate the globality of solutions computed by DCA we proposed a customized B&B algorithm (Algorithm 2) using DC relaxation techniques for lower bounding. Computational experiments showed that DCA is very inexpensive (so scalable) and provides good approximate global solutions. As for global algorithms, B&B always attained global solutions but is very expensive, even though it is by far the best compared with two solvers LaGO and COUENNE whose computed solutions are very bad.

It is worth investigating more sophiscated reformulations of the nonconvex integer program (NIP) in the DC programming framework and more suitable initial point strategies in order to further improve the computational efficiency of DCA. Works in these directions are currently being done.

	BB				DCA			LAGO				COUENNE			
Prob	#	bestLB	bestUB	gap(%)	time(s)	#	ValDCA	gap(%)	time(s)	#	bestUB	gap(%)	time(s)	#	bestUB	gap(%)	time(s)
20.01	210	-37.2110	-37.2110	0.000	4.67	26	-37.2110	0.000	0.28	2400	-37.2097	<1%	99.64	>100000	510.9684	>100%	7202.91
20.02	484	-62.4794	-62.4794	0.000	10.66	16	-61.1309	2.158	0.17	4470	-62.4794	<1%	207.67	– –	124.0707	– –	7171.63
20.03	155	-59.0188	-59.0188	0.000	3.44	4	-59.0188	0.000	0.06	1769	-59.0179	<1%	82.32	– –	535.9168	– –	7215.61
20.04	595	-27.2958	-27.2958	0.000	15.16	20	-27.2958	0.000	0.23	7659	-27.2958	<1%	390.70	– –	206.2992	– –	7190.79
20.05	2256	-63.1283	-63.1283	0.000	49.88	24	-62.0871	1.649	0.25	4493	-63.1277	<1%	216.58	– –	38.1051	– –	7210.26
20.06	205	-51.7268	-51.7268	0.000	5.53	47	-51.3750	0.680	0.70	4291	-51.7268	<1%	214.38	– –	170.4708	– –	7207.84
20.07	73	-20.6690	-20.6690	0.000	1.63	7	-20.6690	0.000	0.09	1599	-20.6690	<1%	67.83	– –	557.5798	– –	7207.54
20.08	390	-29.0576	-29.0576	0.000	11.63	49	-28.1079	3.268	0.52	3854	-29.0576	<1%	180.99	– –	82.8580	– –	7209.23
20.09	471	-28.4991	-28.4991	0.000	16.70	26	-28.4991	0.000	0.36	2632	-28.4978	<1%	116.48	– –	234.4515	– –	7211.88
20.10	999	-20.2093	-20.2093	0.000	23.13	23	-20.2093	0.000	0.25	6948	-20.2091	<1%	361.28	– –	264.1412	– –	7192.05
Aver	583.8			0.000	14.24	24.2		0.776	0.29	4011.5		<1%	193.79	>100000		>100%	7201.97

TABLE 2.1 – Case $n = 20$

	BB				DCA			LAGO				COUENNE			
Prob	#	bestLB	bestUB	gap(%)	time(s)	#	ValDCA	gap(%)	time(s)	#	bestUB	gap(%)	time(s)	#	bestUB	gap(%)	time(s)
50.01	100000	-345.1285	-333.3044	3.426	5727.84	23	-322.5206	6.551	0.52	53092	4916.19	>100%	6149.47	>100000	144.6883	>100%	7178.54
50.02	100000	-218.3728	-215.1854	1.460	4904.02	32	-201.2074	7.861	0.64	62319	11266.4591	– –	7214.22	– –	888.4338	– –	7191.33
50.03	28879	-253.8137	-253.8137	0.000	1217.89	143	-239.8797	5.490	3.19	62936	20005.6186	– –	7214.13	– –	1119.9599	– –	7191.18
50.04	100000	-385.9997	-378.7366	1.881	4738.58	20	-362.5223	6.082	0.41	63147	9618.2333	– –	7211.51	– –	1180.5798	– –	7198.21
50.05	100000	-555.3338	-537.8329	3.152	5201.81	41	-507.9206	8.538	0.84	61060	-537.8295	23.8418	7218.20	– –	310.5000	– –	7200.30
50.06	92755	-158.9058	-158.9058	0.000	4386.64	55	-147.9168	6.915	1.08	69512	-158.9062	53.9058	7213.35	– –	1343.4016	– –	7190.35
50.07	100000	-406.7755	-398.5504	2.022	4454.67	71	-374.3211	7.978	1.53	31489	20239.3475	– –	7206.93	– –	28.7497	– –	7109.78
50.08	45196	-339.2839	-339.2839	0.000	1831.33	24	-325.4041	4.091	0.30	23575	164254.2069	– –	7204.94	– –	520.4522	– –	7198.28
50.09	94920	-400.0557	-400.0557	0.000	4014.84	23	-386.3556	3.425	0.45	23255	171589.7529	– –	7205.23	– –	311.8466	– –	7196.28
50.10	100000	-348.5349	-333.6430	4.273	5010.72	51	-319.6140	8.298	1.02	32287	32703.6389	– –	7205.67	– –	467.8591	– –	7193.13
Aver	86175			1.621	4148.83	48.3		6.523	0.998	48267.2		>87%	6454.67	>100000		>100%	7184.74

TABLE 2.2 – Case $n = 50$

	BB					DCA			LAGO				COUENNE			
Prob	#	bestLB	bestUB	gap(%)	time(s)	#	ValDCA	gap(%)	time(s)	#	bestUB	gap(%)	time(s)	#	bestUB	gap(%)	time(s)
100.01	55217	-3287.0075	-3089.9938	5.994	7200.06	56	-3060.6799	6.886	4.08	20974	419343.8940	>100%	7206.40	5621	-1884.6774	>99%	7188.12
100.02	57355	-2051.4123	-1852.3969	9.701	7200.09	73	-1833.6690	10.614	3.98	21213	296903.2751	– –	7206.28	973414	-1080.0104	– –	7188.32
100.03	60921	-3248.3896	-3087.3787	4.957	7200.09	76	-3046.8523	6.204	4.06	20733	524779.0238	– –	7206.06	14248	-2544.6949	– –	7178.51
100.04	57501	-6454.9409	-6133.5319	4.979	7200.06	27	-6130.1002	5.032	1.67	20876	346301.8705	– –	7206.14	37354	-4915.1015	– –	7175.30
100.05	61485	-2145.9862	-1972.2402	8.096	7200.08	21	-1950.5492	9.107	1.42	20983	421742.6638	– –	7206.53	3231	-1236.5113	– –	7183.69
100.06	55038	-3412.2018	-3190.8935	6.486	7200.02	25	-3112.5748	8.781	1.61	20932	400207.5803	– –	7205.53	985471	-2632.2891	– –	7186.61
100.07	59356	-2163.8764	-1898.5678	12.261	7200.03	57	-1870.9884	13.535	3.16	20733	346525.3730	– –	7206.15	806572	-476.5575	– –	7189.40
100.08	58255	-2141.0330	-1953.9710	8.737	7200.00	29	-1904.0816	11.067	1.59	20474	490532.6343	– –	7205.94	463885	-680.9298	– –	7193.44
100.09	40732	-2102.9127	-1940.4766	7.724	7200.05	48	-1912.2246	9.068	2.80	21073	433686.3576	– –	7206.04	282341	-940.3490	– –	7193.67
100.10	57041	-3381.7469	-3176.9224	6.057	7200.05	54	-3145.0026	7.000	3.19	21148	374341.3665	– –	7205.84	1434411	-2498.3080	– –	7189.45
Aver	56290.1			7.499	7200.05	46.6		8.729	2.76	20913.9		>100%	7206.19	500654.8		>99%	7186.65

TABLE 2.3 – Case $n = 100$

	BB					DCA			LAGO				COUENNE			
Prob	#	bestLB	bestUB	gap(%)	time(s)	#	ValDCA	gap(%)	time(s)	#	bestUB	gap(%)	time(s)	#	bestUB	gap(%)	time(s)
200.01	10244	-42158.9021	-40763.3260	3.310	7200.69	27	-40513.3935	3.903	8.53	7835	661344.8785	>100%	7203.58	275	-39253.0338	>99%	7206.12
200.02	6908	-40289.2479	-39308.5852	2.434	7200.44	49	-39000.5944	3.199	14.81	8219	1127126.4194	– –	7203.66	215	-38694.3279	– –	7203.88
200.03	7376	-32428.2685	-31125.2625	4.018	7200.51	37	-30916.6775	4.661	11.44	7786	1086583.8027	– –	7203.46	519	-30698.3963	– –	7217.12
200.04	7370	-37841.4931	-36635.2493	3.188	7200.14	37	-36118.4054	4.553	16.45	7990	986626.7780	– –	7203.55	375	-35752.7739	– –	7191.91
200.05	7232	-31793.5603	-30968.0309	2.597	7200.09	59	-30378.7273	4.450	18.69	7065	378624.8808	– –	7202.92	361	-29774.3755	– –	7199.66
200.06	10625	-28799.9179	-27646.6818	4.004	7200.45	57	-27436.0667	4.736	25.18	8488	898587.9685	– –	7203.49	603	-26464.4190	– –	7220.66
200.07	9984	-39106.2119	-37716.0304	3.555	7200.30	147	-37490.1729	4.132	65.96	8920	573262.2628	– –	7203.35	386	-36595.5241	– –	7215.26
200.08	10302	-34875.5946	-33784.2557	3.129	7200.62	42	-33237.7733	4.696	14.21	9388	924322.3094	– –	7203.47	293	-32813.1737	– –	7211.60
200.09	7357	-35453.3641	-34249.9487	3.394	7200.31	82	-33824.0514	4.596	28.79	7432	1131789.2314	– –	7203.47	175	-32933.9351	– –	7210.99
200.10	10159	-41430.2266	-40075.2257	3.271	7200.64	82	-39324.4858	5.083	25.66	7488	1465588.8328	– –	7203.38	454	-38779.0514	– –	7214.83
Aver	8755.7			3.290	7200.42	61.9		4.401	22.97	8061.1		>100%	7203.53	365.6		>99%	7209.20

TABLE 2.4 – Case $n = 200$

Chapitre 3

Gestion de portefeuille : Minimisation du coût de transaction non convexe

Ce chapitre est réservé au problème de gestion de portefeuille d'une période qui consiste à minimiser le coût de transaction discontinu soumis aux différentes sortes de contraintes. Ce problème d'optimisation est non convexe et difficile à résoudre. Dans ce chapitre nous étudions des approches basées sur la programmation DC et DCA (algorithme DC) pour les méthodes de résolution.

3.1 Introduction

The mean-variance's model proposed by Markowitz [112] in 1952 is known as a basis for the development of various portfolio selection techniques. While the Markowitz' model is a convex program, extended models taking into account real features such as transaction costs, cardinality constraints, shortselling, buy-in threshold constraints, etc,... are in contrast nonconvex and very difficult to solve in most of the cases. The portfolio optimization problems including transaction costs have been receiving a lot of attention from many researchers (see e.g., [8, 29, 34, 70, 76–79, 108, 109, 145, 156]).

In [145] Speranza investigated a portfolio optimization model including realistic characteristics such as transaction costs, minimum transaction units as well as limits on minimum holdings then designed a heuristic algorithm for its solutions. The problem of selecting portfolio with fixed costs and minimum transaction lots has been studied by Kellerer et al. [70] in which the authors introduced different mixed-integer linear programming models. Due to the high computational complexity of these models, based on the construction and the optimal solution of mixed-integer subproblems, heuristic procedures were proposed. Mean-absolute deviation portfolio optimization problems considering nonconvex transaction costs have been studied by Konno et al. [76–79]. Based on branch and bound algorithms, their algorithms were used for tackling portfolio optimization problems with concave transaction costs [77], with concave transaction costs and minimal transaction unit constraints [78], with concave and piecewise constant transaction costs [79] ; for solving a class of long-short

portfolio optimization problems with concave and DC transaction costs and complementary conditions on variables [76]. In [29], Chen et al. dealt with the portfolio selection problem with transaction costs under the assumption that there exists admissible errors on expected returns and risks of assets. They proposed a new admissible efficient portfolio selection model and developed a heuristic algorithm using particle swarm optimization for solving this model.

In [108] Lobo et al. studied two alternative models for the problem of single-period portfolio optimization. The first consists of maximizing the expected return, taking transaction costs into account, and subject to different types of constraints on the feasible portfolios. They proposed a heuristic method for solving this model where the transaction cost is separable and discontinuous. The second model deals with minimizing the total nonconvex transaction cost subject to feasible portfolio constraints. The authors claimed that their heuristic method for solving the former model can be adapted to solve the later.

The transaction cost which receives many attentions of any investor is a source of concern for portfolio managers. It has an important effect on the portfolio optimization. Ignoring transaction costs would result in inefficient portfolios (see [8]). This fact was also verified by the experimental analysis reported in [156] where the authors solved the portfolio optimization problem subject to V-shaped transaction costs. In the literature, almost all of the proposed solution methods for solving portfolio optimization problems in which transaction costs were included concerned with either a heuristic or a branch and bound procedure.

In this work we are interested in the second model introduced in [108]. We consider a slightly modified model where the constraints include shortselling constraints, limit on expected return, limit on variance, and diversification constraints. The considered transaction cost is assumed to be separable, i.e., the sum of the transaction cost associated with each trade. It is a discontinuous function that results to a difficult nonconvex program. We investigate two deterministic approaches for designing solution methods to this problem. In the first approach, we approximate the discontinuous nonconvex objective function by a DC function and then develop DCA for tackling the resulting DC problem. In the second approach, we introduce binary variables and rewrite the initial problem as a mixed 0-1 programming problem. Later, by using penalty techniques, this mixed 0-1 program is reformulated as a DC programming problem which can be handled by a DC Algorithm. For globally solving the original problem and evaluating the quality of solutions provided by DCA, we propose hybridization algorithms that combine DCA and a Branch-and-Bound (B&B) scheme. Lower bounds of the optimal value are obtained by solving relaxation subproblems.

The rest of the chapter is organized as follows. In the next section, we describe the considered portfolio problem and its mathematical formulation. Section 3.3 is concerned with the first proposed approach in which the DC approximation of the initial problem is constructed and the development of DCA for solving the resulting problem is described. A hybrid Branch-and-Bound-DCA algorithm for tackling the original problem is also presented in this section. Section 3.4 deals with the second approach for solving the original problem : we firstly reformulate it as a mixed zero-one program and then transform the resulting problem into a DC one by penalty techniques. A combined DCA-Branch-and-Bound algorithm is also proposed in the same section. The numerical results are reported in section 3.5, while some conclusions are indicated in the last section.

3.2 Problem description and mathematical formulation

Consider an investment portfolio that consists of holdings in some or all of n assets. The current holdings in each asset are $w = (w_i, \ldots, w_n)^T$. The total current wealth is then $\mathbf{1}^T w$, where $\mathbf{1}$ is a vector with all entries equal to one. The amount transacted in asset i is x_i, with $x_i > 0$ for buying, $x_i < 0$ for selling and $x = (x_1, \ldots, x_n)^T$ is portfolio decision. After transactions, the adjusted portfolio is $w + x$.

The adjusted portfolio $w+x$ is held for a fixed period of time. At the end of that period, the return on asset i is the random variable a_i. We assume knowledge of the first and the second moments of the joint distribution of $a = (a_1, \ldots, a_n)$,

$$\mathbf{E}(a) = \overline{a}, \qquad \mathbf{E}(a - \overline{a})(a - \overline{a})^T = \Sigma.$$

A riskless asset can be included, in this case the corresponding $\overline{a}_i$ equals to its return and the i^{th} row and the i^{th} column of Σ are zero.

The wealth at the end of the period is a random variable, $W = a^T(w+x)$ with expected value and variance given by

$$\mathbf{E}W = \overline{a}^T(w + x), \qquad \mathbf{E}(W - \mathbf{E}W)^2 = (w + x)^T\Sigma(w + x) \tag{3.1}$$

We consider the problem of minimizing the total transaction costs subject to portfolio constraints :

$$\begin{cases} \min \ \phi(x) \\ \text{s.t. } \overline{a}(w + x) \geq r_{\min}, \\ \quad w + x \in \mathcal{S}, \end{cases} \tag{3.2}$$

where $r_{\min}$ is the desired lower bound on the expected return and $\mathcal{S} \subseteq \mathbb{R}^n$ is portfolio constraint set.

The portfolio constraint set $\mathcal{S}$ can be defined from the following convex constraints :

1. *Shortselling constraints :* Individual bounds s_i on the maximum amount of shortselling allowed on asset i are

$$w_i + x_i \geq -s_i, \quad i = 1, \ldots, n. \tag{3.3}$$

   If shortselling is not permitted, the s_i are set to zero. Otherwise, $s_i > 0$.

2. *Variance :* The standard deviation of the end period wealth W is constrained to be less than $\sigma_{\max}$ by the convex quadratic inequality

$$(w + x)^T\Sigma(w + x) \leq \sigma^2_{\max} \tag{3.4}$$

   ((3.4) is a *second-order cone constraint*).

3. *Diversification constraints :* Constraints on portfolio diversification can be expressed in terms of linear inequalities and therefore are readily handled by convex optimization. Individual diversification constraints limit the amount invested in each asset i to a maximum of p_i,

$$w_i + x_i \leq p_i, \quad i = 1, \ldots, n. \tag{3.5}$$

   Alternatively, we can limit the fraction of the total wealth held in each asset,

$$w_i + x_i \leq \lambda_i \mathbf{1}^T(w + x), \quad i = 1, \ldots, n. \tag{3.6}$$

   They are convex inequality constraints on x.

Transaction costs can be used to model a number of costs, such as brokerage fee, bid-ask spread, taxes or even fund loads. In this study, the transaction costs $\phi(x)$ is defined by

$$\phi(x) = \sum_{i=1}^{n} \phi_i(x_i), \tag{3.7}$$

where ϕ_i is the transaction cost function for asset i. We will consider a simple model including fixed plus linear costs. Let β_i be the common fixed costs associated with buying and selling asset i. The fixed plus linear transaction cost function is given by

$$\phi_i(x_i) = \begin{cases} 0, & x_i = 0 \\ \beta_i - \alpha_i^1 x_i, & x_i < 0 \\ \beta_i + \alpha_i^2 x_i, & x_i > 0. \end{cases} \tag{3.8}$$

The function ϕ is nonconvex, unless the fixed costs are zero.

We develop below two approaches based on DC programming and DCA for solving Problem (3.2) with $\mathcal{S}$ being defined in (3.3), (3.4), (3.6) and ϕ being given in (3.7), (3.8).

3.3 DC programming and DCA for solving (3.2)

3.3.1 DC approximation problem

Let C be the feasible set of (3.2). Since ϕ is discontinuous, we will construct a DC approximation function of ϕ. We first compute upper bounds u_i^0 and lower bounds l_i^0 for variables x_i by solving $2n$ convex problems :

$$\min\{x_i \mid x \in C\} \quad (LB_i), \qquad \max\{x_i \mid x \in C\} \quad (UB_i). \tag{3.9}$$

Let $R_0 = \prod_{i=1}^{n} [l_i^0, u_i^0]$. The problem (3.2) can be written as

$$\omega = \min \left\{ \phi(x) = \sum_{i=1}^{n} \phi_i(x_i) \mid x \in C \cap R_0 \right\} \tag{P}$$

For each $i = 1, \ldots, n$, let $\epsilon_i > 0$ be a sufficiently small number chosen as follows :
- $\epsilon_i < \min\{-l_i^0, u_i^0\}$ if $l_i^0 < 0 < u_i^0$,
- $\epsilon_i < u_i^0$ if $l_i^0 = 0 < u_i^0$,
- $\epsilon_i < -l_i^0$ if $l_i^0 < u_i^0 = 0$.

Consider the functions $\overline{\phi_i}, \psi_i : \mathbb{R} \longrightarrow \mathbb{R}$ given by

$$\overline{\phi_i}(x_i) = \begin{cases} \beta_i - \alpha_i^1 x_i, & x_i \leq 0 \\ \beta_i + \alpha_i^2 x_i, & x_i \geq 0 \end{cases}, \qquad \psi_i(x_i) = \begin{cases} -c_i^1 x_i, & x_i \leq 0 \\ c_i^2 x_i, & x_i \geq 0 \end{cases}$$

where $c_i^j = \left(\dfrac{\beta_i}{\epsilon_i} + \alpha_i^j \right)$, $j = 1, 2$, $i = 1, \ldots, n$. By definition, $\overline{\phi_i}, \psi_i$ are convex functions on $\mathbb{R}$. Then, a DC approximation function f of ϕ can be

$$f(x) = \sum_{i=1}^{n} f_i(x_i) \tag{3.10}$$

where $f_i(x_i) = g_i(x_i) - h_i(x_i)$ with g_i, h_i being determined by

- $g_i(x_i) = 0$, $h_i(x_i) = -\beta_i + \alpha_i^1 x_i$ if $l_i^0 < u_i^0 < 0$,
- $g_i(x_i) = 0$, $h_i(x_i) = -\beta_i - \alpha_i^2 x_i$ if $0 < l_i^0 < u_i^0$,
- $g_i(x_i) = 0$, $h_i(x_i) = -\min\{-c_i^1 x_i, \beta_i - \alpha_i^1 x_i\}$ if $l_i^0 < u_i^0 = 0$,
- $g_i(x_i) = 0$, $h_i(x_i) = -\min\{c_i^2 x_i, \beta_i + \alpha_i^2 x_i\}$ if $0 = l_i^0 < u_i^0$,
- $g_i(x_i) = \overline{\phi_i}(x_i) + \psi_i(x_i) = \begin{cases} \beta_i - (\alpha_i^1 + c_i^1)x_i, & x_i \le 0 \\ \beta_i + (\alpha_i^2 + c_i^2)x_i, & x_i \ge 0 \end{cases}$,

  $h_i(x_i) = \max\{\overline{\phi_i}(x_i), \psi_i(x_i)\} = \begin{cases} -c_i^1 x_i, & x_i \le -\epsilon_i \\ \beta_i - \alpha_i^1 x_i, & -\epsilon_i \le x_i \le 0 \\ \beta_i + \alpha_i^2 x_i, & 0 \le x_i \le \epsilon_i \\ c_i^2 x_i, & x_i \ge \epsilon_i \end{cases}$

  if $l_i^0 < 0 < u_i^0$.

It is easy to show that for all cases, g_i, h_i are convex polyhedral functions over $\mathbb{R}$. Therefore, with $g(x) = \sum_{i=1}^{n} g_i(x_i)$ and $h(x) = \sum_{i=1}^{n} h_i(x_i)$, $g - h$ is a DC decomposition of f. In addition,

- $\min\{f(x) \mid x \in C \cap R_0\} \le \min\left\{\phi(x) = \sum_{i=1}^{n} \phi_i(x_i) \mid x \in C \cap R_0\right\}$.
- For each i, the smaller value of ϵ_i, the better approximation of f_i to ϕ_i over $[l_i^0, u_i^0]$.

The problem (P) with ϕ being replaced by f,

$$\mu = \min\{f(x) = g(x) - h(x) \mid x \in C \cap R_0\} \qquad (P_{dc})$$

is a DC approximation problem of (P). We will investigate a DCA scheme for solving this problem.

3.3.2 DCA for solving (P_{dc})

According to the generic DCA scheme presented in Chapter 1, section 1.1, in order to obtain $x^{k+1} \in \partial g^*(y^k)$, at each iteration k, we have to compute a subgradient $y^k \in \partial h(x^k)$ and then solve the following convex program

$$\min\{g(x) - \langle y^k, x\rangle \mid x \in C \cap R_0\}, \tag{3.11}$$

which is equivalent to

$$\min\left\{\sum_{i=1}^{n} t_i - \langle y^k, x\rangle \mid g_i(x_i) \le t_i, \forall i = 1, \ldots, n,\ x \in C \cap R_0\right\}. \tag{3.12}$$

A subgradient $y^k \in \partial h(x^k)$ is computed by : for $i = 1, \ldots, n$,

- if $l_i^0 < u_i^0 < 0 : y_i^k = \alpha_i^1$;
- if $0 < l_i^0 < u_i^0 : y_i^k = -\alpha_i^2$;
- if $l_i^0 < u_i^0 = 0 : y_i^k = \begin{cases} \alpha_i^1, & \text{if } x_i^k < -\epsilon_i \\ c_i^1, & \text{if } x_i^k > -\epsilon_i \\ \in [\alpha_i^1, c_i^1], & \text{if } x_i^k = -\epsilon_i. \end{cases}$
- if $0 = l_i^0 < u_i^0 : y_i^k = \begin{cases} -c_i^2, & \text{if } x_i^k < \epsilon_i \\ -\alpha_i^2, & \text{if } x_i^k > \epsilon_i \\ \in [-c_i^2, -\alpha_i^2], & \text{if } x_i^k = \epsilon_i. \end{cases}$

$$\circ \text{ if } l_i^0 < 0 < u_i^0 : y_i^k = \begin{cases} -c_i^1, & \text{if } x_i^k < -\epsilon_i \\ \in [-c_i^1, -\alpha_i^1], & \text{if } x_i^k = -\epsilon_i \\ -\alpha_i^1, & \text{if } -\epsilon_i < x_i^k < 0 \\ \in [-\alpha_i^1, \alpha_i^2], & \text{if } x_i^k = 0 \\ \alpha_i^2, & \text{if } 0 < x_i^k < \epsilon_i \\ \in [\alpha_i^2, c_i^2], & \text{if } x_i^k = \epsilon_i \\ c_i^2, & \text{if } x_i^k > \epsilon_i. \end{cases}$$

Hence, DCA applied on (P_{dc}) can be described as follows.

Algorithm 1 (DCA applied on (P_{dc})) :

- **Initialization :**
  Let $x^0 \in \mathbb{R}^n$ and ε be a small enough positive number.
  Iteration $k \longleftarrow 0$.
- **Repeat :**
  - $x^k \longrightarrow y^k \in \partial h(x^k)$ as indicated above.
  - Solving the convex program (3.12) to obtain x^{k+1}.
  - $k \longleftarrow k+1$
- **Until :** $|f(x^{k+1}) - f(x^k)| \leq \varepsilon$.

3.3.3 A hybrid algorithm Branch-and-Bound-DCA for solving (P)

To globally solve the problem (P), we propose a hybrid B&B-DCA algorithm.

As DCA is a descent and efficient method for nonconvex programming, DCA will be used to improving upper bounds for ω in B&B scheme while lower bounds will be provided by solving relaxation subproblems.

Since ϕ is separable, the first convex underestimator of the objective function ϕ over the domain $C \cap R_0$ can be constructed as follows :

$$\widetilde{\phi}_{R_0}(x) = \sum_{i=1}^{n} \widetilde{\phi}_i(x_i), \tag{3.13}$$

where $\widetilde{\phi}_i(x_i)$ is given by

$\circ$ if $l_i^0 < u_i^0 < 0 : \widetilde{\phi}_i(x_i) = \beta_i - \alpha_i^1 x_i$;

$\circ$ if $0 < l_i^0 < u_i^0 : \widetilde{\phi}_i(x_i) = \beta_i + \alpha_i^2 x_i$;

$\circ$ if $l_i^0 < u_i^0 = 0 : \widetilde{\phi}_i(x_i) = \left(\dfrac{\beta_i}{l_i^0} - \alpha_i^1\right) x_i$;

$\circ$ if $0 = l_i^0 < u_i^0 : \widetilde{\phi}_i(x_i) = \left(\dfrac{\beta_i}{u_i^0} + \alpha_i^2\right) x_i$;

$$\circ \text{ if } l_i^0 < 0 < u_i^0 : \widetilde{\phi}_i(x_i) = \begin{cases} \left(\dfrac{\beta_i}{l_i^0} - \alpha_i^1\right) x_i, & x_i \leq 0 \\ \left(\dfrac{\beta_i}{u_i^0} + \alpha_i^2\right) x_i, & x_i \geq 0 \end{cases}$$

By solving the convex program

$$\eta(R_0) = \min\{\widetilde{\phi}_{R_0}(x) \mid x \in C \cap R_0\} \tag{R_0cp}$$

we obtain a point $x^{R_0} \in C$ satisfying

$$\eta(R_0) = \widetilde{\phi}_{R_0}(x^{R_0}) \leq \min\{\phi(x) \mid x \in C \cap R_0\}$$

i.e., $\eta(R_0)$ is the first lower bound for ϕ over $C \cap R_0$.

In Branch and Bound scheme, at each iteration, the branching procedure consists in choosing an index i^* and to bisect the set of constraints by setting x_{i^*} either to zero or to be nonzero (in the last case $\phi_{i^*}(x_{i^*}) = \overline{\phi_{i^*}}(x_{i^*})$). The way to choose the index i^* for subdividing as well as the construction of the next node in B&B scheme will be described below.

Suppose that at iteration k, a subproblem of (P) has the form :

$$\min\left\{\sum_{i=1}^{n} \phi_i(x_i) \mid x \in C \cap R_k\right\} \tag{3.14}$$

with $R_k = \{x \in R_0 \mid x_i = 0 \ \forall i \in I_k,\ x_j \neq 0 \ \forall j \in J_k\}$, $I_k, J_k \subset \{1, 2, \ldots, n\}$ (for the first iteration, we have $I_0 = \emptyset, J_0 = \emptyset$). Let x^{R_k} be an optimal solution of the corresponding relaxation problem of (3.14) over $C \cap R_k$. The index i^* satisfying the following condition will be chosen

$$i^* \in \arg\max_i \{\phi_i(x_i^{R_k}) - \widetilde{\phi}_i(x_i^{R_k})\}.$$

In branching procedure, by adding, either $x_{i^*} = 0$ or $x_{i^*} \neq 0$ into constraints, (3.14) will be bisected into two smaller subproblems :

$$\min\left\{\sum_{i=1}^{n} \phi_i(x_i) \mid x \in C \cap R_k,\ x_{i^*} = 0\right\} \text{ and}$$
$$\min\left\{\sum_{i=1}^{n} \phi_i(x_i) \mid x \in C \cap R_k,\ x_{i^*} \neq 0\right\}.$$

Let

$$R_{k_0} = \{x \in R_k \mid x_{i^*} = 0\}, \quad R_{k_1} = \{x \in R_k \mid x_{i^*} \neq 0\}.$$

As we have

$$\min\left\{\sum_{i=1}^{n} \phi_i(x_i) \mid x \in C \cap R_k\right\} = \min\left\{\min\left\{\sum_{i \neq i^*} \phi_i(x_i) \mid x \in C \cap R_k,\ x_{i^*} = 0\right\},\right.$$
$$\left.\min\left\{\overline{\phi_{i^*}}(x_{i^*}) + \sum_{i \neq i^*} \phi_i(x_i) \mid x \in C \cap R_k\right\}\right\}$$

then each subproblem of (P) being considered on $C \cap R_k$ can be written in the ensuing form :

$$\min\left\{\sum_{j \in J_k} \overline{\phi_j}(x_j) + \sum_{i \notin I_k \cup J_k} \phi_i(x_i) \mid x \in C \cap R_0,\ x_i = 0,\ \forall i \in I_k\right\} \tag{P_k}$$

(note that here $R_k = \{x \in R_0 \mid x_i = 0 \ \forall i \in I_k,\ x_j \neq 0 \ \forall j \in J_k\}$).

Thus a relaxation problem of (P_k) is :

$$\min\left\{\sum_{j \in J_k} \overline{\phi_j}(x_j) + \sum_{i \notin I_k \cup J_k} \widetilde{\phi}_i(x_i) \mid x \in C \cap R_0,\ x_i = 0,\ \forall i \in I_k\right\} \tag{$R_k cp$}$$

and an approximated DC problem for (P_k) is constructed as follows :

$$\begin{cases} \min \left(\sum_{j\in J_k} \overline{\phi_j}(x_j) + \sum_{i\notin I_k\cup J_k} g_i(x_i) \right) - \left(\sum_{i\notin I_k\cup J_k} h_i(x_i) \right) \\ \text{s.t. } x \in C \cap R_0 \\ \qquad x_i = 0,\ \forall i \in I_k \end{cases} \tag{R_k^{dc}}$$

where $G_k(x) = \sum\limits_{j\in J_k} \overline{\phi_j}(x_j) + \sum\limits_{i\notin I_k\cup J_k} g_i(x_i)$ and $H_k(x) = \sum\limits_{i\notin I_k\cup J_k} h_i(x_i)$ are convex functions.

Applying DCA on (R_k^{dc}) leads to constructing two sequences $\{x^s\}$ and $\{y^s\}$ such that

$\diamond$ $x^s \longrightarrow y^s \in \partial H_k(x^s)$,

$y_i^s = 0$, if $i \in I_k \cup J_k$,

$$y_i^s = \begin{cases} -c_i^1, & x_i^s < -\epsilon_i \\ \in [-c_i^1, -\alpha_i^1], & x_i^s = -\epsilon_i \\ -\alpha_i^1, & -\epsilon_i < x_i^s < 0 \\ \in [-\alpha_i^1, \alpha_i^2], & x_i^s = 0\ , \quad \text{if } i \notin I_k \cup J_k. \\ \alpha_i^2, & 0 < x_i^s < \epsilon_i \\ \in [\alpha_i^2, c_i^2], & x_i^s = \epsilon_i \\ c_i^2, & x_i^s > \epsilon_i \end{cases}$$

$\diamond$ $y^s \longrightarrow x^{s+1} \in \partial G_k^*(y^s) \Longleftrightarrow x^{s+1}$ solves the convex program below :

$$\min \{G_k(x) - \langle y^s, x\rangle \mid x \in C \cap R_0,\ x_i = 0,\ \forall i \in I_k\} \tag{3.15}$$

Solving $(R_k cp)$ is equivalent to solving

$$\begin{cases} \min \ \sum_{i=1}^{n} t_i \\ \text{s.t. } \beta_j - \alpha_j^1 x_j \le t_j,\ \forall j \in J_k, \\ \qquad \beta_j + \alpha_j^2 x_j \le t_j,\ \forall j \in J_k, \\ \qquad \left(\frac{\beta_i}{l_i} - \alpha_i^1\right) x_i \le t_i,\ \forall i \notin I_k \cup J_k, \\ \qquad \left(\frac{\beta_i}{u_i} + \alpha_i^2\right) x_i \le t_i,\ \forall i \notin I_k \cup J_k, \\ \qquad x \in C \cap R_0, \\ \qquad x_i = 0,\ \forall i \in I_k, \\ \qquad t_i = 0,\ \forall i \in I_k, \\ \qquad t_j \ge \beta_j,\ \forall j \in J_k, \\ \qquad t_i \ge 0,\ \forall i \notin I_k \cup J_k. \end{cases} \tag{3.16}$$

And solving (3.15) is equivalent to solving

$$\begin{cases} \min \ -\langle y^s, x\rangle + \sum_{i=1}^{n} t_i \\ \text{s.t. } \beta_j - \alpha_j^1 x_j \le t_j, \ \forall j \in J_k, \\ \quad \beta_j + \alpha_j^2 x_j \le t_j, \ \forall j \in J_k, \\ \quad \beta_i - (c_i^1 + \alpha_i^1) x_i \le t_i, \ \forall i \notin I_k \cup J_k, \\ \quad \beta_i + (c_i^2 + \alpha_i^2) x_i \le t_i, \ \forall i \notin I_k \cup J_k, \\ \quad x \in C \cap R_0, \\ \quad x_i = 0, \ \forall i \in I_k, \\ \quad t_i = 0, \ \forall i \in I_k, \\ \quad t_i \ge \beta_i, \ \forall i \notin I_k. \end{cases} \tag{3.17}$$

Let x^{R_k} be an optimal solution of $(R_k cp)$. Since x^{R_k} is a feasible solution to (P), $\phi(x^{R_k})$ is an upper bound for the global optimal value ω of (P). To use DCA for finding a better upper bound for ω, we construct an approximated DC problem (R_k^{dc}) for (P_k) as above and launch DCA from x^{R_k} for solving (R_k^{dc}). Note that we do not restart DCA at every iteration of B&B scheme but only when $\phi(x^{R_k})$ is smaller than the best current upper bound.

We are now in position to describe our hybrid algorithm for solving (P).

Algorithm 2 (DCA-BB 1)

- **Initialization :**
  Compute the first bounds $[l_i^0, u_i^0]$ for variables x_i and the first rectangle $R_0 = \prod_{i=1}^{n} [l_i^0, u_i^0]$.
  Construct the convex underestimator function $\widetilde{\phi}_{R_0}$ of ϕ over R_0 then solve the convex program
  $$\min\{\widetilde{\phi}_{R_0}(x) \mid x \in C \cap R_0\} \tag{$R_0 cp$}$$
  to obtain an optimal solution x^{R_0} and the optimal value $\eta(R_0)$ of $(R_0 cp)$.
  Launch DCA from x^{R_0} for solving the corresponding DC approximation problem (P_{dc}). Let $\overline{x}^{R_0}$ be a solution obtained by DCA.
  Set $\mathcal{R}_0 := \{R_0\}$ and $\eta_0 := \eta(R_0)$.
  If $\phi(\overline{x}^{R_0}) < \phi(x^{R_0})$, set $\omega_0 := \phi(\overline{x}^{R_0})$ and $x^* := \overline{x}^{R_0}$.
  Otherwise, set $\omega_0 := \phi(x^{R_0})$ and $x^* := x^{R_0}$.
- **Iteration** $k = 0, 1, 2, \ldots$:
  $k.1$ Delete all $R \in \mathcal{R}_k$ if $\eta(R) \ge \omega_k$ or if the corresponding relaxation problem is infeasible on R. Let $\mathcal{P}_k$ be the set of remaining rectangles. If $\mathcal{P}_k = \emptyset$ then STOP : x^* is a global optimal solution.
  $k.2$ Otherwise, select $R_k \in \mathcal{P}_k$ such that
  $$\eta_k := \eta(R_k) = \min\{\eta(R) \mid R \in \mathcal{P}_k\}$$
  and let i^* be the selected index according to the subdivision process. Subdivide R_k into R_{k_0}, R_{k_1} by setting x_{i^*} either to zero or to be nonzero.

$k.3$ For each R_{k_j}, $j = 0, 1$, with corresponding sets of indices I_{k_j}, J_{k_j} such that $x_i = 0, \forall i \in I_{k_j},\ x_i \neq 0, \forall i \in J_{k_j}$, construct the subproblem (P_{k_j}) of (P) at this node and construct the relaxation problem $(R_{k_j}cp)$ of (P_{k_j}) :

$$\min \left\{ \sum_{i \in J_{k_j}} \overline{\phi_i}(x_i) + \sum_{i \notin I_{k_j} \cup J_{k_j}} \widetilde{\phi_i}(x_i) \mid x \in C \cap R_0,\ x_i = 0,\ \forall i \in I_{k_j} \right\} \qquad (R_{k_j}cp)$$

Solving $(R_{k_j}cp)$ to obtain $x^{R_{k_j}}$ and $\eta(R_{k_j})$.
If $\phi(x^{R_{k_j}}) < \omega_k$, i.e., the best current upper bound is improved on R_{k_j} then construct a DC approximation problem $(R_{k_j}^{dc})$ for (P_{k_j})

$$\left\{ \begin{array}{l} \min \quad \left(\displaystyle\sum_{i \in J_{k_j}} \overline{\phi_i}(x_i) + \sum_{i \notin I_{k_j} \cup J_{k_j}} g_i(x_i) \right) - \left(\sum_{i \notin I_{k_j} \cup J_{k_j}} h_i(x_i) \right) \\ \quad \text{s.t. } x \in C \cap R_0, \\ \qquad x_i = 0,\ \forall i \in I_{k_j}. \end{array} \right. \qquad (R_{k_j}^{dc})$$

and launch DCA from $x^{R_{k_j}}$ for solving it.
Let $\overline{x}^{R_{k_j}}$ be a solution obtained by DCA. Let

$$\gamma_k = \min\{\phi(x^{R_{k_j}}), \phi(\overline{x}^{R_{k_j}})\}.$$

$k.4$ Update ω_{k+1} and the best solution known so far x^*.
$k.5$ Set $\mathcal{R}_{k+1} = (\mathcal{P}_k \setminus R_k) \bigcup \{R_{k_1}, R_{k_2}\}$ and go to the next iteration.

We will present in the next section the second approach for solving (P) by introducing binary variables into (P). The equivalent model of (P) is a nonconvex mixed zero-one programming problem.

3.4 Solving (P) by a zero-one approach

Consider now the case where the cost rate associated with selling and buying are the same : $\alpha_i^1 = \alpha_i^2 = \alpha_i, \forall i = 1, \ldots, n$,

$$\phi_i(x_i) = \begin{cases} 0, & x_i = 0 \\ \beta_i + \alpha_i |x_i|, & x_i \neq 0. \end{cases} \qquad (3.18)$$

Recall that the problem (P) is the following

$$\omega = \min \left\{ \phi(x) = \sum_{i=1}^{n} \phi_i(x_i) \mid x \in C,\ l_i^0 \leq x_i \leq u_i^0, \forall i = 1, \ldots, n \right\} \qquad (P)$$

We present below the reformulation of (P) as a mixed zero-one programming problem and then describe a DCA scheme for tackling the resulting program.

3.4.1 A mixed zero-one formulation

Suppose lower bounds l_i^0 and upper bounds u_i^0 for x_i are calculated. We introduce n binary variables y_i such that $y_i = 0$ if and only if $x_i = 0$, and $y_i = 1$ if $x_i \neq 0$. Then $\phi_i(x_i) = (\beta_i + \alpha_i|x_i|)y_i,\ \forall i = 1, \ldots, n$, and we have

$$\forall i,\ l_i^0 \leq x_i \leq u_i^0 \Longleftrightarrow \begin{cases} y_i \in \{0,1\}, \\ l_i^0 y_i \leq x_i \leq u_i^0 y_i. \end{cases} \tag{3.19}$$

Each ϕ_i is now considered as a function of two variables x_i, y_i and we can replace $\phi_i(x_i)$ by $\varphi_i(x_i, y_i) := (\beta_i + \alpha_i|x_i|)y_i$.

The mixed 0-1 programming formulation of (P) is

$$\begin{cases} \omega = \min\ \varphi(x,y) := \displaystyle\sum_{i=1}^{n} \varphi_i(x_i, y_i) \\ \text{s.t. } x \in C, \\ \quad y_i \in \{0,1\},\ \forall i = 1, \ldots, n, \\ \quad l_i^0 y_i \leq x_i \leq u_i^0 y_i,\ \forall i = 1, \ldots, n. \end{cases} \tag{$Q01$}$$

3.4.2 DC programming and DCA for solving $(Q01)$

Firstly we claim that the objective function of $(Q01)$ is a DC function on $\mathbb{R}^n \times [0,1]^n$. Indeed, we have

$$\begin{aligned} \varphi_i(x_i, y_i) &= (\beta_i + \alpha_i|x_i|)y_i = \alpha_i|x_i|y_i + \beta_i y_i \\ &= \frac{\alpha_i}{2}[(|x_i| + y_i)^2 - (x_i^2 + y_i^2)] + \beta_i y_i \\ &= \Big(\frac{\alpha_i}{2}(|x_i| + y_i)^2\Big) - \Big(\frac{\alpha_i}{2}(x_i^2 + y_i^2) - \beta_i y_i\Big) \end{aligned}$$

Let

$$\theta_i(x_i, y_i) = \frac{\alpha_i}{2}(|x_i| + y_i)^2 \text{ and } \kappa_i(x_i, y_i) = \frac{\alpha_i}{2}(x_i^2 + y_i^2) - \beta_i y_i$$

then θ_i, κ_i are convex functions on $\mathbb{R} \times [0,1]$ since $|x_i|, y_i \geq 0\ \forall (x_i, y_i) \in \mathbb{R} \times [0,1]$. Let

$$\theta(x,y) = \sum_{i=1}^{n} \theta_i(x_i, y_i) \text{ and } \kappa(x,y) = \sum_{i=1}^{n} \kappa_i(x_i, y_i)$$

then $\theta - \kappa$ is a DC decomposition of φ.

Furthermore, let

$$A := \{(x,y) \in \mathbb{R}^n \times [0,1]^n \mid x \in C,\ l_i^0 y_i \leq x_i \leq u_i^0 y_i,\ \forall i = 1, \ldots, n\}$$

and define the function

$$p : \mathbb{R}^n \times \mathbb{R}^n \longrightarrow \mathbb{R},\ p(x,y) := \sum_{i=1}^{n} y_i(1 - y_i)$$

then p is a nonnegative concave function on the convex set A and the feasible set of $(Q01)$ is

$$\{(x,y) \in A \mid y_i \in \{0,1\}, \forall i\} = \{(x,y) \in A \mid p(x,y) = 0\} = \{(x,y) \in A \mid p(x,y) \leq 0\}.$$

Thus ($Q01$) becomes

$$\min\{\varphi(x,y) = \theta(x,y) - \kappa(x,y) \mid (x,y) \in A,\ p(x,y) \leq 0\}. \tag{3.20}$$

Note that the objective function of (3.20) is a DC function and (3.20) contains a reverse convex constraint : $p(x,y) \leq 0$. In order to overcome these difficulties when solving (3.20), we propose to use the penalty technique. More precisely, we will transform (3.20) into a DC program and then use DCA for solving the resulting problem.

With a scalar $t > 0$, we define a penalty function F_t of φ on $\mathbb{R}^n \times \mathbb{R}^n$ by

$$F_t(x,y) = \varphi(x,y) + tp(x,y) = \theta(x,y) - (\kappa(x,y) - tp(x,y)) \tag{3.21}$$

then the penalized problem for (3.20) can be

$$\min\{F_t(x,y) = \theta(x,y) - (\kappa(x,y) - tp(x,y)) \mid (x,y) \in A\} \tag{3.22}$$

or in a natural DC form

$$\min\{G(x,y) - H(x,y) \mid (x,y) \in \mathbb{R}^n \times \mathbb{R}^n\}$$

where

$$G(x,y) = \theta(x,y) + \chi_A(x,y) \text{ and } H(x,y) = \kappa(x,y) - tp(x,y)$$

(here χ_A is the indicator function of A).

Applying DCA to solve (3.22) leads to compute $(z^k, v^k) \in \partial H(x^k, y^k)$ at each iteration k, and then solve the convex program

$$\min\{G(x,y) - \langle (z^k, v^k), (x,y)\rangle \mid (x,y) \in \mathbb{R}^n \times \mathbb{R}^n\}$$

to obtain $(x^{k+1}, y^{k+1}) \in \partial G^*(z^k, v^k)$. This convex program is equivalent to the following

$$\min\{\theta(x,y) - \langle (z^k, v^k), (x,y)\rangle \mid (x,y) \in A\} \tag{3.23}$$

Since we have

$$\begin{aligned} H(x,y) &= \kappa(x,y) - tp(x,y) \\ &= \sum_{i=1}^{n} \Big(\frac{\alpha_i}{2}(x_i^2 + y_i^2) - \beta_i y_i\Big) - t\sum_{i=1}^{n} y_i(1 - y_i) \\ &= \sum_{i=1}^{n} \frac{\alpha_i}{2} x_i^2 + \sum_{i=1}^{n} \Big((\frac{\alpha_i}{2} + t)y_i^2 - (\beta_i + t)y_i\Big) \end{aligned}$$

then a subgradient of H, $(z^k, v^k) \in \partial H(x^k, y^k)$, is computed by

$$(z^k, v^k) \in \partial H(x^k, y^k) \Longleftrightarrow \begin{cases} z_i^k = \alpha_i x_i^k,\ \forall i = 1, \ldots, n, \\ v_i^k = (\alpha_i + 2t)y_i^k - (\beta_i + t),\ \forall i = 1, \ldots, n. \end{cases} \tag{3.24}$$

In order to use a standard CPLEX solver for solving the convex program (3.23) which is exactly

$$\min\left\{\sum_{i=1}^{n} \frac{\alpha_i}{2}(|x_i| + y_i)^2 - \langle (x,y), (z^k, v^k)\rangle \mid (x,y) \in A\right\}, \tag{3.25}$$

we introduce nonnegative variables $\zeta_i, i = 1, \ldots, n$, and transform (3.25) into an equivalent quadratically constrained quadratic program

$$\begin{cases} \min \sum_{i=1}^{n} \frac{\alpha_i}{2}(y_i + \zeta_i)^2 - \langle (x, y, \zeta), (z^k, v^k, 0) \rangle \\ \text{s.t. } (x, y) \in A, \\ \quad x_i + \zeta_i \geq 0, \ \forall i = 1, \ldots, n, \\ \quad -x_i + \zeta_i \geq 0, \ \forall i = 1, \ldots, n, \\ \quad \zeta_i \geq 0, \ \forall i = 1, \ldots, n. \end{cases} \tag{3.26}$$

We describe the DCA applied on (3.22) as follows.

Algorithm 3 (DCA for 01 model) :

- **Initialization :**
  Let $(x^0, y^0) \in \mathbb{R}^n \times [0, 1]^n$ and ε be a small enough positive number.
  Iteration $k \longleftarrow 0$.
- **Repeat :**
  - $\diamond$ Calculate $z_i^k = \alpha_i x_i^k$ and $v_i^k = (\alpha_i + 2t)y_i^k - (\beta_i + t), \ \forall i = 1, \ldots, n$.
  - $\diamond$ Solve (3.25) to obtain (x^{k+1}, y^{k+1}).
  - $\diamond$ $k \longleftarrow k + 1$
- **Until :** $|F(x^{k+1}, y^{k+1}) - F(x^k, y^k)| \leq \varepsilon$ or $\|x^{k+1} - x^k\| + \|y^{k+1} - y^k\| \leq \varepsilon$.

3.4.3 A combined DCA-Branch and Bound algorithm for solving ($Q01$)

To evaluate the globality of solutions computed by DCA in Algorithm 3, we propose to solve ($Q01$) or its equivalent problem (3.20) by a B&B algorithm. The subdivision of the combined algorithm is performed in the way that either $y_i = 0$ or $y_i = 1$.

For lower bounding, we use the same method as presented in Algorithm 2, that means constructing and solving the relaxation problem ($R_k cp$) of ϕ on $C \cap R_k$ at iteration k in the B&B scheme.

Suppose that at iteration k in the B&B scheme, we have two sets of indices $I_k, J_k \subset \{1, \ldots, n\}$ such that $y_i = 0 \ \forall i \in I_k$, $y_j = 1 \ \forall j \in J_k$. The corresponding DC objective function at this iteration is

$$\begin{aligned} \varphi^k(x, y) &= \sum_{i \notin I_k \cup J_k} \varphi_i(x_i, y_i) + \sum_{j \in J_k} (\alpha_j |x_j| + \beta_j) \\ &= \left(\sum_{i \notin I_k \cup J_k} \theta_i(x_i, y_i) + \sum_{j \in J_k} (\alpha_j |x_j| + \beta_j) \right) - \left(\sum_{i \notin I_k \cup J_k} \kappa_i(x_i, y_i) \right) \end{aligned} \tag{3.27}$$

The penalty function of φ^k at iteration k, with $t > 0$,

$$\begin{aligned} F_t^k(x, y) &= \varphi^k(x, y) + tp(x, y) \\ &= \left(\sum_{i \notin I_k \cup J_k} \theta_i(x_i, y_i) + \sum_{j \in J_k} (\alpha_j |x_j| + \beta_j) \right) \end{aligned}$$

$$- \left(\sum_{i \notin I_k \cup J_k} (\kappa_i(x_i, y_i) + t y_i(y_i - 1)) \right)$$

is a DC function with DC components

$$\begin{aligned} G^k(x,y) &= \sum_{i \notin I_k \cup J_k} \theta_i(x_i, y_i) + \sum_{j \in J_k} (\alpha_j |x_j| + \beta_j) \\ &= \sum_{i \notin I_k \cup J_k} \frac{\alpha_i}{2} (|x_i| + y_i)^2 + \sum_{j \in J_k} (\alpha_j |x_j| + \beta_j), \end{aligned}$$

$$\begin{aligned} H^k(x,y) &= \sum_{i \notin I_k \cup J_k} (\kappa_i(x_i, y_i) + t y_i(y_i - 1)) \\ &= \sum_{i \notin I_k \cup J_k} \left(\frac{\alpha_i}{2} (x_i^2 + y_i^2) - \beta_i y_i + t y_i(y_i - 1) \right). \end{aligned}$$

The penalty problem of (3.20) at iteration k can be given by

$$\min\{F_t^k(x,y) \mid (x,y) \in A,\ y_i = 0,\ \forall i \in I_k,\ y_j = 1,\ \forall j \in J_k\} \tag{3.28}$$

Solving (3.28) by DCA leads to determine two sequences $\{(x^s, y^s)\}$, $\{(z^s, v^s)\}$ in $\mathbb{R}^n \times \mathbb{R}^n$ satisfying

$$(z^s, v^s) \in \partial H^k(x^s, y^s) \Longleftrightarrow \begin{cases} z_i^s = \alpha_i x_i^k, \text{ if } i \notin I_k \cup J_k, \\ z_i^s = 0, \text{ if } i \in I_k \cup J_k, \\ v_i^s = (\alpha_i + 2t) y_i^k - (\beta_i + t), \text{ if } i \notin I_k \cup J_k, \\ v_i^s = 0, \text{ if } i \in I_k \cup J_k, \end{cases} \tag{3.29}$$

and

$(x^{s+1}, y^{s+1}) \in \partial (G^k)^*(z^s, v^s) \Longleftrightarrow (x^{s+1}, y^{s+1})$ solves the convex program

$$\begin{cases} \min\ G^k(x,y) - \langle (z^s, v^s), (x,y) \rangle \\ \text{s.t. } (x,y) \in A, \\ \quad y_i = 0, i \in I_k, \\ \quad y_j = 1, j \in J_k \end{cases} \tag{3.30}$$

or

$$\begin{cases} \min \displaystyle\sum_{i \notin I_k \cup J_k} \frac{\alpha_i}{2} (|x_i| + y_i)^2 + \sum_{j \in J_k} (\alpha_j |x_j| + \beta_j) - \langle (z^s, v^s), (x,y) \rangle \\ \text{s.t. } (x,y) \in A, \\ \quad y_i = 0, i \in I_k, \\ \quad y_j = 1, j \in J_k \end{cases} \tag{3.31}$$

By introducing nonnegative variables $\zeta_i, i = 1, \ldots, n$, solving (3.31) is equivalent to

solving the convex quadratically constrained quadratic program :

$$\begin{cases} \min \dfrac{1}{2} \displaystyle\sum_{i \notin I_k \cup J_k} \alpha_i (y_i + \zeta_i)^2 + \sum_{j \in J_k} (\alpha_j \zeta_j + \beta_j) - \langle (z^s, v^s, 0), (x, y, \zeta) \rangle \\ \text{s.t. } (x, y) \in A, \\ \quad x_i + \zeta_i \geq 0, \ \forall j \notin I_k, \\ \quad -x_i + \zeta_i \geq 0, \ \forall j \notin I_k, \\ \quad y_i = 0, \ \forall i \in I_k, \\ \quad y_j = 1, \ \forall j \in J_k, \\ \quad \zeta_i = 0, \ \forall i \in I_k, \\ \quad \zeta_i \geq 0, \ \forall i \notin I_k. \end{cases} \tag{3.32}$$

If solving the penalty problem (3.28) by DCA provides $(\overline{x}, \overline{y})$ as a solution then $\varphi(\overline{x}, \overline{y})$ is an upper bound for ω.

Therefore, we can describe below a combined algorithm for solving (3.20).

Hybrid approach for solving ($Q01$)

Algorithm 4 (DCA-BB for 01 model)

- **Initialization :**
  Compute the first bounds $[l_i^0, u_i^0]$ for variables x_i and the first rectangle $R_0 = \prod_{i=1}^{n} [l_i^0, u_i^0]$.
  $I_0 := \emptyset$, $J_0 := \emptyset$, iteration $k \longleftarrow 0$.
  The optimal value μ_0 of the relaxation problem (R_0cp) of (P) provides the first lower bound for ω.
- **Iteration k :**
  We apply DCA described in Algorithm 3 inside B&B algorithm. DCA is used in the first time at the end of the first iteration of Branch and Bound scheme (iteration 0). And then, in the branch and bound process, we restart DCA when the current upper bound is updated. More precisely, the DCA using inside the branch and bound algorithm is carried on as follows :
  1. Construct the current relaxation problem (R_kcp) of $(Q01)$ at the node k. Solve (R_kcp) to obtain a lower bound for ω at this node and a solution $x^{R_k} \in C \cap R_0$.
  2. If $\phi(x^{R_k})$ is smaller than the current upper bound then construct the penalty problem of the form (3.28) and launch DCA for solving it.
  3. Let $(\widetilde{x}^1, \widetilde{y}^1)$ be the solution obtained by DCA. Let ϵ be a sufficiently small positive number. For each $i \notin I_k \cup J_k$, if $\widetilde{y}_i^1 \leq \epsilon$ then add the constraints $x_i = 0$ into the set of constraints of (R_kcp) and add this index i into I_k. Name the new problem (P'_{re}).
  4. Solve (P'_{re}).
     - If (P'_{re}) provides a solution then launch DCA for solving the corresponding penalty problem of $(Q01)$ (constructed with the new I_k) to obtain $(\widetilde{x}^2, \widetilde{y}^2)$. Update the upper bound, the best current solution known so far by comparing $\phi(x^{R_k})$ with $\phi(\widetilde{x}^1)$ and $\phi(\widetilde{x}^2)$ then return to the branch and bound algorithm.
     - If (P'_{re}) is infeasible, update the upper bound, the best current solution and return to the branch and bound algorithm.

5. Continue the branch and bound process until the convergence.

Numerical experiments in the next section show the efficiency of the proposed algorithms.

3.5 Computational results

The algorithms are coded in C and run on a PC equipped with Window 7 Intel(R) Core(TM) i5-2540M CPU 2.60GHz, 8.00 Go RAM. To solve the convex quadratically constrained linear/quadratic programs, we use CPLEX solver version 12.4.

We have tested four algorithms on the set of data used in [108]. The portfolio selection consists of $(n-1)$ risky and one riskless assets (the riskless asset corresponds to the n^{th}-asset in the portfolio decision). The mean and covariance of $(n-1)$ risky assets were estimated from daily closing prices of S&P 500 stocks (for the tests with $n \leq 101$, we chose the first $(n-1)$ stocks, alphabetically by ticker, with a full year of data from January 9, 1998 to January 8, 1999; for $n > 101$, the first $(n-1)$ stocks were chosen with the data from January 01, 2005 to January 01, 2007). The mean of riskless asset is set to be 0.1.

The results presented in the table 3.1 have been computed using the values

$$\begin{aligned}
&w_i = 1/n,\ \forall i = 1, \ldots, n\\
&\alpha_i^1 = \alpha_i^2 = \alpha_i = 0.01,\ \forall i = 1, \ldots, n-1, \quad \alpha_n^1 = \alpha_n^2 = \alpha_n = 0\\
&\beta_i = 0.1/(n-1),\ \forall i = 1, \ldots, n-1, \quad \beta_n = 0\\
&s_i = 5\beta_i,\ \forall i = 1, \ldots, n-1, \quad s_n = 0.5\\
&\lambda_i = 0.5,\ \forall i = 1, \ldots, n.
\end{aligned}$$

We have tested Algorithm 1 (denoted DCA), Algorithm 2 (denoted DCA-BB), Algorithm 3 (denoted DCA for 01 model) and Algorithm 4 (denoted DCA-BB for 01 model) also the branch and bound algorithm without DCA (denoted BB). The tolerance ε for stopping DCA is equal to 10^{-8}. The stopping criteria of the branch and bound algorithm (with DCA or without DCA) is either the CPU time (in seconds) is greater than 1 hour or the difference between the best upper bound and the best lower bound is smaller than $\epsilon := 10^{-8}$. In our numerical tests, two hybrid algorithms and the BB algorithm always provide an ϵ-optimal solution. In Table 3.1, the number of iterations for each algorithm as well as the ϵ-optimal values found by BB, DCA-BB, "DCA-BB for 01 model" and the CPU time are presented. The values obtained by DCA are shown in the same table.

Comments on the numerical results.

From numerical results we observed that

- DCA and "DCA for 01 model" provide usually a good approximation of the optimal solution within a very short running time (less than 5 seconds) and the number of iterations of DCA and "DCA for 01 model" are less than 5.
- The combined algorithms DCA-BB and "DCA-BB for 01 model" (in which the number of restarting DCA is less than 10) provide the same optimal values in comparison with the classical branch and bound algorithm (BB) within a bit larger CPU time when $n \leq 161$. However, in the last four cases, when $n = 171, 181, 191, 201$, respectively, we can observe the performance of DCA when combining it with BB : it

	BB			DCA-BB				DCA			DCA-BB for 01 model				DCA for 01 model		
n	iter	Opt.val	CPU	iter	rest.	Opt.val	CPU	iter	valDCA	CPU	iter	rest.	Opt.val	CPU	iter	valDCA	CPU
11	51	0.104149	2.699	51	7	0.104149	3.198	2	0.104490	0.078	50	4	0.104149	2.684	2	0.104544	0.140
21	35	0.089724	1.077	35	2	0.089724	1.263	2	0.089742	0.046	84	4	0.089724	3.447	4	0.089742	0.593
31	31	0.086269	1.685	31	2	0.086269	1.996	2	0.089051	0.124	30	2	0.086269	1.981	4	0.087762	0.390
41	41	0.085362	3.260	41	4	0.085362	3.728	2	0.088558	0.109	40	3	0.085362	4.557	4	0.086853	0.889
51	51	0.084845	7.005	51	3	0.084845	7.161	2	0.087046	0.172	50	3	0.084845	7.035	3	0.086337	0.514
61	70	0.084487	11.732	70	6	0.084487	12.691	2	0.085581	0.219	100	2	0.084487	15.418	3	0.085977	0.983
71	159	0.084228	27.877	159	7	0.084228	28.611	2	0.085427	0.265	242	4	0.084228	35.100	4	0.085719	0.827
81	181	0.083878	35.631	181	8	0.083878	39.343	2	0.086047	0.249	428	4	0.083878	76.767	3	0.085530	1.029
91	3803	0.083601	956.046	3803	8	0.083601	982.918	3	0.086280	0.437	3802	4	0.083601	786.803	3	0.085382	1.045
101	4877	0.083683	1573.743	4877	9	0.083683	1609.035	2	0.086818	0.374	4876	3	0.083683	1109.761	4	0.085208	1.310
111	189	0.084263	51.995	189	9	0.084263	56.690	2	0.084967	0.640	1116	3	0.084263	305.000	2	0.085251	1.560
121	316	0.084130	106.190	316	8	0.084130	116.230	2	0.084734	0.765	1403	4	0.084130	430.733	2	0.085169	1.794
131	298	0.083862	101.924	298	6	0.083862	113.374	2	0.084424	1.778	561	3	0.083862	225.102	2	0.085097	2.153
141	495	0.083775	181.563	495	3	0.083775	191.194	2	0.084254	0.936	627	5	0.083775	272.205	2	0.085039	2.262
151	364	0.083678	139.508	364	4	0.083678	142.824	2	0.084109	1.640	666	4	0.083678	297.383	2	0.084989	2.793
161	612	0.083593	281.261	612	4	0.083593	288.858	2	0.083982	1.912	628	5	0.083593	329.627	2	0.084945	2.730
171	721	0.083516	519.326	525	4	0.083516	428.666	2	0.083867	2.158	661	5	0.083516	389.767	2	0.084905	2.715
181	986	0.083447	788.477	563	3	0.083447	448.256	2	0.083765	2.586	874	5	0.083447	590.711	2	0.084869	3.291
191	1135	0.083386	982.932	713	2	0.083386	759.186	2	0.0836755	2.898	915	6	0.083386	689.646	2	0.084837	4.337
201	1643	0.083329	1781.377	751	2	0.083329	848.047	2	0.083592	2.839	1348	5	0.083329	981.273	2	0.084807	4.103

TABLE 3.1 – Minimize Transaction costs

greatly reduces the number of iterations of the branch-and-bound process and the computation time of BB (without DCA) is really more expensive.

3.6 Conclusion

In this chapter, we have rigorously studied the model and solution methods for solving a hard portfolio selection problem where the total transaction cost function is nonconvex, discontinuous. Attempting to use DC programming and DCA, an efficient approach in nonconvex programming, in the first approach, we have constructed an appropriate DC approximation of the objective function, and then investigated a DCA scheme for solving the resulting DC program. The DCA based algorithm is quite simple : at each iteration we have to minimize a linear function under linear constraints and one convex quadratic constraint for which the powerful CPLEX solver can be used. To get a global minimizer of the original problem we combined DCA with a Branch and Bound scheme. The way to compute lower bounds leads to the same type of convex subproblems in DCA, say linear program with additional one convex quadratic constraint.

In the second approach, by introducing binary variables, we transformed the original problem into a nonconvex mixed zero-one program, and based on DC programming and DCA and penalty techniques, the solution method for the resulting problem has been developed. In this approach, CPLEX solver was used for minimizing a quadratically constrained quadratic program. A hybrid algorithm DCA-BB for solving 0-1 model was also proposed.

The computational aspects of the proposed approaches show the efficiency of DCA, its inexpensiveness and also the positive influence of DCA on branch and bound algorithm, specially for large-scale problems.

Chapitre 4

Programmation linéaire multi-objectif en variables mixtes zéro-un

---

Dans ce chapitre nous présentons une nouvelle méthode qui peut être considérée comme une généralisation de la méthode de Benson pour la résolution exacte du problème linéaire multi-objectif en variables mixtes zéro-un.

4.1 Introduction

A multiobjective mixed 0-1 linear program with p objective functions ($p \geq 2$) has the following form :

$$\begin{cases} \min\ f(x) := (f_1(x), \ldots, f_p(x)) = Cx \\ \text{s.t. } Ax \leqq b, \\ \quad x_i \geq 0, \forall i = 1, \ldots, n, \\ \quad x_j \in \{0,1\}, j \in J. \end{cases} \tag{P}$$

with $f_k(x) = c_k^T x,\ k = 1, \ldots, p$, the decision matrix $C \in \mathbb{R}^{p \times n}$ having c_k^T as k^{th}-row vectors, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $J \subset \{1, \ldots, n\}$ and componentwise orders in $\mathbb{R}^p$ are defined as follows : let $y^1, y^2 \in \mathbb{R}^p$,

$$\begin{aligned} &y^1 \leqq y^2 \text{ or } y^2 \geqq y^1 \text{ if } y_k^1 \leq y_k^2 \ \ \forall k = 1, \ldots, p \\ &y^1 \leq y^2 \text{ or } y^2 \geq y^1 \text{ if } y_k^1 \leq y_k^2 \ \ \forall k = 1, \ldots, p \text{ and } y^1 \neq y^2 \\ &y^1 < y^2 \text{ or } y^2 > y^1 \text{ if } y_k^1 < y_k^2 \ \ \forall k = 1, \ldots, p. \end{aligned}$$

Define $\mathbb{R}^p_{\geqq} := \{y \in \mathbb{R}^p \mid y \geqq 0\}$ and analogously for $\mathbb{R}^p_{\geq}$ and $\mathbb{R}^p_{>}$.

Let

$$\mathcal{X} = \left\{ x \in \mathbb{R}^n_{\geqq} \mid Ax \leqq b, x_j \in \{0,1\}, \forall j \in J \right\}.$$

$\mathcal{X}$ is called *feasible set* of (P).

The *outcome set* $\mathcal{Y}$ of (P) is defined by $\mathcal{Y} := \{y \in \mathbb{R}^p \mid y = Cx, x \in \mathcal{X}\}$. Assume that no feasible solution minimizes simultaneously all objectives.

Let $x^* \in \mathcal{X}$ be a feasible solution of the (P) and let $y^* = Cx^*$.

Definition 4.1 [39] *x^* is called* efficient solution *or* Pareto optimal *of (P) if there is no other feasible solution $x \in \mathcal{X}$ such that $Cx \leq Cx^*$. If x^* is efficient, $y^* = Cx^*$ is called* nondominated point*. If $x^1, x^2 \in \mathcal{X}$ are such that $Cx^1 \leq Cx^2$, we say that x^1 dominates x^2 and Cx^1 dominates Cx^2. If $Cx^1 = Cx^2$, x^1 and x^2 are* equivalent*. The set of all efficient solutions $x^* \in \mathcal{X}$ is denoted by $\mathcal{X}_E$ and is called the* efficient set*. The set of all nondominated points $y^* = Cx^* \in \mathcal{Y}$, where $x^* \in \mathcal{X}_E$, denoted by $\mathcal{Y}_N$ and is called the* nondominated set*.*

x^* is called weakly efficient solution *if there is no $x \in \mathcal{X}$ such that $Cx < Cx^*$. The set of all weakly efficient solutions and weakly nondominated points are denoted by $\mathcal{X}_{wE}$ and $\mathcal{Y}_{wN}$ respectively.*

It is known that for multiobjective linear programming problems

$$\min\{Cx \mid Ax \leqq b, x \geqq 0\},$$

the set of efficient solutions is exactly the set of solutions of the linear programs

$$\min\left\{\sum_{k=1}^{p} \lambda_k c_k^T x \mid Ax \leqq b, x \geqq 0\right\},$$

where $\sum_{k=1}^{p} \lambda_k = 1, \lambda_k > 0, k = 1, \ldots, p$, (see e.g. [66]). But the discrete structure of (P) makes this result invalid for solving (P). For problem (P), an optimal solution of the following mixed zero-one linear program

$$\min\left\{\sum_{k=1}^{p} \lambda_k c_k^T x \mid x \in \mathcal{X}\right\},$$

with $\sum_{k=1}^{p} \lambda_k = 1, \lambda_k > 0, k = 1, \ldots, p$, is called *supported efficient solution*. The other efficient solutions of (P), which are not optimal for any weighted sum of the objectives, are called *nonsupported (*or *unsupported) efficient solutions*. There are usually many more nonsupported efficient solutions than supported efficient solutions in numerical results (see e.g. [153]). In addition, numerical results presented in [153] show that the numbers of supported efficient solutions grows linearly with the problem size while the number of nonsupported ones grows exponentially. Moreover, the existence of the nonsupported solutions causes essentially the difficulty of (P).

If x^* is a supported efficient solution then $y^* = Cx^*$ is called *supported nondominated point*. The set of all supported efficient points and supported nondominated points are denoted by $\mathcal{X}_{sE}$ and $\mathcal{Y}_{sN}$, respectively. All supported nondominated points are located on the boundary of the convex hull of $\mathcal{Y}$ (conv$\mathcal{Y}$). The set of all nonsupported efficient points is denoted by $\mathcal{X}_{nE}$. Nonsupported nondominated points $\mathcal{Y}_{nN}$ are located in the interior of conv$\mathcal{Y}$.

In the context of solution methods for multiobjective mixed 0-1 linear programs, there are exact methods and approximation methods (heuristic and metaheuristic). The most commonly used techniques to find efficient solutions is scalarization, i.e., the transformation of the multiobjective problem into a single objective problem (except evolutionary and genetic algorithms which directly evaluate objective function vector $y = Cx$), see e.g. [40] for a discussion of scalarization techniques for multiobjective integer programming problems. Other methods which do not use scalarization techniques comprise, e.g., the work of [17, 18], multiobjective branch and bound methods presented in [113, 153].

We summarize below some of common scalarization techniques that have been proposed in the literature.

The weighted sum method : The scalarization of this method is a convex combination of the p objective functions where the feasible set is unchanged :

$$\min\left\{\sum_{k=1}^{p}\lambda_k c_k^T x = \lambda^T(Cx) \mid x \in \mathcal{X}\right\}. \tag{4.1}$$

It is well known that if $\lambda \in \mathbb{R}^p_>$ then an optimal solution of (4.1) is efficient. However, for multiobjective mixed 0-1 linear problems, as the feasible set $\mathcal{X}$ is not convex, there exists (may be many) efficient solutions to which corresponding nondominated points occur in the interior of conv$\mathcal{Y}$; these efficient solutions can not be found by weighted sum method for any $\lambda \in \mathbb{R}^p_>$.

The ε-constraint method : In the ε-constraint method (described in detail in [27]), one of the p objective functions is considered for minimization and the other $(p-1)$ objectives are taken into constraints :

$$\min\left\{c_j^T x \mid c_k^T x \leq \varepsilon_k,\ \forall k \neq j,\ x \in \mathcal{X}\right\}. \tag{4.2}$$

where $\varepsilon = (\varepsilon_k) \in \mathbb{R}^p$.

If the optimal solution x^* of (4.2) is unique, it is efficient. Otherwise, it is weakly efficient (that means there is no $x \in \mathcal{X}$ such that $Cx < Cx^*$). All efficient solutions can be found by appropriated the right hand side values ε_k : x^* is efficient if and only if it is an optimal solution of (4.2) for all $j = 1, \ldots, p$, where $\varepsilon_k = c_k^T x^*, k \neq j$.

Benson's method : This method (presented in [13] for maximizing multiobjective linear problem) is closed related to the ε-constraint method. Given x^0 a feasible solution. The scalarized problem is formulated as

$$\max\left\{\sum_{k=1}^{p} z_k \mid c_k^T x^0 - z_k - c_k^T x = 0,\ \forall k,\ z \geq 0,\ x \in \mathcal{X}\right\}. \tag{4.3}$$

The x-component of an optimal solution (x^*, z^*) of (4.3) is efficient and all efficient solutions can be found.

The augmented weighted Chebychev method : This is a very popular method in interactive procedures [147] which considers the distance of a feasible point Cx in criterion space and the ideal point y^I defined by $y_k^I := \min\{c_k^T x \mid x \in \mathcal{X}\}$.

$$\min_{x\in\mathcal{X}} \max_{k=1,\ldots,p} \nu_k(c_k^T x - y_k^I) + \gamma \sum_{k=1}^{p}(c_k^T x - y_k^I), \tag{4.4}$$

where $\nu > 0$ is a vector of weights. If $\gamma > 0$ an optimal solution x^* of (4.4) is efficient. In addition, all efficient solutions can be found by appropriately specifying the parameters ν and γ.

The approximation methods have been developed and improved in the last two decades, usually called heuristic and metaheuristic methods (see e.g. [41]). Often heuristics are problem-specific that means a heuristic method being used for one problem cannot work for a different one. In contrast, metaheuristics are applicable generally to a large number of problems. Metaheuristics include, but not limited to, genetic algorithms (see e.g. [140, 146, 160]), simulated annealing (see e.g. [120, 142, 150, 151]), tabu search (see e.g. [45, 46,

55]), variable neighborhood search (see e.g. [22, 111]), evolutionary methods [135, 161],... The success of these methods lies in the fact that they can solve in practice some hard combinatorial problems. However, the important question of how to evaluate the quality of approximation solutions is always proposed.

In this work we investigate a method for solving (P) directly. This solution method which can be considered as a generalization of Benson's method is based also on scalarization but it can be used to find all efficient solutions (like Benson's method, unlike the weighted sum method). This approach can be also used to easily verify whether a feasible solution is Pareto optimal. In addition, due to our proposed method, optimizing over the Pareto optimal set of a multiobjective mixed 0-1 linear problem can be formulated as a mathematical programming problem.

The rest of the chapter is organized as follows. In section 4.2 we present the proposed approach for finding an efficient solution (supported or nonsupported) of (P). This method leads to solve only one mixed 0-1 linear program. The next section deals with two applications of the proposed method to nurse rostering problems and planning franchise expansion problems. The chapter will be concluded with some comments.

4.2 Methodological approach

Suppose that $\mathcal{X} \neq \emptyset$. Given a feasible point $x^0 \in \mathcal{X}$ and a positive arbitrary vector $\lambda \in \mathbb{R}^p_>$. Consider the following mixed 0-1 linear program :

$$q_\lambda(x^0) = \min\{\lambda^T(Cx - Cx^0) \mid Cx \leqq Cx^0, x \in \mathcal{X}\} \qquad (P_\lambda x^0)$$

It can easily be shown that :
(i) $q_\lambda(x^0) \leq 0$, $\forall x^0 \in \mathcal{X}$, $\forall \lambda \in \mathbb{R}^p_>$;
(ii) $x^0 \in \mathcal{X}$ is an efficient point of (P) if and only if the optimal value of $(P_\lambda x^0)$ is zero, i.e. $q_\lambda(x^0) = 0$.

The following theorem shows that solving the multiobjective mixed 0-1 linear program (P) amounts to solving the mixed 0-1 linear programming $(P_\lambda x^0)$.

Theorem 4.1 *Suppose that $q_\lambda(x^0)$ is finite. Let x^* be an optimal solution of the mixed 0-1 linear problem $(P_\lambda x^0)$. Then $q_\lambda(x^*) = 0$ and x^* is an efficient solution of the multiobjective problem (P).*

Proof *Suppose that $q_\lambda(x^0)$ is finite and $q_\lambda(x^0) = \lambda^T(Cx^* - Cx^0)$. So we have : $Cx^* \leqq Cx^0$ and*

$$\begin{aligned}
q_\lambda(x^*) &= \min\{\lambda^T(Cx - Cx^*) \mid Cx \leqq Cx^*, x \in \mathcal{X}\} \\
&= \min\{\lambda^T(Cx) \mid Cx \leqq Cx^*, x \in \mathcal{X}\} - \lambda^T(Cx^*) \\
&\geq \min\{\lambda^T(Cx) \mid Cx \leqq Cx^0, x \in \mathcal{X}\} - \lambda^T(Cx^*) \quad (\textit{because } Cx^* \leqq Cx^0) \\
&= \min\{\lambda^T(Cx - Cx^0) \mid Cx \leqq Cx^0, x \in \mathcal{X}\} + \lambda^T(Cx^0) - \lambda^T(Cx^*) \\
&= q_\lambda(x^0) + \lambda^T(Cx^0) - \lambda^T(Cx^*) = 0.
\end{aligned}$$

This yields that x^* is an efficient solution of (P). □

From this theorem, for obtaining a Pareto optimal solution of (P), firstly, we choose arbitrarily a positive vector $\lambda \in \mathbb{R}^p$ and find a feasible solution x^0 of this problem. We then

construct the correspondingly mixed 0-1 linear program $(P_\lambda x^0)$. Solving the last problem provides an efficient solution to (P).

The approach enjoys several advantages :

1. For any given scalar vector $\lambda > 0$, the solution set of the resulting single objective is a subset of the Pareto optimal solution set of the original multiobjective program.
2. The Pareto optimal set can be expressed by the function q_λ and, as a result, optimizing over the Pareto optimal set can be formulated as a mathematical programming problem.
3. It makes possible to check if a feasible solution x^0 of the multiobjective program is Pareto optimal or not : it suffices to compute $q_\lambda(x^0)$ and compare it with 0 (for any $\lambda > 0$).
4. It is worth noting that the multiobjective program can be equivalently reformulated as maximizing the function q_λ on the feasible set : this problem is NP-hard. For example, the related problem is a concave minimization on a polyhedral convex set if one is concerned with multiobjective linear programming. Fortunately, we have proved that it suffices to compute $q_\lambda(x)$ with a feasible solution x to compute a Pareto optimal solution.

Talking a little more about the second advantage mentioned above, let consider the following problem of minimizing a real valued function over the efficient set of (P) :

$$\min\{F(x) \text{ s.t. } x \in \mathcal{X}_E\} \tag{4.5}$$

where F is a real valued function on $\mathbb{R}^n$ and $\mathcal{X}_E$ is the efficient set of (P). Since we can express $\mathcal{X}_E = \{x \in \mathcal{X} \mid q_\lambda(x) = 0\}$ where $\lambda \in \mathbb{R}^p_>$ chosen arbitrary, (4.5) is equivalent to

$$\min\{F(x) \text{ s.t. } q_\lambda(x) = 0,\ x \in \mathcal{X}\} \tag{4.6}$$

The problem of maximizing over the efficient set of a multiobjective linear program was studied in [95, 96].

4.3 Applications

In this section, we present two applications of the proposed method for solving nurse rostering and planning franchise expansion problems. These problems are firstly formulated as multiobjective mixed 0-1 linear programs and then one Pareto optimal will be found based on the above method.

4.3.1 Nurse rostering problems

Staff scheduling has been extensively studied by many researchers for more than 40 years [28,42]. This type of problem could be understood as assigning employees to shifts over a scheduling period such that certain constraints are satisfied. Nurse rostering is a resource-allocation problem, in which the workload needs to be assigned to nurses periodically, taking into account a number of constraints and requirements. There are two types of constraints : hard and soft constraints. Hard constraints are those that must be satisfied in order to have a feasible schedule. They are often generated by physical resource restrictions and legislation. When requirements are desirable but not obligatory, they are referred to soft constraints. The objective of nurse rostering problems is to minimize the value of deviations

of these soft constraints. Hence it can be formulated as a multiobjective mathematical programming problem.

In the literature, a wide variety of methodologies and models have been developed to deal with the nurse rostering problem. An overview of the area can be found in some survey papers [24, 28]. There are two main classes of algorithms : exact algorithms (using integer programming (IP) [9, 51]) and (meta)heuristics (using genetic algorithms, tabu search, variable neighborhood search, simulated annealing,..., see [41]). Some other algorithms (see e.g, [23, 25]) combine these two approaches. However, to reduce complexity, almost authors avoid directly treating the multiobjective program.

We first describe below the nurse rostering problem then formulate three benchmark problems as multiobjective mixed zero-one linear programs. The proposed approach in section 4.2 will be used to find an efficient solution for each problem.

Problem description and mathematical formulation

The nurse rostering problems involve usually four types of shifts : Early(E), Day(D), Late(L) and Night(N). Some hard constraints as follows must be satisfied by all feasible solutions :

- HC1 : Daily coverage requirement of each shift type ;
- HC2 : For each day, a nurse may not start more than one shift ;
- HC3 : Maximum number of total working days during the period ;
- HC4 : Maximum number of night shifts during the period ;
- HC5 : Maximum number of consecutive night shifts ;
- HC6 : Maximum number of consecutive working days ;
- HC7 : Minimum number of free days after a series of night shifts ;
- HC8 : Maximum number of on-duty weekends during the period ;
- HC9 : No night shift before a free weekend.

In addition, beside the personal demands, the problem has some soft constraints which should be satisfied as much as possible :

- SC1 : No single night shift ;
- SC2 : No stand-alone shift ;
- SC3 : No single shift in weekends (complete weekend) ;
- SC4 : Avoid a single day off ;
- SC5 : Maximum/minimum number of shifts per week ;
- SC6 : Stint length of consecutive working days for each nurse.

Note that according to practical situations, one can exchange or add more some soft constraints and hard constraints.

We address three benchmark problems originally provided by ORTEC, an international consultancy company specializing in planning, optimization and decision support solutions. They are called "GPost", "GPostB" and "ORTEC01" (see `http://www.cs.nott.ac.uk/~tec/NRP/index.html`).

"GPost" is a small problem for eight nurses across a rostering period of exactly four full weeks. There are only two nurse contracts, full time (who works 36 hours/week) and part time (who works 20 hours/week) and two shift types, day(D) and night(N). For each day, we need 3 day shifts and 1 night shift. "GPostB" is a relaxation of "GPost" without the request on the first two days (which is described by the constraint HC0 in the model).

"ORTEC01" is a larger problem for 16 nurses for 31 days of January 2003. There are four shift types : Early(E), Day(D), Late(L) and Night(N). For E, D, L shifts, we need 3

Hard constraint		GPost/GPostB	ORTEC01
HC1	(depends on each problem)		
HC2		1	1
HC3	+ for 36 hours/week contract	18	20
	+ for 32 hours/week contract		18
	+ for 20 hours/week contract	10	11
HC4		4	3
			(in 5 weeks)
HC5		3	3
HC6		6	6
HC7		2	2
HC8		2 of 3	3
		consecutive WK	(in 5 weeks)

TABLE 4.1 – Hard constraints

shifts for each type on each day from Monday to Friday and we need 2 shifts for each type on Saturday and Sunday, while each day there starts only 1 N shift.

The set of rostering rules can be described in Table 4.1 and Table 4.2.

Mathematical formulation :

We use the notations mentioned in Table 4.3.

Decision variables $x_{i,j,k}$ where $i \in I$, $i \in J$, $k \in K$ are defined by :

$$x_{i,j,k} = \begin{cases} 1 & \text{if nurse } i \text{ is assigned shift } k \text{ for day } j, \\ 0 & \text{otherwise.} \end{cases}$$

We introduce nonnegative slack variables $s^l_{i,j}$ which represent the violation of soft constraints (for SC1 to SC4 and SC8) or the deviation from the preferred range of soft constraints (for SC5 to SC7).

For example :

- For SC1 (No single night shift), slack variables $s^1_{i,j}$ are introduced, we have the following constraints :

$$\begin{aligned} & x_{i,1,k^*} - x_{i,2,k^*} - s^1_{i,1} \leq 0 \quad \forall i \in I, \\ & - x_{i,j,k^*} + x_{i,j+1,k^*} - x_{i,j+2,k^*} - s^1_{i,j} \leq 0 \ \forall i \in I, \forall j \in \{1, \ldots, |J| - 2\}. \end{aligned}$$

  If SC1 is satisfied for nurse i on day j then $s^1_{i,j} = 0$. Otherwise, $s^1_{i,j} = 1$.

- For SC5, slack variables $s^5_{i,t}$ and $s^6_{i,t}$ are introduced for nurse i to t^{th}-week in the period ($i \in I$).

  The constraint SC5 for nurses who work with full-time contract (I_1) in the second week can be expressed as :

$$\sum_{j=w_2-6}^{w_2} \sum_{k \in K} x_{i,j,k} - s^5_{i,2} \leq 5 \quad \forall i \in I_1,$$

	Soft constraint	Preferred range
SC5	Number of shifts per (full) week (from Monday to Sunday)	
	+ for 36 or 32 hours/week contract	4 to 5
	+ for 20 hours/week contract	2 to 3
SC6	Stint length	
	+ for 36 or 32 hours/week contract	4 to 6
	+ for 20 hours/week contract	2 to 3
	(SC7 and SC8 concern ORTEC01)	
SC7	Series of E or L shifts	2 to 3
SC8	Avoid : E shift follows D/L shift	
	Avoid : N shift follows E shift	
	Avoid : D shift follows L shift	

TABLE 4.2 – Soft constraints

$I = I_1 \cup I_2$	set of indices of available nurses I_1 : set of indices of full-time nurses I_2 : set of indices of part-time nurses
GPost/GPostB	$I_1 = \{1, 2, 3, 4\}$, $I_2 = \{5, 6, 7, 8\}$
ORTEC01	$I_1 = I_{11} \cup I_{12}$ $I_{11} = \{1, \ldots, 12\}$, indices of nurses for 36 hours/week contract $I_{12} = \{13\}$, index of nurses with 32 hours/week contract $I_2 = \{14, 15, 16\}$,
J	set of indices of days in the scheduling period For GPost/GPostB : $J = \{1, \ldots, 28\}$ For ORTEC01 : $J = \{1, \ldots, 31\}$
K	set of shift types, k^* designs for N shift For GPost/GPostB : $K = \{1(\text{D}), 2(\text{N})\}$, $k^* = 2$ For ORTEC01 : $K = \{1(\text{E}), 2(\text{D}), 3(\text{L}), 4(\text{N})\}$, $k^* = 4$
W	set of indices of Sunday in the scheduling period For GPost/GPostB : $W = \{w_1, w_2, w_3, w_4\} = \{7, 14, 21, 28\}$ For ORTEC01 : $W = \{w_1, w_2, w_3, w_4\} = \{5, 12, 19, 26\}$
$d_{k,j}$	requirement about shift k on day j, $k \in K, j \in J$

TABLE 4.3 – Notations

$$\sum_{j=w_2-6}^{w_2} \sum_{k\in K} x_{i,j,k} + s^6_{i,2} \geq 4 \quad \forall i \in I_1.$$

If the constraint SC5 is satisfied for nurse $i \in I_1$ then $s^5_{i,2} = 0$, $s^6_{i,2} = 0$. Otherwise, for instance, nurse i works just on 2 days during second week then $s^5_{i,2} = 0$, $s^6_{i,2} = 2$.

We present now hard and soft constraints for "GPost" and "GPostB".

Constraints for GPost/GPostB :

$$\text{HC1} \quad \sum_{i\in I} x_{i,j,k} = d_{k,j} \quad \forall j \in J, k \in K; \tag{4.7}$$

$$\text{HC2} \quad \sum_{k\in K} x_{i,j,k} \leq 1 \quad \forall i \in I, j \in J; \tag{4.8}$$

$$\text{HC3} \quad \sum_{j\in J} \sum_{k\in K} x_{i,j,k} \leq m_i \quad \forall i \in I; \tag{4.9}$$

$$\text{HC4} \quad \sum_{j\in J} x_{i,j,k^*} \leq n_1 \quad \forall i \in I; \tag{4.10}$$

$$\text{HC5} \quad \sum_{j=r}^{r+3} x_{i,j,k^*} \leq 3 \quad \forall i \in I, r \in \{1,\ldots,|J|-3\}; \tag{4.11}$$

$$\text{HC6} \quad \sum_{j=r}^{r+6} \sum_{k\in K} x_{i,j,k} \leq 6 \quad \forall i \in I, r \in \{1,\ldots,|J|-6\}; \tag{4.12}$$

$$\text{HC7} \quad x_{i,j,k^*} - \sum_{k\neq k^*} x_{i,j+1,k} + \sum_{k\neq k^*} x_{i,j+2,k} \leq 1 \tag{4.13}$$

$$\forall i \in I, \forall j \in \{1,\ldots,|J|-2\};$$

$$x_{i,j,k^*} + \sum_{k\neq k^*} x_{i,j+1,k} - \sum_{k\neq k^*} x_{i,j+2,k} \leq 1 \tag{4.14}$$

$$\forall i \in I, \forall j \in \{1,\ldots,|J|-2\};$$

$$x_{i,j,k^*} + \sum_{k\neq k^*} x_{i,j+1,k} + \sum_{k\neq k^*} x_{i,j+2,k} \leq 2 \tag{4.15}$$

$$\forall i \in I, \forall j \in \{1,\ldots,|J|-2\}; \tag{4.16}$$

$$\text{HC8} \quad \sum_{t=r}^{r+2} \sum_{k\in K} x_{i,w_t,k} \leq 2 \quad \forall i \in I, r \in \{1,2\}; \tag{4.17}$$

$$\text{HC9} \quad x_{i,w_t-2,k^*} - \sum_{k\in K} (x_{i,w_t-1,k} + x_{i,w_t,k}) \leq 0 \ \forall i \in I, t \in \{1,2,3,4\}; \tag{4.18}$$

$$\text{SC1} \quad x_{i,1,k^*} - x_{i,2,k^*} - s^1_{i,1} \leq 0 \quad \forall i \in I; \tag{4.19}$$

$$- x_{i,j,k^*} + x_{i,j+1,k^*} - x_{i,j+2,k^*} - s^1_{i,j} \leq 0 \tag{4.20}$$

$$\forall i \in I, \forall j \in \{1,\ldots,|J|-2\};$$

$$\text{SC2} \quad \sum_{k\in K} x_{i,1,k} - \sum_{k\in K} x_{i,2,k} - s^2_{i,1} \leq 0 \quad \forall i \in I; \tag{4.21}$$

$$- \sum_{k\in K} x_{i,j,k} + \sum_{k\in K} x_{i,j+1,k} - \sum_{k\in K} x_{i,j+2,k} - s^2_{i,j} \leq 0 \tag{4.22}$$

$$\forall i \in I; \forall j \in \{1, \ldots, |J| - 2\};$$

SC3
$$\sum_{k \in K} x_{i,w_t-1,k} - \sum_{k \in K} x_{i,w_t,k} - s^3_{i,t} \leq 0 \quad \forall i \in I, t \in \{1,2,3,4\}; \tag{4.23}$$

$$-\sum_{k \in K} x_{i,w_t-1,k} + \sum_{k \in K} x_{i,w_t,k} - s^3_{i,t} \leq 0 \quad \forall i \in I, t \in \{1,2,3,4\}; \tag{4.24}$$

SC4
$$-\sum_{k \in K} x_{i,1,k} + \sum_{k \in K} x_{i,2,k} - s^4_{i,1} \leq 0 \quad \forall i \in I; \tag{4.25}$$

$$\sum_{k \in K} x_{i,j,k} - \sum_{k \in K} x_{i,j+1,k} + \sum_{k \in K} x_{i,j+2,k} - s^4_{i,j} \leq 1 \tag{4.26}$$

$$\forall i \in I, \forall j \in \{1, \ldots, |J| - 2\};$$

SC5
$$\sum_{j=w_t-6}^{w_t} \sum_{k \in K} x_{i,j,k} - s^5_{i,t} \leq 5 \quad \forall i \in I_1, t \in \{1,2,3,4\}; \tag{4.27}$$

$$\sum_{j=w_t-6}^{w_t} \sum_{k \in K} x_{i,j,k} - s^5_{i,t} \leq 3 \quad \forall i \in I_2, t \in \{1,2,3,4\}; \tag{4.28}$$

$$-\sum_{j=w_t-6}^{w_t} \sum_{k \in K} x_{i,j,k} - s^6_{i,t} \leq -4 \quad \forall i \in I_1, t \in \{1,2,3,4\}; \tag{4.29}$$

$$-\sum_{j=w_t-6}^{w_t} \sum_{k \in K} x_{i,j,k} - s^6_{i,t} \leq -2 \quad \forall i \in I_2, t \in \{1,2,3,4\}; \tag{4.30}$$

SC6
$$3\sum_{k \in K} x_{i,1,k} - 3\sum_{k \in K} x_{i,2,k} - s^7_{i,1} \leq 0 \quad \forall i \in I_1; \tag{4.31}$$

$$3\sum_{k \in K} x_{i,1,k} - \sum_{k \in K} x_{i,2,k} - 2\sum_{k \in K} x_{i,3,k} - s^7_{i,1} \leq 0 \quad \forall i \in I_1; \tag{4.32}$$

$$3\sum_{k \in K} x_{i,1,k} - \sum_{r=2}^{r=4} \sum_{k \in K} x_{i,r,k} - s^7_{i,1} \leq 0 \quad \forall i \in I_1; \tag{4.33}$$

$$3\sum_{k \in K} (-x_{i,j,k} + x_{i,j+1,k}) - 3\sum_{k \in K} x_{i,j+2,k} - s^7_{i,j+1} \leq 0; \tag{4.34}$$

$$\forall i \in I_1, \forall j \in \{1, \ldots, |J| - 4\};$$

$$3\sum_{k \in K} (-x_{i,j,k} + x_{i,j+1,k}) - \sum_{k \in K} x_{i,j+2,k} - 2\sum_{k \in K} x_{i,j+3,k} - s^7_{i,j+1} \leq 0; \tag{4.35}$$

$$\forall i \in I_1, \forall j \in \{1, \ldots, |J| - 4\};$$

$$3\sum_{k \in K} (-x_{i,j,k} + x_{i,j+1,k}) - \sum_{r=2}^{r=4} \sum_{k \in K} x_{i,j+r,k} - s^7_{i,j+1} \leq 0; \tag{4.36}$$

$$\forall i \in I_1, \forall j \in \{1, \ldots, |J| - 4\};$$

$$\sum_{r=j}^{j+3} \sum_{k \in K} x_{i,r,k} - s^8_{i,j} \leq 3 \quad \forall i \in I_2, j \in \{1, \ldots, |J| - 3\}. \tag{4.37}$$

For GPost, the request on the first two days is presented by the constraint (HC0) below :

$$\text{HC0} \quad \begin{cases} x_{1,1,1} = 1; \\ x_{1,2,1} = 1; \\ x_{3,1,1} = 1; \\ x_{3,2,1} = 1; \\ x_{4,1,2} = 1; \\ x_{4,2,2} = 1; \\ x_{5,1,1} = 1; \\ x_{5,2,1} = 1. \end{cases}$$

For ORTEC01, (HC0) will be replaced by (HC0*) :

$$\text{HC0}^* \quad \begin{cases} x_{10,j,3} = 0 \quad \forall j \in J; \\ x_{1,6,2} = 1. \end{cases}$$

For ORTEC01, we replace (4.17) with (4.38) :

$$\text{HC8}^* \quad \sum_{t=1}^{4} \sum_{k \in K} x_{i,w_t,k} \leq 3 \quad \forall i \in I \tag{4.38}$$

And the constraints from (4.27) to (4.30) will be replaced with the following constraints :

$$\text{SC5}^* \quad \sum_{j=1}^{w_1} \sum_{k \in K} x_{i,j,k} - s^5_{i,1} \leq g_{i1} \quad \forall i \in I; \tag{4.39}$$

$$\sum_{j=w_t-6}^{w_t} \sum_{k \in K} x_{i,j,k} - s^5_{i,t} \leq g_{it} \quad \forall i \in I, t \in \{2,3,4\}; \tag{4.40}$$

$$\sum_{j=w_4}^{31} \sum_{k \in K} x_{i,j,k} - s^5_{i,5} \leq g_{i5} \quad \forall i \in I; \tag{4.41}$$

$$-\sum_{j=1}^{w_1} \sum_{k \in K} x_{i,j,k} - s^6_{i,1} \leq -h_{i1} \quad \forall i \in I; \tag{4.42}$$

$$-\sum_{j=w_t-6}^{w_t} \sum_{k \in K} x_{i,j,k} - s^6_{i,t} \leq -h_{it} \quad \forall i \in I, t \in \{2,3,4\}; \tag{4.43}$$

$$-\sum_{j=w_4}^{31} \sum_{k \in K} x_{i,j,k} - s^6_{i,5} \leq -h_{i5} \quad \forall i \in I; \tag{4.44}$$

where g_{it}, h_{it} are maximum/minimum number of shifts per t^{th}-week for nurse i. In addition, for ORTEC01, we consider also soft constraints SC7, SC8 below :

$$\text{SC7} \quad \sum_{r=j}^{j+3} x_{i,r,k} - s^9_{i,j,k} \leq 3 \quad \forall i \in I, j \in \{1, \ldots, |J| - 3\}, k \in \{1,3\}; \tag{4.45}$$

$$x_{i,1,k} - x_{i,2,k} - s^{10}_{i,1,k} \leq 0 \quad \forall i \in I, \forall k \in \{1,3\}; \tag{4.46}$$

$$- x_{i,j,k} + x_{i,j+1,k} - x_{i,j+2,k} - s^{10}_{i,j,k} \leq 0 \quad (4.47)$$
$$\forall i \in I, \forall j \in \{1, \ldots, |J| - 2\}, \forall k \in \{1, 3\};$$

SC8
$$x_{i,j,2} + x_{i,j+1,1} - s^{11}_{i,j,1} \leq 1 \quad \forall i \in I, \forall j \in \{1, \ldots, |J| - 1\}; \quad (4.48)$$
$$x_{i,j,1} + x_{i,j+1,4} - s^{11}_{i,j,2} \leq 1 \quad \forall i \in I, \forall j \in \{1, \ldots, |J| - 1\}; \quad (4.49)$$
$$x_{i,j,3} + x_{i,j+1,1} - s^{11}_{i,j,3} \leq 1 \quad \forall i \in I, \forall j \in \{1, \ldots, |J| - 1\}; \quad (4.50)$$
$$x_{i,j,3} + x_{i,j+1,2} - s^{11}_{i,j,4} \leq 1 \quad \forall i \in I, \forall j \in \{1, \ldots, |J| - 1\}; \quad (4.51)$$

Target functions : let x be the vector of size n containing all decision variables $x_{i,j,k}$ and slack variables $s^l_{i,j}$. The target functions are defined by :

$$g_1(x) = \sum_{i \in I} \sum_{j} s^1_{i,j}; \qquad g_2(x) = \sum_{i \in I} \sum_{j} s^2_{i,j};$$
$$g_3(x) = \sum_{i \in I} \sum_{t} s^3_{i,t}; \qquad g_4(x) = \sum_{i \in I} \sum_{j} s^4_{i,j};$$
$$g_5(x) = \sum_{i \in I} \sum_{t} (s^5_{i,t} + s^6_{i,t}); \qquad g_6(x) = \sum_{i \in I_1} \sum_{j} s^7_{i,j};$$
$$g_7(x) = \sum_{i \in I_2} \sum_{j} s^8_{i,j}; \qquad g_8(x) = \sum_{i \in I} \sum_{j} \sum_{k \in \{1,3\}} s^9_{i,j,k};$$
$$g_9(x) = \sum_{i \in I} \sum_{j} \sum_{k \in \{1,3\}} s^{10}_{i,j,k}; \qquad g_{10}(x) = \sum_{i \in I} \sum_{j} \sum_{r=1}^{4} s^{11}_{i,j,r}.$$

The nurse rostering problems "GPost", "GPostB", "ORTEC01" can be formulated as :

$$\begin{cases} \min \ F_1(x) = (g_1(x), \ldots, g_7(x)) \\ \text{s.t. (HC0)} - \text{(HC9)}, \\ \quad \text{(SC1)} - \text{(SC6)}. \end{cases} \quad \text{(GPost)}$$

$$\begin{cases} \min \ F_2(x) = (g_1(x), \ldots, g_7(x)) \\ \text{s.t. (HC1)} - \text{(HC9)}, \\ \quad \text{(SC1)} - \text{(SC6)}. \end{cases} \quad \text{(GPostB)}$$

$$\begin{cases} \min \ F_3(x) = (g_1(x), \ldots, g_{10}(x)) \\ \text{s.t. (HC1)} - \text{(HC7)}, \\ \quad \text{(HC8}^*\text{)}, \text{(HC9)}, \\ \quad \text{(SC1)} - \text{(SC5}^*\text{)}, \\ \quad \text{(SC6)} - \text{(SC8)}. \end{cases} \quad \text{(ORTEC01)}$$

Since g_i $(i = 1, \ldots, 11)$ are linear functions, these problems are multiobjective mixed 0-1 linear programs that can be rewritten as

$$\min\{Cx, \text{ s.t. } x \in \mathcal{X}\}, \quad \text{(NRP)}$$

where $C \in \mathbb{R}^{p \times n}$ is the matrix whose i^{th} row verifies $C_i x = g_i(x)$, $i = 1, \ldots, p$ and $\mathcal{X}$ is the feasible set.

The size of these problems is presented in Table 4.4.

Computational results

We have written the code in C and implemented these codes on a PC equipped with Window 7 Intel(R) Core(TM) i5-2540M CPU 2.60GHz, 8.00 Go RAM. CPLEX version 12.4 is used for solving mixed 0-1 linear programming subproblems.

Calculated time for finding an efficient solution of "GPost", represented in Table 4.5, is 23 mins approximately.

Calculated time for finding an efficient solution of "GPostB", represented in Table 4.6, is 21 mins approximately.

Calculated time for finding an efficient solution of "ORTEC01", represented in Table 4.7, is 35 mins approximately.

Problem	Objectives	Variables	01 variables	Constraints
GPost	7	1368	448	2528
GPostB	7	1368	448	2520
ORTEC01	10	7792	1984	9684

Table 4.4 – Problem size

S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S
N1	D	D	D	D					D	D	D	N	N	N			D	D	D					D	D	D	D	D	D
N2			D	D	D	D	D	D		D	D	D			D	D	D	D				D	D			N	N	N
N3	D	D	D	D	D			N	N				D	D	D	D	N	N				D	D	D			D	D
N4	N	N				D	D	D	D	D	D				D	D	D			D	D	N	N			D	D	D
N5	D	D										D	D	D					N	N	N			D	D			
N6					D	D	D			N	N								D	D	D			N	N			
N7					N	N	N					D	D	D				D	D			D	D					
N8			N	N				D	D						N	N				D	D				D	D		

Table 4.5 – GPost

S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S
N1	D	D	N	N				D	D	D	D				D	D	D			D	D	N	N			D	D	D
N2	D	D	D	D				D	D	N	N				N	N			D	D	D	D	D			D	D	D
N3	D	D	D			D	D	D	D	D			D	D	D	D	N	N				D	D	N	N			
N4			D	D	D	D	D			D	D	D					D	D	N	N	N			D	D	D	D	D
N5					D	D	D					D	D		D	D										N	N	N
N6				D	D							N	N	N					D	D	D			D	D			
N7					N	N	N					D	D	D			D	D						D	D			
N8	N	N						N	N				D	D				D	D			D	D					

Table 4.6 – GPostB

S	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F
N1			D	D	D	D	D			N	N	N			E	D	D				D	D	E	E	E	D				L	L
N2	D	D	L					D	D	E	E	D			D	L	L	D	D	D				D	D	L	N	N			E
N3	E	E	E	D	D			D	E	D					L	L	L				E	D	E	L	L	E	E			D	D
N4	L	L	N	N	N			D	L	L	D	D	E				L	L	E	L	L	L							D	D	E
N5	L	L	L			D	E	E			L	L	N	N			E	E	E				E	E	L	L	L	E			L
N6			E	E	L	N	N			D	D	E	E			D	E	L	L	L	L		L	E			D	L	L		
N7				E	E	E	L	L	D					E	D	E	E			E	E	L	L	L			D	D	L	L	N
N8				L	L	E	E	L	L				D	D	N	N				E	D	D	L	L			D	D	L	L	L
N9				L	E	E	D	E	E					E	L	D	D	D	D			D	D	N	N	N			E	E	
N10	E	E	D			D	D	N	N				D	D	E	E	D			E	E						E	E	D	D	D
N11	E	E	D					E	E	E	L	E	E			E	N	N	N			E	D	D	D	D	D	E			D
N12	D	D	L			L	E	L	D	D			L	D	D	L				D	L	L	D	D					E	E	E
N13										E	E	L	L	L	E			E	L	L	E	E			E	E	L	L	N	N	
N14	L	L	E						L	L			D	E						N	N						L	L			
N15	D	D				L	L			L				L	L						N	N						D	D		
N16	N	N				L	L						L	L						D	D							E	E	E	

Table 4.7 – ORTEC01

4.3.2 Facility location

In the past 40 years, there has been a tremendous growth in franchise systems. 70 different industries use the franchising business model and according to the International Franchising Association the sector earns more than $1.5 trillion in revenues each year ([65]). Franchising is a business model in which many different owners share a single brand name. A parent company (a franchisor) allows entrepreneurs (franchisees) to use the company's strategies and trademarks ; in exchange, the franchisee pays an initial fee plus royalties based on his or her revenue (equals to a certain percentage of the store's monthly or yearly sales). For example, the initial fee to open a Hilton hotel is over $85000 and Intercontinental Hotels Group (IHG) franchisees are required to pay the company 5% of their yearly sales. As part of the franchising agreement, the franchisor also provides the franchisee with training and support as well as regional and/or national advertising. A discussion of the promises and problems of franchising is given in [64].

The objective of the franchisor is to maximize system-wide market coverage in such a way that each individual franchisee has sufficient demand to obtain normal profits ([159]) because the franchisor's system revenue will be increased, while the goal of the franchisee, on the other hand, is to maximize his or her individual market share. The introduction of new franchisees into an existing franchise network may cause conflict ([49, 69, 159]). For example, the new franchisees would be interested in locations that attract a maximum number of customers from existing stores and competing stores. However, the existing franchisees would not agree to a plan that reduces their revenue. Although frequently the franchisor ultimately makes the decisions but in the franchisor's long-term profitability and to maintain a mutually beneficial relationship, the rationalization of given decisions is very important.

Several works in the literature study the problem of locating franchisees. In order to de-

sign franchise network, the authors of [31] presented a multiobjective integer programming model in which three objectives were considered : maximizing the total number of outlets to be sited in attempt to maximize total market share ; maximizing the total demand within the threshold of an outlet (these two first objectives for the franchisor were presented in [159]) ; minimizing the total demand within the range of service of more than one outlet. An approach for generating a set of non-inferior (efficient solutions) was proposed.

The authors of [69] discussed about the conflict appearing in the franchise expansion. They then introduced a model for locating a new store on existing franchisees. [49] also considered this conflict and presented three 0-1 linear programming models to maximize the corresponding revenues of the franchisor, existing franchisees and new franchisees. A feasible space reduction method was used to solve the multiobjective problem.

The paper [74] deals with the problem of locating new franchisees in a region which already has some franchisees and other competing stores by using a multiobjective 0-1 linear programming model for making trade-offs. The STEP method (STEM) (developed in [12]) which is considered as a feasible region reduction method, was applied to reach a compromise solution.

In this work, we are interested in the problem of locating new franchisees into an existing franchise network with the multiobjective 0-1 linear model being introduced and solved in [74]. In the following, we describe the facility location problem and its mathematical formulation, then we use the method being proposed in section 4.2 for finding an efficient solution of the facility location problem. Computational results are reported at the end of the section.

Problem description and its mathematical formulation

Let the region in which the franchisees are to be located be divided into zones. The distance between every pair of zones should be obtained. We use the notations below :

S set of zones considered
E set of zones with existing franchisees
T set of zones with competing stores
R set of all zones excluding E and T, $E \cap T = \emptyset$, $R = S \setminus (E \cup T)$
a_i population of zone i, $i \in S$
d_{ij} distance between pair of zones $i, j \in S$
S_{ik} distance from zone i to its k^{th} closest store before the introduction of new franchisees
b_{ik} fraction of demand from zone i which receives service from the k^{th} closest store, $\sum_{k=1}^{m} b_{ik} = 1, \ \forall i \in S$
(m is the maximum number of facilities and in most of real situations $b_{ik} = 0, \ \forall k > 2$)

Suppose that the zones are designed such that each zone contains no more than one competing store or one existing franchisee. A franchisor seeks to locate new franchisees in this region. New stores may be located in any zone in R. Assume that the customers are more likely to patronize stores that are closer to them. In reality, customers may patronize their closest, or the second closest, or the k^{th} closest facility. This situation may be arisen if the closest store is busy, or if the customer prefers the second closest due to the store's

reputation, prices, quality of service, or other reasons. For all problems discussed in a later section, we assume that 70% of demand from a zone is satisfied by the closest store and the remaining 30% is satisfied by the next closest store, i.e., $m = 2,\ b_{i1} = 0.7,\ b_{i2} = 0.3,\ \forall i \in S$.

When new franchisees are introduced, some customers who patronized existing franchisees or competing stores are attracted maybe to the new franchisees. It severely affects the revenue of existing franchisees and a conflict occurs. Hence, the franchisor should also consider to minimize the conflict system in the franchise expansion process.

The following objectives are considered :

1. Maximizing the number of customers attracted from competing stores and existing franchisees.
2. Maximizing the number of customers who visit the franchisor's outlets.
3. Minimizing the number of customers attracted from the existing franchisees.
4. Minimizing the number of customers lost from each of the existing franchisees.

To formulate the location problem, the 0-1 decision variables are introduced :

$$x_{ijk} = \begin{cases} 1 & \text{if any customers in zone } i \text{ visit store in zone } j \\ & \text{which is the } k^{th} \text{ closest store,} \\ 0 & \text{otherwise,} \end{cases}$$

where $i \in S,\ j \in S,\ k = 1, \ldots, m$.

The following model is constructed :

$$\begin{cases} \max & z_1 = \sum\limits_{i \in T \cup E} \sum\limits_{j \in R} \sum\limits_{k=1}^{m} c_{ijk} x_{ijk} \\ \max & z_2 = \sum\limits_{i \in T} \sum\limits_{j \in R} \sum\limits_{k=1}^{m} c_{ijk} x_{ijk} \\ \min & z_3 = \sum\limits_{i \in E} \sum\limits_{j \in R} \sum\limits_{k=1}^{m} c_{ijk} x_{ijk} \\ \min & z_{i+3} = \sum\limits_{j \in R} \sum\limits_{k=1}^{m} c_{ijk} x_{ijk},\ i \in E \\ \text{s.t.} & \sum\limits_{j \in S} x_{ijk} = 1,\ \forall i \in S,\ k = 1, \ldots, m, \\ & x_{jj1} \geq \sum\limits_{k=1}^{m} x_{ijk},\ \forall i, j \in S,\ i \neq j, \\ & \sum\limits_{j \in S} x_{j,j,1} = r, \\ & x_{jj1} = 1,\ \forall j \in E \cup T, \\ & x_{ijk} \in \{0, 1\},\ \forall i, j \in S,\ k = 1, \ldots, m, \end{cases} \tag{4.52}$$

where r is the sum of the number of competing, existing stores and new stores to be established, and

$$c_{ijk} = \begin{cases} a_i b_{ik} & \text{if } d_{ij} < S_{ik}, \\ \dfrac{a_i b_{ik}}{2} & \text{if } d_{ij} = S_{ik}, \\ 0 & \text{if } d_{ij} > S_{ik}. \end{cases}$$

The problem (4.52) can be rewritten as follows :

$$\begin{cases} \min & z = (-z_1, -z_2, z_3, z_{i+3}),\ i \in E \\ \text{s.t.} & \sum\limits_{j\in S} x_{ijk} = 1, \forall i \in S,\ k = 1, \ldots, m, \\ & x_{jj1} \geq \sum\limits_{k=1}^{m} x_{ijk}, \forall i, j \in S,\ i \neq j, \\ & \sum\limits_{j\in S} x_{j,j,1} = r, \\ & x_{jj1} = 1, \forall j \in E \cup T, \\ & x_{ijk} \in \{0,1\}, \forall i, j \in S,\ k = 1, \ldots, m. \end{cases} \tag{4.53}$$

In this problem, the number of considered zones is $n := |S|$ and the number of existing franchisees is $n' := |E|$. Denoted by m the maximum number of facilities, then the number of variables is $N = n^2m$, the number of constraints is $n(m+n-1)+n'+1$ and the number of objective functions is $p = 3 + n'$. This is a multiobjective 0-1 linear programming problem.

According to the theorem 4.1, solving (4.53) leads to solving one 0-1 linear program of the form $(P_\lambda x^0)$, with $x^0 \in \mathcal{X}$ where

$$\mathcal{X} = \left\{ x \in \mathbb{R}^N \;\middle|\; \begin{array}{l} \sum\limits_{j\in S} x_{ijk} = 1, \forall i \in S,\ k = 1, \ldots, m, \\ x_{jj1} \geq \sum\limits_{k=1}^{m} x_{ijk}, \forall i, j \in S,\ i \neq j, \\ \sum\limits_{j\in S} x_{j,j,1} = r, \\ x_{jj1} = 1, \forall j \in E \cup T, \\ x_{ijk} \in \{0,1\}, \forall i, j \in S,\ k = 1, \ldots, m. \end{array} \right\}$$

(4.53) can be rewritten in the form of (P) :

$$\min\{Cx,\ \text{s.t. } x \in \mathcal{X}\},$$

where $C \in \mathbb{R}^{p\times N}$. With $x^0 \in \mathcal{X}$ and $\lambda \in \mathbb{R}^p,\ \lambda > 0$ chosen arbitrarily, we solve the following 0-1 linear programming problem :

$$\min\{\lambda^T(Cx - Cx^0) \mid Cx \leq Cx^0,\ x \in \mathcal{X}\}.$$

Each solution of the last 0-1 linear program is an efficient solution of (4.53).

Computational results

We have written the computer codes in C and implemented on a PC equipped with Window 7 Intel(R) Core(TM) i5-2540M CPU 2.60GHz, 8.00 Go RAM. CPLEX 12.4 solver is used for solving 0-1 linear programming problems.

The data for testing problem (4.53) was derived from [31]. Table 4.8 and Table 4.9 show the distance between every pair of zones and the population of each zone in the region.

Six models were considered as shown in Table 4.10. The number of new franchisees to be located is 1. The numbers of constraints and variables depend on the number of sites considered for the location of the new franchisee store. The number of objectives depends on the number of existing stores. The problem sizes of six models are shown in Table 4.11. Finally Table 4.12 provides the computational results, say the site for locating one new store for each problem in Table 4.10, and CPU time in seconds.

zone	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	0.0	1.4	2.2	1.0	1.7	4.2	3.1	2.9	3.0	3.5	4.7	4.1	4.3	5.9	5.8	5.8	5.1	5.3	7.5	7.0	6.5
2	1.4	0.0	3.2	1.2	1.8	5.1	4.0	2.8	2.2	2.3	5.3	4.3	3.6	6.3	5.9	5.8	4.7	4.8	7.8	7.0	6.1
3	2.2	3.2	0.0	2.0	1.6	1.9	0.9	2.2	3.2	4.2	2.5	2.4	4.1	3.9	4.0	4.2	4.0	4.8	5.5	5.3	5.3
4	1.0	1.2	2.0	0.0	0.7	3.8	2.7	1.9	2.0	2.6	4.1	3.3	3.3	5.2	4.9	4.8	4.1	4.2	6.7	6.1	5.5
5	1.7	1.8	1.6	0.7	0.0	3.3	2.2	1.2	1.9	2.6	3.4	2.5	2.8	4.4	4.1	4.1	3.4	3.7	5.8	5.3	4.8
6	4.2	5.1	1.9	3.8	3.3	0.0	1.1	3.3	4.6	5.7	1.3	2.3	5.0	2.8	3.2	3.7	3.4	5.4	4.2	4.5	5.3
7	3.1	4.0	0.9	2.7	2.2	1.1	0.0	2.3	3.5	4.6	1.6	1.7	4.1	3.0	3.2	3.5	3.7	4.6	4.6	4.5	4.9
8	2.9	2.8	2.2	1.9	1.2	3.3	2.3	0.0	1.3	2.4	2.9	1.7	1.8	3.5	3.1	2.9	2.1	2.6	5.0	4.2	3.5
9	3.0	2.2	3.2	2.0	1.9	4.6	3.5	1.3	0.0	1.1	4.2	2.9	1.4	4.7	4.1	3.8	2.6	2.4	6.0	5.0	3.9
10	3.5	2.3	4.2	2.6	2.6	5.7	4.6	2.4	1.1	0.0	5.3	4.0	1.7	5.7	5.1	4.7	3.3	2.6	7.0	5.8	4.4
11	4.7	5.3	2.5	4.1	3.4	1.3	1.6	2.9	4.2	5.3	0.0	1.3	4.2	1.5	1.9	2.4	3.2	4.4	3.0	3.1	4.0
12	4.1	4.3	2.4	3.3	2.5	2.3	1.7	1.7	2.9	4.0	1.3	0.0	2.9	1.9	1.6	1.8	2.0	3.1	3.4	2.9	3.1
13	4.3	3.6	4.1	3.3	2.8	5.0	4.1	1.8	1.4	1.7	4.2	2.9	0.0	4.3	3.6	3.1	1.6	1.0	5.4	4.1	2.6
14	5.9	6.3	3.9	5.2	4.4	2.8	3.0	3.5	4.7	5.7	1.5	1.9	4.3	0.0	0.8	1.4	2.8	4.1	1.5	1.7	3.1
15	5.8	5.9	4.0	4.9	4.1	3.2	3.2	3.1	4.1	5.1	1.9	1.6	3.6	0.8	0.0	0.6	2.0	3.3	1.9	1.3	2.3
16	5.8	5.8	4.2	4.8	4.1	3.7	3.5	2.9	3.8	4.7	2.4	1.8	3.1	1.4	0.6	0.0	1.5	2.8	2.2	1.2	1.7
17	5.1	4.7	4.0	4.1	3.4	3.4	3.7	2.1	2.6	3.3	3.2	2.0	1.6	2.8	2.0	1.5	0.0	1.3	3.7	2.5	1.4
18	5.3	4.8	4.8	4.2	3.7	5.4	4.6	2.6	2.4	2.6	4.4	3.1	1.0	4.1	3.3	2.8	1.3	0.0	5.0	3.6	1.8
19	7.5	7.8	5.5	6.7	5.8	4.2	4.6	5.0	6.0	7.0	3.0	3.4	5.4	1.5	1.9	2.2	3.7	5.0	0.0	1.5	3.5
20	7.0	7.0	5.3	6.1	5.3	4.5	4.5	4.2	5.0	5.8	3.1	2.9	4.1	1.7	1.3	1.2	2.5	3.6	1.5	0.0	2.0
21	6.5	6.1	5.3	5.5	4.8	5.3	4.9	3.5	3.9	4.4	4.0	3.1	2.6	3.1	2.3	1.7	1.4	1.8	3.5	2.0	0.0

TABLE 4.8 – Distance matrix

zone	1	2	3	4	5	6	7	8	9	10	11
a_i	250	250	350	250	250	300	250	250	350	400	250
zone	12	13	14	15	16	17	18	19	20	21	
a_i	350	200	450	250	400	250	250	400	200	400	

TABLE 4.9 – Population

Problem	Sites considered	Competitor sites	Existing sites
1	all	7	4, 17
2	except 8, 9, 10	7	4, 17
3	except 10	4, 11	5, 17
4	all	14	4, 13
5	except 8, 9, 10	14	4, 13
6	all	11	5, 13

TABLE 4.10 – Location status

Problem	Variables	Constraints	Objectives
1	882	466	5
2	648	346	5
3	800	425	5
4	882	466	5
5	648	346	5
6	882	466	5

TABLE 4.11 – Problem sizes

Problem	New store	time (s)
1	3	0.000000
2	3	0.000000
3	3	0.000000
4	21	0.000000
5	21	0.000000
6	3	0.000000

TABLE 4.12 – New site for location

4.4 Conclusion

We have introduced in this chapter a new approach for solving the multiobjective mixed 0-1 linear programs and two applications of this approach in nurse rostering and planning franchise expansion problems. The proposed method is also based on scalarization techniques but all supported and nonsupported solutions of the multiobjective mixed 0-1 linear program can be found, say the Pareto set, by varying feasible solutions and considering different positive vectors λ. Unlike the weighted sum method which consists of transforming the multiobjective problem into mono one where the feasible set is unchanged, the proposed method takes into account new constraints which allows us to check whether a feasible solution of the multiobjective problem is efficient. Moreover, due to the proposed method, the multiobjective problem can be equivalently reformulated as maximizing the function q_λ on the feasible set $\mathcal{X}$. Therefore, optimizing over the efficient set of a multiobjective problem can be formulated as a mathematical program. In addition, our approach could be extended, from linear objective functions, to quadratic objective functions. Works in this direction are in progress.

Chapitre 5

Programmation DC et DCA pour la résolution du problème de moindres carrés linéaires en variables entières bornées/ factorisation en matrices non négatives

---

Nous proposons dans ce chapitre des approches basées sur la programmation DC et DCA pour résoudre le problème de moindres carrés linéaires en variables entières bornées et celui de factorisation en matrices non négatives (Nonnegative Matrix Factorization (NMF)).

5.1 Problème de moindres carrés linéaires en variables entières bornées

5.1.1 Introduction

Etant donné un vecteur y de $\mathbb{R}^m$ et une matrice A de $\mathbb{R}^{m\times n}$ de rang n, le problème de moindres carrés linéaires en variables entières (ILS- *« integer least squares problem »* en anglais) est défini comme suit :

$$\min\{\|Ax-y\|_2^2 \mid x\in\mathbb{Z}^n\}. \tag{5.1}$$

Ce problème résulte de nombreuses applications telles que la télécommunication sans fil, la cryptographie, le codage treillis, l'imagerie radar, ect. (voir e.g., [6, 56, 114] et ses références). Il a été prouvé que le problème (5.1) est NP-difficile dans [152]. Dans certaines applications de télécommunication sans fil, il faut que x soit borné (voir e.g., [20,33]). Cela implique le problème suivant :

$$\min\{\|Ax-y\|_2^2 \mid x\in\mathbb{Z}^n,\ l\le x\le u\} \tag{5.2}$$

où $l, u \in \mathbb{Z}^n$. Ce problème est nommé problème de moindres carrés linéaires en variables entières bornées (BILS- *« box-constrained integer least squares problem »* en anglais).

Dans la littérature, la résolution du problème ILS (5.1) ainsi que celle de BILS (5.2) comprend habituellement deux volets : la réduction et la recherche. La stratégie de réduction typique pour résoudre (5.1) est la réduction bien connue Lenstra-Lenstra-Lovász (LLL reduction) [100]. Une excellente étude sur les algorithmes de recherche ainsi que des méthodes de réduction typiques pour la résolution de (5.1) peut être trouvée dans [6]. Dans [20], la mise en œuvre de la stratégie de recherche de Schnorr-Euchner (voir [141]) présentée dans [6] pour la résolution de ILS (5.1), a été modifiée pour résoudre le problème BILS (5.2). Ensuite, deux algorithmes de recherche basés sur la stratégie de recherche de Schnorr-Euchner et sur la stratégie de recherche de Pohst (voir [133, 154]) respectivement, ont été introduits pour traiter le problème BILS (5.2). Dans [33], trois stratégies de réduction ont été proposées pour résoudre (5.2). Cependant les stratégies de réduction mentionnées ci-dessus n'ont utilisé que des informations de la matrice A. Ce n'est pas le cas de [26] dans lequel les auteurs ont utilisé toutes les informations disponibles du problème (5.2) et ils ont présenté une nouvelle stratégie de réduction pour sa résolution. Il a été prouvée cette stratégie est plus efficace que celles proposées dans [20] et [33].

Dans cette section, nous proposons une approche basée sur la programmation DC et DCA, qui est sensiblement différente des algorithmes existants, pour résoudre le problème BILS (5.2). Nous utilisons tout d'abord la technique de pénalisation en programmation avec des variables entières afin de reformuler le problème (5.2) comme un programme DC, puis développons une version de DCA pour la résolution du problème résultant.

5.1.2 Programmation DC et DCA pour la résolution de (BILS)

Considérons la fonction de pénalisation suivante :

$$p : \mathbb{R}^n \longrightarrow \mathbb{R},\ p(x) := \sum_{i=1}^{n} \sin^2(\pi x_i) \tag{5.3}$$

Cette fonction continue p qui est introduite au Chapitre 1 et utilisée dans le Chapitre 2, vérifie : $p(x) \leq 0 \Longleftrightarrow p(x) \equiv 0 \Longleftrightarrow x \in \mathbb{Z}^n$, et $p(x) \geq 0, \forall x \in \mathbb{R}^n$. Par ailleurs, p est une fonction DC avec décomposition $p = \phi - \varphi$ où

$$\phi(x) := \left[\frac{\eta}{2} x^T x\right] \text{ et } \varphi(x) := \left[\sum_{i=1}^{n} \left(\frac{\eta}{2} x_i^2 - \sin^2(\pi x_i)\right)\right] \text{ avec } \eta \geq 2\pi^2.$$

Le problème BILS (5.2) équivaut donc au problème ci-dessous :

$$\min\{f(x) := \|Ax - y\|_2^2 \mid l \leq x \leq u,\ p(x) = 0\}. \tag{5.4}$$

Le problème de pénalisation de (5.4) est construit comme suit

$$\min\{F_t(x) := f(x) + tp(x) \mid l \leq x \leq u\} \tag{5.5}$$

où t est un nombre réel strictement positif. (5.5) est un programme DC car F_t peut être exprimé comme la différence de deux fonctions convexes : $F_t(x) = G(x) - H(x)$ où l'on peut choisir

$$\text{soit } \begin{cases} G(x) := f(x) + t\phi(x) = \|Ax - y\|_2^2 + \dfrac{t\eta}{2}\|x\|_2^2 \\ H(x) := t\varphi(x) = \displaystyle\sum_{i=1}^{n} \left(\frac{t\eta}{2} x_i^2 - t\sin^2(\pi x_i)\right) \end{cases} \quad \text{avec } \eta \geq 2\pi^2 \tag{5.6}$$

$$\text{soit } \begin{cases} G(x) := \dfrac{\rho}{2}\|x\|_2^2 \\ H(x) := \dfrac{\rho}{2}\|x\|_2^2 - \|Ax-y\|_2^2 - \displaystyle\sum_{i=1}^{n} t\sin^2(\pi x_i) \end{cases} \qquad \text{avec } \rho \geq 2\lambda_n(A^TA) + 2t\pi^2 \tag{5.7}$$

ici, $\lambda_n(A^TA)$ est la plus grande valeur propre de la matrice symétrique A^TA.

Comme nous avons précisé dans le chapitre 2, le problème (5.5) est équivalent au problème (5.2) dans le sens donné par le théorème ci-dessous :

Théorème 5.1 *Soit ϵ un positif réel, $0 < \epsilon < \dfrac{1}{2}$. Il existe un nombre $t > 0$ (qui dépend de ϵ), assez grand tel que si $\widetilde{x}$ est une solution globale du problème (5.5) et $\widetilde{x}$ se trouve dans un ϵ-voisinage d'un entier $x^* \in [l,u]$ alors x^* résout le problème (5.2).*

Dans l'algorithme DC (DCA) construit ci-dessous pour la résolution du problème (5.5), nous choisissons les composantes DC de F_t présentées dans (5.7).

Algorithm 1

- ***Initialisation :***
  Choisir un point initial $x^0 \in \mathbb{R}^n$.
  Soient θ_1 et θ_2 deux nombres strictement positifs suffisamment petits.
  Soit t un nombre réel strictement positif.
  Itération $k \longleftarrow 0$.
- ***Répéter :***
  ◇ Calculer $z^k \in \partial H(x^k)$:

  $$z^k \in \partial H(x^k) \Longleftrightarrow z^k = \rho x^k - (2(A^TA)x^k - 2y^TA) - v^k,\ \forall i = 1,\ldots,n$$

  où $v_i^k = \pi t \sin(2\pi x_i^k),\ \forall i$.
  ◇ Résoudre le programme quadratique convexe suivant

  $$\min\left\{\frac{\rho}{2}\|x\|_2^2 - \langle z^k, x\rangle \mid l \leq x \leq u\right\} \tag{5.8}$$

  pour obtenir $x^{k+1} \in \partial G^*(z^k)$.
  ◇ $k \longleftarrow k+1$.
- ***Jusqu'à :*** Si $\|x^k - x^{k+1}\|_2 \leq \theta_1$ ou $|F_t(x^k) - F_t(x^{k+1})| \leq \theta_2(1 + |F_t(x^k)|)$ alors **STOP** : x^{k+1} est une solution du problème (5.5). Prenons $\widehat{x} \in \mathbb{Z}^n$ tel que $\widehat{x}_i$ est l'entier le plus proche de x_i^{k+1}. Le point $\widehat{x}$ sera donc une solution réalisable de (5.2).

Notons que à chaque itération k dans Algorithme 1, la solution du problème (5.8) est explicite,

$$x_i^{k+1} = \begin{cases} \dfrac{z_i^k}{\rho} & \text{si } l_i \leq \dfrac{z_i^k}{\rho} \leq u_i \\ l_i & \text{si } \dfrac{z_i^k}{\rho} \leq l_i \\ u_i & \text{si } \dfrac{z_i^k}{\rho} \geq u_i \end{cases} \qquad \forall i = 1,\ldots,n \tag{5.9}$$

et il ne demande aucun solveur pour la résolution.

5.2 Problème de factorisation en matrices non négatives (NMF)

5.2.1 Introduction

Le problème de factorisation en matrices à coefficients non négatifs s'exprime comme suit : étant donnée une matrice $\mathbf{V}$ de dimensions $m \times n$ à coefficients non négatifs et un rang désiré $r < \min\{m, n\}$, la NMF consiste à calculer pour $\mathbf{V}$ une approximation $\widehat{\mathbf{V}}$ qui peut être exprimée comme le produit de deux matrices $\mathbf{W}$ et $\mathbf{H}$,

$$\mathbf{V} \approx \mathbf{WH} = \widehat{\mathbf{V}} \tag{5.10}$$

où $\mathbf{W} \in \mathbb{R}^{m\times r}$ et $\mathbf{H} \in \mathbb{R}^{r\times n}$ sont non négatifs. L'ordre du modèle r est généralement choisi au départ tel que $(m+n)r \ll mn$, de façon à réduire la dimension des données.

La NMF a été introduite par Paatero et Tapper [119] en 1994. Elle a été étudiée intensivement dès que l'article de Lee et Seung [97] été publié en 1999. Il y a de nombreuses applications de la NMF dans plusieurs domaines dont nous pouvons citer ci-dessous quelques exemples :

- Traitement d'image [53, 97, 104],
- Traitement de texte [14, 35, 97],
- Economie [36],
- Biologie [72, 107],
- Gastronomie [157],
- Transcription de musique polyphonique [15, 143].

Le problème de NMF peut être formalisé comme la minimisation d'une fonction de coût définie par

$$D(\mathbf{V}|\widehat{\mathbf{V}}) = \sum_{i=1}^{m}\sum_{j=1}^{n} d(v_{ij}|\widehat{v}_{ij}) \tag{5.11}$$

où $\widehat{v}_{ij} = \sum_{k=1}^{r} w_{ik}h_{kj}$ et d est une divergence scalaire, c'est à dire une fonction telle que $d(a|b) \geq 0, \forall a, b \in \mathbb{R}_+$, et $d(a|b) = 0$ si et seulement si $b = a$. Plus précisément, le problème (5.10) est souvent reformulé comme le problème d'optimisation suivant

$$\min\{D(\mathbf{V}|\mathbf{WH}),\ \text{s.t. } \mathbf{W}, \mathbf{H} \geq 0\}. \tag{5.12}$$

Les fonctions de coût les plus populaires pour la NMF sont la distance euclidienne (EUC) et la divergence généralisée de Kullback-Leibler (KL), qui ont été particulièrement popularisées par Lee et Seung [97, 98]. La distance euclidienne (EUC) correspond à

$$d_{EUC}(a|b) = \frac{1}{2}(a-b)^2 \tag{5.13}$$

et la divergence généralisée de Kullback-Leibler (KL) correspond à :

$$d_{KL}(a|b) = a\log(\frac{a}{b}) - a + b \tag{5.14}$$

Il faut citer aussi la divergence d'Itakura-Saito (IS),

$$d_{IS}(a|b) = \frac{a}{b} - \log(\frac{a}{b}) - 1 \tag{5.15}$$

qui a été utilisée dans les applications audio dans [44]. En outre, les β-divergences, introduites par [38], généralisent les trois divergences précédentes. Elles sont définies pour $\beta \in \mathbb{R} \setminus \{0, 1\}$ comme suit :

$$d_\beta(a|b) = \frac{1}{\beta(\beta-1)}(a^\beta + (\beta-1)b^\beta - \beta a b^{\beta-1}) \tag{5.16}$$

La distance euclidienne EUC correspond à $\beta = 2$. Les divergences KL et IS sont respectivement obtenues par passage à la limite lorsque $\beta \longrightarrow 1$ et $\beta \longrightarrow 0$. De plus, la fonction $d_\beta(a|b)$ est convexe par rapport à b si et seulement si $\beta \in [1, 2]$, ainsi la distance EUC et la divergence KL sont convexes mais la divergence IS ne l'est pas.

En général, la fonction de coût $D(\mathbf{V}|\widehat{\mathbf{V}})$ n'est pas convexe par rapport au couple $(\mathbf{W}, \mathbf{H})$. Pourtant, si la divergence $d(a|b)$ est convexe par rapport à b, alors la fonction de coût $D(\mathbf{V}|\widehat{\mathbf{V}})$ est elle-même convexe par rapport à $\mathbf{W}$ (et par rapport à $\mathbf{H}$) lorsque $\mathbf{H}$ (resp. $\mathbf{W}$) est fixé.

On voit que pour tout $\beta > 0$, on a la limite $\lim\limits_{b \to +\infty} d_\beta(a|b) = +\infty$, et de plus la fonction $d_\beta(a|b)$ est continue, donc le problème de NMF possède au moins une solution.

Afin de minimiser les fonctions de coût précédentes, de nombreux algorithmes ont été proposés. Le plus populaire est les règles de mise à jour multiplicatives des facteurs $\mathbf{W}$ et $\mathbf{H}$, présentées par Lee et Seung [97] pour la distance EUC et la divergence KL, qui garantissent la non négativité de $\mathbf{W}$ et $\mathbf{H}$ à chaque itération ainsi que la décroissance monotone de la fonction de coût $D(\mathbf{V}|\mathbf{WH})$ ([98]). En considérant la distance EUC, la mise à jour des facteurs $\mathbf{W}$ et $\mathbf{H}$ est comme suit

$$\begin{cases} \mathbf{H} \leftarrow \mathbf{H} \otimes \dfrac{\mathbf{W}^T\mathbf{V}}{\mathbf{W}^T(\mathbf{WH})} \\ \mathbf{W} \leftarrow \mathbf{W} \otimes \dfrac{\mathbf{VH}^T}{(\mathbf{WH})\mathbf{H}^T} \end{cases} \tag{5.17}$$

où le symbole $\otimes$ et la barre de fraction désignent respectivement le produit et la division matriciels terme à terme. En considérant la divergence KL, la mise à jour est

$$\begin{cases} \mathbf{H} \leftarrow \mathbf{H} \otimes \dfrac{\mathbf{W}^T(\mathbf{V} \oslash (\mathbf{WH}))}{\mathbf{W}^T\mathbf{1}} \\ \mathbf{W} \leftarrow \mathbf{W} \otimes \dfrac{(\mathbf{V} \oslash (\mathbf{WH}))\mathbf{H}^T}{\mathbf{1}\mathbf{H}^T} \end{cases} \tag{5.18}$$

où $\oslash$ désigne aussi la division matricielle terme à terme. Ces algorithmes ont été ensuite généralisés pour la β-divergence dans [75] où l'auteur a également prouvé la décroissance de la fonction objectif pour $\beta \in [1, 2]$ à chaque itération.

Par ailleurs, il existe dans la littérature d'autres approches pour résoudre le problème de NMF dont nous pouvons citer ci-dessous quelques unes :

- Méthode du gradient projeté [106],
- Méthode du gradient conjugué [158],
- Algorithmes de moindres carrés alternés [43],
- Algorithmes de quasi-Newton [30].

Ces approches aident à résoudre plus rapidement avec de meilleurs valeurs de l'objectif que les règles de Lee et Seung mais ils ne conservent pas naturellement la non négativité. Il demande donc d'y ajouter des étapes supplémentaires pour assurer cette contrainte.

Concernant la convergence de l'algorithme multiplicatif proposé par Lee et Seung, bien que la fonction de coût décroisse à chaque itération, la convergence de la suite des valeurs de $\mathbf{W}$ et $\mathbf{H}$ n'est pas garantie. De plus, la décroissance des valeurs de $D(\mathbf{V}|\mathbf{WH})$ ne permet pas d'affirmer que le point limite $(\mathbf{W}, \mathbf{H})$ est un minimum local, ni un point stationnaire de la fonction objectif.

Dans ce travail, nous nous sommes intéressés au choix de la distance EUC pour exprimer la fonction de coût. Nous proposons dans la suite une approche basée sur la programmation DC et l'algorithme DCA pour la résolution du problème de NMF (5.12). Grâce à la convergence de DCA, cette approche assure également la décroissance de la fonction de coût, et de plus la suite des valeurs $(\mathbf{W}, \mathbf{H})$ générées par l'algorithme converge vers un point KKT (Karush-Kuhn-Tucker) généralisé.

5.2.2 Décomposition DC de la fonction de coût

Considérons la fonction de coût euclidienne EUC,

$$D(\mathbf{V}|\mathbf{WH}) = \sum_{i=1}^{m}\sum_{j=1}^{n} d\left(v_{ij}|\sum_{k=1}^{r} w_{ik}h_{kj}\right) = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}\left(v_{ij} - \sum_{k=1}^{r} w_{ik}h_{kj}\right)^2 \tag{5.19}$$

Le problème (5.12) est réécrit comme suit

$$\min\left\{ f(\mathbf{W}, \mathbf{H}) := \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}\left(v_{ij} - \sum_{k=1}^{r} w_{ik}h_{kj}\right)^2,\ \text{s.t. } \mathbf{W}, \mathbf{H} \geq 0 \right\} \tag{5.20}$$

Comme nous savons, quand la matrice $\mathbf{W}$ (resp. $\mathbf{H}$) est fixée, la fonction $f(\mathbf{W}, \mathbf{H})$ sera convexe en $\mathbf{H}$ (resp. en $\mathbf{W}$), et le problème (5.20) deviendra un problème de moindres carrés linéaires. Néanmoins, la fonction objectif f de (5.20) n'est pas convexe par rapport au couple $(\mathbf{W}, \mathbf{H})$. Dans ce qui suit, nous montrons que f est une fonction DC en explicitant une simple décomposition DC.

En effet, pour tout triplet (i, j, k), on peut exprimer le produit $w_{ik}h_{kj}$ comme la différence de deux fonctions convexes comme suit

$$w_{ik}h_{kj} = \left(\frac{1}{2}(w_{ik} + h_{kj})^2\right) - \left(\frac{1}{2}(w_{ik}^2 + h_{kj}^2)\right) \tag{5.21}$$

car $w_{ik}, h_{kj} \geq 0, \forall i, j, k$. Ainsi,

$$\begin{aligned}
f(\mathbf{W}, \mathbf{H}) &= \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}\left(v_{ij} - \sum_{k=1}^{r} w_{ik}h_{kj}\right)^2 \\
&= \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}\left[\left(v_{ij} + \sum_{k=1}^{r}\frac{1}{2}(w_{ik}^2 + h_{kj}^2)\right) - \left(\sum_{k=1}^{r}\frac{1}{2}(w_{ik} + h_{kj})^2\right)\right]^2 \\
&= \sum_{i=1}^{m}\sum_{j=1}^{n}\left[\left(v_{ij} + \sum_{k=1}^{r}\frac{1}{2}(w_{ik}^2 + h_{kj}^2)\right)^2 + \left(\sum_{k=1}^{r}\frac{1}{2}(w_{ik} + h_{kj})^2\right)^2\right] \\
&\quad - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}\left[\left(v_{ij} + \sum_{k=1}^{r}\frac{1}{2}(w_{ik}^2 + h_{kj}^2)\right) + \left(\sum_{k=1}^{r}\frac{1}{2}(w_{ik} + h_{kj})^2\right)\right]^2
\end{aligned}$$

Posons

$$\mathfrak{g}(\mathbf{W},\mathbf{H}) := \sum_{i=1}^{m}\sum_{j=1}^{n}\left[\left(v_{ij}+\sum_{k=1}^{r}\frac{1}{2}(w_{ik}^2+h_{kj}^2)\right)^2+\left(\sum_{k=1}^{r}\frac{1}{2}(w_{ik}+h_{kj})^2\right)^2\right] \tag{5.22}$$

et

$$\mathfrak{h}(\mathbf{W},\mathbf{H}) := \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}\left[\left(v_{ij}+\sum_{k=1}^{r}\frac{1}{2}(w_{ik}^2+h_{kj}^2)\right)+\left(\sum_{k=1}^{r}\frac{1}{2}(w_{ik}+h_{kj})^2\right)\right]^2 \tag{5.23}$$

alors $\mathfrak{g},\mathfrak{h}$ sont des fonctions convexes par rapport à $(\mathbf{W},\mathbf{H})$ (car v_{ij} sont des constants non négatifs et les variables $\mathbf{W},\mathbf{H}\geq 0$) et $f=\mathfrak{g}-\mathfrak{h}$ est une décomposition DC de la fonction de coût.

Le problème (5.20) est donc un programme DC

$$\min\{f(\mathbf{W},\mathbf{H}) := \mathfrak{g}(\mathbf{W},\mathbf{H})-\mathfrak{h}(\mathbf{W},\mathbf{H}) \mid \mathbf{W},\mathbf{H}\geq 0\} \tag{5.24}$$

En mettant la correspondance entre une matrice $\mathbf{A}$ de dimensions $m\times n$ et un vecteur

$$vect(\mathbf{A}) = \begin{pmatrix} A_{.,1}\\ A_{.,2}\\ \vdots\\ A_{.,n}\end{pmatrix}$$

de $\mathbb{R}^{mn}$, où $A_{.,\ell}$ est la $\ell^{\text{ème}}$ colonne de la matrice $\mathbf{A}$, les fonctions $f,\mathfrak{g},\mathfrak{h}$ sont désormais considérées comme des fonctions de $(m+n)r$ variables. Cette correspondance est utilisée dans le calcul des sous-gradients de $\mathfrak{h}$ au point $(\mathbf{W},\mathbf{H})$ et dans la résolution des programmes convexes dans le schéma DCA.

Nous décrivons maintenant une version de DCA pour la résolution de (5.24).

5.2.3 Programmation DC et DCA pour la résolution de NMF

Algorithme 2

- **Initialisation**
  Choisir deux matrices $\mathbf{W}^0\in\mathbb{R}^{m\times r}$ et $\mathbf{H}^0\in\mathbb{R}^{r\times n}$.
  Choisir la tolérance $\epsilon>0$ suffisamment petit.
  Itération $\ell\longleftarrow 0$.
- **Répéter**
  - ◇ Calculer $(\mathbf{X}^\ell,\mathbf{Y}^\ell)\in\partial\mathfrak{h}(\mathbf{W}^\ell,\mathbf{H}^\ell)$.
  - ◇ Résoudre le programme convexe suivant pour obtenir $(\mathbf{W}^{\ell+1},\mathbf{H}^{\ell+1})$:

$$\min\{\mathfrak{g}(\mathbf{W},\mathbf{H})-\langle(\mathbf{X}^\ell,\mathbf{Y}^\ell),(\mathbf{W},\mathbf{H})\rangle \mid \mathbf{W},\mathbf{H}\geq 0\} \tag{5.25}$$

  Ici, le produit $\langle(\mathbf{X}^\ell,\mathbf{Y}^\ell),(\mathbf{W},\mathbf{H})\rangle$ est calculé via l'identification par les colonnes entre l'espace vectoriel des matrices $m\times n$ et l'espace des vecteurs mn (m,n sont donnés auparavant). Ce produit égale exactement à $(\mathbf{X}^\ell,\mathbf{Y}^\ell)\otimes(\mathbf{W},\mathbf{H})$ où le symbole $\otimes$ désigne le produit matriciel terme à terme.
- **Jusqu'à** $|f(\mathbf{W}^{\ell+1},\mathbf{H}^{\ell+1})-f(\mathbf{W}^\ell,\mathbf{H}^\ell)|\leq\epsilon$

Dans cet algorithme, à chaque itération ℓ, un sous-gradient $(\mathbf{X}^\ell, \mathbf{Y}^\ell)$ de $\mathfrak{h}$ au point $(\mathbf{W}^\ell, \mathbf{H}^\ell)$ est calculé comme suit : $\forall i = 1, \ldots, m$, $\forall s = 1, \ldots, r$, $\forall j = 1, \ldots, n$,

$$\begin{aligned} x_{is}^\ell &= \sum_{j=1}^{n} (2w_{is}^\ell + h_{sj}^\ell) \left(v_{ij} + \sum_{k=1}^{r} \frac{1}{2}((w_{ik}^\ell)^2 + (h_{kj}^\ell)^2) + \sum_{k=1}^{r} \frac{1}{2}(w_{ik}^\ell + h_{kj}^\ell)^2 \right) \\ &= \sum_{j=1}^{n} (2w_{is}^\ell + h_{sj}^\ell) \left(v_{ij} + \sum_{k=1}^{r} \left[(w_{ik}^\ell)^2 + (h_{kj}^\ell)^2 + w_{ik}^\ell h_{kj}^\ell \right] \right) \end{aligned} \tag{5.26}$$

et

$$\begin{aligned} y_{sj}^\ell &= \sum_{i=1}^{m} (w_{is}^\ell + 2h_{sj}^\ell) \left(v_{ij} + \sum_{k=1}^{r} \frac{1}{2}((w_{ik}^\ell)^2 + (h_{kj}^\ell)^2) + \sum_{k=1}^{r} \frac{1}{2}(w_{ik}^\ell + h_{kj}^\ell)^2 \right) \\ &= \sum_{i=1}^{m} (w_{is}^\ell + 2h_{sj}^\ell) \left(v_{ij} + \sum_{k=1}^{r} \left[(w_{ik}^\ell)^2 + (h_{kj}^\ell)^2 + w_{ik}^\ell h_{kj}^\ell \right] \right) \end{aligned} \tag{5.27}$$

5.3 Conclusion

Dans ce chapitre, nous avons considéré deux problèmes : le problème de moindres carrés linéaires en variables entières bornées et celui de factorisation en matrices non négatives (le dernier utilise la distance euclidienne pour exprimer la fonction de coût). Nous avons suggéré d'utiliser la programmation DC et DCA pour leur résolution. Concernant le problème de moindres carrés linéaires en variables entières bornées, nous avons utilisé toutes les informations données du problème mais pas forcément sa structure spécifique. Quant au problème de factorisation en matrices non négatives, l'algorithme proposé assure théoriquement la non négativité des facteurs générés, la décroissance de la fonction de coût ainsi que la convergence de la suite des facteurs vers des points KKT généralisés. La mise en œuvre de ces approches fait l'objet de notre travail en cours.

Conclusion générale

Cette thèse est consacrée au développement de la programmation DC et DCA pour l'optimisation non convexe et l'optimisation globale en variables mixtes entières. Avec, comme applications, la modélisation et la résolution numérique d'une classe NP-difficile des programmes DC non convexes non linéaires en variables mixtes entières.

Après une présentation des fondements théoriques et algorithmiques de la programmation DC et DCA, et les techniques de globalisation Branch-and-Bound dont nécessite l'ensemble de nos travaux de recherche, nous avons développé la programmation DC et DCA ainsi que la méthode de Branch-and-Bound (B&B) pour modéliser et résoudre les programmes non convexes suivants : problème de gestion de portefeuille sous le coût de transaction concave, problème de minimisation du coût de transaction non convexe discontinu en gestion de portefeuille sous deux formes (la première est un programme DC obtenu en approximant la fonction objectif du problème original par une fonction DC polyédrale et la deuxième est un programme DC mixte 0-1 équivalent), problème multi-objectif en variables mixtes binaires, problème de moindres carrés linéaires en variables entières bornées et celui de factorisation en matrices non négatives (Nonnegative Matrix Factorization (NMF)).

Les simulations numériques montrent la robustesse, la versatilité, la rapidité (et donc la scalabilité), la flexibilité, la performance et la globalité de DCA par rapport aux méthodes existantes.

Nous sommes convaincus qu'il est possible d'améliorer les qualités de la programmation DC et DCA par une résolution plus adaptée des sous-programmes convexes générés par DCA et une implémentation plus sophistiquée de l'algorithme, tout en étudiant une stratégie de choix de points initiaux en fonction des structures spécifiques du programme considéré. Ces problématiques font l'objet de nos travaux en cours.

Bibliographie

[1] COUENNE solver. `https://projects.coin-or.org/Couenne`.

[2] CPLEX solver. `http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/`.

[3] LaGO solver. `https://projects.coin-or.org/LaGO`.

[4] K. Abhishek, S. Leyffer, and J. T. Linderoth. FilMINT : an outer-approximation based solver for nonlinear mixed integer programs. *INFORMS Journal on Computing*, 22 :555–567, 2010.

[5] C. S. Adjiman, I. P. Androulakis, and C. A. Floudas. Global optimization of mixed-integer nonlinear problems. *AIChE Journal*, 46(9) :1769–1797, 2000.

[6] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger. Closest point search in lattice. *IEEE Transactions on Information Theory*, 48(8) :2201–2214, August 2002.

[7] I. P. Androulakis, C.D. Maranas, and C. A. Floudas. α-BB : global optimization method for general constrained nonconvex problems. *Journal of Global Optimization*, 7 :337–363, 1995.

[8] R. D. Arnott and W. H. Wagner. The Measurement and Control of Trading Costs. *Financial Analysts Journal*, 46 :73–80, November-December 1990.

[9] N. Beaumont. Scheduling staff using mixed integer programming. *European Journal of Operation Research*, 98 :473–484, 1997.

[10] P. Belotti. Couenne : a user's manual. Technical report, Lehigh University, 2009.

[11] P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Wächter. Branching and bounds tightening techniques for non-convex MINLP. *Optimization Methods and Software*, 24(4-5) :597–634, 2009.

[12] R. Benayoun, J. De Montgolfier, J. Tergny, and O. Larichev. Linear programming with multiple objective functions : STEP method (STEM). *Mathematical Programming*, 1 :366–375, 1971.

[13] H.P. Benson. Existence of Efficient Solutions for Vector Maximization Problems. *Journal of Optimization : Theory and Applications*, 26(4) :569–580, December 1978.

[14] M. W. Berry and M. Browne. Email surveillance using non-negative matrix factorization. *Computational & Mathematical Organization Theory*, 11 :249–264, October 2005.

[15] N. Bertin. *Les factorisation en matrices non-négatives. Approches contraintes et probabilistes, application à la transcription automatique de musique polyphonique.* PhD thesis, Ecole Nationale Supérieure des Télécomunications, France, Octobre 2009.

[16] D. Bienstock. Computational Study of a Family of Mixed-Integer Quadratic Programming Problems. *Mathematical Programming*, 74(2) :121–140, 1996.

[17] G. R. Bitran. Theory and algorithms for linear multiple objective programs with zero-one variables. *Mathematical Programming*, 17 :362–390, 1979.

[18] G.R. Bitran. Linear multiple objective programs with zero-one variables. *Mathematical Programming*, 13 :121–139, 1977.

[19] P. Bonami, L. T Biegler, A. R. Conn, G. Cornuéjols, I. E. Grossmann, C. D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya, and A. Wächter. An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization*, 5 :186–204, 2008.

[20] J. Boutros, N. Gresset, L. Brunel, and M. Fossorier. Soft-input soft-output lattice sphere decoder for linear channels. In *Proceedings of the IEEE GLOBECOM03*, pages 1583–1587, San Francisco, 2003.

[21] S. Burer and A. N. Letchford. Non-convex mixed-integer nonlinear programming : A survey. *Surveys in Operations Research and Management Science*, 17 :97–106, 2012.

[22] E. K. Burke, P. De Causmaecker, S. Petrovic, G. Vanden Berghe, and H. Van Landeghem. Variable neighborhood search for nurse rostering problems. In *Metaheuristics : Computer Decision-Making*, pages 153–172. Kluwer Academic Publishers B.V., 2003.

[23] E. K. Burke, P. De Causmaecker, and G. Vanden Berghe. A Hybrid Tabu Search Algorithm for the Nurse Rostering Problem. *Simulated Evolution and Learning*, 1585 :187–194, 1999. Lecture Notes in Computer Science.

[24] E. K. Burke, P. De Causmaecker, G. Vanden Berghe, and H. Van Landeghem. The state of the art of nurse rostering. *Journal of Scheduling*, 7 :441–499, 2004.

[25] E. K. Burke, J. Li, and R. Qu. A hybrid model of integer programmming and variable neighbourhood search for highly-constrained nurse rostering problems. *European Journal of Operation Research*, 203 :484–493, 2010.

[26] X.-W. Chang and Q. Han. Solving Box-Constrained Integer Least Squares Problems. *IEEE Transactions on Wireless Communications*, 7(1) :277–287, January 2008.

[27] V. Chankong and Y. Y. Haimes. *Multiobjective Decision Making : Theory and Methodology*. Elsevier Science Publishing Co., New York, NY, 1983.

[28] B. Cheang, H. Li, A. Lim, and B. Rodrigues. Nurse rostering problems - A bibliographic survey. *European Journal of Operation Research*, 151 :447–460, 2003.

[29] W. Chen and W. G. Zhang. The admissible portfolio selection problem with transaction costs and improved PSO algorithm. *Physica A*, 389 :2070–2076, 2010.

[30] A. Cichocki, R. Zdunek, and S. Amari. Nonnegative matrix and tensor factorization. *IEEE Signal Processing Magazine*, 25(1) :142–145, 2008.

[31] J. R. Current and J. E. Storbeck. A Multiobjective Approach to Design Franchise Outlet Networks. *Journal of Operational Research Society*, 45(1) :71–81, January 1994.

[32] C. D'Ambrosio, A. Frangioni, L. Liberti, and A. Lodi. Experiments with a feasibility pump approach for nonconvex MINLPs. In P. Festa, editor, *Proceedings of the 9th Symposium on Experimental Algorithms, Lecture Notes in Computer Science*, volume 6049, Springer, Berlin, 2010.

[33] M. O. Damen, H. El Gamal, and G. Caire. On maximum-likelihood detection and the search for the closest lattice point. *IEEE Transactions on Information Theory*, 49(10) :2389–2402, October 2003.

[34] M. H. A. David and A. R. Norman. Portfolio selection with transaction costs. *Mathematics of Operations Research*, 15(4) :676–713, November 1990.

[35] C. Ding, T. Li, and W. Peng. On the equivalence between non-negative matrix factorization and probalistic latent semantic indexing. *Computational Statistics and Data Analysis*, 52(8) :3913–3927, Avril 2008.

[36] K. Drakakis, S. Rickard, and A. De Fréin, R. Cichocki. Analysis of financial data using non-negative matrix factorization. *International Mathematical Forum*, 3(37-40) :1853–1870, 2008.

[37] M. A. Duran and I. E. Grossmann. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical Programming*, 36 :307–339, 1986.

[38] S. Eguchi and Y. Kano. Robustifying maximum likelihood estimation. Technical report, Tokyo Institute of Statistical Mathematics, Tokyo, Japon, 2001.

[39] M. Ehrgott. *Multicriteria Optimization*. Springer, Berlin Heidelberg, 2005.

[40] M. Ehrgott. A discussion of scalarization techniques for multiple objective integer programming. *Annals of Operation Research*, 147 :343–360, 2006.

[41] M. Ehrgott and X. Gandibleux. Approximative Solution Methods for Multiobjective Combinatorial Optimization. *Sociedad de Estadística e Investigación Operativia, Top*, 12(1) :1–89, 2004.

[42] A.T. Ernst, H. Jiang, M. Krishnamoorthy, and D. Sier. Staff scheduling and rostering : A review of applications, methods and models. *European Journal of Operation Research*, 53 :3–27, 2004.

[43] L. Finesso and P. Spreij. Approximate nonnegative matrix factorization via alternating minimization. In *Proceedings of the 16th International Symposium on Mathematical Theory of Networks and Systems (MTNS)*, Louvain, Belgique, 2004.

[44] C. Févotte, N. Bertin, and J-.L. Durrieu. Nonnegative matrix factorization with the itakura-saito divergence. with application to music analysis. *Neural Computation*, 21(3) :793–830, march 2009.

[45] X. Gandibleux and A. Fréville. Tabu search based procedure for solving the 0-1 multiobjective knapsack problem : the two objectives case. *Journal of Heuristics*, 6 :361–383, 2000.

[46] X. Gandibleux, N. Mezdaoui, and A. Fréville. A tabu search procedure to solve multiobjective combinatorial optimization problems. In R. Caballero, F. Ruiz, and R. Steuer, editors, *Advances in Multiple Objective and Goal Programming, Lecture Notes in Economics and Mathematical Systems*, volume 455, pages 291–300. Springer Verlag, Berlin, 1997.

[47] R. P. Ge and C. B. Huang. A Continuous Approach to Nonlinear Integer Programming. *Applied Mathematics and Computation*, 34 :39–60, 1989.

[48] A. M. Geoffrion. Generalized Benders Decomposition. *Journal of Optimization Theory and Applications*, 10(4) :237–260, 1972.

[49] A. Ghosh and C. S. Craig. FRANSYS : a franchise distribution system location model. *Journal of Retailing*, 67(4) :466–495, 1991.

[50] J. Ch. Gilbert. Optimisation Différentiable : Théorie et Algorithmes, 2008. Syllabus de cours à l'ENSTA, Paris, France.

[51] C. A. Glass and R. A. Knight. The nurse rostering problem : A critical appraisal of the problem structure. *European Journal of Operation Research, Discrete Optimization*, 202 :379–389, 2010.

[52] I. E. Grossmann, J. Viswanathan, A. Vecchietti, R. Raman, and E. Kalvelagen. GAMS/DICOPT : A discrete continuos optimization package. *Optimization and Engineering*, 2002.

[53] D. Guillamet, B. Schiele, and Vitrià. Color histogram classification using NMF. Technical report, Centre de Visió Per Computador, Université autonome de Barcelone, October 2001.

[54] O. K. Gupta and V. Ravindran. Branch and Bound experiments in convex nonlinear integer programming. *Management Science*, 31 :1533–1546, 1985.

[55] M. P. Hansen. Tabu search for multiobjective combinatorial optimization : TAMOCO. *Control and Cybernetics*, 29 :799–818, 2000.

[56] A. Hassibi and S. Boyd. Integer parameter estimation in linear models with applications to GPS. *IEEE Transactions on Signal Processing*, 46(11) :2938–2952, November 1998.

[57] R. Hemmecke, M. Köppe, J. Lee, and R. Weismantel. Nonlinear Integer Programming. In M. Jünger, T. Liebling, D. Naddef, G. Nemhauser, W. Pulleyblank, G. Reinelt, G. Rinaldi, and Wolsey L., editors, *50 Years of Integer Programming 1958-2008 : The Early Years and State-of-the-Art Surveys*, pages 561–618. Springer, 2010.

[58] J. B. Hiriart-Urruty and C. Lemarechal. *Convex Analysis and Minimization algorithms*. Springer, Berlin, 1993.

[59] R. Horst. A General Class of Branch-and-Bound Methods in Global Optimisation with Some New Approaches for Concave Minimization. *Journal of Optimization Theory and Application*, 51(2) :271–291, November 1986.

[60] R. Horst. Deterministic Global Optimisation with Partition Sets Whose Feasibility Is Not Known : Application to Concave Minimization, Reverse Convex Constraints, DC-Programming, and Lipschitzian Optimization. *Journal of Optimization Theory and Application*, 58(1) :11–37, July 1988.

[61] R. Horst, P. M. Pardalos, and N. V. Thoai. *Introduction to global optimization*. Second edition, Kluwer Academic Publishers, Netherlands, 2000.

[62] R. Horst, N. V. Thoai, and H. Benson. Concave minimization via conical partitions and polyhedral outer approximation. *Mathematical Programming*, 50 :259–274, 1991.

[63] R. Horst and H. Tuy. *Global optimization*. Third edition, Springer, Berlin, 1996.

[64] S. D. Hunt. Franchising : promises, problems, prospects. *Journal of Retailing*, 53(3) :71–84, 1977.

[65] IFA. Economic Impact of Franchised Businesses. `http://www.franchise.org/Files/EIS6_2.pdf`, 2004. A Study for the International Franchise Association Educational Foundation.

[66] H. Isermann. Proper efficiency and the linear vector maximum problem. *Operation Research*, 22(1) :189–191, 1974.

[67] B. Kalantari and J.B. Rosen. An Algorithm for Global Minimization of Linearly Constrained Concave Quadratic Functions. *Mathematics of Operation Research*, 12 :544–561, 1987.

[68] R. Karuppiah and I. E. Grossmann. A Lagrangian based branch-and-cut algorithm for global optimization of nonconvex mixed-integer nonlinear programs with decomposable structures. *Journal of Global Optimization*, 41(2) :163–186, June 2008.

[69] P. J. Kaufmann and V. K. Rangan. A Model for Managing System Conflict during Franchise Expansion. *Journal of Retailing*, 66(2) :155–173, 1990.

[70] H. Kellerer, R. Mansini, and M. G. Speranza. Selecting Portfolios with Fixed Costs and Minimum Transaction Lots. *Annals of Operations Research*, 99 :287–304, 2000.

[71] P. Kesavan, R. J. Allgor, E. P. Gatzke, and P. I. Barton. Outer approximation algorithms for separable non-convex mixed-integer nonlinear programs. *Mathematical Programming*, 100 :517–535, 2004.

[72] S-.P. Kim, Y. N. Rao, D. Ergodmus, J. C. Sanchez, M.A.L. Nicolelis, and J. C. Principe. Determining pattern in neural activity for reaching movements using non-negative matrix factorization. *EURASIP Journal on Applied Signal Processing*, 19 :3113–3121, 2005.

[73] G. R. Kocis and I. E. Grossmann. Computational experience with DICOPT solving MINLP problems in process systems engineering. *Computer & Chemical Engineering*, 13 :307–315, 1989.

[74] S. Kolli and G. W. Evans. A multiple objective integer programming approach for planning franchise expansion. *Computers & Industrial Engineering*, 37 :543–561, 1999.

[75] R. Kompass. A Generalized Divergence Measure for Nonnegative Matrix Factorization. *Neural Computation*, 19(3) :780–791, March 2007.

[76] H. Konno, K. Akishino, and R. Yamamoto. Optimization of a Long-Short Portfolio under Nonconvex Transaction Cost. *Computational Optimization and Applications*, 32 :115–132, 2005.

[77] H. Konno and A. Wijayanayake. Mean-absolute deviation portfolio optimization model under transaction costs. *Journal of the Operation Research, Society of Japan*, 42(4) :422–435, December 1999.

[78] H. Konno and A. Wijayanayake. Portfolio optimization problems under concave transaction costs and minimal transaction unit constraints. *Mathematical Programming*, 89 :233–250, 2001.

[79] H. Konno and R. Yamamoto. Global Optimization Versus Integer Programming in Portfolio Optimization under Nonconvex Transaction Costs. *Journal of Global Optimization*, 32 :207–219, 2005.

[80] H. Konno and R. Yamamoto. Integer programming approaches in mean-risk models. *CMS*, 2 :339–351, 2005.

[81] H. Konno and H. Yamazaki. Mean-Absolute Deviation Portfolio Optimization Model and Its Application to Tokyo Stock Market. *Management Science*, 37(5) :519–531, May 1991.

[82] M. Köppe. On the complexity of nonlinear mixed-integer optimization. In J. Lee and S. Leyffer, editors, *Mixed-Integer Nonlinear Programming, in : IMA Volumes in Mathematics and its Applications*, volume 154, pages 533–558. Springer, Berlin, 2011.

[83] A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28 :497–520, 1960.

[84] H. A. Le Thi. DC programming and DCA. `http://lita.sciences.univ-metz.fr/~lethi/`.

[85] H. A. Le Thi. *Analyse numérique des algorithmes de l'optimisation D.C. : Approches locale et globale. Codes et simulations numériques en grande dimension. Applications.* PhD thesis, Université de Rouen, France, Décembre 1994.

[86] H. A. Le Thi. Contribution à l'optimisation non convexe et l'optimisation globale : théorie, algorithmes et applications. *Habilitation à Diriger de Recherches*, Université de Rouen :France, Juillet 1997.

[87] H. A. Le Thi and T. Pham Dinh. DC (Difference of convex functions) optimization algorithm (DCA) for globally minimizing nonconvex quadratic forms on euclidean balls and spheres. *Operations Reseach Letters*, 19 :207–216, 1996.

[88] H. A. Le Thi and T. Pham Dinh. Solving a class of linearly constrained indefinite quadratic problems by DC algorithms. *Journal of Global Optimization*, 11(3) :253–285, 1997.

[89] H. A. Le Thi and T. Pham Dinh. A Branch-and-Bound method via D.C. Optimization Algorithm and Ellipsoidal technique for Box Constrained Nonconvex Quadratic Programming Problems. *Journal of Global Optimization*, 13 :171–206, 1998.

[90] H. A. Le Thi and T. Pham Dinh. A continuous approach for globally solving linearly constrained quadratic zero-one programming problems. *Optimization : A Journal of Mathematical Programming and Operations Research*, 50(1–2) :93–120, 2001.

[91] H. A. Le Thi and T. Pham Dinh. A continuous approach for large-scale constrained quadratic zero-one programming. *Optimization*, 45(3) :1–28, 2001. (In honor of Professor ELSTER, Founder of the Journal Optimization).

[92] H. A. Le Thi and T. Pham Dinh. Large-Scale Molecular Optimization from Distance Matrices by a D.C. Optimization Approach. *SIAM Journal on Optimization*, 14(1) :77–114, 2003.

[93] H. A. Le Thi and T. Pham Dinh. The DC (Difference of convex functions) Programming and DCA Revisited with DC Models of Real World Nonconvex Optimization Problems. *Annals of Operations Research*, 133 :23–46, 2005.

[94] H. A. Le Thi, T. Pham Dinh, and V. N. Huynh. Exact penalty and error bounds in DC programming. *Journal of Global Optimization*, 52 :509–535, 2012.

[95] H. A. Le Thi, T. Pham Dinh, and L. D. Muu. Numerical solution for optimization over the efficient set by DC optimization algorithms. *Operations Reseach Letters*, 19 :117–128, 1996.

[96] H. A. Le Thi, T. Pham Dinh, and N. V. Thoai. Combination between global and local methods for solving an optimization problem over the efficient set. *European Journal of Operational Research*, 142 :258–270, 2002.

[97] D. D. Lee and S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401 :788–791, October 1999.

[98] D. D. Lee and S. Seung. Algorithms for non-negative matrix factorization. *Advances in Neural Information Proceesing systems*, 13 :556–562, 2001.

[99] S. Lee and I. E. Grossmann. A global optimization algorithm for non-convex generalized disjunctive programming and applications to process systems. *Computer & Chemical Engineering*, 25 :1675–1697, 2001.

[100] A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261 :515–534, 1982.

[101] S. Leyffer. Integrating SQP and branch-and-bound for mixed integer nonlinear programming. *Computational Optimization and Applications*, 18 :295–309, 2001.

[102] D. Li, X. L. Sun, and J. Wang. Optimal lot solution to cardinality constrained mean-variance formulation for portfolio selection. *Mathematical Finance*, 16(1) :83–101, January 2006.

[103] H. L Li and J. F. Tsai. A distributed computation algorithm for solving portfolio problems with integer variables. *European Journal of Operation Research*, 186 :882–891, 2008.

[104] S. Z. Li, X. Hou, H. Zhang, and Q. Cheng. Learning spatially localized parts-based representation. In *Proceedings IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 207–212, Hawaii, USA, 11-13 december 2001, 2001.

[105] C. C Lin and Y. T. Liu. Genetic algorithms for portfolio selection problems with minimum transaction lots. *European Journal of Operation Research*, 185 :393–404, 2008.

[106] C-.J. Lin. Projected Gradient Methods for Nonnegative Matrix Factorization. *Neural Computation*, 19(10) :2756–2779, 2007.

[107] W. Liu and K. Yuan. Sparse p-norm Nonnegative Matrix Factorization for clustering gene expression data. *Internaional Journal of Data Mining and Bioinformatics*, 2(3) :236–249, july 2008.

[108] M. S Lobo, M. Fazel, and S. Boyd. Portfolio optimization with linear and fixed transaction costs. *Annals of Operations Research*, 157 :341–365, 2007.

[109] M. J. P. Magill and G. M. Constantinides. Portfolio Selection with Transaction Costs. *Journal of Economic Theory*, 13 :245–263, 1976.

[110] R. Mansini and M. G. Speranza. Heuristic algorithms for portfolio selection problem with minimum transaction lots. *European Journal of Operation Research*, 114 :219–233, 1999.

[111] R. Marett and M. Wright. A comparison of neighborhood search techniques for multi-objective combinatorial problems. *Computers and Operations Research*, 23 :465–483, 1996.

[112] H. Markowitz. Portfolio selection. *The Journal of Finance*, 7(1) :77–91, March 1952.

[113] G. Mavrotas and D. Diakoulaki. A branch and bound algorithm for mixed zero-one multiple objective linear programming. *European Journal of Operation Research*, 107(3) :530–541, 1998.

[114] W. H. Mow. Universal lattice decoding : principle and recent advances. *Wireless Communications Mobile Computing*, 3 :553–569, 2003.

[115] L. D. Muu, T. Q. Phong, and T. Pham Dinh. Decomposition methods for solving a class of nonconvex programming problems dealing with bilinear and quadratic functions. *Computational Optimization and Application*, 4 :203–216, 1995.

[116] G. Nannicini and P. Belotti. Rounding based heuristics for nonconvex MINLPs. *Mathematical Programming Computation*, 4(1) :1–31, 2012.

[117] Y. S Niu. *Programmation DC & DCA en Optimisation Combinatoire et Optimisation Polynomiale via les Techniques de SDP*. Ph.D. diss., INSA de Rouen, Rouen, France, 2010.

[118] I. Nowak and S. Vigerske. LaGO - a (heuristic) branch and cut algorithm for non-convex MINLPs. *CEJOR*, 16 :127–138, 2008.

[119] P. Paatero and U. Tapper. Positive matrix factorization : A non-negative factor model with optimal itilization of error estimates of data values. *Environmetrics*, 5 :111–126, 1994.

[120] G. Parks and A. Suppapitnarm. Multiobjective optimization of PWR reload core designs using simulated annealing. In *Proceedings of the International Conference on Mathematics and Computation, Reactor Physics and Environmental Analysis in Nuclear Applications*, volume 2, pages 1435–1444, Madrid, Spain, September 1999.

[121] T. Pham Dinh. Algorithmes de calcul d'une forme quadratique sur la boule unité de la norme maximum. *Numerische Mathematik*, 45 :377–440, 1985.

[122] T. Pham Dinh. Algorithms for solving a class of nonconvex optimization problems. Methods of subgradients. *Mathematics for Optimization*, 1986. Fermat days 85.

[123] T. Pham Dinh. Duality in DC (difference of convex functions) optimization. Subgradient methods. *Trends in Mathematical Optimization, International Serie of Numer Math, Birkhäuser*, 84 :277–293, 1988.

[124] 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*, t. 318, Série I :379–384, 1994.

[125] T. Pham Dinh and H. A. Le Thi. Lagrangian stability and global optimization in nonconvex quadratic minimization over euclidean balls and spheres. *Journal of Convex Analysis*, 2 :263–276, 1995.

[126] T. Pham Dinh and H. A. Le Thi. Convex analysis approach to DC programming : Theory, Algorithms and Applications. *Acta Mathematica Vietnamica*, 22(1) :289–355, 1997. dedicated to Professor Hoang Tuy on the occasion of his 70th birthday.

[127] T. Pham Dinh and H. A. Le Thi. A DC optimazation algorithm for solving the trust region subproblem. *SIAM Journal of Optimization*, 8(2) :476–505, May 1998.

[128] T. Pham Dinh, N. Nguyen Canh, and H. A. Le Thi. An efficient combined DCA and B&B using DC/SDP relaxation for globally solving binary quadratic programs. *Journal of Global Optimization*, 48 :595–632, 2010.

[129] A. T. Phillips and J. B. Rosen. A parallel algorithm for constrained concave quadratic global minimization. *Mathematical Programming*, 42 :412–448, 1988.

[130] T. Q. Phong. *Analyse Numérique des Méthodes d'Optimisation Globale. Codes et Simulations numériques. Applications*. Ph.D. dissertation, Université de Rouen, France, Décembre 1994.

[131] T. Q. Phong, H. A. Le Thi, and T. Pham Dinh. Decomposition branch and bound method for globally solving linearly constrained indefinite quadratic minimization problems. *Operations Research Letters*, 17 :215–222, 1996.

[132] T. Q. Phong, H. A. Le Thi, and T. Pham Dinh. On the global solution of linearly constrained infinite quadratic minimization problems by decomposition branch and bound method. *RAIRO, Operation Research*, 30 :31–49, 1996.

[133] M. Pohst. On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications. *ACM SIGSAM bulletin*, 15(1) :37–44, Feb. 1981.

[134] I. Quesada and I. E. Grossmann. An LP/NLP based branch and bound algorithm for convex MINLP optimization problems. *Computer & Chemical Engineering*, 16 :937–947, 1992.

[135] I. Radziukynienė and A. Žilinskas. Evolutinary Methods for Multi-objective Portfolio Optimization. In *Proceedings of the World Congress on Engineering, WCE 2008*, volume II, pages 3–7, London, UK, July 2-4, 2008.

[136] R. T. Rockafellar. *Convex Analysis*. Princeton New Jersey, Princeton University Press, 1970.

[137] J. B. Rosen and P. M. Pardalos. Global minimization of large scale constrained quadratic problem by separation programming. *Mathematical Programming*, 34(2) :163–174, 1986.

[138] H. Ryoo and N. V. Sahinidis. Global optimization of nonconvex NLPs and MINLPs with applications in process design. *Computer & Chemical Engineering*, 19 :551–566, 1995.

[139] H. Ryoo and N. V. Sahinidis. A branch-and-reduce approach to global optimization. *Journal of Global Optimization*, 8 :107–138, 1996.

[140] J. D. Schaffer. *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*. PhD thesis, Vanderbilt University, Nashville, TN, 1984.

[141] C. P. Schnorr and M. Euchner. Lattice basis reduction : Improved practical algorithms and solving subset sum problems. *Mathematical Programming*, 66 :181–199, 1994.

[142] P. Serafini. Simulated annealing for multiobjective optimization problems. In *Proceedings of the 10th International Conference on Multiple Criteria Decision Making*, volume 1, pages 87–96, Taipei-Taiwan, 1992.

[143] P. Smaragdis and J. C. Brown. Non-Negative Matrix Factorization for Polyphonic Music Transcription. *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA'03)*, pages 177–180, 2003. New Paltz, New York, USA, 19-22 October.

[144] E. M. B. Smith and C. C. Pantelides. Global optimisation of non-convex MINLPs. *Computer & Chemical Engineering*, 21 :S791–S796, May 1997.

[145] M. G. Speranza. A heuristic algorithm for a portfolio optimization model applied to the Milan stock market. *Computer Ops. Res.*, 23(5) :433–441, 1996.

[146] N. Srinivas and K. Deb. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3) :221–248, 1994.

[147] R. E. Steuer and E. U. Choo. An Interactive Weighted Tchebycheff Procedure for Multiple Objective Programming. *Mathematical Programming*, 26 :326–344, 1983.

[148] R. A. Stubbs and S. Mehrotra. A branch-and-cut method for 0-1 mixed convex programming. *Mathematical Programming*, 86 :515–532, 1999.

[149] H. Tuy. *Convex Analysis and Global Optimization*. Kluwer Academic Publisher, 1998.

[150] E. L. Ulungu. *Optimisation combinatoire multicritère : Détermination de l'ensemble des solutions efficaces et méthodes interactives*. Ph.D. dissertation, Université de Mons-Hainaut, Belgium, 1993.

[151] E. L. Ulungu and J. Teghem. Heuristic for multi-objective combinatorial optimization problems with simulated annealing. *Presented at the EURO XII Conference, Helsinki*, 1992.

[152] P. Van Emde Boas. Another NP-complete partition problem and the complexity of computing short vectors in a lattice. Technical report, Mathematische Instituut, University of Amsterdam, 1981.

[153] M. Visée, J. Teghem, M. Pirlot, and E. L. Ulungu. Two-Phases Method and Branch and Bound procedures to Solve the Bi-Objective Knapsack Problem. *Journal of Global Optimization*, 12 :139–155, 1998.

[154] E. Viterbo and E. Biglieri. A universal decoding algorithm for lattice codes. In *Proceedings of the 14th Colloque GRETSI-Juan-Les-Pins*, pages 611–614, Torino, Italy, September 13-16, 1993.

[155] T. Westerlund and F. Pettersson. A cutting plane method for solving convex MINLP problems. *Computer & Chemical Engineering*, 19 :S131–S136, 1995.

[156] A. Yoshimoto. The mean-variance approach to portfolio optimization subject to transaction costs. *Journal of the Operations Research, Society of Japan*, 39(1) :99–117, March 1996.

[157] S. S. Young, P. Fogel, and D. Hawkins. Clustering Scotch Whiskies using Non-Negative Matrix Factorization. *Joint Newsletter for the Section on Physical and Engineering Science and the Quality and Productivity Section of the American Statistical Association*, 14(1) :11–13, 2006.

[158] R. Zdunek and A. Cichocki. Nonnegative matrix factorization with constrained second-order optimization. *Signal Processing*, 87(8) :1904–1916, 2007.

[159] R. E. Zeller, D. D. Achabal, and L. A. Brown. Market penetration and locational conflict in franchise systems. *Decision Sciences*, 11 :58–80, 1980.

[160] G. Zhou and M. Gen. Genetic algorithm approach on multi-criteria minimum spanning tree problem. *European Journal of Operational Research*, 114 :141–152, 1999.

[161] E. Zitzler. *Evolutionary Alorithms for Multiobjective Optimization : Methods and Applications*. Ph.D. dissertation, Swiss Federal Institute of Technology Zurich, November 1999.