

THÈSE

en vue de l'obtention du titre de

DOCTEUR DE L'UNIVERSITÉ PAUL VERLAINE-METZ

Discipline : Informatique

présentée par :

MOEINI Mahdi

Titre de la thèse :

La programmation DC et DCA pour l'optimisation de portefeuille

Date de soutenance : 27 Juin 2008

Composition du Jury :

Président	Tao PHAM DINH	*Professeur, INSA de Rouen*
Rapporteurs	Gérard PLATEAU	*Professeur, Université Paris Nord*
	Adnan YASSINE	*Professeur, Université du Havre*
Examinateurs	Van Thoai NGUYEN	*Professeur, Université de Trèves*
	Raymond BISDORFF	*Professeur, Université du Luxembourg*
Directrice	Hoai An LE THI	*Professeur, Université Paul Verlaine-Metz*

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



AVERTISSEMENT

Cette thèse est le fruit d'un long travail approuvé par le jury de soutenance et disponible à l'ensemble de la communauté universitaire élargie.
Elle est soumise à la propriété intellectuelle de l'auteur au même titre que sa version papier. Ceci implique une obligation de citation, de référencement dans la rédaction de tous vos documents.
D'autre part, toutes contrefaçons, plagiats, reproductions illicites entraînent une poursuite pénale.
Enfin, l'autorisaton de diffusion a été accordée jusqu' à nouvel ordre.

- Contact SCD Metz : daniel.michel@scd.univ-metz.fr

Ecrire au doctorant:

Code de la Proriété Intellectuelle. articles L 122. 4
Code de la Proriété Intellectuelle. articles L 335.2- L 335.10
http://www.cfcopies.com/V2/leg/leg_droi.php
http://www.culture.gouv.fr/culture/infos-pratiques/droits/protection.htm

Remerciements

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

C'est une vieille tradition que de remercier au début d'un tel travail tous ceux qui, de loin ou de près, directement ou indirectement, ont contribué à le rendre possible. C'est avec mon enthousiasme le plus vif et le plus sincère que je voudrais rendre mérite à tous ceux qui à leur manière m'ont aidé à mener à bien cette thèse.

Je remercie en premier lieu Madame le Professeur Hoai An LE THI, ma directrice de thèse pour ses précieux conseils, son aide constant ainsi que pour la confiance qu'elle m'a accordée durant la préparation de la thèse.

Je tiens ensuite à remercier particulièrement Monsieur le Professeur Tao PHAM DINH, directeur de l'équipe Modélisation et Optimisation Appliquée de l'INSA de Rouen pour ses conseils et son attention constante. Je lui exprime toutes ma reconnaissance pour sa sympathie.

Messieurs les Professeurs Gérard PLATEAU et Adnan YASSINE m'ont fait l'honneur d'accepter d'être rapporteurs de ma thèse et de juger mon travail. Je les en remercie ici très vivement.

Je tiens aussi à remercier Messieurs les Professeurs Van Thoai NGUYEN et Raymond BISDORFF leur disponibilité et pour d'avoir accepté d'être membres du jury.

Je me permets également de remercier la Ministère de la Science, de la Recherche et de la Technologie (MSRT) de l'Iran pour l'aide financière qu'elle m'a attribuée.

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

Je remercie tous mes collègues et mes amis français, iranians et vietnamiens rencontrés à Metz pour les moments agréables lors de mon séjour à Metz. Je remercie particulièrement Romuald, Thomas, Belaïd et Damien. Enfin je remercie tous ceux qui m'ont aidé de près ou de loin et tous ceux qui m'ont motivé même inconsciemment.

Table des matières

I Méthodologie **15**

1 Introduction à la programmation DC et DCA **17**

1.1 Eléments de base de l'analyse DC 18

1.1.1 Notations et propriétés 18

1.1.2 Fonctions convexes polyédrales 20

1.1.3 Fonction DC 20

1.2 Optimisation DC 22

1.2.1 Dualité DC 23

1.2.2 Optimalité globale en optimisation DC 24

1.2.3 Optimalité locale en optimisation DC 25

1.3 DCA 27

1.3.1 Principe de DCA 27

1.3.2 Existence des suites générées 28

1.3.3 Calcul des sous-gradients 29

1.3.4 Optimisation DC polyédrale 30

1.3.5 Interprétations de DCA 31

2 L'algorithme par Séparation et Evaluation (SE) **33**

2.1 Introduction 33

2.2 Le cas général de SE 34

2.2.1 Méthode de résolution et convergence 35

2.2.2 Séparation et Evaluation avec des ensembles non réalisables 39

2.2.3 Réalisation 42

2.3 Les cas particuliers de SE 46
2.3.1 Le domaine est non convexe 46
2.3.2 La fonction objectif est non convexe 48

II Gestion de portefeuille en finance : modèles et méthodes 53

3 Gestion de portefeuille : les notions de base 55
3.1 Introduction 55
3.1.1 Les rentabilités des actifs financiers 56
3.1.2 Le risque 57
3.2 Les critères 58
3.2.1 Le critère de Maximum d'Espérance d'Utilité 59
3.2.2 Dominance stochastique 61
3.2.3 La cohérence 63
3.3 Les mesures classiques de risque 63
3.3.1 La variance 64
3.3.2 La mesure de risque de baisse 67
3.3.3 Le modèle de Mean-Absolute Deviation (MAD) 67
3.3.4 Value at risk (VaR) et Value at Risk Conditionnel (CVaR) 68
3.4 La classification des mesures de risque 70
3.4.1 Une approche générale pour la représentation des mesures de risque . 70

4 Gestion de portefeuille sous les contraintes de seuil, de seuil d'achat et de cardinalité 73
4.1 Introduction 73
4.2 Le modèle moyenne-variance (MV) 75
4.3 Contraintes de seuil d'achat 76
4.3.1 Programmation DC et DCA pour la résolution du problème 76
4.4 Contraintes de cardinalité 81
4.4.1 Programmation DC et DCA pour la résolution du problème 84
4.5 Contraintes de seuil 90

4.5.1 Reformulation 91
4.5.2 Résolution de (P_{seuil}^{DC}) par DCA 95
4.5.3 Résolution globale de (P_{seuil}) 97
4.6 Conclusion 100

5 Gestion de portefeuille avec la mesure de risque de baisse sous les contraintes de cardinalité 107

5.1 Introduction 107
5.2 Description et formulation 108
5.3 Contraintes de cardinalité 114
5.4 Programmation DC et DCA pour la résolution du problème 115
5.4.1 Reformulation 115
5.4.2 Résolution de (P_{DC}) par DCA 116
5.5 Résolution globale 118
5.6 Expériences numériques 118
5.7 Conclusion 122

6 Gestion de portefeuille sous les fonctions de coûts de transaction non convexes 123

6.1 Introduction 123
6.2 Description et formulation 125
6.2.1 Modèle de Déviation Moyenne-Absolue (MAD) 125
6.2.2 Coûts de transaction 128
6.2.3 Le modèle de choix de portefeuille sous les coûts de transaction . . . 129
6.2.4 Formulation $0-1$ du modèle de choix de portefeuille sous les coûts de transaction en escalier 130
6.3 Approximation des fonctions de coûts de transaction 131
6.3.1 Approximation de la fonction f_u par des fonctions DC polyédrales . . 131
6.3.2 Approximation de la fonction objectif de (P) 135
6.4 Résolution de (P_{DC}) par DCA 135
6.4.1 Calcul de $\partial H(x, \psi, \phi)$ 136
6.4.2 Calcul de $\partial \left((G - \sum_{u=1}^{n} r_u x_u) + \chi_{\mathcal{D}} \right)^* (y^k, \xi^k, \eta^k)$ 136

6.4.3 Algorithme de DCA pour résoudre (P_{DC}) 137
6.4.4 Trouver un bon point de départ pour DCA 137
6.5 Résolution globale du problème (P) 138
6.6 Expériences numériques 139
6.6.1 Commentaires sur les résultats 143
6.6.2 Influence des fonctions de coût de transaction 145
6.7 Conclusion 146

7 Une analyse au pire des cas pour l'investissement robuste en gestion de portefeuille en présence de contraintes de cardinalité 149
7.1 Introduction 149
7.2 Description et formulation 150
7.2.1 Modèle MV 151
7.2.2 Modèle min-max pour les décisions robustes 152
7.2.3 Robustesse du modèle min-max 153
7.3 Contraintes de cardinalité 154
7.4 Programmation DC et DCA pour la résolution du problème 155
7.4.1 Reformulation 155
7.4.2 Résolution de (P_{DC}) par DCA 156
7.5 Expériences numériques 156
7.5.1 Résultats numériques 159
7.6 Conclusion 166

Table des figures

2.1 Un schéma de base pour un algorithme par SE pour les problèmes mixtes . . 47

2.2 Un schéma de base pour un algorithme par SE révisé (combiné avec DCA) . 49

3.1 Exemple de la courbe des portefeuilles efficients 65

3.2 VaR et CVaR en $\beta\%$ de confiance et la fonction de densité f 69

4.1 Les optimums globaux et les valeurs optimales fournies par DCA 100

4.2 CPU en seconde 103

4.3 Le nombre d'itération de chaque algorithme 104

4.4 Le nombre de relance de DCA pendant l'exécution de l'approche combinée (SE-DCA) 105

6.1 Fonctions des coûts de transaction 128

6.2 Fonction des coûts de transaction non convexe et constante par morceaux . 129

6.3 Les points supplémentaires 132

6.4 Les fonctions convexes polyédrales 132

6.5 Les fonctions linéaires par morceaux $F_u(x_u)$ 133

6.6 Enveloppe convexe de fonction de coûts de transaction en escalier 139

6.7 Séparation sur un intervalle 139

6.8 Le nombre d'actifs composant le portefeuille, sans et avec les coûts de transaction 145

6.9 Le nombre d'actifs composant le portefeuille, sans et avec les coûts de transaction 147

7.1 Les strategies min-max avec un seul scénario de rendement et avec multiples scénarios de rendement 163

7.2 Evaluation croisée avec un seul scénario de rendement 164

7.3 Non-infériorité de min-max 165
7.4 Analyse au pire des cas 167

Liste des tableaux

4.1 La performance de l'algorithme pour le premier jeu de données en utilisant deux algorithmes par SE 82

4.2 La performance de l'algorithme pour le deuxième jeu de données en utilisant l'algorithme par SE (le deuxième) 83

4.3 La performance des algorithmes pour les différentes valeurs de cardinalité . . 89

4.4 La performance des algorithmes pour le premier jeu de données (S&P 100) en utilisant l'algorithme par SE et l'approche combinée 101

4.5 La performance des algorithmes pour le deuxième jeu de données (Dax 100) en utilisant l'algorithme par SE et l'approche combinée 102

5.1 La performance des algorithmes pour le premier ensemble de paramètres . . 120

5.2 La performance des algorithmes pour le deuxième ensemble de paramètres . 121

6.1 Les coûts de transaction 140

6.2 La performance des algorithmes pour le premier jeu de données ($S\&P$) . . . 141

6.3 La performance des algorithmes pour le deuxième jeu de données 142

6.4 La performance des algorithmes pour le troisième jeu de données ($Russell$) . 144

7.1 Les résultats pour différentes valeurs de $card$. Puisque $\alpha = 1.0$ alors les valeurs de $Risque$ correspondent aux valeurs optimales de fonction objectif. 160

7.2 Comparaisons des résultats pour un seul scénario 161

7.3 Comparaisons des résultats pour trois scénarios 162

Liste des Publications et Conférences

MOEINI Mahdi

Article avec comité de lecture

- H.A Le Thi, M. Moeini and T. Pham Dinh, *Portfolio Selection under Downside Risk Measures and Cardinality Constraints based on DC Programming and DCA*, to appear in *Computational Management Science*.

- H.A Le Thi, M. Moeini and T. Pham Dinh, *DC programming Approach for Portfolio Optimization under Step Increasing Transaction Costs*, to appear in *Optimization*.

- H.A Le Thi, M. Moeini, *Optimization of a Long-Short Portfolio under Threshold Constraints using DC Programming and DCA*, submitted to "Operations Research".

- N. Gulpinar, H.A Le Thi, and M. Moeini, *Robust Investment Strategies with Discrete Asset Choice Constraints Using DC Programming and DCA*, submitted to "Journal of Global Optimization".

- H.A Le Thi, and M. Moeini, *Portfolio Selection Under Buy-In Threshold Constraints Using DC Programming and DCA*, Proceeding de third International Conference on Service Systems and Service Management (SSSM06/IEEE), Troyes, October 2006, pp. 296-300.

Communications aux colloques internationaux avec actes publiés

- H.A. Le Thi, M. Moeini, and T. Pham Dinh, *A DC programming approach for Downside risk portfolio selection problem under cardinality constraints*, Third International Conference on Computational Management Science, May 2006, Amsterdam.

- H.A. Le Thi, M. Moeini, and T. Pham Dinh, *Optimization of a Long-Short Portfolio Under Threshold Constraints*, 4th International Conference on Computational Management Science, 20-22 Avril 2007, Geneva.

- H.A. Le Thi, M. Moeini, and T. Pham Dinh, *Portfolio Optimization Under Step Increasing Transaction Costs Using DC Programming and DCA*, 22nd European Conference

on Operational Research, 8-11 July 2007, Prague.

- N. Gulpinar, H.A Le Thi, and M. Moeini, *Worst-case Robust Investment Strategies with Discrete Asset Choice Constraints Using DCA*, The International Conference on Non Convex Programming : Local and Global Approaches (NCP07), 17-21 December 2007, National Institute for Applied Sciences - Rouen, France.

- N. Gulpinar, H.A Le Thi, and M. Moeini, *A DC Programming Approach for Portfolio Selection Models with Discrete Asset Choice Constraints*,The International Conference on Non Convex Programming : Local and Global Approaches (NCP07), 17-21 December 2007, National Institute for Applied Sciences - Rouen, France.

Communications aux colloques nationaux avec actes publiés

- H.A Le Thi and M. Moeini, *Une approche de programmation D.C. pour le problème de la gestion du portefeuille de risque de chute du cours sous les contraintes de cardinalité*, 7ème congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision, Lille, Février 2006.

- H.A Le Thi, M. Moeini, and T. Pham Dinh, *Gestion de portefeuille sous les contraintes de seuil*, Conférence Scientifique conjointe en Rechercher Opérationnelle et Aide à la décision FRANCORO/ROADEF 07, 20-23 Février 2007, Grenoble, France.

Introduction générale

Contexte général et problématique

L'optimisation est à la fois une science et un outil largement utilisée dans divers domaines scientifiques, en ingénierie comme en industrie, qui nous aide à prendre la meilleure décision pour un grand nombre de décisions possibles. L'optimisation s'efforce à la fois de construire des méthodes de calcul pour trouver des solutions optimales, d'explorer les propriétés théoriques et d'étudier l'efficacité numérique des algorithmes.

Il n'est exagéré de dire que chacun utilise l'optimisation dans sa vie d'une façon ou d'une autre. Parmi les éminentes applications d'optimisation nous pouvons citer la gestion de portefeuille, la planification de production, les réseaux informatiques, différentes filières d'ingénierie, etc. Pour résoudre ces problèmes, l'optimisation offre un cadre algorithmique très riche. Dans cette étude, il nous faut d'abord distinguer deux filières d'optimisation :

- les modèles d'optimisation stochastique,
- les modèles d'optimisation déterministe.

Cette thèse se cadre dans le contexte d'optimisation déterministe qui à son tour, se divise à deux branches : la programmation convexe et la programmation non convexe. Un programme convexe ou un problème d'optimisation convexe est celui de la minimisation d'une fonction (objectif) convexe sous un ensemble convexe des contraintes. Lorsque la double convexité chez l'objectif et les contraintes n'est pas vérifiée, nous sommes en face d'un problème d'optimisation non convexe. La double convexité d'un programme convexe permet d'établir des caractérisations (sous forme de conditions nécessaires et suffisantes) de solutions optimales et ainsi de construire des méthodes itératives convergeant vers des solutions optimales. Théoriquement nous pouvons résoudre tout programme convexe. L'absence de cette double convexité rend la résolution d'un programme non convexe difficile voire impossible dans l'état actuel des choses. Contrairement à la programmation convexe, les solutions optimales locales et globales sont à distinguer dans un programme non convexe. D'autre part si nous disposons des caractérisations d'optimalité locale utilisables, au moins pour la classe des programmes non convexes assez réguliers, qui permettent la construction des méthodes convergeant vers des solutions locales (algorithmes locaux) il n'y a par contre pas de caractérisations d'optimalité globale sur lesquelles sont basées les méthodes itératives convergeant vers des solutions globales (algorithmes globaux). L'analyse et l'optimisation convexes modernes se voient ainsi contrainte à une extension logique et naturelle à la non convexité et à la non différentiabilité.

Les méthodes numériques conventionnelles de l'optimisation convexe ne fournissent que des minima locaux bien souvent éloignés de l'optimum global.

L'optimisation non convexe connaît une explosion spectaculaire depuis d'une quinzaine d'années car dans les milieux industriels, on a commencé à remplacer les modèles convexes par des modèles non convexes plus complexes mais plus fiables qui présentent mieux la nature des problèmes étudiés. Durant ces dernières années, la recherche en optimisation non convexe a largement bénéficié des efforts des chercheurs et s'est enrichie de nouvelles approches. Nous pouvons distinguer deux approches différentes mais complémentaires en programmation non convexe :

i) Approches globales combinatoires : elles sont basées sur les techniques combinatoires de la Recherche Opérationnelle. Elles consistent à localiser les solutions optimales à l'aide des méthodes d'approximation, des techniques de coupe, des méthodes de décomposition, des algorithmes par séparation et évaluation. Elles ont connu de très nombreux développements importants au cours de ces dernières années à travers les travaux de H. Tuy (reconnu comme le pionnier), R. Horst, P. Pardalos et N. V. Thoai ([47, 48, 49, 50]). L'inconvénient majeur des méthodes globales est leur lourdeur (encombrement en places-mémoires) et leur coût trop important. Elles ne sont pas applicables aux problèmes d'optimisation non convexes réels qui sont souvent de très grande dimension.

ii) Approches locales et globales d'analyse convexe qui sont basées sur l'analyse et l'optimisation convexe. Ici la programmation DC (Différence de deux fonctions Convexes) et DCA (DC Algorithmes) jouent le rôle central car la plupart des problèmes d'optimisation non convexe sont formulés/reformulés sous la forme DC. Sur le plan algorithmique, l'essentiel repose sur les algorithmes de l'optimisation DC (DCA) introduits par T. Pham Dinh en 1985 à l'état préliminaire et développés intensivement à travers de nombreux travaux communs de H.A Le Thi et T. Pham Dinh depuis 1993 pour devenir maintenant classiques et de plus en plus utilisés par des chercheurs et praticiens de par le monde, dans différents domaines des sciences appliquées.

Les travaux de cette thèse se situent dans le cadre de la programmation non convexe. Ils s'appuient principalement sur la programmation DC et DCA. Cette démarche est motivée par la robustesse et la performance de la programmation DC et DCA comparées à des méthodes existantes, leur adaptation aux structures des problèmes traités et leur capacité de résoudre des problèmes industriels de très grande dimension. A notre connaissance, DCA fait actuellement partie des rares algorithmes de la programmation non convexe étant capables de traiter des problèmes (différentiables ou non) de très grande dimension.

Un programme DC est de la forme

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

où $g, h \in \Gamma_0(\mathbb{R}^n)$, le cône convexe de toutes les fonctions convexes semi-continues inférieurement et propres sur $\mathbb{R}^n$. Une telle fonction f est appelée fonction DC et g et h

des composantes DC de f. La programmation DC est une extension de la Programmation Convexe : cette extension est assez large pour couvrir la quasi-totalité des programmes non convexes. DCA est une approche locale qui travaille avec les deux fonctions convexes g et h (dont la différence est la fonction objectif elle-même du programme DC) et non avec la fonction objectif, i.e., f. Puisqu'une fonction DC admet une infinité de décompositions DC, il y a une infinité de DCA appliqués à un programme DC. Et les impacts de ces décompositions DC sur les qualités des DCA correspondants (rapidité, robustesse, globalité, ...) sont importants. La résolution d'un problème concret par DCA devrait répondre aux deux questions cruciales :

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

Cadre de la thèse, objets et objectifs, motivations

Cette thèse est consacrée à la modélisation et l'optimisation non convexe basées sur *la programmation DC et DCA* pour certains problèmes en *finance*.

Il y a plus d'un demi-siècle, la gestion de portefeuille était à un carrefour avec la publication de l'article de Harry Markowitz [112]. Il a été le pionnier du premier traitement rigoureux du dilemme de l'investisseur, à savoir comment atteindre de plus grands profits tout en minimisant le risque. Pour son approche Moyenne-Variance (MV) dans la sélection de portefeuille, H. Markowitz a reçu le prix Nobel d'économie en 1990 (partagé avec M.H. Miller et W. Sharpe). Depuis ce temps, l'analyse de moyenne-variance demeure un sujet qui génère beaucoup d'intérêt parmi les chercheurs et les praticiens.

Avec les progrès technologiques des dernières années, le développement des algorithmes exploitant les nombreuses découvertes en mathématiques financières est en pleine expansion. Il nous suffit de consulter les articles sur internet pour en mesurer la portée.

Le modèle MV de Markowitz utilise la variance pour mesurer le risque. Après l'introduction du modèle MV par Markowitz, des nombreux modèles ont été mis en œuvre qui utilisent d'autres mesures de risque comme semivariance ou le modèle Mean-Absolute Deviation (MAD), etc ([62], [113]). Chaque modèle a ses avantages et ses inconvénients. Grâce aux progrès technologiques, ces modèles se résolvent facilement, même pour les problèmes de grande taille. Si nous voulons rendre les modèles plus réalistes, nous aurons besoin d'introduire des termes non convexes ou des variables discrètes ([33], [54]). Les coûts de transaction et les contraintes de cardinalité sont deux exemples bien connus. Pour résoudre les problèmes qui contiennent des termes non convexes, nous avons besoin d'utiliser les approches d'optimisation globale.

Malheureusement, il existe relativement peu d'application de ces méthodes en finance. La raison principale est du fait que depuis longtemps, jusqu'au milieu des années 80, la programmation non convexe était négligée sous prétexte que la résolution d'un programme non convexe de façon déterministe est limitée aux applications très particulières. Les méthodes heuristiques étaient reconnues comme les seuls moyens permettant de traiter des problèmes non convexes sans forme particulière [54]. Donc, les chercheurs en finance ne sont pas au courant des progrès récents en optimisation non convexe. Par conséquent, soit ils formulent les problèmes sous la forme de programmation convexe, soit ils appliquent des méthodes heuristiques. Dans le but de corriger cet esprit, le document présent se propose de développer des outils à la prise de décision financière. Surtout, le but de cette thèse est de présenter des applications réussies des méthodes de programmation DC et DCA en gestion de portefeuille.

Notre travail en Programmation DC et DCA pour la modélisation, la conception et la réalisation des DCA bien adaptés aux structures spécifiques des problèmes choisis est composé de :

- Etude approfondie des modèles d'optimisation non convexe et la modélisation DC des problèmes ; Formulations et reformulations des programmes DC équivalents, choix des décompositions DC les mieux adaptées, choix de meilleur point initial pour démarrer DCA.
- Mis en œvre des schéma de DCA correspondant.
- Combinaison de DCA et d'autres approches pour chercher les bons points initiaux pour DCA et/ou pour prouver la globalité des solutions obtenues par DCA.
- Implémentations et simulations numériques comparatives.

Les modèles traités au cours de la thèse sont :

- *Gestion de portefeuille sous les contraintes de seuil d'achat, de seuil et de cardinalité* : ce sont trois modèles tels que chacun d'entre eux est une généralisation du modèle MV de Markowitz. Les contraintes de cardinalité limitent le nombre d'actifs composant le portefeuille et les contraintes de seuil et seuil d'achat empêchent des très petits investissements dans chaque actif. Les modèles généralisés sont non convexes et par conséquent difficiles à résoudre par des méthodes classiques.
- *Gestion de portefeuille avec la mesure de risque de baisse sous les contraintes de cardinalité* : la mesure de risque choisie est de type *de baisse*. La présence des contraintes de cardinalité exigent une formulation de programmation mixte en variables binaires. L'utilisation de la mesure de risque de baisse est liée aux défauts du modèle MV.
- *Gestion de portefeuille avec les fonctions des coûts de transaction en escalier* : Il s'agit de la présence des fonctions de coûts de transaction concave et plus précisément les fonctions constantes par morceaux. Ces fonctions des coûts de transaction sont utilisées pour les transaction sur Internet. Le travail consiste à résoudre le problème pour un nombre important d'actifs et un nombre suffisamment grand de morceaux de fonctions de coût.
- *Investissement robuste en gestion de portefeuille en présence des contraintes de cardinalité* : un défi important en gestion de portefeuille est la robustesse des stratégies d'investissement proposés par des modèles. Le modèle min-max étudié dans ce travail est une

généralisation du modèle MV de Markowitz tel que plusieurs scénarios d'investissement sont examinés au lieu d'un seul. C'est la raison pour laquelle la stratégie proposée par le modèle est robuste dans le sens où si un autre scénario se réalise la stratégie d'investissement sera la meilleure. De plus le modèle contient des contraintes de cardinalité et des contraintes de borne telles que l'on puisse avoir du contrôle sur l'investissement dans chaque actif.

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

1. *La programmation linéaire/quadratique en variables mixtes binaires sous des contraintes linéaires/quadratiques* (la gestion de portefeuille sous les contraintes de seuil d'achat et de cardinalité, l'investissement robuste en gestion de portefeuille sous les contraintes de cardinalité). Bien qu'il s'agisse des problèmes d'optimisation combinatoire, nous les reformulons, grâce à la pénalité exacte, comme un problème d'optimisation continue qui est en fait un programme DC. En plus de DCA, nous développons une méthode combinée de DCA et SE pour résoudre le problème.
2. *La programmation quadratique en variables mixtes binaires sous les contraintes de complémentarité* (la gestion de portefeuille sous les contraintes de seuil). La présence des contraintes de complémentarité rend le problème de plus en plus difficile. Nous proposons une nouvelle fonction de pénalité qui prend en compte non seulement les variables binaires mais aussi les contraintes de complémentarité. Ensuite, nous reformulons le modèle, grâce à la pénalité exacte, comme un problème de programmation DC. Nous développons une approche combinée de DCA et SE pour résoudre le problème.
3. *La minimisation d'une fonction non convexe et non lisse sur un ensemble convexe* (la gestion de portefeuille avec les fonctions des coûts de transaction en escalier). Le problème est non convexe et non lisse à cause des fonctions constantes par morceaux. La méthode traditionnelle pour résoudre les problèmes prenant en compte les fonctions en escalier consiste à les reformuler en introduisant des variables binaires, ce qui rend le problème lourd et de plus en plus difficile à résoudre même avec les logiciels les plus puissants. Notre approche est de proposer des fonctions DC polyédrales afin d'estimer les fonctions de coûts, ensuite nous utilisons DCA pour résoudre le programme DC polyédral. Une procédure combinée de DCA et d'un algorithme par séparation et évaluation est proposée.

Organisation de la thèse

La thèse est divisée en deux parties et est composée de sept chapitres. Dans la première partie intitulée "Méthodologie" nous présentons des outils théoriques et algorithmiques servant des références aux autres. Le premier chapitre concerne la programmation DC et DCA tandis que le deuxième porte sur les algorithmes par séparation et évaluation. Dans la deuxième partie nous développons la programmation DC et DCA pour la résolution des problèmes en finance. Nous commençons par une introduction à la gestion de portefeuille (le chapitre trois). Le

Chapitre 4 est dédié aux généralisations du modèle MV de Markowitz, où nous étudions le modèle MV sous les contraintes de seuil d'achat, de seuil et de cardinalité. Le Chapitre 5 est consacré à la mesure de risque de baisse et les contraintes de cardinalité. Le Chapitre 6 porte sur le problème de choix de portefeuille avec les fonctions des coûts de transaction en escalier. L'investissement robuste en gestion de portefeuille sous les contraintes de cardinalité est développé dans le dernier chapitre.

Première partie

Méthodologie

Chapitre 1

Introduction à la programmation DC et DCA

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

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

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

Le pionnier de cette approche est H. Tuy dont le premier travail remonte à 1964. Ses travaux sont abondants, citons les livres de Horst-Tuy ([174, 175]) qui présentent la théorie, algorithmes et applications de l'optimisation globale. Viennent ensuite les principales contributions de l'Ecole Américaine (P. M. Pardalos, J. B. Rosen,...), Allemande (R. Horst, ...), Française (Le Thi Hoai An, Pham Dinh Tao,...) et l'Ecole Vietnamienne (Phan Thien Thach, Le Dung Muu, ...).

La seconde approche repose sur l'arsenal puissant d'analyse et l'optimisation convexes. Son premier travail dû à Pham Dinh Tao (1975) concerne le calcul des normes matricielles (problème fondamental en analyse numérique) qui est un problème de maximisation d'une fonction convexe sur un convexe. Le travail de Toland (1978) ([168]) sur la dualité et l'optimalité locale en optimisation DC généralise de manière élégante les résultats établis par Pham en maximisation convexe. La théorie de l'optimisation DC est ensuite développée notamment par Pham Dinh Tao, J. B. Hiriart Urruty, Jean - Paul Penot, Phan Thien Thach, Le Thi Hoai An. Sur le plan algorithmique dans le cadre de la seconde approche, on dispose actuellement des DCA (DC Algorithms) introduits par Pham Dinh Tao (1986), qui sont basés sur les conditions d'optimalité et de dualité en optimisation DC. Mais il a fallu attendre les travaux communs de Le Thi Hoai An et Pham Dinh Tao (voir [70]-[104] et [135]-[140]) pour qu'il s'impose définitivement en optimisation non convexe comme étant des algorithmes les plus simples et performants, capables de traiter des problèmes de grande taille.

Nous reportons dans ce chapitre les principaux résultats relatifs à la programmation DC et DCA qui nous seront les plus utiles pour nos travaux. Ces résultats sont extraits de ceux présentés dans H. A. Le Thi 1994 ([70]), H. A. Le Thi 1997 ([71]). Pour une étude détaillée nous nous référons à ces deux références (voir également [70]-[104] et [135]-[140]).

1.1 Eléments de base de l'analyse DC

1.1.1 Notations et propriétés

Ce paragraphe est consacré à un rapide rappel d'analyse convexe pour faciliter la lecture de certains passages. Pour plus de détails, on pourra se référer aux ouvrages de P.J Laurent ([67]), de R.T Rockafellar ([146]) et d'A. Auslender ([6]). Dans toute la suite X désigne l'espace euclidien $\mathbb{R}^n$, muni du produit scalaire usuel noté $\langle .,. \rangle$ et de la norme euclidienne associée $\|x\| = \langle x, x\rangle^{\frac{1}{2}}$ et Y l'espace vectoriel dual de X relatif au produit scalaire, que l'on peut identifier à X. On note par $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$ muni d'une structure algébrique déduite de celle de $\mathbb{R}$ avec la convention que $\infty - (+\infty) = +\infty$ ([146]). Étant donnée une fonction $f : S \longrightarrow \overline{\mathbb{R}}$ définie sur un ensemble S convexe de X, on appelle domaine effectif de f l'ensemble

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

et épigraphe de f

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

1.1.2 Fonctions convexes polyédrales

Une partie convexe C est dite convexe polyédrale si

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

Une fonction est dite convexe polyédrale si

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

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

Proposition 1.2 *([146])*
- *Soit f une fonction convexe polyédrale. f est partout finie si et seulement si $C = X$,*
- *Si f est polyédrale alors f^* l'est aussi. De plus si f est partout finie alors*

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

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

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

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

1.1.3 Fonction DC

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

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

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

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

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

Ces résultats se généralisent aux cas des fonctions à valeur dans $\mathbb{R} \cup \{+\infty\}$ ([71]). Il en résulte que l'ensemble des fonctions DC sur Ω est un espace vectoriel ($DC(\Omega)$) : c'est le plus petit espace vectoriel contenant l'ensemble des fonctions convexes sur $\Omega(Co(\Omega))$.

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

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

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

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

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

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

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

localement DC).

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

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

1.2 Optimisation DC

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

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

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

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

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

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

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

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

Problème (2) est communément appelé *la programmation DC*. Elle est d'un intérêt majeur aussi bien d'un point de vue pratique que théorique. Du point de vue théorique, on peut souligner que, comme on a vu en haut, la classe des fonctions DC est remarquablement stable par rapport aux opérations fréquemment utilisées en optimisation. En outre, on dispose d'une élégante théorie de la dualité ([70, 71, 83, 129, 130, 168, 178]) qui, comme en

optimisation convexe, a de profondes répercussions pratiques sur les méthodes numériques.

Sur le plan algorithmique, les algorithmes de l'optimisation DC (DCA) dus à Pham Dinh Tao ([134, 135]) constituent une nouvelle approche originale basée sur la théorie DC. Ces algorithmes représentent en fait une généralisation des algorithmes de sous-gradients étudiés par le même auteur sur la maximisation convexe ([129, 134]). Cependant, il a fallu attendre les travaux communs de Le Thi et Pham au cours de ces quinze dernières années (voir [35], [70]-[104] et [135]-[140]) pour que les DCA deviennes maintenant classiques et populaires.

1.2.1 Dualité DC

En analyse convexe, le concept de la dualité (fonctions conjuguées, problème dual, etc.) est une notion fondamentale très puissante. Pour les problèmes convexes et en particulier linéaires, une théorie de la dualité a été développée depuis déjà plusieurs décennies ([146]). Plus récemment, en analyse non convexe d'importants concepts de dualité ont été proposés et développés, tout d'abord, pour les problèmes de maximisation convexe, avant de parvenir aux problèmes DC. Ainsi la dualité DC introduite par Toland (1978) peut être considérée comme une généralisation logique des travaux de Pham Dinh Tao (1975) sur la maximisation convexe. On va présenter ci-dessous les principaux résultats (en optimisation DC) concernant les conditions d'optimalité (locale et globale) et la dualité DC. Pour plus de détails, le lecteur est renvoyé au document de Le Thi (1997) (voir également [83]).

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

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

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

et le problème dual

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

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

Ce résultat de dualité DC défini à l'aide des fonctions conjuguées donne une importante relation en optimisation DC ([168]).

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

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

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

Preuve :
(i)

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

(ii) cf. Toland ([168]).

□

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

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

1.2.2 Optimalité globale en optimisation DC

En optimisation convexe, x^0 minimise une fonction $f \in \Gamma_0(X)$ si et seulement si : $0 \in \partial f(x^0)$. En optimisation DC, la condition d'optimalité globale suivante ([179]) est formulée à l'aide des ϵ-sous-différentiels de g et h. Sa démonstration (basée sur l'étude du comportement du ϵ-sous-différentiel d'une fonction convexe en fonction du paramètre ϵ) est compliquée. La démonstration dans [71] est plus simple et convient bien au cadre de l'optimisation DC : elle exprime tout simplement que cette condition d'optimalité globale est une traduction géométrique de l'égalité des valeurs optimales dans les programmes DC primal et dual.

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

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

Remarque 1.3 –
(i) Si $f \in \Gamma_0(X)$, on peut écrire $f = g - h$ avec $f = g$ et $h = 0$. Dans ce cas l'optimalité globale dans (P) - qui est identique à l'optimalité locale car (P) est un problème convexe - est caractérisée par,

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

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

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

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

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

1.2.3 Optimalité locale en optimisation DC

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

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

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

où $V_x^\bullet$ désigne un voisinage de $x^\bullet$.

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

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

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

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

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

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

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

Remarquons que pour un certain nombre de problème DC et en particulier pour h polyédrale, la condition nécessaire (1.9) est également suffisante, comme nous le verrons un peu plus loin. On dit que $x^\bullet$ est un point critique de $g-h$ si $\partial h(x^\bullet) \cup \partial g(x^\bullet)$ est non vide ([168]). C'est une forme affaiblie de l'inclusion sousdifférentielle. La recherche d'un tel point critique est à la base de DCA (forme simple) qui sera étudiée dans la section suivante. En général DCA converge vers une solution locale d'un problème d'optimisation DC. Cependant sur le plan théorique, il est important de formuler des conditions suffisantes pour l'optimalité locale.

Théorème 1.3 ***(Condition suffisante d'optimalité locale ([71, 83]))*** *Si x^* admet un voisinage V tel que*

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

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

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

$$\partial h(x^*) \subset int(\partial g(x^*)),$$

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

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

Preuve : Ce résultat généralise le premier obtenu par C. Michelot dans le cas où $g, h \in \Gamma_0(X)$ sont finies partout et h convexe polyédrale (cf. ([71, 83])). □

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

Corollaire 1.4 ***(Transport par dualité DC des minima locaux ([71, 83]))*** *Supposons que $x^\bullet \in dom(\partial h)$ soit un minimum local de $g-h$, soient $y^\bullet \in \partial h(x^\bullet)$ et $V_{x^\bullet}$ un voisinage de $x^\bullet$ tel que $g(x)-h(x) \geq g(x^\bullet)-h(x^\bullet), \quad \forall x \in V_{x^\bullet} \cap dom(g)$. Si*

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

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

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

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

1.3 DCA

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

1.3.1 Principe de DCA

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

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

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

Les DCA reposent sur la construction de deux suites $\{x^k\}$ et $\{y^k\}$ qui sont améliorées a chaque itération de sorte que leur limite respective x^* et y^* soient candidates pour être les optima locaux du problème primal et du problème dual respectivement. Ces deux suites sont liées par dualité et vérifient les propriétés suivantes :

- les suites $\{g(x^k) - h(x^k)\}$ et $\{h^*(y^k) - g^*(y^k)\}$ sont décroissantes,
- et si $(g - h)(x^{k+1}) = (g - h)(x^k)$ alors l'algorithme s'arrête à la $(k + 1)^{ieme}$ itération et le point x^k (resp. y^k) est un point critique de $g - h$ (resp. $h^* - g^*$),
- sinon toute valeur d'adhérence $x^\bullet$ de $\{x^k\}$ (resp. $y^\bullet$ de $\{y^k\}$) est un point critique de $g - h$ (resp. $h^* - g^*$).

L'algorithme cherche en définitif un couple $(x^\bullet, y^\bullet) \in X \times Y$ tel que $x^\bullet \in \partial g^*(y^\bullet)$ et $y^\bullet \in \partial h(x^\bullet)$.

Schéma de DCA simplifié

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

On construit ainsi :

Algorithme 1. [DCA]

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

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

1.3.2 Existence des suites générées

L'algorithme DCA est bien défini si on peut effectivement construire les deux suites $\{x^k\}$ et $\{y^k\}$ comme ci-dessus à partir d'un point initial arbitraire x^0.
– Par construction, si $x^0 \in dom(\partial h)$, alors $y^0 \in \partial h(x^0)$ est bien défini.
– Pour $k \geq 1$, y^k est bien défini si et seulement si x^k est défini et contenu dans $dom(\partial h)$, par suite, x^k et y^k sont bien définis si et seulement si $\partial g^*(y^{k+1}) \cap dom(\partial h)$ est non vide, ce qui entraîne que $y^{k+1} \in dom(\partial g^*)$.

Lemme 1.1 *([83]) Les suites $\{x^k\}$, $\{y^k\}$ dans DCA sont bien définies si et seulement si*

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

La convergence de l'algorithme est assurée par les résultats suivants ([83]) :

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

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

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

(ii)

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

où $dx^k = x^{k+1} - x^k$.

Corollaire 1.5 *([83])**(Convergence)***
1.
$$\begin{aligned}(g-h)(x^{k+1}) &\leq (h^*-g^*)(y^k) - \tfrac{\rho_2}{2}\|dx^k\|^2\\ &\leq (g-h)(x^k) - [\tfrac{\rho_2}{2}\|dx^{k-1}\|^2 + \tfrac{\rho_1^*}{2}\|dy^k\|^2]\end{aligned}$$
2.
$$\begin{aligned}(g-h)(x^{k+1}) &\leq (h^*-g^*)(y^k) - \tfrac{\rho_2^*}{2}\|dx^k\|^2\\ &\leq (g-h)(x^k) - [\tfrac{\rho_2^*}{2}\|dx^{k-1}\|^2 + \tfrac{\rho_1^*}{2}\|dy^k\|^2]\end{aligned}$$
3.
$$\begin{aligned}(h^*-g^*)(y^{k+1}) &\leq (g-h)(x^{k+1}) - \tfrac{\rho_1^*}{2}\|dy^k\|^2\\ &\leq (h^*-g^*)(y^k) - [\tfrac{\rho_1^*}{2}\|dy^k\|^2 + \tfrac{\rho_2^*}{2}\|dx^k\|^2]\end{aligned}$$
4.
$$\begin{aligned}(h^*-g^*)(y^{k+1}) &\leq (g-h)(x^{k+1}) - \tfrac{\rho_1}{2}\|dy^{k+1}\|^2\\ &\leq (h^*-g^*)(y^k) - [\tfrac{\rho_1}{2}\|dx^{k+1}\|^2 + \tfrac{\rho_2}{2}\|dx^k\|^2]\end{aligned}$$

Corollaire 1.6 *([83]) Si les égalités ont lieu, il vient :*
1. $(g-h)(x^{k+1}) = (h^*-g^*)(y^k) \Longleftrightarrow y^k \in \partial h(x^{k+1})$
2. $(g-h)(x^{k+1}) = (g-h)(x^k) \Longleftrightarrow x^k \in \partial g^*(y^k), \quad y^k \in \partial h(x^{k+1})$
3. $(h^*-g^*)(y^k) = (g-h)(x^k) \Longleftrightarrow x^k \in \partial g^*(y^k)$
4. $(h^*-g^*)(y^{k+1}) = (h^*-g^*)(y^k) \Longleftrightarrow y^k \in \partial h(x^{k+1}), \quad x^{k+1} \in \partial g^*(y^{k+1})$.

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

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

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

1.3.3 Calcul des sous-gradients

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

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

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

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

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

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

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

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

Et similairement, par dualité

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

1.3.4 Optimisation DC polyédrale

L'optimisation DC polyédrale survient lorsque l'une des composantes convexes g ou h est convexe polyédrale. A l'instar des problèmes d'optimisation convexe polyédrale, cette classe de problème d'optimisation DC se rencontre fréquemment en pratique et possèdent d'intéressantes propriétés. Nous allons voir que la description de DCA y est particulièrement simple ([70, 71, 83]).

Soit le programme DC

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

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

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

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

Théorème 1.5 ***(Convergence finie)***

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

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

1.3.5 Interprétations de DCA

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

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

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

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

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

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

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

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

qui coïncide en x^k avec f,

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

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

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

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

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

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

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

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

Pour une étude complète de la programmation DC et DCA, se reporter aux [70]-[104] et [135]-[140] et références incluses. Le traitement d'un programme non convexe par une approche DC et DCA devrait comporter donc deux tâches : la recherche d'une décomposition DC adéquate et celle d'un bon point initial. Pour un programme DC donné, la question de décomposition DC optimale reste ouverte, en pratique on cherche des décompositions DC bien adaptées à la structure spécifiques du programme DC étudié pour lesquelles les suites $\{x^k\}$ et $\{y^k\}$ sont faciles à calculer, si possible explicites pour que les DCA correspondants soient moins coûteux en temps et par conséquent capables de supporter de très grandes dimensions.

Chapitre 2

L'algorithme par Séparation et Evaluation (SE)

Résumé *Ce chapitre concerne l'algorithme de Séparation et Evaluation (SE). La séparation et évaluation est un algorithme exhaustif qui converge vers la solution globale sous certaines conditions. Ce chapitre est consacré aux éléments de base de cet algorithme ainsi que les conditions de convergence. De plus, deux cas particuliers de cet algorithme sont présentés.*

2.1 Introduction

Un algorithme par séparation et évaluation (SE) est une méthode générique de résolution de problèmes d'optimisation, et plus particulièrement d'optimisation combinatoire ou discrète non convexes. Dans les méthodes par séparation et évaluation, la *séparation* permet d'obtenir une méthode générique pour localiser toutes les solutions optimales tandis que l'*évaluation* évite l'énumération systématique de toutes les solutions.

Séparation

La phase de séparation consiste à diviser le problème en un certain nombre de sous-problèmes qui ont chacun leur ensemble de solutions réalisables de telle sorte que tous ces ensembles forment une partition de l'ensemble de toutes les solutions possibles. Ainsi, en résolvant tous les sous-problèmes et en prenant la meilleure solution trouvée, on est assuré d'avoir résolu le problème initial. Ce principe de séparation peut être appliqué de manière récursive à chacun des sous-ensembles de solutions obtenus, et ceci tant qu'il y a des ensembles contenant plusieurs solutions. Les ensembles de solutions (et leurs sous-problèmes associés) construits ont une hiérarchie naturelle en arbre, souvent appelée *arbre de recherche* ou *arbre de décision*.

Evaluation

L'évaluation d'un nœud de l'arbre de recherche a pour but de déterminer l'optimum de l'ensemble des solutions réalisables associé au nœud en question ou, au contraire, de prouver mathématiquement que cet ensemble ne contient pas de solution intéressante pour la résolution du problème (typiquement, qu'il n'y a pas de solution optimale). Lorsqu'un tel nœud est identifié dans l'arbre de recherche, il est donc inutile d'effectuer la séparation de son espace de solutions. Pour déterminer qu'un ensemble de solutions réalisables ne contient pas de solution optimale, la méthode la plus générale consiste à déterminer une borne inférieure pour tous les problèmes contenus dans l'ensemble (s'il s'agit d'un problème de minimisation). Si on arrive à trouver une borne inférieure associée à un sous-problème supérieur à la meilleure solution trouvée jusqu'à présent, on a alors l'assurance que le sous-ensemble ne contient pas l'optimum. Les techniques les plus classiques pour le calcul de bornes sont basées sur l'idée de relaxation de certaines contraintes : relaxation continue, relaxation lagrangienne, etc.

L'algorithme par Séparation et Evaluation est peut-être la technique la plus populaire en optimisation globale. SE est plus connu sous son nom anglais *Branch and Bound (B&B) algorithm*. L'avantage principale de cette approche est dû à sa capacité de résoudre une grande variété des problèmes non convexes. Théoriquement, pour chaque problème d'optimisation, quelque soit le problème, on peut construire un algorithme de type SE pour le résoudre [39]. D'après Murty [120], l'approche SE a été développée indépendamment par Land et Doig en 1960 [66] et aussi par Murty, Karel et Little en 1962 [119]. Mais d'après Gupta [39] les premières applications de SE datent à 1958 ([26], [39]) pourtant le mot Branch and Bound a été premièrement utilisé par Little et al. en 1963 [109]. En tout cas, Land et Doig [66] ont utilisé leur méthode pour résoudre un problème mixte pour lequel certaines variables sont entières. Murty et al. [119] avaient développé un algorithme pour un cas particulier d'optimisation en variables discrètes.

Dans ce chapitre, nous allons d'abord présenter le schéma général de SE et le théorème de convergence. Ensuite, nous allons présenter deux cas particuliers de l'approche SE.

2.2 Le cas général de SE

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

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

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

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

L'idée de base de la méthode SE consiste en une 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 fonction objectif sur cet ensemble afin d'éliminer les parties non prometteuses et de sélectionner un ensemble que l'on devrait diviser par la suite.

Définition 2.2.1 *Soit M un compact dans $\mathbb{R}^n$ et soit I un ensemble fini des indices. Un ensemble $M_i : i \in I$ de sous-ensemble compacts est dit 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. □

2.2.1 Méthode de résolution et convergence

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

Prototype SE

Initialisation :

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

   $$S_{0,i} \subset M_{0,i} \cap S, S_{0,i} \neq \emptyset$$

   et

   $$\gamma_{0,i} = \gamma(M_{0,i}) := min\, f(S_{0,i}), \quad x^{0,i} \in argmin\, f(S_{0,i}).$$

3. Pour chaque $i \in I_0$ déterminer

   $$\beta_{0,i} = \beta(M_{0,i}) \leq min\, f(S \cap M_{0,i}).$$

4. Calculer

   $$\gamma_0 = min_{i \in I_0}\, \gamma_{0,i}, \tag{2.2}$$

   $$x^0 \in argmin\, \{f(x^{0,i}, i \in I_0)\}, \tag{2.3}$$

   $$\beta_0 = min_{i \in I_0} \beta_{0,i}. \tag{2.4}$$

Itération k :

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

$$\beta_{k,i} \geq \gamma_0$$

ou pour lequel on sait que $min\, f(S)$ ne peut pas avoir lieu dans $M_{k,i}$. Soit $\mathcal{R}_k$ le collection des éléments restant $M_{k,i} \in \mathcal{M}_k$.
Si $\mathcal{R}_k = \emptyset$ alors s'arrêter, x^k est une solution.

k.2 Sélectionner $M_{k,i_k} \in \mathcal{R}_k$, choisir un ensemble fini des indices J_{k+1} et construire une partition

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

de M_{k,i_k} telle que $M_{k+1,i} \cap S \neq \emptyset$.

k.3 Pour chaque $i \in J_{k+1}$ déterminer

$$S_{k+1,i} \subset M_{k+1,i} \cap S, S_{k+1,i} \neq \emptyset$$

et

$$\gamma_{k+1,i} = \gamma(M_{k+1,i}) := min\, f(S_{k+1,i}),\ x^{k+1,i} \in argmin\, f(S_{k+1,i}).$$

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

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

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

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

est la partition actuelle.

k.6 Calculer

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

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

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

et retourner à l'itération $k+1$.

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

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

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

Condition de la convergence

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

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

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

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

Définition 2.2.2 *Une estimation de borne est dite cohérente si, pour une suite décroissante quelconque* $M_{k_q,i_{k_q}}$ *générée par la procédure de séparation, i.e.*

$$M_{k_{q+1},i_{k_{q+1}}} \subset M_{k_q,i_{k_q}}, \tag{2.11}$$

on a

$$\lim_{q \longrightarrow \infty} (\gamma_{k_q,i_{k_q}} - \beta_{k_q,i_{k_q}}) = 0. \tag{2.12}$$

□

Puisque $\beta_{k_q,i_{k_q}} \leq \gamma_{k_q} \leq \gamma_{k_q,i_{k_q}}$, la condition (2.12) peut s'écrire

$$\lim_{q \longrightarrow \infty} (\alpha_{k_q} - \beta_{k_q,i_{k_q}}) = 0. \tag{2.13}$$

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

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

Définition 2.2.3 *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", si au moins après chaque nombre fini d'itérations, on a

$$M_{k,i_k} \in argmin\{\beta(M) : M \in \mathcal{R}_k\}. \tag{2.14}$$

□

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

$$\gamma := \lim_{k \longrightarrow \infty} \alpha_k = \lim_{k \longrightarrow \infty} f(x^k) = min\ f(S). \tag{2.15}$$

– *(ii) Si la sélection est borne améliorante, alors*

$$\beta := \lim_{k \longrightarrow \infty} \beta_k = min\ f(S). \tag{2.16}$$

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

Preuve :
– (i) Puisque $\gamma \geq inf\ f(S)$, il suffit de montrer que $f(x) \geq \beta, \quad \forall x \in S$.
Si x appartient à un ensemble qui est éliminé dans l'itération k où à un ensemble

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

(i.e. M reste inchangé) alors, on a

$$f(x) \geq \beta_{k,i} > \gamma_k \geq \gamma$$

ou

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

car la sélection est complète. Donc $f(x) \geq \gamma$. Sinon, il existe une suite décroissante $\{M_{k_q,i_{k_q}}\}$ telle que $x \in M_{k_q,i_{k_q}}, \forall q$. Du fait que $f(x) \geq \beta_{k_q,i_{k_q}}, \forall q$ et l'ensemble de borne est cohérente on a $lim\ \gamma_{k_q} = \alpha = lim\ \beta_{k_q,i_{k_q}}$, ce qui implique $f(x) \geq \gamma$.
– (ii) Il s'ensuit de (2.14) que $\{M_{k,i_k}\}$doit contenir une sous-suite décroissante $\{M_{k_q,i_{k_q}}\}$ avec $\beta_{k_q} = \beta_{k_q,i_{k_q}}$. Puisque l'estimation de borne est cohérente, on a

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

– (iii) L'ensemble

$$S(x^0) := \{x \in S : f(x) \leq f(x^0)\}$$

est borné et, puisque S est fermé et f est continue, $S(x^0)$ est fermé. Comme on a déjà vu, $f(x^k) \leq f(x^0)$, on a $\{x^k\} \subset S(x^0)$. Donc x^k possède des points d'accumulation et (iii) est une conséquence de (i).

□

Noter que si l'estimation de borne est cohérente alors la sélection qui améliore des bornes est aussi complète, parce que $f(x) \leq \beta_{k_q}, \forall x \in S$ et lorsque (2.14) est appliqué, on a

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

pour tout ensemble M.

Dans le prototype SE, il est nécessaire que $M \cap S \neq \emptyset$ pour chaque élément M de la partition. Or, dans des cas concrets il n'y a pas de règles concrètes pour décorer de façon définitive si $M \cap S$ est vide ou non. En effet, pour définir un ensemble M, souvent un polyèdre, il suffit de connaître l'ensemble $V(M)$ de sommets de M. Mais évidemment, cela ne suffit pas pour donner une décision concrète sur $M \cap S \neq \emptyset$ ou $M \cap S = \emptyset$, même dans un cas très simple. Cette déficience, en effet limite l'applicabilité de la méthode. Dans ce qui suit, on va utiliser des règles qui ne garantissent pas que les ensembles vérifiant $M \cap S = \emptyset$ seront supprimés mais seulement une partie suffisante d'eux, tout en assurant la convergence de l'algorithme.

2.2.2 Séparation et Evaluation avec des ensembles non réalisables

Définition 2.2.4 *Un ensemble M tel que $M \cap S = \emptyset$ est appelé "non réalisable". Un ensemble M tel que $M \cap S \neq \emptyset$ est appelé "réalisable". Un ensemble M est dit "incertain" quand nous ne savons pas si M est réalisable ou non.* □

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

– $-\infty < \beta(M) \leq min\, f(M \cap S)$,si M est réalisable,
– $-\infty < \beta(M) \leq min\, f(M)$,si M est incertain.

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

Prototype 2

Initialisation :

1. Choisir $S \subset M_0$ un compact, un ensemble fini des 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

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

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

– $-\infty < \beta(M_{0,i}) \leq min\ f(M_{0,i} \cap S)$,si $M_{0,i}$ est réalisable,
– $-\infty < \beta(M_{0,i}) \leq min\ f(M_{0,i})$,si $M_{0,i}$ est incertain.

4. Calculer

$$\alpha_0 = min_{i \in I_0}\ \alpha_{0,i}, \tag{2.17}$$

$$x^0 \in argmin\ \{f(x^{0,i}, i \in I_0)\}, \tag{2.18}$$

$$\beta_0 = min_{i \in I_0} \beta_{0,i}. \tag{2.19}$$

Itération k :

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

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

ou pour lequel on sait que $min\ f(S)$ ne peut pas atteindre $M_{k,i}$. Soit $\mathcal{R}_k$ la collection des éléments restant $M_{k,i} \in \mathcal{M}_k$.
Si $\mathcal{R}_k = \emptyset$ alors s'arrêter, x^k est une solution.

k.2 Sélectionner $M_{k,i_k} \in \mathcal{R}_k$; choisir un ensemble fini des indices J_{k+1} et construire une partition

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

de M_{k,i_k}. Appliquer des règles pour éliminer les sous-ensembles qu'on sait ils sont non réalisables.

k.3 Pour chaque $i \in J_{k+1}$ déterminer

$$S_{k+1,i} \in M_{k+1,i} \cap S, S_{k+1,i} \neq \emptyset$$

et

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

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

k.4 Pour chaque $i \in J_{k+1,i}$ 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

$$\mathcal{M}_{k+1} = (\mathcal{R}_k \setminus M_{k,i_k}) \cup M_{k,i_k}. \tag{2.20}$$

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

k.6 Calculer

$$\alpha_{k+1} = min_{i\in I_{k+1}}\, \alpha_{k+1,i}, \tag{2.21}$$

$$\beta_{k+1} = min_{i\in I_{k+1}}\, \beta_{k+1,i}. \tag{2.22}$$

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

Définition 2.2.5 *Soit $\{M_{k_q}\}$ une suite décroissante d'ensembles générés par la procédure de division. Une procédure de division est dite "exhaustive" si pour chaque suite décroissante $\{M_{k_q}\}$ des ensembles générés par cette procédure, la suite de diamètres $d(M_{k_q})$ associés à M_{k_q} vérifie*

$$\lim_{q\longrightarrow\infty} d(M_{k_q}) = 0. \tag{2.23}$$

□

Évidemment, pour une suite décroissante des ensembles générés par une division exhaustive, on a

$$\lim_{q\longrightarrow\infty} M_{k_q} = \cap_q M_{k_q} = \{\overline{x}\}, \quad \overline{x} \in \mathbb{R}^n. \tag{2.24}$$

Définition 2.2.6 *L'opération d'estimation de borne inférieure est appelée "fortement cohérente" si pour n'importe quelle suite décroissante $\{M_{k_q}\}$ d'ensembles générée par une division exhaustive telle que*

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

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

$$\beta(M_{k'_q}) \longrightarrow f(\overline{x}), \quad q \longrightarrow \infty. \tag{2.25}$$

□

Définition 2.2.7 *L'élimination-par-non réalisabilité est appelée "certaine à la limite" si pour n'importe quelle suite décroissante $\{M_{k_q}\}$ d'ensembles générées par une division exhaustive telle que $M_{k_q} \longrightarrow \{\overline{x}\}$ on a*

$$\overline{x} \in S. \tag{2.26}$$

□

On désigne Y^α l'ensemble des points d'accumulation de la suite y^k de point correspondant à β_k. Soit $X^* = argmin\, f(S)$ l'ensemble de solution optimales de (P). On a le théorème suivant.

Théorème 2.2 *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

$$\beta := \lim\ \beta_k = \min\ f(S) \tag{2.27}$$

et

$$Y^\alpha \subset X^*. \tag{2.28}$$

Preuve : Supposons que la procédure ne se termine pas après un nombre fini d'itérations. On considère la suite de borne inférieure β_k. Soit, pour chaque k, y^k le point correspondant à β_k qui est généré par la procédure. Soit $M_k \in argmin\{\beta(M) : M \in \mathcal{R}_k\}$ un élément dans $\{\mathcal{R}_k\}$ tel que

$$y^k \in M_k, \quad \beta_k = \beta(M_k).$$

Par construction, β_k est une suite non décroissante et bornée supérieurement par $min(S)$, donc on a l'existence de

$$\beta = \lim_{k \longrightarrow \infty} \beta_k \leq min(S).$$

Soient $\overline{y} \in Y^\alpha$ et $\{y^r\}$ une sous-suite de $\{y^k\}$ telle que $y^r \longrightarrow \overline{y}$. Alors en vertu de (i) et (ii) on peut conclure qu'il existe une suite décroissante $\{M_q \subset M_r\}$ des éléments de division telle que

$$y^q \in M_q, \quad \beta_q = \beta(M_q)$$

et $M_q \longrightarrow \{\overline{y}\}$. Il découle de (iii) l'existence d'une sous-suite $\{M_{q'}\}$ de M_q telle que $\beta(M_{q'}) \longrightarrow f(\overline{y})$. En vertu de (iv) on a $\overline{y} \in S$. On voit donc

$$\beta = \lim\ \beta(M_{q'}) = f(\overline{y}) \leq min\ f(S)$$

et par conséquent on a

$$\beta = f(\overline{y}) = min\ f(S),$$

ce qui achève la preuve du théorème. □

2.2.3 Réalisation

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

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

On va aborder par la suite ces opérations.

2.2.3.1 Stratégie de division

Par idée, les éléments de partition M_k doivent être très simple pour qu'on puisse les manipuler facilement. Naturellement, on utilise les plus simples polyèdres comme des simplexes, des rectangles et des cônes (polyédraux). Il faut également diviser ces polyèdres de telle manière que la procédure de division soit exhaustive, ce qui est nécessaire pour assurer la convergence de la méthode SE (cf. Théorème 2.2).

(a) Subdivision simpliciale

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

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

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

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

et ainsi on peut construire une subdivision, appelée *radiale*, de M. Très souvent, on choisit s comme le milieu de la plus longue arrête de M et M est divisé ainsi en deux simplexes. Dans ce cas, on a une bisection de M. Il est démontré que la bisection est exhaustive. Pourtant, on constate que les procédures exhaustives de division (en particulier bisection) ne sont pas très efficaces ; la convergence de la méthode est assez lente. On suggère alors d'utiliser comme s un point ωobtenu 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 c'est qu'on ne peut plus assurer que la procédure de division soit exhaustive. Récemment, certaines stratégies plus flexibles sont étudiées. L'idée est d'utiliser le plus souvent possible la ω-subdivision et de faire intervenir la bisection pour empêcher la dégradation. Une stratégie heuristique mais pratique, a été proposée dans [45]. Considérons un simplexe M et un point $\omega \in M$, qui s'écrit comme $\omega = \sum_{i=0}^{n} \lambda_i v^i$. Étant fixé un nombre $\delta > 0$.
Si $min\{\lambda_i : \lambda_i > 0, i \in \{1, 2, \ldots, n\}\} > \delta$ alors appliquer ω-subdivision. Sinon utiliser bisection.
Les expériences ont indiqué que $\delta = \frac{1}{2}n^2$ est un choix convenable.

(b) 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 serré qui contient S (convexe) peut être déterminé en résolvant $2n$ problèmes convexes

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

Les processus de subdivision rectangulaire jouent un rôle important dans des méthodes de Séparation et Evaluation. L'approche de Phillips et Rosen [141] (voir également Kalantari et Rosen [53]) emploie la subdivision exhaustive, i.e. toutes les suites décroissantes des rectangles générées par l'algorithmes tendra à un point. Une bisection rectangulaire adaptative prétendue proposée dans Muu,L.D. [121] semble être plus efficace parce que l'exhaustion n'est pas nécessaire pour la convergence. Dans Horst et Tuy [49], un concept de la subdivision rectangulaire normale (NRS) a été présenté pour la classe des problèmes concaves séparables de minimisation qui inclut l'approche de Kalantari-Rosen et une subdivision proposée plus tôt dans Falk et Soland [28]. Intuitivement, la variante des algorithmes rectangulaires en utilisant ω-subdivision et subdivision adaptative devrait converger plus rapidement que ceux qui emploient la subdivision exhaustive, parce qu'ils tiennent compte des conditions du sous-problème relaxé courant.

(c) Subdivision conique

Supposons que S possède un point intérieure ω. Soit M_0 un n-simplexe tel que $S \subseteq M_0, M_0$ a $n+1$ faces $F_{0,i}, i = 1, 2, \ldots, n+1$ de dimension $n-1$ qui sont des $(n-1)$-simplexes. Les cônes polyèdraux (appélés 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 provoquer une division de cône C en cônes $\{C_j : j \in J\}$ correspondant aux F_j. En particulier, la bisection de cônes est induite par la bisection de simplexes. Si la procédure de division de simplexe est exhaustive alors la procédure de division de cônes sera aussi exhaustive dans le sens que chaque suite décroissante de cône $\{C_k\}$ (générée par cette procédure) va tendre ver un rayon sortant de ω. Les cônes sont très utiles quand une solution globale se trouve sur la frontière d'un convexe. En effet le premier algorithme conique a été proposé par Tuy [165] pour la minimisation d'une fonction concave sur un polyèdre.

2.2.3.2 Règles de sélection

Naturellement, on peut choisir à chaque itération

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

qui satisfait (2.14), i.e. cette sélection améliore des bornes. Pourtant, il y a bien d'autres règles qui n'utilisent pas explicitement cette propriété. Par exemple (cf. Tuy et al. [177]) :

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

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

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

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

2.2.3.3 Estimation de borne

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

Définition 2.2.8 *Soit $\{T_k\}$ une suite d'ensemble de $\mathbb{R}^n$. Alors*

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

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

$$T = \lim_{k\longrightarrow\infty} T_k \Longleftrightarrow T = \overline{\lim}_{k\longrightarrow\infty} T_k = \underline{\lim}_{k\longrightarrow\infty} T_k.$$

□

Une division va générer des suites décroissantes $\{M_k\}$ qui convergent vers un compact $M := \cap_k M_k$. Notons $M_k \longrightarrow M$.

Lemme 2.1 *([44]) Soit $S \in \mathbb{R}^n$ un compact et soit $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ continue. Alors l'estimation de borne est cohérente si, pour toute suite décroissante de compacts M_k,on a*

- *(i) $M_k \longrightarrow M$ compact, $M \cap S \neq \emptyset$.*
- *(ii) Il existe une suite de compacts T_k telle que*

$$M_k \supseteq T_k \supseteq M_k \cap S, T_k \longrightarrow M \cap S.$$

- *(iii)*

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

- *(iv)*

$$\alpha(M_k) \longrightarrow min\ f(M \cap S).$$

L'estimation de borne présente toujours un dilemme entre la convergence et l'efficacité. Un algorithme SE va converger plus vite si on peut estimer des bornes d'une façon plus précise. Or, cela devrait coûter plus cher ce qui peut rendre l'algorithme moins efficace.

2.3 Les cas particuliers de SE

Depuis la création de SE, il a été largement utilisé et étudié. Dans cette section, nous allons présenter deux cas particuliers de SE. Ce sont les algorithmes de type SE appliqués aux problèmes

(a) dont le domaine des solutions admissibles est non convexe : Nous nous intéressons aux problèmes linéaires ou quadratiques pour lesquels certaines variables sont discrètes,

(b) dont la fonction objectif est non convexe : Ce sont des problèmes pour lesquels la fonction objectif est non convexe mais elle est séparable.

2.3.1 Le domaine est non convexe

Dans cette section, nous nous concentrons sur l'approche SE qui est appliquée afin de résoudre des problèmes mixtes en variables binaires. SE peut également être utilisé pour résoudre des problèmes d'optimisation en variables entières. Que ce soient les variables binaires ou entières, SE fait une énumération implicite de toutes les solutions possibles et il construit un arbre de recherche dont chaque nœud est associé à un problème continu basé sur le problème original. La racine de l'arbre est le problème original dont toutes les variables binaires (ou entières) sont relaxées.

A chaque itération, SE choisit un nœud de l'extrémité et résout le sous-problème correspondant. Quatres cas sont possibles (voir [167]) :

Cas 1. Le sous-problème n'est pas réalisable alors on peut le supprimer.

Cas 2. Le sous-problème a une valeur d'optimal supérieure à celle de la meilleure solution obtenue jusqu'au présent alors toutes les restrictions supplémentaires ne vont pas attribuer une meilleure solution. Nous supprimons le sous-problème.

Cas 3. Si la solution du sous-problème satisfait toutes contraintes binaires et de plus elle a une valeur inférieure à celle de la meilleure solution obtenue jusqu'au présent, alors nous sauvegardons la solution et nous faisons une mise à jour de la valeur optimale.

Cas 4. Si aucun des cas cités ci-dessus n'est pas réalisé alors (au moins) une des variables binaires est fractionnelle. Nous considérons deux nouveaux sous-problèmes. Pour chacun des nouveaux sous-problèmes, la variable fractionnelle est fixée à zero ou un. Ensuite, nous ajoutons ces sous-problèmes à l'arbre de recherche comme les fils du sous-problème.

Enfin, nous sélectionnons le prochain sous-problème à résoudre. L'algorithme s'arrête s'il n'y plus de problème à sonder.

La figure 2.1 montre un diagramme de l'algorithme SE pour un problème en variables mixtes.

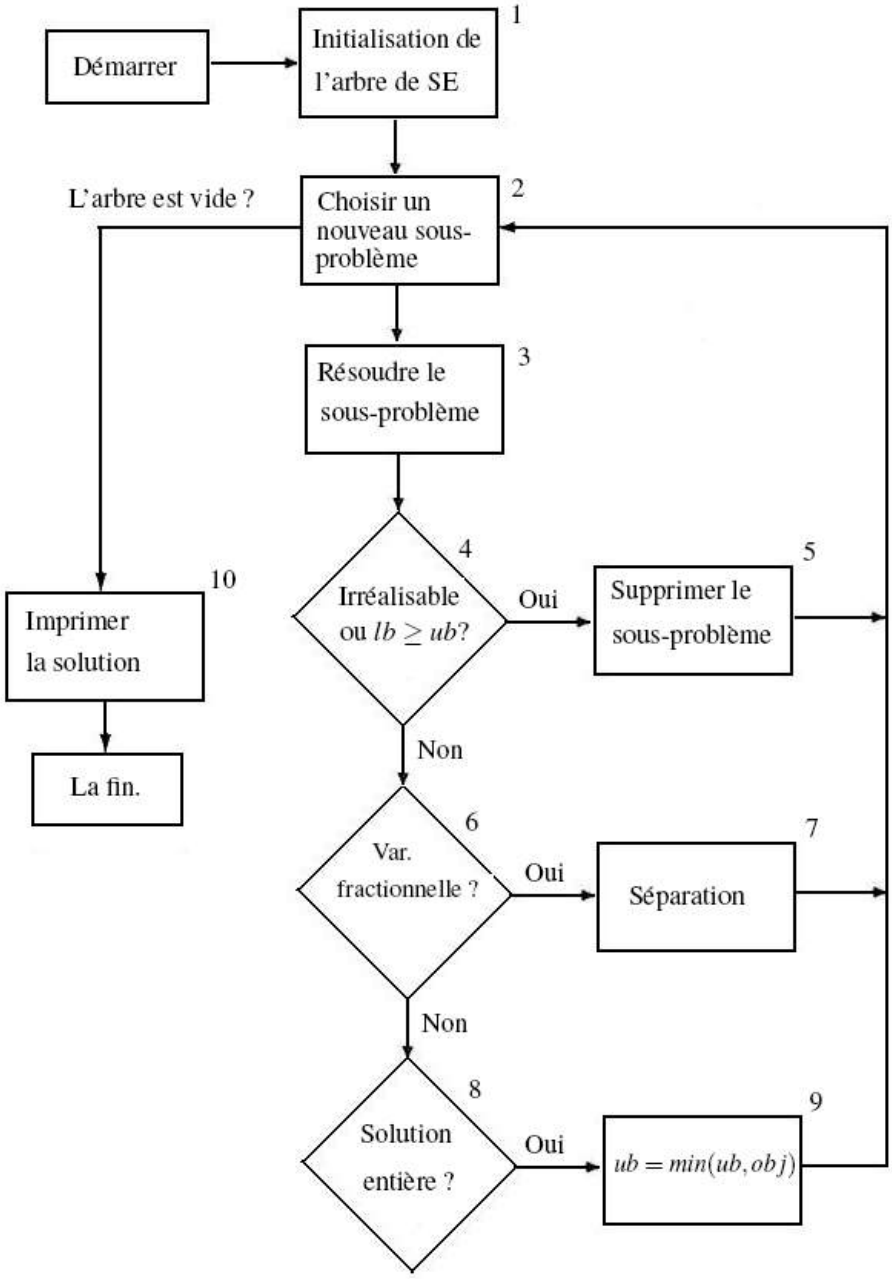


FIG. 2.1 – Un schéma de base pour un algorithme par SE pour les problèmes mixtes

A l'étape 2, nous choisissons le sous-problème suivant. Il existe plusieurs façon de choisir un sous-problème parmi les tous. Deux d'entre elles sont les plus utilisées. La première consiste à choisir le dernier problème ajouté à l'arbre de recherche. Ce choix est dû aux problèmes qui ont le plus grand nombre de variables entières. De cette façon, nous allons plus de chance d'obtenir une solution entière. Deuxième approche est de choisir le sous-problème qui a la

plus petite valeur de fonction objectif. Le but est d'améliorer la borne inférieure le plus vite possible.

A la troisième étape, nous devons choisir la variable sur laquelle la séparation s'effectue. Nous pouvons choisir la variable qui est la plus proche d'un nombre entier ou choisir celle de premier, c'est-à-dire la variable qui a la plus petite indice d'ordre parmi les autres.

Pendant l'énumération implicite, si nous arrivons à bien diminuer le nombre de séparation ou la taille de l'arbre de recherche, alors l'approche par SE sera plus efficace. Pour atteindre ce but, nous développerons des algorithmes combinés à la base de SE qui

- cherchent à trouver des variables (qui doivent être binaires) qui s'approchent à 0 ou 1. Après les avoir localisées, nous allons les fixer à 0 ou 1, respectivement. La méthode DCA servira à trouver ces variables.
- utilisent des sous-approches locales (DCA) afin de trouver des solutions binaires plus vite. De cette façon, nous essayons d'améliorer la borne supérieure. Ayant une meilleure borne supérieure nous permet de supprimer plus de sous-problèmes (inutiles) et par conséquent d'avoir une convergence plus rapide.

La Figure (2.2) montre un diagramme de l'algorithme par SE révisé. A l'étape trois, après avoir résolu le sous-problème, nous utilisons sa solution pour démarrer DCA. Deux cas sont possibles :

- **La solution fournie par DCA est entière :** nous pouvons l'utiliser pour savoir s'il s'agit d'une solution améliorante. Si la valeur de fonction objectif associée à cette solution est inférieure à la valeur d'optimale (trouvée jusqu'à présent), nous faisons une mise à jour de la borne supérieure ainsi que de la valeur optimale.
- **La solution fournie par DCA n'est pas entière :** si certaines variables se sont approchées vers 0 ou 1, nous les fixons à 0 ou 1, respectivement. Ensuite, nous redémarrons DCA pour obtenir une solution entière. Puisque certaines variables sont déjà fixées à 0 ou 1, alors nous avons plus de chance d'avoir une solution entière. Cela va accélérer la convergence de l'algorithme.

2.3.2 La fonction objectif est non convexe

L'algorithme par séparation et évaluation que nous présentons se trouve dans [127].

Nous considérons le problème suivant

$$(P) : \min \quad \{f(x) : x \in D \cap C \subset \mathbb{R}^n\}.$$

où D est un ensemble convexe et compact, $C := \{x : l_i \leq x_i \leq L_i, i = 1, \ldots, n\}$ est un rectangle. Nous supposons que la fonction objectif est séparable, c'est-à-dire

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

où $f_i(x_i), i = 1, \ldots, n$, sont des fonctions non convexes. Dans cette section nous nous intéressons à la résolution du problème (P) en faisant une sous-estimation de la fonction

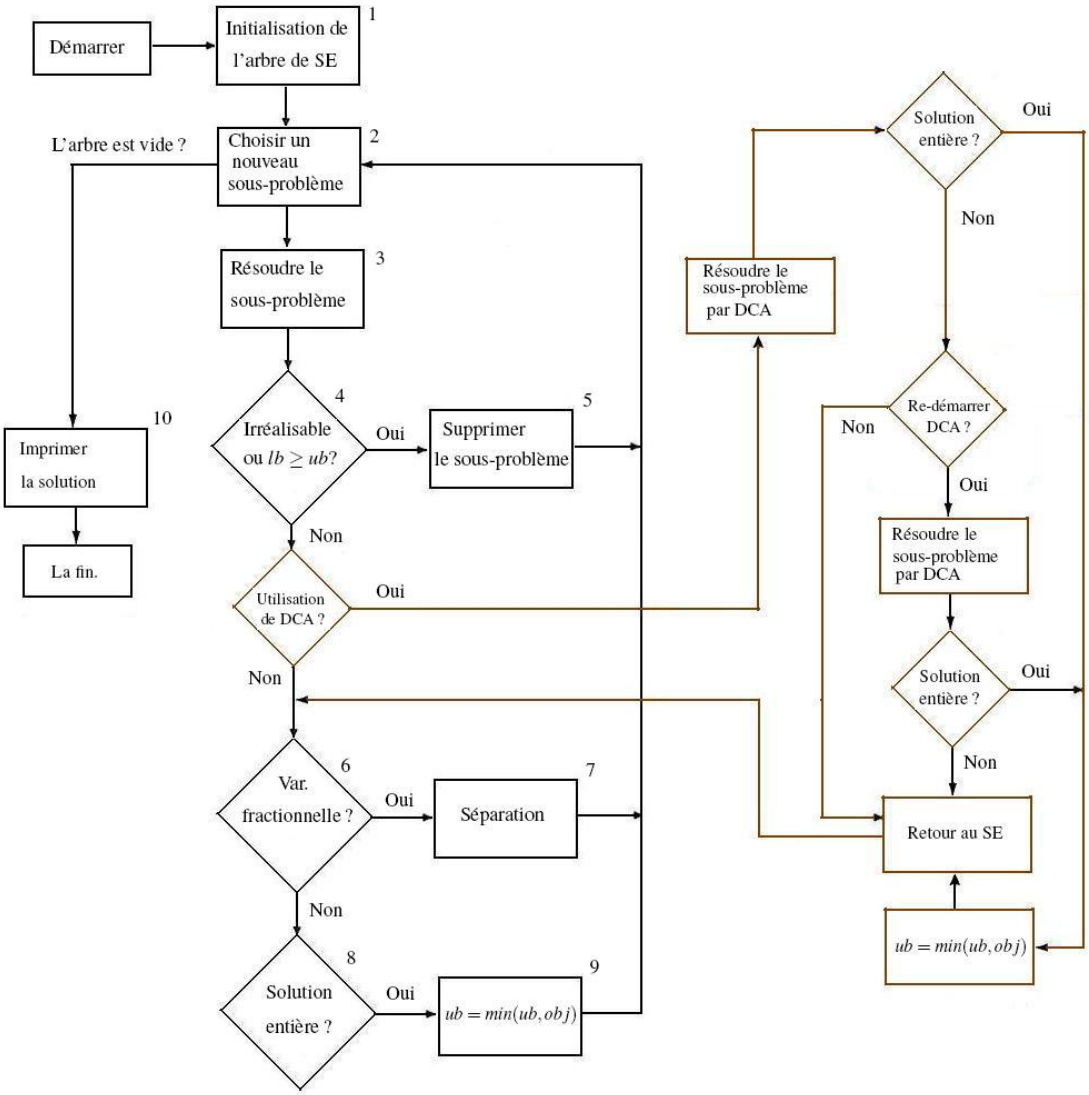


FIG. 2.2 – Un schéma de base pour un algorithme par SE révisé (combiné avec DCA)

objectif par d'envelopes convexes. Soit ϕ_i l'envelope convexe de f_i sur l'intervalle $[l_i, L_i]$ alors $\phi = \sum_{i=1}^{n} \phi_i$ est l'envelope convexe de $f(x)$ sur le rectangle C.

Nous allons utiliser un algorithme de type SE afin de résoudre (P). Pendant cet algorithme, on construit deux suites comme $\{\alpha_k\}, \{\beta_k\}$, celle de bornes inférieures et celle de bornes supérieures, respectivement. Pour cela, on construit une suite de points x_k telle que chacun de ces points est la solution d'un sous-problème comme P_{kv}. Soit $C_{kv} \subset C$ un rectangle tel que

$$C_{kv} = \{x : l^{kv} \leq x \leq L^{kv}\}.$$

La fonction objectif du problème P_{kv} est une envelope convexe de f sur C_{kv} et se définit par

$$\phi^{kv}(x) := \sum_{i=1}^{n} \phi_i^{kv}(x)$$

où f_i a été remplacé par son envelope convexe ϕ_i^{kv} sur l'intervalle $[l_i^{kv}, L_i^{kv}]$. Le sous-problème P_{kv} associé au rectangle C_{kv} est

$$(P_{kv}) : \min \quad \{\phi^{kv}(x) : x \in D \cap C_{kv}\}.$$

Soit x^{kv} la solution optimale de P_{kv}, alors $\phi^{kv}(x^{kv})$ est une borne inférieure pour f sur $D \cap C_{kv}$ et $f(x^{kv})$ est une borne supérieure pour la valeur optimale de (P).

L'algorithme se résume comme la suite :

Initialisation de l'algorithme par SE

Nous allons construire les premières bornes inférieure et supérieure. Soit ϕ_i l'envelope convexe de f_i sur l'intervalle $[l_i, L_i]$. Alors $\phi = \sum_{i=1}^{n} \phi_i$ est l'envelope convexe de $f(x)$ sur le rectangle C, et nous résolvons le problème suivant

$$\min \left\{ \sum_{i=1}^{n} \phi_i(x) : x \in D \cap C \right\}.$$

Soit x^0 la solution optimale de ce problème, alors la première borne inférieure est $\alpha_0 = \sum_{i=1}^{n} \phi_i(x^0)$ et la première borne supérieures est $\beta_0 = \sum_{i=1}^{n} f(x^0)$. Si $\beta_0 - \alpha_0 \leq \epsilon$ alors x^0 est une solution ϵ-optimale, sinon nous devons choisir un intervalle à diviser. Nous choisissons cet intervalle de façon suivante, soit $[l_t, L_t]$ l'intervalle pour lequel

$$f_t(x_t^0) - \phi_t(x_t^0) = \max\{f_i(x_i^0) - \phi_i(x_i^0) : i = 1, \ldots, n\}$$

alors nous divisons l'intervalle $[l_t, L_t]$ en deux, i.e., $[l_t, (l_t + L_t)/2]$ et $[(l_t + L_t)/2, L_t]$ et nous ajoutons deux problèmes à l'ensemble des problèmes qui doivent être résolus par la suite. Ce sont deux problèmes suivants

$$\min \{f(x) : x \in D \cap C^1\} \tag{2.29}$$

et

$$\min \{f(x) : x \in D \cap C^2\} \tag{2.30}$$

où

$$C^1 := \{x : l_t \leq x_t \leq (l_t + L_t)/2, l_i \leq x_i \leq L_i : i \neq t\}$$

et

$$C^2 := \{x : (l_t + L_t)/2 \leq x_t \leq L_t, l_i \leq x_i \leq L_i : i \neq t\}.$$

Itération k

A l'itération k de SE, il existe p_k rectangles $\{C_{k1}, \ldots, C_{kp_k}\}$. Associé à chaque rectangle, il y a une envelope convexe qui sous-estime la fonction objectif, f. A la base de chaque rectangle, comme C_{kv}, nous construisons un problème relaxé dit (P_{kv}) où $v = 1, \ldots, p_k$. Ce problème se définit par

$$(P_{kv}) : \min \{\phi^{kv}(x) : x \in D \cap C_{kv}\}.$$

Soit x^{kv} la solution optimal de P_{kv}, alors $\phi^{kv}(x^{kv})$ est une borne inférieure pour f sur $D \cap C_{kv}$ et $f(x^{kv})$ est une borne supérieure sur $D \cap C_{kv}$. Choisissons le rectangle qui a la plus petite valeur de fonction objectif. Alors,

$$\phi^{kv_k}(x^{kv_k}) = \min_v \phi^{kv}(x^{kv}), \quad v = 1, \ldots, p_k.$$

Si $f(x^{kv_k}) = \phi^{kv_k}(x^{kv_k})$ alors nous avons trouvé la solution optimal de (P) sinon nous passons à la prochaine étape qui consiste à diviser le rectangle C_{kv_k} en deux ou plusieurs. Pour simplifier la notation, soit $y^k := x^{kv_k}$. L'intervalle à être divisé est choisi de façon que

$$f_t(y_t^k) - \phi_t^{kv_k}(y_t^k) = \max\{f_i(y_i^k) - \phi_i^{kv_k}(y_i^k) : i = 1, \ldots, n\}$$

et nous divisons l'intervalle $[l_t^{kv_k}, L_t^{kv_k}]$ en deux, i.e. $[l_t^{kv_k}, (l_t^{kv_k}+L_t^{kv_k})/2]$ et $[(l_t^{kv_k}+L_t^{kv_k})/2, L_t^{kv_k}]$. Ensuite, nous considérons deux nouveaux rectangles :

$$C_{kv_k}^1 = \left(\prod_{i=1, i\neq t}^{n} [l_i^{kv_k}, L_i^{kv_k}] \right) \times [l_t^{kv_k}, (l_t^{kv_k} + L_t^{kv_k})/2]$$

et

$$C_{kv_k}^2 = \left(\prod_{i=1, i\neq t}^{n} [l_i^{kv_k}, L_i^{kv_k}] \right) \times [(l_t^{kv_k} + L_t^{kv_k})/2, L_t^{kv_k}].$$

Enfin nous ajoutons deux problèmes qui correspondent à deux nouveaux rectangles à l'ensembles des problèmes qui doivent être résolus par la suite.

Chaque fois que nous obtenons une nouvelle borne qu'elle soit inférieure ou supérieure, nous la comparons avec les meilleures bornes et si c'est une borne améliorante alors nous la remplaçons. S'il s'agit de borne supérieure alors nous sauvegardons la solution qui lui correspond.

Lorsque la différence de la borne supérieure et la borne inférieure est suffisamment petite, l'algorithme s'arrête et la solution actuelle est l'optimum.

Deuxième partie

Gestion de portefeuille en finance : modèles et méthodes

Chapitre 3

Gestion de portefeuille : les notions de base

Résumé *Ce chapitre concerne une introduction à la gestion de portefeuille, plus particulièrement aux différentes mesures de risque utilisées dans la littérature financière. La gestion de portefeuille consiste à gérer les capitaux confiés dans le respect des contraintes réglementaires et contractuelles et appliquant les politiques d'investissements définies en interne, pour en tirer le meilleur rendement possible en fonction du risque choisi. Dans ce chapitre, nous allons d'abord définir la rentabilité et le risque. Ensuite, la théorie d'utilité, la notion de dominance stochastique et la notion de cohérence seront présentées. Ces notions seront utilisées pour comparer des différentes mesures de risque que nous allons présenter par la suite.*

3.1 Introduction

Imaginez un investisseur qui dispose d'un capital et qui a l'opportunité d'investir dans un certain nombre d'actifs financiers. Typiquement un tel investisseur doit prendre une décision importante. Comment répartir son capital parmi les actifs ? La gestion de portefeuille nous apporte la réponse. En gestion de portefeuille, nous nous intéressons aux problèmes de choix d'actifs financiers en présence de risque. Par exemple, il peut s'agir de choisir entre plusieurs actifs, ceux qui permettent au mieux, soit de minimiser le risque pour un rendement fixé, soit de maximiser le rendement pour un niveau fixé de risque. Alors les deux dimensions fondamentales d'un investissement financier sont le rendement et le risque. La rentabilité (ou le rendement) est la variation de la valeur accumulée d'un actif ou d'un portefeuille sur une période donnée. Alors que la rentabilité est simple à évaluer, il n'existe pas de façon unique d'évaluer le "risque".

Afin d'évaluer le risque d'un investissement, nous avons besoin des *informations* qui concernent les actifs composant le portefeuille. Ce sont les rentabilités des actifs, leur distribution et certaines propriétés stochastiques comme leur rentabilité moyenne et/ou la variance entre les rentabilités des actifs. Les informations peuvent aussi concerner les préférences des inves-

tisseurs. Ensuite, en utilisant ces informations nous nous servons des *critères* selon lesquels les actifs financiers sont classifiés par rapport à leur performance. Les critères dont nous disposons sont *Maximum d'Espérance d'Utilité (MEU)*, la *Dominance Stochastique* (DS) et la *Cohérence*. Enfin, ce sont les *procédures de calcul*, par lesquelles nous utilisons les informations afin de trouver les portefeuilles qui satisfont ces critères. Les procédures de calcul sont les différents modèles de choix de portefeuille que nous allons étudier dans ce chapitre. *Les informations*, *les critères* et *les procédures de calcul* caractérisent l'analyse de portefeuille.

Dans ce qui suit, nous allons d'abord présenter brièvement la rentabilité des actifs financiers et la notion de risque. Les critères de classification des portefeuilles seront présentés en section 3.2. La section 3.3 est consacrée aux procédures de calcul (les modèles de choix de portefeuille et les différentes mesures de risque). Une classification de différentes mesures de risque sera présentée en section 3.4.

3.1.1 Les rentabilités des actifs financiers

La rentabilité est une notion fondamentale en finance et elle apparaît dans l'expression de la plupart des modèles de gestion de portefeuille.

Définition 3.1.1 *(La rentabilité)*
La rentabilité mesure l'appréciation (ou dépréciation) relative de la valeur d'un actif financier ou d'un portefeuille d'actifs financiers entre deux instants successifs [2]. □

Soient t et $t+1$ deux instants successifs. Nous notons P_t et P_{t+1} les valeurs (prix) de l'actif aux instants t et $t+1$, respectivement. Nous calculons la rentabilité réalisée dans l'intervalle $[t, t+1]$ par

$$r_t = (P_{t+1} - P_t)/(P_t).$$

Si un flux financier D_{t+1} tel qu'un dividende est reçu entre t et $t+1$, cette formule devient

$$r_t = (P_{t+1} - P_t + D_{t+1})/(P_t).$$

Par la suite, nous supposerons que de tels flux sont incorporés à la valeur finale P_{t+1}.

De nombreux modèles financiers utilisent des séries historiques de cours boursiers pour estimer les propriétés stochastiques des rentabilités correspondantes comme leur rentabilité moyenne. Plusieurs méthodes sont disponibles. Celle que nous avons adoptée consiste en le calcul des rentabilités de sous-périodes, ensuite nous utilisons la formule suivante

$$\overline{r} = (1/T) \sum_{t=1}^{T} r_t.$$

Nous considérons, en général, que les rentabilités des cours boursiers possèdent des densités de probabilités normales et identiquement distribuées. Soit 'a' un actif avec la rentabilité

aléatoire r qui possède une telle distribution avec la moyenne μ et l'écart-type σ c'est-à-dire $r \sim N(\mu, \sigma)$. Dans ce cas, la densité de probabilité de la variable aléatoire r s'écrit

$$f(r) = \frac{1}{\sigma\sqrt{2\pi}} exp\left[-\frac{1}{2}\left(\frac{r-\mu}{\sigma}\right)^2\right].$$

L'hypothèse selon laquelle les rentabilités des actifs financiers sont normalement distribuées est souvent faite dans la littérature financière. Mais ceci ne correspond pas à la réalité. En général, les rentabilités sur les marchés ne suivent pas de loi gaussienne. Dans ce cas deux autres propriétés stochastiques entrent en jeu, ce sont la *skewness* et la *kurtosis*.

La *skewness* (ou *coefficient d'asymétrie*) d'une variable aléatoire se définit par

$$Sk = \frac{\mathbb{E}[r - \mathbb{E}(r)]^3}{\sigma(r)^3}.$$

Pour calculer la coefficient d'asymétrie d'une série de m observations, on utilise la formule

$$Sk = \frac{(\frac{1}{m-1}) \sum_{t=1}^{m} (r_t - \mu)^3}{\hat{\sigma}(r)^3}$$

où μ est la moyenne des rentabilités et $\hat{\sigma}(r)$ l'estimateur de leur écart-type. Pour une distribution normale, Sk est égale à zéro.

La *kurtosis* (ou *indice d'aplatissement*) d'une variable aléatoire est

$$K = \frac{\mathbb{E}[r - \mathbb{E}(r)]^4}{\sigma(r)^4}$$

et de la même manière que la coefficient d'asymétrie, l'indice d'aplatissement d'une série de m observations se calcule en utilisant la relation

$$K = \frac{(\frac{1}{m-1}) \sum_{t=1}^{m} (r_t - \mu)^4}{\hat{\sigma}(r)^4}.$$

Pour une distribution normale $K = 3$.

3.1.2 Le risque

Le risque est défini de différentes manières ([5], [43], [65]). L'une d'elles que nous allons adopter est celle de ATHEARN [5] et CROWE et HORN [20] dont le risque est défini comme étant lié à l'incertitude et d'autre part, causé par les écarts non attendus des résultats par rapport à l'objectif attendu. C'est-à-dire que nous fixons l'objectif et nous comparons les réalisations des rentabilités des actifs par rapport à cet objectif. La définition des réalisations qu'elles soient attendues ou non attendues dépend de l'investisseur et/ou du modèle. D'après cette définition, le risque porte sur deux aspects importants. D'abord, le risque est de nature

incertaine et de plus, il n'est pas attendu. Cette définition distingue les réalisations qui portent sur des profits et celles qui indiquent les pertes. Parfois les réalisations qui sont les plus proches de l'objectif fixé en amont sont considérées comme les réalisations attendues et parfois nous souhaitons seulement les réalisations qui sont supérieures à un objectif précis.

Dès que nous parlons de risque, nous pouvons distinguer deux types d'actifs financiers. Les actifs sans risques et les actifs risqués. Un actif sans risque est celui qui atteint le résultat, x, avec certitude, c'est-à-dire avec la probabilité de 1, ou $p(x) = 1$ où p est la fonction de probabilité. A l'opposé, nous avons l'actif risqué pour un tel actif, il y a plusieurs résultats possibles, comme $x_1, x_2, \ldots, x_n$, parmi lesquels il y a au moins un x_i avec $0 < p(x_i) < 1$.

Plutôt que d'analyser les risques au niveau des actifs individuels, nous voulons mesurer les risques au niveau du portefeuille. Car pour un investisseur, il n'est pas raisonnable d'investir tout son capital dans un seul actif mais il doit, au contraire, investir sur un ensemble d'actifs, c'est-à-dire un portefeuille d'actifs. En effet, le portefeuille garantit un taux de rendement moyen élevé et présente moins de fluctuations négatives de ses actifs pour le même niveau de risque, cette propriété s'appelle la *diversification* [128]. La diversification réduit le risque. S'il y a une bonne diversification, le risque d'un portefeuille est inférieur à la somme de risques de chacun des titres le composant. En fait, le risque d'un actif se compose, d'après le modèle d'évaluation des actifs financiers (MEDAF) (ou Capital Asset Pricing Model (CAPM)), du risque systématique et du risque spécifique. Le risque spécifique est le risque propre à l'actif considéré, c'est un risque diversifiable, c'est-à-dire qu'il peut être éliminé avec une bonne diversification du portefeuille. Le risque systématique ou risque de marché est dû aux fluctuations générales du marché et ne peut pas être éliminé par la diversification. Ce dernier doit être supporté par les investisseurs [128].

L'analyse du risque de portefeuille est plus compliquée que celle d'un seul actif car il faut observer le comportement de différents actifs, notamment à cause de la disparité de leur rendement respectif. De plus, il y a fort probablement des corrélations entre les différents actifs. Autrement dit, le risque de l'investissement est en relation avec les autres. En outre, l'investisseur ne peut pas négliger l'influence des décisions des autres investisseurs sur les siennes. En général, un investisseur peut avoir trois attitudes différentes :

1. **risque-averse (riscophobe)** : l'investisseur qui évite le risque,
2. **risque-neutre** : l'investisseur qui est indifférent à la prise de risque,
3. **riscophile** : l'investisseur qui n'hésite pas à prendre le risque.

Alors chaque investisseur éprouve une certaine aversion au risque.

3.2 Les critères

Dans cette section, nous allons présenter quelques notions basées sur des axiomes. Ces notions vont nous aider à bien comparer les différentes mesures de risque. Chacune d'entre elles est un ensemble de critères qu'un portefeuille doit satisfaire pour qu'il soit classifié comme celui que l'investisseur préfère.

Pour un investisseur, l'objectif principal est la rentabilité et la profitabilité. Alors nous allons d'abord étudier une notion qui est basée sur l'hypothèse selon laquelle l'investisseur connaît bien la loi de distribution des rendements.

3.2.1 Le critère de Maximum d'Espérance d'Utilité

La théorie d'utilité fournit une façon d'exprimer la sensibilité de l'individu au risque. Pour cela, la théorie d'utilité étudie les préférences des individus et leurs représentations numériques au sein des fonctions d'utilité.

En 1947, von Neumann et Morgenstern ont mis au point un ensemble d'axiomes concernant les préférences des individus. Dans leur étude, von Neumann et Morgenstern ont adopté une convention. Ils supposent que l'individu connaît la loi de distribution des revenus aléatoires. Ici, il est supposé que l'individu est rationnel, c'est-à-dire que son comportement est défini au moyen de cet ensemble d'axiomes.

Pour chaque individu (rationnel) on peut définir une fonction d'utilité dont l'argument est la richesse et l'individu peut utiliser l'espérance mathématique afin de classifier tous les investissements risqués. Plus précisément, une fonction d'utilité est une règle par laquelle on associe un indice numérique à chacun des investissements de sorte que les préférences de l'individu se manifestent par le fait que cet indice d'"utilité" est d'autant plus élevé que la préférence est grande.

Soit $L = \{p_1A_1, p_2A_2, \ldots, p_nA_n\}$ un *investissement risqué simple* ou plus simplement un *investissement (risqué)*, où les A_i sont des n résultats possibles avec les probabilités p_i. Les axiomes du critère de Maximum d'Espérance d'Utilité (MEU) sont les suivants [108] :

Axiome 1 : *Comparabilité.* D'après cet axiome, en face de deux revenus monétaires, l'investisseur doit choisir entre A_i et A_j. Soit il préfère A_i à A_j ($A_i \succ A_j$), soit il préfère A_j à A_i ($A_j \succ A_i$) soit A_i et A_j lui sont indifférents ($A_i \sim A_j$).

Axiome 2 : *Continuité (valeur intermédiaire).* Si l'investisseur préfère A_3 à A_2 et A_2 à A_1, alors il existe une probabilité $U(A_2)$ ($0 \leq U(A_2) \leq 1$) telle que,

$$L = \{(1 - U(A_2))A_1, (U(A_2))A_3\} \sim A_2.$$

En fait, l'individu ne doit pas exprimer de discontinuité dans ses choix, par exemple, si l'investisseur préfère A_3 à A_1, on veut que A_2 qui est extrêmement proche de A_3 soit préféré à A_1.

Axiome 3 : *Interchangeabilité.* Supposons que $L_1 = \{p_1A_1, p_2A_2, p_3A_3\}$ soit un investissement. Si l'investisseur est indifférent entre A_2 et B, tels que $B = \{q_1A_1, (1 - q_n)A_3\}$, alors l'axiome d'interchangeabilité dit que l'investisseur est indifférent entre L_1 et L_2 où $L_2 = \{p_1A_1, p_2B, p_3A_3\}$.

Axiome 4 : *Transitivité.* Soient L_1, L_2, L_3 les investissements possibles et $L_1 \succ L_2$ et $L_2 \succ L_3$. D'après l'axiome de transitivité $L_1 \succ L_3$. De la même manière, si $L_1 \sim L_2$ et

$L_2 \sim L_3$ alors $L_1 \sim L_3$. Cet axiome est typiquement un axiome de rationalité puisqu'il établit que l'individu doit faire preuve d'une cohérence interne dans ses choix.

Axiome 5 : *Décomposabilité.* Un investissement complexe est un investissement dont les résultats se composent d'investissements simples. Soit L^* un investissement complexe tel que

$$L^* = \{qL_1, (1-q)L_2\}$$

où L_1 et L_2 sont les investissements simples tels que

$$L_1 = \{p_1A_1, (1-p_1)A_2\},$$

$$L_2 = \{p_2A_1, (1-p_2)A_2\}.$$

Selon l'axiome de Décomposabilité, L^* peut être décomposé en un investissement simple comme L qui contient seulement A_1 et A_2. Plus précisément,

$$L^* \sim L = \{p^*A_1, (1-p^*)A_2\}$$

avec $p^* = qp_1 + (1-q)p_2$.

Axiome 6 : *Monotonicité.* Soient $L_1 = \{pA_1, (1-p)A_2\}$ et $L_2 = \{pA_1, (1-p)A_3\}$. Si $A_3 > A_2$, et alors $A_3 \succ A_1$, D'après cet axiome on a $L_2 \succ L_1$.

Nous pouvons prouver (voir [108]) qu'en acceptant ces axiomes, le critère de (MEU) est la meilleure méthode de prise de décision et les investissements doivent être classifiés selon leur espérance d'utilité.

D'après les idées de von Neumann et Morgenstern, la première propriété de la fonction d'utilité est la richesse. C'est-à-dire qu'entre deux investissements existants, celui qui donne plus de revenu est préféré. La deuxième propriété de la fonction d'utilité est l'attitude de l'individu vis à vis du risque. Ce sont les trois positions différentes de l'individu que nous avons citées ci-dessus, c'est-à-dire : riscophobe, risque-neutre et riscophile.

Soient $U(w)$ la fonction d'utilité (en fonction de la richesse w) et $U''(w)$ la dérivée seconde de la fonction d'utilité. Chaque individu rationnel veut maximiser l'espérance mathématique de $U(w)$, i.e., $E[U(w)]$ (d'où vient le nom Maximum d'Espérance d'Utilité (MEU)) [24].

Définition 3.2.1 *(le critère d'espérance d'utilité) [24]*
Les préférences d'un individu satisfont au critère d'espérance d'utilité s'il existe une fonction croissante U appelée fonction d'utilité telle que l'individu préfère le rendement aléatoire w_1 au rendement aléatoire w_2 si et seulement si l'espérance d'utilité de w_1 est supérieure à celle de w_2 :

$$w_1 \text{ préféré à } w_2 \Longleftrightarrow E[U(w_1)] \geq E[U(w_2)].$$

□

La croissance de la fonction d'utilité exprime simplement que l'individu aime la richesse.

En utilisant la fonction d'utilité, les comportements des individus à l'égard du risque se traduisent comme suit :

1. **risque-averse (riscophobe)** : Si l'investisseur est risque-averse, alors $U(w) > E[U(w)]$, tel que $U''(w) < 0$,
2. **risque-neutre** : Si l'investisseur est risque-neutre, alors $U(w) = E[U(w)]$, tel que $U''(w) = 0$,
3. **riscophile** : Si l'investisseur est riscophile, alors $U(w) < E[U(w)]$, tel que $U''(w) > 0$.

Quelques années après les travaux de von NEUMANN et de MORGENSTERN, MARKOWITZ [1952,1959] et TOBIN [1958] ont utilisé la théorie d'utilité afin de résoudre le problème de choix de portefeuille.

Même si le critère d'espérance d'utilité a un fondement solide, il présente certains défauts. Soient A et B deux investissements risqués et U la fonction d'utilité d'un investisseur. En utilisant le critère d'espérance d'utilité, nous pouvons associer un nombre à chacun de ces investissements (les nombres peuvent être égaux). Soient $U(A) = a$ et $U(B) = b$, a et b peuvent nous aider à classifier les investissements A et B selon la préférence de l'investisseur. Ils n'ont pas d'autre signification. C'est-à-dire qu'il ne peuvent pas nous dire la magnitude de préférence. Par exemple, si $a = 150$ et $b = 100$, cela ne dit pas que l'investissement A est 50% mieux que l'investissement B. En bref, le critère d'espérance d'utilité est un critère plutôt *ordinal* que *cardinal*. Ce fait est le premier défaut du critère de MEU [108].

L'autre défaut du critère de MEU est lié à la richesse. Soit w la richesse initiale de l'investisseur et notons x le revenu sur la richesse initiale à la fin de la période de l'investissement. Alors la richesse totale est $w + x$. Contrairement au caractère constant de w, nous savons que x est une variable aléatoire. L'inclusion de w dans l'utilité est cruciale. Mais contrairement à l'importance de w en théorie d'utilité, les investisseurs prennent leurs décisions en se concentrant plus particulièrement sur les fluctuations de x, alors que w reste en arrière plan ou même w est négligé. Cela signifie que les investisseurs prennent leurs décisions sur $U(x)$ au lieu de $U(w + x)$, où U est la fonction d'utilité de l'investisseur [108].

Prendre la décision selon la fluctuation de la richesse est en contradiction avec les fondements de base de l'espérance d'utilité [108]. Ce n'est pas le cas pour la notion de la *Dominance Stochastique (DS)* même si DS est une dérivée de la théorie de l'espérance d'utilité. En utilisant DS, on partitionne l'ensemble des investissements possible en deux, celui des investissements efficients et celui des investissements inefficaces. Cette classification est invariante par rapport à w. DS est la notion que nous allons aborder dans la section suivante.

3.2.2 Dominance stochastique

La *Dominance Stochastique* (DS) est basée sur les axiomes des préférences des individus à l'égard de l'aversion au risque [30]. Tout d'abord, DS était la généralisation des travaux en théorie de majorisation (Hardy et al. [40]) par HANOCH et LEVY [41] ; ROTHSCHILD et

STIGLITZ, [151]. En suite, il a été largement utilisé en économie et en finance ([11], [107] et [125] et les références incluses).

En gestion de portefeuille, la notion de DS est utilisée pour mettre en ordre les portefeuilles et nous supposons que nous connaissons la distribution des rendement aléatoires des actifs. A la base de cette convention sur les distributions, DS met en ordre les rendements des actifs tels que les rendements dominants ont une espérance d'utilité supérieure à celles des rendements dominés. Ce raisonnement s'applique sans avoir besoin d'information sur la fonction d'utilité des investisseurs.

Dans la suite, nous allons présenter brièvement la dominance stochastique en premier, deuxième et troisième ordre [108].

Nous supposons qu'il existe deux distributions des rentabilités, nommées F et G, sur le domaine $D \subset [a, b)$. Ici, $F\ DSP\ G$ veux dire que "F" domine "G" stochastiquement en premier ordre, $F\ DSD\ G$ montre que "G" est dominé par "F" stochastiquement en deuxième ordre. Finalement, $F\ DST\ G$ c'est-à-dire que "F" domine "G" stochastiquement en troisième ordre. Mathématiquement [65] :

- $F\ DSP\ G$: si
  $F(r) \leq G(r) \quad \forall r \in D$ et $\exists r^* : \quad F(r^*) < G(r^*)$
  sous l'hypothèse que $U' > 0$. Cela veut dire que $\mathbb{E}(U(r|F)) > \mathbb{E}(U(r|G))$.
- $F\ DSD\ G$: si
  $\int_a^r F(p)d(p) \leq \int_a^r G(p)d(p) \quad \forall r \in D$ et $\exists r^* : \quad \int_a^{r^*} F(p)d(p) \leq \int_a^{r^*} G(p)d(p)$
  sous l'hypothèse que $U' > 0$ et $U'' < 0$. Cela veut dire que si $F\ DSD\ G$ alors $\mathbb{E}(U(r|F)) > \mathbb{E}(U(r|G))$.
- $F\ DST\ G$: si
  $\int_a^r (\int_a^q F(p)d(p))dq \leq \int_a^r (\int_a^q G(p)d(p))dq \quad \forall r \in D$ et
  $\exists r^* : \int_a^{r^*} (\int_a^q F(p)d(p))dq < \int_a^{r^*} (\int_a^q G(p)d(p))dq$, sous l'hypothèse que $U' > 0$ et $U'' < 0$ et $U''' > 0$ et $\mathbb{E}(r|F) \geq \mathbb{E}(r|G)$. Cela veut dire que si $F\ DST\ G$ alors $\mathbb{E}(U(r|F)) > \mathbb{E}(U(r|G))$.

Plus l'ordre est grand, plus le nombre de portefeuilles dominés est grand. Ceci résulte du fait que si un portefeuille est efficient au sens du critère du troisième ordre, il est aussi efficient pour le deuxième et le premier ordre mais la réciproque n'est pas forcément vérifiée.

En fait, DSP, DSD, et DST partitionnent l'ensemble des actifs risqués en deux sous-ensembles : l'ensemble des portefeuilles efficients et celui des portefeuilles inefficaces.

La notion de dominance stochastique est moins restrictive que l'hypothèse quadratique pour les fonctions d'utilités [65]. Cette hypothèse est fondamentale pour l'approche Moyenne-Variance que nous allons aborder dans la section (3.3). Puisque le concept de DS est exhaustif, alors une mesure de risque, compatible avec DS, est préférée. Ceci est vrai pour les mesures LPM dont on parlera en sections (3.3).

3.2.3 La cohérence

Malheureusement, les mesures de risque les plus utilisées en pratique ne reflètent pas de façon correcte les préférences des investisseurs. Afin de surmonter ce problème, des mesures de risque cohérentes ont été proposées. ARTZNER et al. [4] ont défini quatre propriétés qu'une mesure de risque doit satisfaire pour qu'elle soit cohérente.

Définition 3.2.2 *Soit V un ensemble de variables aléatoires avec des valeurs réelles, une mesure de risque est une fonction à valeurs réelles comme ρ, telle que*

$$\rho : V \longrightarrow \mathbb{R}.$$

□

Soit ρ une mesure de risque, pour $v, v' \in V$, ρ est dite cohérente si elle est

1. **sous-additive** : $\rho(v + v') \leq \rho(v) + \rho(v')$.
   La mesure de risque d'une somme de deux portefeuilles est inférieure à la somme des mesures de risque de ces deux portefeuilles. Ce résultat est dû à la corrélation qui peut exister entre ces derniers.
   En effet, cette propriété reflète le gain de diversification.
2. **positivement homogène** : $\rho(\lambda v) = \lambda\rho(v) \quad \forall\lambda \geq 0$.
   La propriété d'homogénéité positive est *un cas limite* de la propriété de sous-additivité qui représente l'absence de diversification.
3. **invariante par translation** : $\rho(v + c) = \rho(v) - c, \quad \forall c \in \mathbb{R}$.
   La propriété d'invariance par translation signifie que l'addition (ou la soustraction) d'un montant initial sûr (i.e., sans risque) 'c' au portefeuille initial décroît (accroît) simplement la mesure du risque ρ par c.
   On constate que l'addition d'un montant initial égal au $\rho(v)$ réduit le risque à 0 soit $\rho(v + \rho(v)) = \rho(v) - \rho(v) = 0$.
4. **monotone** : Si $v \leq v'$ alors $\rho(v) \geq \rho(v')$.
   D'après cette propriété, si un portefeuille a un capital requis supérieur à celui d'un autre, alors le risque associé au portefeuille dont le capital requis est le plus élevé est inférieur à celui de l'autre.

La sous-additivité favorise la diversification. La sous-additivité et la positivement homogénéité garantissent la convexité de la mesure de risque, ce qui est un avantage en gestion de portefeuille [34].

3.3 Les mesures classiques de risque

Dans cette section nous allons aborder la quantification de risque. Pour cela nous allons exposer certaines mesures de risque.

3.3.1 La variance

Une mesure classique de risque est la variance, Var, et sa racine carrée dite *écart-type*, σ. Il est bien connu que *Markowitz* est la première personne qui a utilisé la variance comme la mesure de risque ([112, 113]). Markowitz a bien marqué le début de la théorie moderne de portefeuille où pour la première fois, le problème de choix de portefeuille a été clairement mis au point et résolu. Markowitz a proposé un modèle basé sur deux notions de probabilité, l'espérance mathématique des rendements et la variance entre les rendements [29]. Le modèle de Markowitz essaie de partager la richesse parmi les actifs tout en réduisant la variance de portefeuille.

Définition 3.3.1 *Soit f la fonction de densité d'une variable aléatoire r, nous définissons la variance de r par*

$$Var(r) = \sigma^2(r) := \int_{-\infty}^{\infty} (r - \mathbb{E}(r))^2 f(r) dr \tag{3.1}$$

où $\mathbb{E}$ est l'opérateur d'espérance mathématique et r est une variable continue. Pour le cas où r est une variable discrète, la variance de r se définit par

$$Var(r) = \sigma^2(r) := \sum_{r} (r - \mathbb{E}(r))^2 f(r) \tag{3.2}$$

où la somme est effectuée sur toutes les valeurs possibles de r. □

Si r est le rendement d'un portefeuille, alors la variance de rendement sera le carré de l'écart-type du rendement par rapport à l'espérance mathématique du rendement.

Supposons qu'une liste des rendements historiques soit disponible. La liste contient les rendements historiques de n actifs pendant m périodes. A partir de cette liste nous définissons le rendement moyen de chaque actif en utilisant la formule suivante

$$r_i := \frac{1}{m} \sum_{j=1}^{m} r_{ij}$$

où r_{ij} est le rendement de l'actif i en période j tel que $i = 1, \ldots, n$ et $j = 1, \ldots, m$.

Soient $x_1, x_2, \ldots, x_n$, les n actifs sous risque (risqués) qui forment les portefeuilles. De plus nous supposons que $x_i \geq 0$ et $\sum_{i=1}^{n} x_i = 1$ et les variables x_i correspondent aux rendements aléatoires $\mathbf{r} = (r_1, r_2, \ldots, r_n)^t$. Le rendement du portefeuille $\mathbf{x} = (x_1, x_2, \ldots, x_n)^t$ est $R = \mathbf{x}^t\mathbf{r}$ et le risque dédié au portefeuille est défini par $\sigma^2 = \mathbf{x}^t\mathbf{Q}\mathbf{x}$. $\mathbf{Q}$ est la matrice de Variance-Covariance dont l'élément (i, j) est calculé par

$$\sigma_{i,j} := \frac{1}{m} \sum_{k=1}^{m} (r_{ik} - r_i)(r_{jk} - r_j). \tag{3.3}$$

En fait, $\sigma_{i,j}$ présente la covariance entre les actifs i et j. Pour le cas où $i = j$, la formule (4.1) définit la variance de l'actif i.

Markowitz a démontré que pour un investisseur rationnel qui cherche à maximiser l'espérance mathématique de la fonction d'utilité, le portefeuille efficient est celui qui a le plus haut rendement espéré pour un niveau fixé de risque et celui qui a le moindre de risque pour un niveau fixé de rendement.

Définition 3.3.2 *(le critère de Moyenne-Variance (MV))*
Les préférences d'un individu satisfont au critère moyenne-variance s'il existe une fonction $U(.,.)$ *définie sur* $\mathbb{R} \times \mathbb{R}_+$ *telle que l'individu préfère le rendement* r_1 *au rendement* r_2 *si et seulement si*

$$U(E(r_1), var(r_1)) > U(E(r_2), var(r_2)).$$

□

En général, nous supposons que U croissante par rapport à la première variable et décroissante par rapport à la seconde, afin d'exprimer l'amour de la richesse, et l'aversion pour le risque. Dans ce cas un individu choisira toujours un portefeuille MV-efficace, c'est-à-dire le portefeuille qui minimise la variance pour l'espérance de rendement donnée [24].

HANOCH et LEVY [41] ont démontré que le critère de MV est un critère fiable quelque soit la fonction d'utilité de l'individu si la distribution est la Gaussienne.

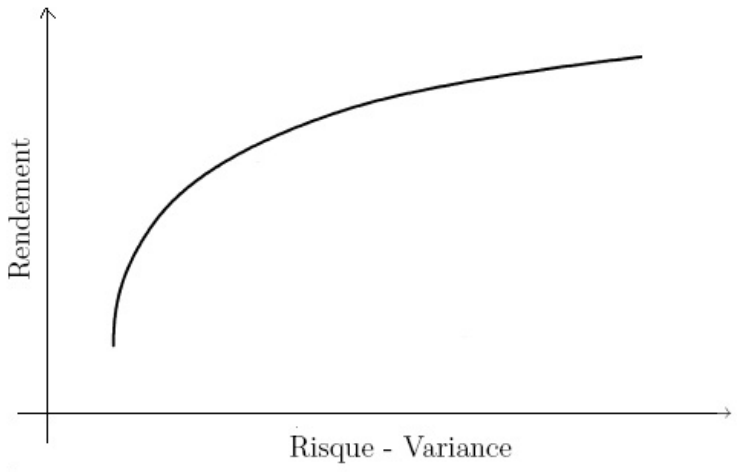


FIG. 3.1 – Exemple de la courbe des portefeuilles efficients

L'idée de Markowitz nous permet de tracer la courbe des portefeuilles efficients. Cette courbe est composée des portefeuilles qui attribuent le risque minimum (V) pour un gain fixé (R) ou

les portefeuilles qui ont le gain maximal pour un risque fixé (V) (Voir la Fig. 3.1). Pour tracer cette courbe nous pouvons résoudre le problème quadratique convexe suivant qui minimise le risque (x^tQx) pour un gain fixé (R) :

$$\begin{cases} \min \mathbf{x}^t\mathbf{Q}\mathbf{x} \\ s.c. \begin{cases} \sum_{j=1}^{n} r_j x_j = R, \\ \sum_{j=1}^{n} x_j = 1, \\ x_j \geq 0, \qquad j = 1, \ldots, n. \end{cases} \end{cases} \tag{3.4}$$

En pratique, le modèle suivant est très utilisé

$$\begin{cases} \min \lambda \mathbf{x}^t\mathbf{Q}\mathbf{x} - (1-\lambda) \sum_{j=1}^{n} r_j x_j \\ s.c. \begin{cases} \sum_{j=1}^{n} x_j = 1, \\ x_j \geq 0, \qquad j = 1, \ldots, n. \end{cases} \end{cases} \tag{3.5}$$

où λ est le paramètre d'aversion au risque et $0 \leq \lambda \leq 1$. $\lambda = 1$ correspond à l'individu purement risque-averse et $\lambda = 0$ correspond à celui qui n'a aucune peur de prendre le risque. De la même manière, si nous faisons varier λ entre 0 et 1, nous pourrons tracer la courbe des portefeuilles efficients.

L'utilisation de la variance comme mesure de risque a des avantages et des inconvénients. L'avantage de cette mesure de risque est sa simplicité. De plus, en utilisant la matrice de variance-covariance, nous pouvons construire un modèle de programmation quadratique convexe ([7],[29]) afin de calculer les portefeuilles efficients. Actuellement, un problème quadratique convexe se résout facilement même pour des dimensions très grandes comme 10000 [60]. Mais cela ne suffit pas car certaines contraintes du monde réel ne sont pas prises en compte dans le modèle de base. Nous parlerons de ces contraintes non prises en compte dans les prochains chapitres.

L'approche de Markowitz est plus connue sous le nom de l'approche Moyenne-Variance (MV). Cette approche connaît aussi certains inconvénients. D'abord, elle suppose que, soit les rendements suivent une distribution normale (ou multi-normale), soit la fonction d'utilité est quadratique [24]. Ces deux cas sont malheureusement assez peu réalistes. La deuxième critique est due au manque de sensibilité du modèle quant aux différences entre les gains et les pertes car l'approche MV pénalise de la même manière les gains ou les pertes s'éloignant trop de la moyenne [7]. L'approche MV n'est pas, en général, compatible avec la dominance stochastique [59]. Finalement, l'approche MV n'est pas cohérente car elle ne respecte pas les axiomes de monotonie et d'invariance par translation [34].

Après le travail de Markowitz, les modèles alternatifs ont été développés pour pallier aux défauts du modèle MV.

3.3.2 La mesure de risque de baisse

En 1959, Markowitz s'était rendu compte des limites de l'approche MV. Il a proposé la *semivariance*. La *semivariance* est une mesure de type dite *les mesures de risque de baisse*. Ce type de mesure se concentre essentiellement sur les pertes. Les mesures de risque de baisse sont aussi utiles quand les rendements suivent une distribution non-gaussienne [27].

La semivariance prend en compte seulement les écarts qui sont inférieurs à un *objectif* [27].

Mathématiquement, la mesure de semivariance est définie comme suit ([34],[65]) :

$$SV_T := \sum_r ([r - T]^-)^2 f(r) \tag{3.6}$$

où T est l'objectif et f est la fonction de densité de rendement aléatoire r. Markowitz suppose que T est égal à l'espérance de rendement, $\mathbb{E}(r)$, d'où vient le nom *semivariance*. Dans ce cas nous avons :

$$SV_R := \sum_r ([r - \mathbb{E}(r)]^-)^2 f(r). \tag{3.7}$$

La semivariance prend en compte seulement les écarts associés aux valeurs de r qui sont inférieures à l'espérance des rendements.

L'idée des mesures de risque de baisse est à l'origine du fait que les investisseurs sont favorables aux rentabilités supérieures à l'objectif [7]. Malgré l'apparence logique de ce type de mesures de risque, elles provoquent certaines critiques. Tous les investisseurs ne sont pas d'accord avec cette méthode, certains ne croient pas que cette méthode puisse mesurer correctement les risques. De plus, tous ne sont pas d'accord sur le même objectif. En effet, cette méthode est très *subjective* [108]. Enfin, la semivariance n'est pas une mesure de risque cohérente.

3.3.3 Le modèle de Mean-Absolute Deviation (MAD)

Konno et Yamazaki [62] ont mis au point un modèle afin de remplacer celui de MV par leur modèle de programmation linéaire. Considérons R_j comme une variable aléatoire qui représente le taux d'intérêt de l'actif j. Nous supposons que $(R_1, R_2, \ldots, R_n)$ est distribué sur un ensemble fini de points $(r_{1t}, r_{2t}, \ldots, r_{nt})$ où $t = 1, \ldots, T$ et que

$$\wp_t = Pr\{(R_1, R_2, \ldots, R_n) = (r_{1t}, r_{2t}, \ldots, r_{nt})\} \tag{3.8}$$

soient connus auparavant pour $t = 1, \ldots, T$. Nous utilisons les notations suivantes
- M = le capital,
- x_j = la portion du capital investi en actif j,
- α_j = la borne supérieure pour l'investissement en actif j,
- ω = le niveau de risque autorisé.

La déviation absolue $W(x)$ du portefeuille $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ se définit par

$$W[R(x)] = E[\mid R(x) - E[R(x)] \mid] = \sum_{t=1}^{T} \wp_t \mid \sum_{j=1}^{n} (r_{jt} - r_j)x_j \mid \tag{3.9}$$

où $r_j = \sum_{t=1}^{T} \wp_t r_{jt}$ est la valeur espérée de R_j.

Le modèle de Mean-Absolute Deviation (MAD) est comme suit

$$\max \left\{ \sum_{j=1}^{n} r_j x_j : W[\sum_{j=1}^{n} R_j x_j] \leq wM, \sum_{j=1}^{n} x_j = M, 0 \leq x_j \leq \alpha_j, j = 1, \ldots, n \right\}. \tag{3.10}$$

Ce modèle peut s'écrire sous la forme d'un modèle de programmation linéaire.

Konno et Yamazaki déclarent que le modèle MAD est plus crédible que MV ([33],[59],[125]). Leurs arguments sont les suivants :

1. Dans la formulation de MAD nous n'avons pas besoin de la matrice de Variance-Covariance.
2. MAD est un modèle de programmation linéaire alors que MV est un programme quadratique. Les programmes linéaires sont moins coûteux que ceux de quadratiques surtout pour les problèmes de grandes tailles.
3. Les portefeuilles efficients fournis par MAD ont moins d'actifs que ceux de MV. Cette propriété est un avantage quand il y a des coûts de transactions.
4. Le modèle MAD est compatible avec la dominance stochastique en deuxième ordre quelle que soit la distribution des rendements mais nous savons que, en général, le modèle MV n'est pas compatible avec la dominance stochastique.

SIMAAN [160] a publié un article sur les avantages et les inconvénients du modèle MAD. Il a démontré qu'en éliminant la matrice de variance-covariance le modèle MAD surestime les risques, par conséquent, il élimine certains profits.

3.3.4 Value at risk (VaR) et Value at Risk Conditionnel (CVaR)

Récemment, une mesure de risque est de plus en plus utilisée dans les établissements financiers ([34],[52],[65], [118]). Cette mesure de risque est nommée *Valeur exposée au risque* ou *risque potentiel de perte* ou bien plus connue sous le nom *Value at Risk* (VaR_β). VaR_β désigne la perte potentielle que nous nous autorisons sur un certain horizon T et pour un niveau de probabilité β donné. Plus précisément, pour un niveau de confiance $100\beta\%$ où $\beta \in (0, 1)$, nous définissons VaR_β comme la plus petite perte possible qui est inférieure ou égale à $100(1 - \beta)\%$ sur l'horizon T.

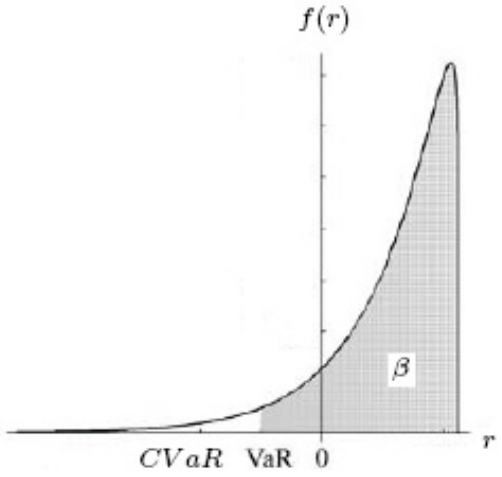


FIG. 3.2 – VaR et CVaR en $\beta\%$ de confiance et la fonction de densité f

Soit $x \in \mathbb{R}^n$, définissons $L(x) \in \mathbb{R}^n$ la variable aléatoire qui représente la perte. Soit $\Psi_L(x, \zeta)$ la fonction de répartition de $L(x)$, alors

$$\Psi_L(x, \zeta) = P(L(x) \leq \zeta).$$

Pour une variable x, Value at Risk en $100\beta\%$ de confiance se définit par

$$VaR_\beta(x) = inf\{\zeta | \Psi_L(x, \zeta) \geq \beta\}.$$

Une alternative à VaR est CVaR qui n'est pas aussi populaire que VaR. Pourtant, CVaR possède certaines propriétés qui la rendent plus logique que VaR ([34],[63]). CVaR est compatible avec la cohérence dont nous avons discuté [1]. Dans ce qui suit, nous présentons d'abord une définition mathématique pour CVaR.

Soit $T_\beta(x)$ la variable aléatoire qui correspond à la β-queue de la perte $L(x)$. Nous munissons la variable $T_\beta(x)$ par la distribution

$$\Psi_{T_\beta}(x, \zeta) = \begin{cases} 0 & : \zeta < VaR_\beta(x), \\ \frac{\Psi_L(x,\zeta)-\beta}{1-\beta} & : \zeta \geq VaR_\beta(x). \end{cases} \tag{3.11}$$

Pour une variable de décision x, le Value at Risk Conditionnel en $100\beta\%$ de confiance est l'espérance mathématiques de $T_\beta(x)$ en prenant en compte la fonction de distribution définie ci-dessus :

$$CVaR_\beta(x) = E(T_\beta(x)).$$

VaR est facile à utiliser à condition que la distribution des données soit gaussienne, mais nous savons bien que ce n'est pas toujours le cas en pratique.

CVaR est une mesure de risque cohérente ([34] et les références incluses) mais VaR, en général, ne satisfait pas les propriétés de cohérence car VaR n'est pas sous-additif. Ce n'est très apprécié en général, car cela signifie que la diversification n'est pas respectée par VaR. Dans le cas où les rendements ont une distribution gaussienne, VaR est sous-additif. Pour *certains cas* où la distribution est elliptique, VaR est sous-additif et cohérent ([34] et les références incluses).

3.4 La classification des mesures de risque

En général, nous pouvons répartir les différentes mesures de risque en deux groupes selon la perception que nous avons du risque [33]. Le premier groupe se compose des mesures de risque qui sont basées sur la notion de dispersion. Plus précisément, il existe un objectif, comme l'espérance de rendement, et nous mesurons le risque par les dispersions pondérées des résultats autour de l'objectif. Ces mesures de risque sont des mesures symétriques de risque. Ce sont les mesures qui pénalisent les écarts négatifs aussi bien que positifs. Deux des mesures les plus connues parmi les mesures symétriques sont le modèle MV de Markowitz et le modèle MAD de Konno et al. Le deuxième groupe est composé des mesures de risque qui considère le risque comme les écarts inférieurs à un objectif. Ce sont les mesures de baisse. L'objectif peut être défini avec consultation de l'investisseur (subjective) ou sans (objective). Ce type de mesure de risque se classifie comme les mesures asymétriques de risque. La semivariance proposée par Markowitz, Value at Risk (VaR) [118] et Conditional Value at Risk (CVaR) [148] sont comprises dans ce groupe.

3.4.1 Une approche générale pour la représentation des mesures de risque

BAWA [9] et FISHBURN [31] ont développé un *modèle* $\alpha - \tau$ afin de définir les mesures de baisse de façon générale. Leur modèle s'appelle *Lower Partial Moments (LPM)*. Soit r la variable aléatoire de rendement des actifs et f la fonction de densité de la distribution des rendements.
Définition : LPM de l'ordre α et l'objectif τ se défini comme la suite

$$LPM_{\tau,\alpha}(r) := \sum_{r \leq \tau}([\tau - r]^{\alpha}) f(r) = E\{(max[0, \tau - r])^{\alpha}\} \quad , \alpha > 0. \tag{3.12}$$

L'introduction de LPM était un progrès considérable dans le domaine de risque car il fournit une représentation générale de risque.

Il existe une formulation encore plus générale. Bernell STONE [163] était la première personne qui a défini une mesure de risque à trois paramètres que nous allons nommer *Stone's*

Risk Measure ou plus simplement *SRM* :

$$SRM_{\tau,\alpha}(r) := \sum_{r\leq\gamma}([\tau - r]^{\alpha})f(r) \quad , \alpha \geq 0 \tag{3.13}$$

où τ est l'objectif par rapport auquel les écarts sont mesurés. α est le paramètre mesurant l'impact des écarts. γ est le paramètre avec lequel nous limitons le domaine où le risque est mesuré.

Nous pouvons rendre la plupart des mesures de risque sous la forme de SRM, que ce soit les mesures de risque symétrique ou asymétrique.

Les mesures symmétriques de risque

Nous considérons deux mesures de risque bien connues, c'est-à-dire MV et MAD. Puisque les rendements supérieurs à l'objectif sont aussi considérés comme faisant partie du risque alors les valeurs associées au risque peuvent fluctuer entre $(-\infty, +\infty)$. Nous posons $\gamma = +\infty$.

MV Le modèle MV de Markowitz est un cas particulier de SRM pour $\tau := E(r) = \overline{r}$ et $\alpha = 2$, dans ce cas

$$\sigma^2 \equiv SRM_{\overline{r},2}(r) := \sum_{r}([\overline{r} - r]^2)f(r) = E[(\overline{r} - r)^2]. \tag{3.14}$$

Dans le cas plus général où l'objectif n'est pas l'espérance de rendement du portefeuille, la représentation devient

$$SRM_{\tau,2}(r) := \sum_{r}([\tau - r]^2)f(r) = E[(\tau - r)^2]. \tag{3.15}$$

MAD Soient $\alpha = 1$ et $\tau := E(r) = \overline{r}$, alors la mesure de risque MAD se formule

$$MAD \equiv SRM_{\overline{r},1}(r) := \sum_{r} \mid \overline{r} - r \mid^1 f(r) = E[\mid \overline{r} - r \mid^1]. \tag{3.16}$$

Les mesures asymmétriques de risque

Nous pouvons démontrer que toutes les mesures asymétriques peuvent se voir comme un cas particulier de SRM [33].

Semivariance : Soit $\alpha = 2$ et $\gamma = \tau := E(r) = \overline{r}$ alors

$$\sigma^{-2} \equiv SRM_{\overline{r},2}(r) := \sum_{r\leq\overline{r}}([\overline{r} - r]^2)f(r) = E\{(max[0, \overline{r} - r])^2\}. \tag{3.17}$$

Risque de baisse : Soit $\gamma = \tau$ et $\alpha = 2$ alors

$$SRM_{\tau,2}(r) := \sum_{r\leq\tau}([\tau - r]^2)f(r) = E\{(max[0, \tau - r])^2\}. \tag{3.18}$$

Value at Risk : Soit $VaR_{\beta}(r) = \theta$ alors il nous suffit de choisir $\alpha = 0$ et $\gamma = \tau = \theta$, alors

$$SRM_{\theta,0}(r) := \sum_{r\leq\theta}([\theta - r]^0)f(r) = E\{(max[0, \theta - r])^0\} = 1 - \beta. \tag{3.19}$$

Chapitre 4

Gestion de portefeuille sous les contraintes de seuil, de seuil d'achat et de cardinalité

Résumé *Dans ce chapitre nous proposons une nouvelle approche continue basée sur la programmation DC et DCA pour la résolution des problèmes de gestion de portefeuille dont l'ensemble des solutions admissibles est non convexe. Ces problèmes sont des extensions du modèle de Markowitz pour les cas plus réalistes. Les extensions consistent à ajouter des contraintes de seuil d'achat, des contraintes de seuil et celles de cardinalité. La présence de ces contraintes rend le problème non convexe, par conséquent très difficile à résoudre par les méthodes classiques. La plupart des méthodes existantes pour résoudre ce problème sont basées sur des heuristiques. Dans ce travail, nous traitons ce problème par DCA via la pénalité exacte en nous basant sur les décompositions DC appropriées. Les simulations numériques sur plusieurs jeux de données empiriques montrent l'efficacité de notre approche par rapport aux méthodes standards.*

4.1 Introduction

En ce qui concerne le problème de choix de portefeuille, étant donnés un nombre d'actifs financiers et une somme à investir, nous devons choisir quelques actifs afin d'investir le capital [29]. Pour résoudre ce problème, Markowitz a présenté un modèle en 1952 [112], le modèle moyenne-variance (MV) c'est un modèle classique en planning de portefeuille. Markowitz a démontré que l'investisseur rationnel qui veut maximiser l'espérance d'utilité, choisit le portefeuille qui est optimal selon l'espérance de rendement et la variance. Il a défini un portefeuille non-dominé comme le portefeuille efficient s'il atteint le plus haut niveau de rendement espéré pour un niveau fixé de risque ou celui qui a le plus faible niveau de risque pour un niveau prévu de rendement. Afin de sélectionner tous les portefeuilles efficients parmi les autres, il nous faut résoudre un problème quadratique. Ce problème contient un paramètre, soit celui du niveau de risque autorisé, soit celui du niveau attendu de

rendement. Après avoir trouvé tous les portefeuilles efficients, nous pouvons tracer la courbe des portefeuilles efficients (la *frontière efficiente*), qui est une courbe lisse et croissante.

Le modèle MV est un modèle général qui ne couvre pas tous les besoins du monde réel ([16],[29],[51]). Parfois, dans les situations réelles, nous devons respecter certaines contraintes d'investissements. Par exemple :

(a) Contraintes de seuil d'achat : Ce sont des contraintes qui nous empêchent d'avoir de très petits investissements dans les actifs [7].

(b) Contraintes de cardinalité : Ce sont des contraintes qui nous obligent à choisir un nombre limité d'actifs parmi les actifs disponibles [29].

(c) Contraintes de seuil : Ces contraintes sont les généralisations des contraintes de seuil d'achat où les ventes-à-découverte sont autorisées [7].

La présence de ces contraintes rend le problème non convexe et par conséquent très difficile à résoudre. En fait, tous les cas peuvent se formuler dans un contexte de programmation en variables mixtes. Les contraintes de seuil d'achat et de cardinalité ont été largement étudiées ([16],[29],[51]). La plupart des méthodes utilisées sont basées sur des heuristiques. Récemment, une méthode appelée *Direct* a été utilisée pour résoudre les problèmes qui contiennent les contraintes de seuil d'achat ([7],[8]). Cette méthode souffre de manque de preuve de convergence. Nous pouvons seulement fixer une limite sur le nombre d'itération de l'algorithme. La méthode *Direct* peut être également utilisée pour résoudre le problème sous les contraintes de seuil. Les contraintes de seuil n'ont pas été bien étudiées dans la littérature financière. La cause est peut-être la non-covexité du modèle et plus particulièrement la présence des contraintes de complémentarité (voir [55]).

Le but de ce chapitre est de présenter une méthode basée sur la programmation DC et DCA afin de résoudre ces problèmes. La méthode consiste à reformuler les modèles dans le contexte de programmation DC via la pénalité exacte en nous basant sur les décompositions DC appropriées. Plus particulièrement, nous allons introduire une nouvelle fonction de pénalité qui prend en considération, non seulement les variables binaires mais aussi les contraintes de complémentarité. Pour chaque contrainte (cardinalité, seuil d'achat ou seuil), le modèle généralisé est résolu par un algorithme par séparation et évaluation (SE) et/ou une méthode combiné de SE et DCA.

Le reste du chapitre est organisé de façon suivante : dans la deuxième section, nous présentons la description et la formulation du modèle MV. La section 4.3 donne une description du modèle MV en prenant en compte les contraintes de seuil d'achat. La section 4.4 décrit la résolution du modèle MV en présence des contraintes de cardinalité. Un cas plus général du modèle MV avec contraintes de seuil d'achat, c'est-à-dire les contraintes de seuil, est étudié en section 4.5. Nous terminons ce chapitre par une discussion sur les résultats et une conclusion.

4.2 Le modèle moyenne-variance (MV)

Dans le premier temps, en introduisant les notations que nous utiliserons au cours ce chapitre, nous rappelons le modèle MV de Markowitz.

Supposons qu'une liste des rendements historiques soit disponible. La liste contient les rendements historiques de n actifs pendant m périodes. A partir de cette liste nous définissons le rendement moyen de chaque actif en utilisant la formule suivante

$$r_i := \frac{1}{m}\sum_{j=1}^{m} r_{ij}$$

où r_{ij} est le rendement de l'actif i dans le période j tel que $i = 1, \ldots, n$ et $j = 1, \ldots, m$.

Soient $x_1, x_2, \ldots, x_n$, les pondérations de n actifs sous risque (risqué) qui forment les portefeuilles. En fait, x_i est la variable de décision qui représente la proportion du capital investi en actif i. De plus nous supposons que $x_i \geq 0$, $\sum_{i=1}^{n} x_i = 1$ et les variables x_i correspondent aux rendements $\mathbf{r} = (r_1, r_2, \ldots, r_n)^t$. Le rendement du portefeuille $\mathbf{x} = (x_1, x_2, \ldots, x_n)^t$ est $R = \mathbf{x}^t\mathbf{r}$ et le risque dédié au portefeuille est défini par $\sigma^2 = \mathbf{x}^t\mathbf{Q}\mathbf{x}$. $\mathbf{Q}$ est la matrice de Variance-Covariance dont l'élément (i, j) est calculé par

$$\sigma_{i,j} := \frac{1}{m}\sum_{k=1}^{m}(r_{ik} - r_i)(r_{jk} - r_j). \tag{4.1}$$

En fait, $\sigma_{i,j}$ représente la covariance entre les actifs i et j. Pour le cas où $i = j$, la formule (4.1) définit la variance de l'actif i. En utilisant ces notations, le modèle MV de Markowitz se résume (voir [7]) :

$$\begin{cases} \min \mathbf{x}^t\mathbf{Q}\mathbf{x} \\ s.c. \begin{cases} \sum\limits_{j=1}^{n} r_j x_j = R, \\ \sum\limits_{j=1}^{n} x_j = 1, \\ x_j \geq 0, \qquad j = 1, \ldots, n. \end{cases} \end{cases} \tag{4.2}$$

Dans ce modèle, R, le rendement espéré du portefeuille, est un paramètre fixé par l'investisseur. Puisque $0 \leq x_i \leq 1$, alors R peut varier entre '$\min_j r_j$' et '$\max_j r_j$'. La contrainte $\sum\limits_{j=1}^{n} r_j x_j = R$ peut être remplacée par $\sum\limits_{j=1}^{n} r_j x_j \geq R$, qui présente la préférence de l'investisseur à avoir un gain supérieur ou égal à R.

Le modèle MV est un modèle de programmation quadratique qui peut être résolu de façon très efficace. Pourtant, si nous ajoutons les contraintes de seuil d'achat ou de cardinalité le nouveau modèle ne sera plus facile à résoudre. Les sections suivantes sont consacrées à reformuler et à résoudre les nouveaux modèles par DCA.

4.3 Contraintes de seuil d'achat

Afin d'introduire les contraintes de seuil d'achat dans le modèle MV, nous définissons a_j, b_j comme les bornes inférieure et supérieure sur la proportion du capital investi dans l'actif j, respectivement. Il faut vérifier $a_j > 0$. Le modèle MV en présence des contraintes de seuil d'achat est donc :

$$
\begin{aligned}
(P1): \quad & \\
& \min \ \mathbf{x}^t \mathbf{Q} \mathbf{x} \\
& s.c. \\
& \sum_{j=1}^{n} r_j x_j = R, \\
& \sum_{j=1}^{n} x_j = 1, \\
& x_j \in \{0\} \cup [a_j, b_j] \qquad : j = 1, \ldots, n. \qquad (4.3)
\end{aligned}
$$

A cause des contraintes 4.3, le problème $(P1)$ est non convexe.

4.3.1 Programmation DC et DCA pour la résolution du problème

4.3.1.1 Reformulation

Le problème $(P1)$ peut être reformulé en tant qu'un modèle de programmation binaire. Soit $z_j, j = 1, \ldots, n$, des variables binaires telles que :

$$
z_j = \begin{cases} 1, & x_j \in [a_j, b_j], \\ 0, & sinon. \end{cases} \qquad (4.4)
$$

Avec l'introduction des nouvelles variables, le modèle $(P1)$ se met sous la forme suivante

$$
\begin{aligned}
(P2): \quad & \\
& \min \ \mathbf{x}^t \mathbf{Q} \mathbf{x} \\
& s.c. \\
& \sum_{j=1}^{n} r_j x_j = R, \\
& \sum_{j=1}^{n} x_j = 1, \\
& a_j z_j \leq x_j \leq b_j z_j \qquad : j = 1, \ldots, n, \\
& z_j \in \{0, 1\} \qquad : j = 1, \ldots, n.
\end{aligned}
$$

Soit $\mathcal{A}$ le polyèdre convexe borné et non vide défini par :

$$\mathcal{A} := \left\{ (\mathbf{x}, \mathbf{z}) \in \mathbb{R}^n \times [0,1]^n : \sum_{j=1}^{n} r_j x_j = R, \sum_{j=1}^{n} x_j = 1, a_j z_j \leq x_j \leq b_j z_j, j = 1, \ldots, n \right\}.$$

Alors $(P2)$ s'écrit

$$\min \left\{ \mathbf{x}^t \mathbf{Q} \mathbf{x} : (\mathbf{x}, \mathbf{z}) \in \mathcal{A}, z_j \in \{0,1\} : j = 1, \ldots, n \right\}. \tag{4.5}$$

Considérons la fonction p définie par

$$p(\mathbf{x}, \mathbf{z}) := \sum_{i=1}^{n} z_i (1 - z_i). \tag{4.6}$$

Clairement, la fonction p est concave et finie sur $\mathcal{A}$, de plus $p(\mathbf{x}, \mathbf{z}) \geq 0$ pour tout $(\mathbf{x}, \mathbf{z}) \in \mathcal{A}$, et

$$\{(\mathbf{x}, \mathbf{z}) \in \mathcal{A} : \mathbf{z} \in \{0,1\}^n\} = \{(\mathbf{x}, \mathbf{z}) \in \mathcal{A} : p(\mathbf{x}, \mathbf{z}) \leq 0\}.$$

Par conséquent (4.5) peut être réécrite

$$\min \left\{ \mathbf{x}^t \mathbf{Q} \mathbf{x} : (\mathbf{x}, \mathbf{z}) \in \mathcal{A}, p(\mathbf{x}, \mathbf{z}) \leq 0 \right\}. \tag{4.7}$$

A partir du théorème 4.1, ci-dessous, nous obtenons, pour un nombre positif et suffisamment grand comme t (tel que $t \geq t_0$), un problème de minimisation convexe-concave qui est équivalent à $(P2)$:

$$(P2 - DC) : \quad \min \left\{ \mathbf{x}^t \mathbf{Q} \mathbf{x} + tp(\mathbf{x}, \mathbf{z}) : (\mathbf{x}, \mathbf{z}) \in \mathcal{A} \right\}. \tag{4.8}$$

Théorème 4.1 *([105]) Soient $\mathbb{K}$ un polyèdre convexe borné et non vide, f une fonction finie convexe sur $\mathbb{K}$ et p une fonction finie concave non négative sur $\mathbb{K}$. Il existe $t_0 \geq 0$ tel que pour tout $t \geq t_0$, deux problèmes ci-dessous sont équivalents :*

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

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

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

Il est clair que $\mathcal{A}$ est un polyèdre convexe, non vide et borné dans $\mathbb{R}^n \times \mathbb{R}^n$. Alors, $(P2)$ peut s'exprimer sous la formule de $(P2 - DC)$.

4.3.1.2 Résolution de $(P2-DC)$ par DCA

D'abord, on constate que $(P2-DC)$ est un problème de minimisation d'une fonction qui est convexe par rapport à $\mathbf{x}$ et concave par rapport à $\mathbf{z}$, alors la fonction objectif du modèle $(P2-DC)$ est une fonction DC.

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

$$g(\mathbf{x},\mathbf{z})=\mathbf{x}^t\mathbf{Q}\mathbf{x}+\chi_{\mathcal{A}}(\mathbf{x},\mathbf{z}) \quad \text{et} \quad h(\mathbf{x},\mathbf{z})=-t\sum_{i=1}^{n} z_i(1-z_i). \tag{4.9}$$

Par conséquent, g et h sont des fonctions convexes et le problème $(P2-DC)$ est un problème de programmation DC sous la forme

$$\min\{g(\mathbf{x},\mathbf{z})-h(\mathbf{x},\mathbf{z}) \ : \ (\mathbf{x},\mathbf{z})\in\mathbb{R}^n\times\mathbb{R}^n\}. \tag{4.10}$$

Selon la description générale de DCA, la résolution de $(P2-DC)$ via la formulation (4.10) par DCA consiste en la détermination de deux suites

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

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

$$(\mathbf{u},\mathbf{v})\in\partial h(\mathbf{x},\mathbf{z}) \Longleftrightarrow \mathbf{u}=\mathbf{0}, \quad \mathbf{v}=t(2\mathbf{z}-\mathbf{1}). \tag{4.11}$$

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

$$\min\left\{\mathbf{x}^t\mathbf{Q}\mathbf{x}-\langle(\mathbf{u}^k,\mathbf{v}^k),(\mathbf{x},\mathbf{z})\rangle : (\mathbf{x},\mathbf{z})\in\mathcal{A}\right\}. \tag{4.12}$$

Algorithme 4.1 *Schéma de DCA*

Initialisation :
- *Choisir* $(\mathbf{x}^0,\mathbf{z}^0)\in\mathbb{R}^n\times\mathbb{R}^n$ *et* $k=0$.
- *Choisir la tolérance* ϵ *positive suffisamment petite.*

Répéter
- *Calculer* $(\mathbf{u}^k,\mathbf{v}^k)\in\partial h(\mathbf{x}^k,\mathbf{z}^k)$ *via (4.11).*
- *Calculer* $(\mathbf{x}^{k+1},\mathbf{z}^{k+1})\in\partial g^*(\mathbf{u}^k,\mathbf{v}^k)$ *en résolvant le programme quadratique (4.12).*
- $k+1\leftarrow k$.

Jusqu'à $\left\|(\mathbf{x}^{k+1},\mathbf{z}^{k+1})-(\mathbf{x}^k,\mathbf{z}^k)\right\|\leq\epsilon$.

La convergence de l'algorithme 4.1 est donnée par le prochain théorème ([91, 94, 139]).

Théorème 4.2 *(Propriétés de la convergence de l'algorithme 4.1)*
(i) L'algorithme 4.1 génère une suite $\{(\mathbf{x}^k,\mathbf{z}^k)\}$ *tel que la suite* $\{(\mathbf{x}^k)^t\mathbf{Q}\mathbf{x}^k+tp(\mathbf{x}^k,\mathbf{z}^k)\}$ *est décroissante.*

(ii) Il existe un nombre non négatif t tel que pour chaque $t \geq t_0$ la suite $\{p(\mathbf{x}^k, \mathbf{z}^k)\}$ est décroissante. En particulier, si $(\mathbf{x}^r, \mathbf{z}^r)$ est une solution admissible de $(P2)$ alors $(\mathbf{x}^k, \mathbf{z}^k)$, pour tout $k \geq r$, est admissible également.
(iii) DCA a un taux de convergence linéaire pour le problème $(P2 - DC)$.
(iv) La suite $\{(\mathbf{x}^k, \mathbf{z}^k)\}$ converge à $(\mathbf{x}^*, \mathbf{z}^*)$ où le point $(\mathbf{x}^*, \mathbf{z}^*)$ est un point critique du problème $(P2 - DC)$.

Preuve : (i), (iii) et (iv) sont des conséquences immédiates du théorème de la convergence de DCA pour une programmation DC générale (voir [87, 94, 139, 140]).
(ii) Soit $\mathcal{V}(\mathcal{A})$ l'ensemble des sommets de $\mathcal{A}$. Si $\mathcal{V}(\mathcal{A})$ est contenu dans l'ensemble des solutions admissibles de $(P2)$ alors l'affirmation est triviale avec $t_0 = 0$. Sinon, posons

$$\xi := min\{p(\mathbf{x}', \mathbf{z}') - p(\mathbf{x}, \mathbf{z}) : ((\mathbf{x}, \mathbf{z}), (\mathbf{x}', \mathbf{z}')) \in \mathcal{V}(\mathcal{A}) \times \mathcal{V}(\mathcal{A}), p(\mathbf{x}', \mathbf{z}') > p(\mathbf{x}, \mathbf{z})\},$$

$$\eta := max\{\mathbf{x}'^t\mathbf{Q}\mathbf{x}' - \mathbf{x}^t\mathbf{Q}\mathbf{x} : ((\mathbf{x}, \mathbf{z}), (\mathbf{x}', \mathbf{z}')) \in \mathcal{V}(\mathcal{A}) \times \mathcal{V}(\mathcal{A})\}$$

donc $0 < \xi < +\infty$ et $0 \leq \eta < +\infty$ car l'ensemble $\mathcal{V}(\mathcal{A})$ est fini. Considérons maintenant le nombre non négatif t_0 défini par $t_0 := \frac{\eta}{\xi}$ et $t > t_0$. Soit $\{(\mathbf{x}^k, \mathbf{z}^k)\}$ un ensemble généré par **Algorithme 4.1** appliqué au $(P2 - DC)$ à partir de cette valeur t. Supposons qu'il existe $r \geq 1$ tel que $p(\mathbf{x}^{r+1}, \mathbf{z}^{r+1}) > p(\mathbf{x}^r, \mathbf{z}^r)$. Puisque $t > t_0$ alors

$$t[p(\mathbf{x}^{r+1}, \mathbf{z}^{r+1}) - p(\mathbf{x}^r, \mathbf{z}^r)] > t_0[p(\mathbf{x}^{r+1}, \mathbf{z}^{r+1}) - p(\mathbf{x}^r, \mathbf{z}^r)]$$

ou

$$t[p(\mathbf{x}^{r+1}, \mathbf{z}^{r+1}) - p(\mathbf{x}^r, \mathbf{z}^r)] > \frac{\eta}{\xi}[p(\mathbf{x}^{r+1}, \mathbf{z}^{r+1}) - p(\mathbf{x}^r, \mathbf{z}^r)].$$

Par la définition de ξ,

$$\xi \geq p(\mathbf{x}^{r+1}, \mathbf{z}^{r+1}) - p(\mathbf{x}^r, \mathbf{z}^r),$$

alors

$$t[p(\mathbf{x}^{r+1}, \mathbf{z}^{r+1}) - p(\mathbf{x}^r, \mathbf{z}^r)] > \eta$$

en prenant en compte la définition de η, nous avons

$$t[p(\mathbf{x}^{r+1}, \mathbf{z}^{r+1}) - p(\mathbf{x}^r, \mathbf{z}^r)] > \mathbf{x}^{r,t}\mathbf{Q}\mathbf{x}^r - \mathbf{x}^{r+1,t}\mathbf{Q}\mathbf{x}^{r+1}$$

i.e.

$$\mathbf{x}^{r+1,t}\mathbf{Q}\mathbf{x}^{r+1} + tp(\mathbf{x}^{r+1}, \mathbf{z}^{r+1}) \geq \mathbf{x}^{r,t}\mathbf{Q}\mathbf{x}^r + tp(\mathbf{x}^r, \mathbf{z}^r),$$

ce qui contredit la décroissance de la suite $\mathbf{x}^t\mathbf{Q}\mathbf{x} + tp(\mathbf{x}, \mathbf{z})$. □

4.3.1.3 Résolution globale

Pour trouver la solution globale du problème de choix de portefeuille sous les contraintes de seuil d'achat et ainsi afin de comparer les résultats obtenus par DCA, nous avons utilisé deux algorithmes par Séparation et Évaluation (SE) afin de résoudre le problème.

Le premier algorithme par SE s'applique au modèle $(P1)$ de façon que pour calculer la borne inférieure les contraintes de seuil d'achat sont relaxées, autrement dit, les contraintes 4.3 sont remplacées par $x_j \in [0, b_j], j = 1, \ldots, n$. Le résultat de ce remplacement est un modèle de programmation quadratique qui se résout facilement. La borne supérieure est mise à jour si l'on trouve une meilleure solution pour le modèle $(P1)$. La séparation est effectuée sur une variable comme $x_j \in]0, a_j[$ pour que nous ayons soit $x_j = 0$, soit $a_j \leq x_j \leq b_j$.

Le deuxième algorithme par SE est utilisé sur le modèle $(P2)$, où la borne inférieure est calculée en relaxant les contraintes binaires par celles linéaires, autrement dit les contraintes $z_j \in \{0, 1\}$ sont remplacées par $0 \leq z_j \leq 1$. La borne supérieure est mise à jour si nous trouvons une meilleure solution pour le modèle $(P2)$. La séparation est effectuée sur la variable fractionnelle z_j telle que soit $z_j = 0$, soit $z_j = 1$.

4.3.1.4 Expériences numériques

Les algorithmes ont été codés avec le langage C++ et exécutés sur un ordinateur Pentium de 1.6GHz, 512Mo RAM. Pour résoudre les programmes quadratiques, nous avons utilisé le logiciel CPLEX en version 9.1. Les tests ont été faits sur deux jeux de données qui avaient été utilisés dans les articles publiés par différents chercheurs ([16], [29], [51], [154]). Les données correspondent aux prix hebdomadaires de deux différents indices de Mars 1992 à Septembre 1997. Les indices sont de Dax 100 en Allemagne et de Nikkei 225 au Japon. Il y a 85 actifs pour le premier jeu de données et 225 actifs pour le deuxième. Les bornes inférieure et supérieure sur les investissements sont 0.05 et 1.0, respectivement. Le paramètre de pénalité, t, est égal à 0.01 pour le premier jeu de données et 0.02 pour le deuxième. $\epsilon = 10^{-7}$ et les tests ont été effectués pour différentes valeurs de R.

Recherche d'un bon point initial pour DCA

Une question importante qui se pose est de choisir un bon point initial pour la méthode DCA. L'approche que nous avons adoptée consiste d'abord en la résolution du problème relaxé du modèle $(P2)$. Ensuite, certaines modifications ont été effectuées sur la solution du problème relaxé. Le processus de choix du point initial se résume ainsi :

1. **Résoudre le problème relaxé :** Nous résolvons le problème relaxé du $(P2)$ afin de trouver $(\widetilde{\mathbf{x}}, \widetilde{\mathbf{z}})$.
2. **Trouver une solution entière :** Si $\widetilde{z}_j$ est non nul, nous l'arrondissons à 1 pour obtenir $(\widetilde{\mathbf{x}}, \widehat{\mathbf{z}})$ de $(\widetilde{\mathbf{x}}, \widetilde{\mathbf{z}})$.

En général, le nouveau point n'est plus une solution admissible du modèle $(P2)$ ni pour le modèle $(P2-DC)$, mais à partir de cette solution DCA trouve une solution admissible juste après une seule itération.

En effet, afin de trouver un bon point initial, nous avons tenté plusieurs choix :

- le point fourni par le processus expliqué ci-dessus ;
- la solution optimale du modèle relaxé de $(P2)$;
- la solution optimale du problème suivant :

$$\min\left\{\sum_{j=1}^{n} z_j(1-z_j) : (\mathbf{x},\mathbf{z}) \in \mathcal{A}\right\}.$$

Le premier choix donne les meilleurs résultats.

Les tableaux 4.1 et 4.2, présentent les résultats. Dans ces tableaux, les valeurs optimales (Val. Opt.) fournies par DCA, par le premier algorithme SE (SE-1) et/ou par le deuxième algorithme SE (SE-2) ont été présentées. Ces tableaux montrent aussi le nombre d'itération (iter.) et le temps de résolution (CPU) en secondes.

Les résultats montrent l'efficacité et supériorité de DCA par rapport aux autres algorithmes. Car DCA donne des solutions de haute précision dans 2, 3 ou 4 itérations et en une ou deux secondes.

4.4 Contraintes de cardinalité

Nous allons étudier un cas général du modèle de Markowitz où nous acceptons que l'investisseur attend un rendement futur supérieur ou égale à R. Il existe des coûts de transaction et la vente est autorisée. Les coûts acceptés que l'investisseur doit payer au cours de ses transactions sont proportionnels aux valeurs d'achat et de vente [114]. Nous acceptons que l'investisseur soit actuellement dans une position précise et prêt à payer les coûts de transaction afin d'arriver à un portefeuille benchmark. De plus, le but de ce travail insiste essentiellement sur la présence d'une contrainte qui rend le modèle plus réaliste. Cette contrainte est nommée, dans la littérature financière, *contrainte de cardinalité*. Elle a pour l'objectif de limiter le nombre d'actifs composant le portefeuille optimal. Afin d'introduire toutes les notions qui viennent d'être citées dans le modèle MV, nous acceptons les notations supplémentaires suivantes, soient :

- $\mathbf{c}_b, \mathbf{c}_s \in \mathbb{R}^n$ sont les vecteurs qui représentent les coûts de transaction pour les achats et pour les ventes, respectivement ;
- $\mathbf{x}_b, \mathbf{x}_s$ correspondent aux variables d'achat et de vente ;
- $\mathbf{p} \in \mathbb{R}^n$ est le vecteur qui représente la position actuelle de l'investisseur ;
- $\overline{\mathbf{x}} \in \mathbb{R}^n$ est le vecteur du portefeuille benchmark ;
- $\mathbf{z}$ est le vecteur des variables binaires, $z_i = 1$ montre que l'actif i est inclus dans le portefeuille et 0 sinon ;

	DCA			Séparation et Évaluation				
R	Val. Opt.	iter	CPU	Val. Opt.	iter(SE1)	CPU(SE1)	iter(SE2)	CPU(SE2)
0.0001	0.000186	2	0.235	0.000174	1348	147.953	1018	97.734
0.0002	0.000189	2	0.234	0.000170	718	77.343	559	54.907
0.0003	0.000193	2	0.218	0.000167	491	53.328	373	36.688
0.0004	0.000182	3	0.266	0.000164	549	59.313	406	39.078
0.0005	0.000174	3	0.266	0.000162	671	72.625	519	49.750
0.0006	0.000173	4	0.312	0.000159	788	86.500	579	55.344
0.0007	0.000170	4	0.313	0.000158	1475	158.547	1161	111.750
0.0008	0.000167	3	0.266	0.000156	1648	175.828	959	91.937
0.0009	0.000167	4	0.313	0.000154	1838	194.860	1004	96.422
0.001	0.000167	4	0.312	0.000153	1980	209.610	1207	114.563
0.002	0.000156	2	0.219	0.000141	204	22.062	161	15.359
0.003	0.000159	2	0.234	0.000147	140	15.875	126	12.266
0.004	0.000207	2	0.203	0.000170	129	14.406	98	9.766

TAB. 4.1 – La performance de l'algorithme pour le premier jeu de données en utilisant deux algorithmes par SE

	DCA			Séparation et Évaluation (SE2)		
R	Val. Opt.	iter	CPU	Val. Opt.	iter	CPU
0.00001	0.000306	2	1.594	0.000305	12	10.953
0.00002	0.000306	2	1.609	0.000305	13	10.656
0.00003	0.000306	2	1.594	0.000305	12	10.516
0.00004	0.000306	2	1.625	0.000305	12	11.703
0.00005	0.000306	2	1.609	0.000305	13	11.610
0.00006	0.000306	2	1.578	0.000305	13	11.547
0.00007	0.000306	2	1.610	0.000305	13	11.672
0.00008	0.000306	2	1.594	0.000305	13	11.813
0.00009	0.000306	2	1.609	0.000305	13	11.813
0.0001	0.000306	2	1.703	0.000305	13	11.890
0.0002	0.000305	2	1.750	0.000305	14	13.110
0.0003	0.000307	2	1.719	0.000306	14	13.110
0.0004	0.000310	2	1.781	0.000308	16	15.407
0.0005	0.000311	2	1.735	0.000310	24	22.844
0.0006	0.000314	2	1.719	0.000312	15	14.250
0.0007	0.000316	2	1.719	0.000315	15	14.328
0.0008	0.000322	2	1.781	0.000319	32	30.563
0.0009	0.000324	2	1.687	0.000322	32	30.563
0.001	0.000328	2	1.781	0.000326	30	29.265
0.002	0.000391	2	1.828	0.000390	12	12.140
0.003	0.000519	2	1.953	0.000517	11	11.657

Tab. 4.2 – La performance de l'algorithme pour le deuxième jeu de données en utilisant l'algorithme par SE (le deuxième)

– a_i, b_i sont les paramètres concernant les bornes inférieure et supérieure ;
– *card* est le paramètre de cardinalité qui montre *le nombre* souhaité d'actifs composant le portefeuille.

En utilisant ces notations, le modèle s'écrit

$$
\begin{aligned}
&(P_{card}) : \\
&\quad \min\ (\mathbf{x}-\overline{\mathbf{x}})^t\mathbf{Q}(\mathbf{x}-\overline{\mathbf{x}}) \\
&\quad s.c. \\
&\qquad (\mathbf{x}-\overline{\mathbf{x}})^t\mathbf{r} - (\mathbf{c}_b^t\mathbf{x}_b + \mathbf{c}_s^t\mathbf{x}_s) \;\geq\; R, && (4.13)\\
&\qquad \mathbf{p} + \mathbf{x}_b - \mathbf{x}_s \;=\; \mathbf{x}, && (4.14)\\
&\qquad \sum_{j=1}^{n} x_j \;=\; 1, && (4.15)\\
&\qquad \sum_{j=1}^{n} z_j \;=\; card, && (4.16)\\
&\qquad a_j z_j \leq x_j \leq b_j z_j \qquad : j=1,\ldots,n, && (4.17)\\
&\qquad z_j \;\in\; \{0,1\} \qquad : j=1,\ldots,n, && (4.18)\\
&\qquad \mathbf{x}_b, \mathbf{x}_s \;\geq\; \mathbf{0}. && (4.19)
\end{aligned}
$$

La contrainte 4.14 montre le passage de l'investisseur de la position actuelle $\mathbf{p}$ au portefeuille composé du vecteur $\mathbf{x}$ en achetant $\mathbf{x}_b$ et vendant $\mathbf{x}_s$. La contrainte 4.13 indique le rendement du portefeuille après avoir payé les coûts de transaction. Les contraintes 4.16, 4.17 et 4.18 concernent la modélisation de la contrainte de cardinalité. En effet, c'est la contrainte de cardinalité qui rend le problème difficile à résoudre. A cause de la présence des variables binaires, le problème (P_{card}) est non convexe. Plusieurs alternatives du modèle de Markowitz avec la contrainte de cardinalité ont été étudiées et ont été résolues. La plupart des méthodes utilisées sont basées sur des heuristiques ([16, 29, 51]). Dans ce travail, la méthode utilisée est basée sur la programmation DC et DCA.

4.4.1 Programmation DC et DCA pour la résolution du problème

4.4.1.1 Reformulation

Le problème (P_{card}) peut être mis sous forme de programmation DC. Soit $\mathcal{A} \subseteq \mathbb{R}^{3n} \times [0,1]^n$ le polyèdre convexe borné et non vide défini par les contraintes 4.13, 4.14, 4.15, 4.16, 4.17 et 4.19. Le problème (P_{card}) s'écrit

$$\min\left\{(\mathbf{x}-\overline{\mathbf{x}})^t\mathbf{Q}(\mathbf{x}-\overline{\mathbf{x}}) : (\mathbf{x},\mathbf{x}_b,\mathbf{x}_s,\mathbf{z}) \in \mathcal{A}, z_j \in \{0,1\} : j=1,\ldots,n\right\}. \tag{4.20}$$

Considérons la fonction p définie par

$$p(\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}) := \sum_{i=1}^{n} z_i(1 - z_i). \tag{4.21}$$

La fonction p est une fonction concave et finie sur $\mathcal{A}$, de plus $p(\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}) \geq 0$ pour tout $(\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}) \in \mathcal{A}$, et

$$\{(\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}) \in \mathcal{A} : \mathbf{z} \in \{0,1\}^n\} = \{(\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}) \in \mathcal{A} : p(\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}) \leq 0\}.$$

Par conséquent (4.20) peut être réécrite ainsi

$$\min\left\{(\mathbf{x} - \overline{\mathbf{x}})^t \mathbf{Q}(\mathbf{x} - \overline{\mathbf{x}}) : (\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}) \in \mathcal{A}, p(\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}) \leq 0\right\}. \tag{4.22}$$

D'après le théorème 4.1, il existe un nombre suffisamment grand t_0 tel que, pour tout t satisfaisant $(t \geq t_0)$, le problème suivant soit équivalent à (P_{card}) :

$$(P_{card} - DC): \quad \min\left\{(\mathbf{x} - \overline{\mathbf{x}})^t \mathbf{Q}(\mathbf{x} - \overline{\mathbf{x}}) + tp(\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}) : (\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}) \in \mathcal{A}\right\}. \tag{4.23}$$

4.4.1.2 Résolution de $(P_{card} - DC)$ par DCA

Le modèle $(P_{card} - DC)$ est un problème de minimisation d'une fonction est convexe par rapport à $\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s$ et concave par rapport à $\mathbf{z}$, alors la fonction objectif du modèle $(P_{card} - DC)$ est une fonction DC.

Une décomposition naturelle de la fonction objectif du modèle $(P_{card} - DC)$, à l'aide des fonctions g et h est

$$g(\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}) = (\mathbf{x} - \overline{\mathbf{x}})^t \mathbf{Q}(\mathbf{x} - \overline{\mathbf{x}}) + \chi_{\mathcal{A}}(\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}) \quad \text{et} \quad h(\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}) = -t \sum_{i=1}^{n} z_i(1 - z_i) \tag{4.24}$$

où $\chi_{\mathcal{A}}$ est la fonction indicatrice sur $\mathcal{A}$, i.e., $\chi_{\mathcal{A}}(\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}) = 0$ si $(\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}) \in \mathcal{A}$, $+\infty$ sinon.

En bref, le problème $(P_{card} - DC)$ est un programme DC sous la forme

$$\min\{g(\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}) - h(\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}) \ : \ (\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}) \in \mathbb{R}^{4n}\} \tag{4.25}$$

pour lequel les fonctions g et h sont définies ci-dessus.

Selon la description de DCA, la résolution de $(P_{card} - DC)$ via la formulation (4.25) par DCA consiste à construire deux suites

$$(\mathbf{u}^k, \mathbf{u}_b^k, \mathbf{u}_s^k, \mathbf{v}^k) \in \partial h\ (\mathbf{x}^k, \mathbf{x}_b^k, \mathbf{x}_s^k, \mathbf{z}^k) \quad \text{et} \quad (\mathbf{x}^{k+1}, \mathbf{x}_b^{k+1}, \mathbf{x}_s^{k+1}, \mathbf{z}^{k+1}) \in \partial g^*(\mathbf{u}^k, \mathbf{u}_b^k, \mathbf{u}_s^k, \mathbf{v}^k).$$

La fonction h est différentiable et son gradient au point $(\mathbf{x}^k, \mathbf{x}_b^k, \mathbf{x}_s^k, \mathbf{z}^k)$ est calculé de façon suivante

$$(\mathbf{u}^k, \mathbf{u}_b^k, \mathbf{u}_s^k, \mathbf{v}^k) \in \partial h(\mathbf{x}^k, \mathbf{x}_b^k, \mathbf{x}_s^k, \mathbf{z}^k) \Longleftrightarrow \mathbf{u}^k = \mathbf{u}_b^k = \mathbf{u}_s^k = \mathbf{0}, \quad \mathbf{v}^k = t(2\mathbf{z}^k - \mathbf{1}). \tag{4.26}$$

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

$$\min \left\{ (\mathbf{x} - \overline{\mathbf{x}})^t \mathbf{Q} (\mathbf{x} - \overline{\mathbf{x}}) - \langle (\mathbf{u}^k, \mathbf{u}_b^k, \mathbf{u}_s^k, \mathbf{v}^k), (\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}) : (\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}) \in \mathcal{A} \right\}. \tag{4.27}$$

Algorithme 4.2 *Schéma de DCA*

Initialisation :
- *Choisir* $(\mathbf{x}^0, \mathbf{x}_b^0, \mathbf{x}_s^0, \mathbf{z}^0) \in \mathbb{R}^{3n} \times \mathbb{R}^n$ *et* $k = 0$.
- *Choisir la tolérance* ϵ *positive suffisamment petite.*

Répéter
- *Calculer* $(\mathbf{u}^k, \mathbf{u}_b^k, \mathbf{u}_s^k, \mathbf{v}^k) \in \partial h(\mathbf{x}^k, \mathbf{x}_b^k, \mathbf{x}_s^k, \mathbf{z}^k)$ *via (4.26).*
- *Calculer* $(\mathbf{x}^{k+1}, \mathbf{x}_b^{k+1}, \mathbf{x}_s^{k+1}, \mathbf{z}^{k+1}) \in \partial g^*(\mathbf{u}^k, \mathbf{u}_b^k, \mathbf{u}_s^k, \mathbf{v}^k)$ *en résolvant le programme quadratique (4.27).*
- $k + 1 \leftarrow k$.

Jusqu'à $\left\|(\mathbf{x}^{k+1}, \mathbf{x}_b^{k+1}, \mathbf{x}_s^{k+1}, \mathbf{z}^{k+1}) - (\mathbf{x}^k, \mathbf{x}_b^k, \mathbf{x}_s^k, \mathbf{z}^k)\right\| \leq \epsilon$.

Le théorème de convergence de l'algorithme 4.2 se résume à l'aide du théorème suivant dont la démonstration n'est pas donnée à cause de sa ressemblance au théorème 4.2 (voir aussi [91, 94, 139]).

Théorème 4.3 *(Propriétés de la convergence de l'algorithme 4.2)*
(i) L'algorithme 4.2 génère une suite $\{(\mathbf{x}^k, \mathbf{x}_b^k, \mathbf{x}_s^k, \mathbf{z}^k)\}$ *telle que la suite* $(\mathbf{x}^k - \overline{\mathbf{x}})^t \mathbf{Q}(\mathbf{x}k - \overline{\mathbf{x}}) + tp(\mathbf{x}^k, \mathbf{x}_b^k, \mathbf{x}_s^k, \mathbf{z}^k)$ *est décroissante.*
(ii) Il existe un nombre non négatif t *tel que pour chaque* $t \geq t_0$ *la suite* $\{p(\mathbf{x}^k, \mathbf{x}_b^k, \mathbf{x}_s^k, \mathbf{z}^k)\}$ *est décroissante. En particulier, si* $(\mathbf{x}^r, \mathbf{x}_b^r, \mathbf{x}_s^r, \mathbf{z}^r)$ *est une solution admissible de* (P_{card}) *alors* $(\mathbf{x}^k, \mathbf{x}_b^k, \mathbf{x}_s^k, \mathbf{z}^k)$*, pour tout* $k \geq r$*, est admissible également.*
(iii) DCA a un taux de convergence linéaire pour le problème $(P_{card} - DC)$.
(iv) La suite $\{(\mathbf{x}^k, \mathbf{x}_b^k, \mathbf{x}_s^k, \mathbf{z}^k)\}$ *converge à* $(\mathbf{x}^*, \mathbf{x}_b^*, \mathbf{x}_s^*, \mathbf{z}^*)$ *où le point* $(\mathbf{x}^*, \mathbf{x}_b^*, \mathbf{x}_s^*, \mathbf{z}^*)$ *est un point critique du problème* $(P_{card} - DC)$.

4.4.1.3 Résolution globale

Afin de trouver la solution globale du problème (P_{card} et de vérifier la qualité des résultats obtenus par DCA, nous avons utilisé un algorithme par Séparation et Évaluation (SE) et le logiciel CPLEX afin de résoudre le problème. Enfin, une méthode combinée de DCA et l'algorithme par SE a été développée. L'algorithme par SE calcule les bornes inférieures en remplaçant les contraintes binaires $z_j \in \{0, 1\}$ par $0 \leq z_j \leq 1$. La borne supérieure est mise à jour si nous trouvons une meilleure solution pour le modèle (P_{card}). A chaque itération nous choisissons une variable fractionnelle comme z_j et la séparation est effectuée sur cette variable telle que soit $z_j = 0$, soit $z_j = 1$.

La philosophie des méthodes combinées est essentiellement basée sur l'utilisation de DCA afin de trouver une solution, éventuellement, améliorant la borne supérieure. De cette manière,

l'algorithme va effacer plus de sous-problèmes qui ne possèdent pas de solution globale. Cette opération accélère la convergence de l'algorithme par SE.

Le schéma de l'algorithme combiné :

A chaque itération de l'algorithme par SE, après avoir décidé de relancer la méthode DCA, nous continuons de la façon suivante

1. Construire le modèle DC.
2. Prendre la solution du sous-problème afin de résoudre le problème DC par DCA. Avant d'utiliser la solution, il faut arrondir toutes les variables binaires non nulles à 1.
3. Résoudre le problème DC par DCA.
4. Évaluer la solution fournie par DCA. Si la solution est meilleure que la meilleure solution actuelle alors faire une mise à jour de la borne supérieure et de la meilleure solution.
5. Continuer l'algorithme par SE.

Pour tous les algorithmes et les méthodes, la procédure s'arrête soit lorsque les bornes inférieure et supérieure sont suffisamment serrées (proches), soit le temps d'exécution dépasse une limite prévue. On relance DCA lorsque le nombre de composants 0-1 de variables binaires de la solution du sous-problème relaxé est suffisamment grand, par exemple supérieur ou égal à $n/2$.

4.4.1.4 Expérience numériques

Les algorithmes ont été codé avec le langage C++ et exécutés sur un ordinateur Pentium de 3GHz et 1Go RAM. La version 10.1 du logiciel CPLEX a été utilisée afin de résoudre les sous-problèmes relaxés ainsi que le modèle P_{card}. Les tests ont été effectués sur un jeu de données qui avait été déjà utilisé dans les articles publiés par différents chercheurs ([16, 29, 51, 154]). Les données correspondent aux prix hebdomadaires des actifs financiers de Mars 1992 à Septembre 1997. Les actifs ont été choisis parmi l'indice Dax 100 en Allemagne. Le nombre des actif est 85. Le paramètre de pénalité, i.e., t, est égale à 2.0. $\epsilon = 10^{-7}$ et les tests ont été faits sur différentes valeurs de $card$. Ce sont les valeurs pour lesquelles le problème P_{card} devient très difficile à résoudre. Les valeurs choisies sont : $5, 6, \ldots, 15$. Pour les valeurs plus grandes, le problème est facile à résoudre. Les valeurs choisies pour les autres paramètres sont :

– c_{b_j}, c_{s_j} : 0.1% de transaction (achat ou vente) ;
– $p_j = \mathbf{0}$: nous supposons qu'il ne s'agit pas d'un re-balancement de portefeuille ;
– $\overline{x}_j = 1/n$;
– $a_j = 0.05$: les bornes inférieures ;
– $b_j = 1.0$: les bornes supérieures.

Recherche d'un bon point initial pour DCA

La procédure suivante explique la manière par laquelle le point initial de DCA a été choisi :

1. **Résoudre le problème relaxé :** Nous résolvons le problème relaxé du (P_{card}) afin de trouver $(\widetilde{\mathbf{x}}, \widetilde{\mathbf{x}}_b, \widetilde{\mathbf{x}}_s, \widetilde{\mathbf{z}})$. Le problème relaxé se définit par la relaxation des contraintes binaires, i.e. les contraintes $z_j \in \{0, 1\}$ sont remplacées par $0 \leq z_j \leq 1$.
2. **Trouver une solution entière :** Nous arrondissons les variables $\widetilde{\mathbf{z}}$ non nulles à 1 pour arriver de $(\widetilde{\mathbf{x}}, \widetilde{\mathbf{x}}_b, \widetilde{\mathbf{x}}_s, \widetilde{\mathbf{z}})$ à $(\widetilde{\mathbf{x}}, \widetilde{\mathbf{x}}_b, \widetilde{\mathbf{x}}_s, \widehat{\mathbf{z}})$.

En général, le nouveau point n'est plus une solution admissible du modèle (P_{card}) ni pour le modèle $(P_{card} - DC)$. DCA trouve une solution admissible pour le modèle $(P_{card} - DC)$ après une seule itération. En cours des itérations suivantes, DCA va améliorer la solution.

En effet, afin de trouver un bon initial, nous avons testé plusieurs choix :

- le point fourni par le processus expliqué ci-dessus ;
- la solution optimale du modèle relaxé de (P_{card}) ;
- la solution optimale du problème suivant

$$\min \left\{ \sum_{j=1}^{n} z_j(1 - z_j) : (\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}) \in \mathcal{A} \right\}.$$

La première procédure donne de meilleurs résultats.

Le tableau 4.3, présente les résultats pour les différentes valeurs de paramètre de cardinalité (card). Dans ce tableau, les valeurs optimales (Val. Opt.) fournies par DCA, par l'algorithme de SE (SE) et par la méthode combinée (SE-DCA) et par CPLEX ont été présentées. Le tableau présente aussi le nombre d'itération (iter.) de DCA, le temps de résolution (CPU) en secondes pour DCA, le nombre d'appel à la méthode DCA pendant la procédure de SE-DCA (relance). Le temps d'exécution de tous les autres algorithmes était limité à 1200 secondes.

Les résultats montrent que DCA donne de bons résultats. Ils sont encourageants et en fait c'est la raison pour laquelle nous avons combiné l'algorithme SE et DCA afin d'améliorer la performance de SE. A chaque itération de SE-DCA, si les conditions étaient favorables, nous avons relancé DCA. Le rôle de DCA est de trouver une meilleure solution.

La supériorité de la méthode combinée par rapport à SE classique est due à l'efficacité de DCA. Selon les résultats présentés au tableau l'algorithme classique par SE échoue pour la plupart des valeurs de *card*. A l'opposé, les résultats fournis par la méthode combinée peuvent être comparés avec ceux de CPLEX. Plus particulièrement, la méthode combinée fournit une meilleure solution que CPLEX pour $card = 13$.

	DCA			SE-DCA			CPLEX	SE
card	Val. Opt.	iter	CPU	Val. Opt.	iter	relance	Val. Opt.	Val. Opt.
5	0.000114	4	0.343	0.000075	10000	744	0.000071	0.000088
6	0.000078	4	0.344	0.000063	9429	530	0.000057	0.000072
7	0.000072	4	0.360	0.000051	8128	451	0.000050	0.000063
8	0.000060	4	0.375	0.000043	8678	592	0.000041	0.000054
9	0.000056	4	0.344	0.000038	8440	580	0.000037	0.000052
10	0.000101	4	0.359	0.000035	7894	515	0.000030	Échoué
11	0.000068	4	0.360	0.000031	8324	567	0.000029	Échoué
12	0.000083	4	0.344	0.000029	8403	581	0.000027	Échoué
13	0.000050	4	0.359	0.000025	8434	583	0.000026	Échoué
14	0.000041	4	0.375	0.000025	8382	581	0.000021	Échoué
15	0.000038	4	0.359	0.000024	8338	573	0.000020	Échoué

TAB. 4.3 – La performance des algorithmes pour les différentes valeurs de cardinalité

4.5 Contraintes de seuil

Dans la plupart des marchés boursiers, un investisseur peut vendre un actif qu'il ne possède pas. Ce procédé est nommé *vente-à-découvert*. La mécanique de vente-à-découvert dit que l'actif peut prendre une position négative. Dans ce qui suit, nous allons présenter une description générale de vente-à-découvert qui existe dans la littérature financière. Cette description est simplifiée et elle ne contient pas tous les aspects réels de vente-à-découvert comme l'existence de coûts de transaction [27].

Imaginez que chaque action d'une compagnie soit vendue au prix de 100$ et un investisseur croit que cette action va valoir moins cher (par exemple 95$) au bout d'une période (par exemple, un an). De plus, l'investisseur estime que la compagnie va payer 3$ de dividende à la fin de la période. Si l'investisseur *achète* une action de cette compagnie, son investissement sera -100\$ à la date 0. Au bout de la période, il aura $+3$\$ de dividende et pourra vendre cette action au prix de $+95$\$. Alors, à la fin de période l'investisseur perdra 2$. Clairement, aucun investisseur ne souhaite avoir de telles actions dans son portefeuille, voire il veut investir un montant négatif dans une telle action. La question est la condition d'avoir une telle possibilité. Comment pouvons-nous créer une telle situation ? Supposons qu'un courtier accepte la vente-à-découvert, alors l'investisseur peut vendre l'action, dont nous avons parlé, à 100$. À la fin de la période, l'investisseur doit racheter l'action pour la rendre au courtier, de plus, il va payer le dividende au courtier. Alors, l'investisseur doit payer 95$ pour racheter l'action et 3$ pour le dividende ; soit, 2$ de gagne.

Afin de modéliser la vente-à-découvert, nous utilisons une définition alternative de la vente-à-découvert [27]. D'un point de vue, une vente-à-découvert consiste à considérer un capital potentiel équivalent au montant de la vente-à-découvert. Autrement dit, la vente-à-découvert est une source de capital hors de la richesse que l'investisseur possède. Alors, le capital total investi est la somme de la vente-à-découvert et de la richesse initiale (que l'investisseur investit hors de la vente-à-découvert). Notons x_j comme la portion du capital investi en actif j, si l'investissement est dû à la vente-à-découvert alors, $x_j < 0$ sinon $x_j \geq 0$. D'après cette définition, nous devons avoir la contrainte $\sum_{j=1}^{n} \mid x_j \mid = 1$. En général, à la fin de la période d'investissement, l'investisseur doit payer certains coûts à propos de sa vente-à-découvert, mais ici nous n'allons pas prendre en compte l'existence d'un tel coût. Alors, le revenu total d'investissement est de $\sum_{j=1}^{n} r_j x_j$ où, comme pour le modèle de Markowitz, r_j est le rendement moyen de l'actif j. Le modèle de Markowitz en présence de la vente-à-découvert se résume par

$$
\begin{aligned}
&\min \ \mathbf{x}^t \mathbf{Q} \mathbf{x} \\
&s.c. \\
&\qquad \sum_{j=1}^{n} r_j x_j \ = \ R, \\
&\qquad \sum_{j=1}^{n} \mid x_j \mid \ = \ 1.
\end{aligned}
$$

Après avoir formulé le modèle de Markowitz avec la vente-à-découvert, nous pouvons prendre en considération tout type de contraintes qui sont appliquées dans les situations réelles comme les contraintes de seuil ou bien les contraintes de cardinalité. Les contraintes de seuil (voir [7]) empêchent de très petits investissements dans chaque actif, que ce soit un actif concernant l'achat ou un actif concernant la vente-à-découvert. Dans ce travail, nous avons étudié un modèle qui contient ces deux types d'actifs. De plus, il existe des contraintes de seuil sur les investissements. Soient, a_j, b_j les bornes inférieure et supérieure sur l'investissement dans l'actif j, respectivement, telles que $0 < a_j \leq b_j \leq 1$. De la même manière, soient c_j, d_j les bornes supérieure et inférieure sur la vente-à-découvert dans l'actif j, respectivement, telles que $-1 \leq c_j \leq d_j < 0$. En utilisant ces notations, la généralisation du modèle de Markowitz, dans laquelle la vente-à-découvert est permise et dans laquelle des contraintes de seuil sont présentes, s'exprime par

$$\begin{aligned} &\min \ \mathbf{x}^t\mathbf{Q}\mathbf{x} \\ &s.c. \\ &\quad \sum_{j=1}^{n} r_j x_j \ = \ R, \\ &\quad \sum_{j=1}^{n} \mid x_j \mid \ = \ 1, \\ &\quad\quad x_j \ \in \ \{0\} \cup [a_j, b_j] \cup [c_j, d_j] \qquad : j = 1, \ldots, n. \end{aligned} \tag{4.28}$$

Selon ce modèle, soit on n'investit pas dans l'actif j, i.e. $x_j \in \{0\}$, soit l'investissement est effectué dans la limite des bornes, i.e. $x_j \in [a_j, b_j]$ s'il s'agit d'un achat et $x_j \in [c_j, d_j]$ s'il s'agit d'une vente-à-découvert. Le modèle est non-convexe. Il y a des contraintes de seuil et une contrainte de valeur absolue. Avant de résoudre le programme, nous allons essayer de le reformuler.

4.5.1 Reformulation

Afin de pouvoir supprimer le signe de valeur absolue, nous introduisons deux vecteurs : $\mathbf{y}, \mathbf{y}'$ tels que

$$\mid x_j \mid = y_j - y'_j, \quad y_j y'_j = 0, \quad y_j \geq 0, y'_j \leq 0 : j = 1, \ldots, n$$

où, y_j correspond aux achats et y'_j correspond aux ventes-à-découvert. Avec les nouvelles variables, nous avons

$$x_j = y_j + y'_j \quad : j = 1, \ldots, n.$$

Le changement de variable exige l'introduction des contraintes de complémentarité. L'interprétation de ces contraintes est liée au fait que l'investisseur ne doit pas et ne peut pas acheter et vendre un titre simultanément.

(P_{seuil}) :

$$\min\ (\mathbf{y}+\mathbf{y}')^t\mathbf{Q}(\mathbf{y}+\mathbf{y}')$$

s.c.

$$\begin{aligned}
\sum_{j=1}^{n} r_j(y_j+y'_j) &= R, \\
\sum_{j=1}^{n}(y_j-y'_j) &= 1, \\
y_j &\in \{0\}\cup[a_j,b_j] && : j=1,\dots,n, \qquad (4.29)\\
y'_j &\in \{0\}\cup[c_j,d_j] && : j=1,\dots,n, \qquad (4.30)\\
y_j y'_j &= 0 && : j=1,\dots,n. \qquad (4.31)
\end{aligned}$$

Essentiellement, la résolution d'un modèle avec les possibilités de vente ou de vente-à-découvert est difficile. Les contraintes de complémentarité sont les premières causes. D'un côté, l'élimination de ces contraintes va nous donner des portefeuilles qui franchissent cette règle, d'autre part ces contraintes posent des problèmes au niveau de la résolution efficace du modèle. Konno et al. [56] ont développé un algorithme par séparation et évaluation (SE) pour résoudre ce problème. Dans l'algorithme proposé, les bornes inférieures sont calculées en relaxant les contraintes de complémentarité et la séparation est effectuée sur la variable qui ne respecte pas ces contraintes. Pourtant, la résolution d'un problème par un algorithme de SE n'est pas toujours promettant, surtout lorsqu'il s'agit d'un problème de grande taille. En raison de difficultés qui existent en résolution des problèmes avec les possibilité de vente-à-découvert, ils n'ont pas été bien étudiés dans la littérature financière ([55, 56]). On trouve rarement des travaux dans lesquels un modèle de portefeuille avec la vente-à-découvert soit étudié en acceptant et respectant tous ses aspects théoriques et logiques. En général, les contraintes de complémentarité sont supprimées ([55, 56]). Dans ce travail nous proposons une fonction de pénalité qui remplace les contraintes de seuil et celles de complémentarité. Après avoir utilisé un résultat de pénalité exacte, nous mettons le modèle sous la forme d'un programme DC. Ensuite DCA s'applique pour résoudre le problème DC. Un algorithme par séparation et évaluation, ainsi qu'une méthode combinée de SE et DCA pour résoudre le modèle de façon efficace et globale est proposée. D'abord, nous allons reformuler le modèle (P_{seuil}) sous forme d'un programme binaire. Soient z_j et z'_j des variables binaires telles que pour $j=1,\dots,n$:

$$z_j = \begin{cases} 1 & : \quad y_j \in [a_j,b_j], \\ 0 & : \quad \text{sinon}, \end{cases}$$

et

$$z'_j = \begin{cases} 1 & : \quad y'_j \in [c_j,d_j], \\ 0 & : \quad \text{sinon}. \end{cases}$$

Avec les nouvelles variables, (P_{seuil}) s'exprime sous la forme suivante :

$$
\begin{aligned}
(P^{bin}_{seuil}) :\ & \\
& \min\ (\mathbf{y}+\mathbf{y}')^t\mathbf{Q}(\mathbf{y}+\mathbf{y}') \\
& s.c. \\
& \sum_{j=1}^{n} r_j(y_j+y'_j) = R, \\
& \sum_{j=1}^{n} (y_j-y'_j) = 1, \\
& a_j z_j \le y_j \le b_j z_j \qquad : j=1,\dots,n, \qquad (4.32)\\
& c_j z'_j \le y'_j \le d_j z'_j \qquad : j=1,\dots,n, \qquad (4.33)\\
& z_j z'_j = 0 \qquad : j=1,\dots,n, \qquad (4.34)\\
& z_j, z'_j \in \{0,1\} \qquad : j=1,\dots,n. \qquad (4.35)
\end{aligned}
$$

Les deux modèles (P_{seuil}) et (P^{bin}_{seuil}) sont équivalents dans le sens où
- Si $(\mathbf{y},\mathbf{y}')$ est une solution optimale de (P_{seuil}), alors nous pouvons trouver $\mathbf{z}$ et $\mathbf{z}'$ tels que $(\mathbf{y},\mathbf{y}',\mathbf{z},\mathbf{z}')$ soit une solution optimale de (P^{bin}_{seuil}) .
- Si $(\mathbf{y},\mathbf{y}',\mathbf{z},\mathbf{z}')$ est une solution optimale de (P^{bin}_{seuil}) alors $(\mathbf{y},\mathbf{y}')$ est une solution optimale de (P_{seuil}).

Notons $\mathcal{A}$ l'ensemble défini par

$$
\mathcal{A} := \left\{ \begin{array}{l} (\mathbf{y},\mathbf{y}',\mathbf{z},\mathbf{z}') \in \mathbb{R}^n\times\mathbb{R}^n\times[0,1]^n\times[0,1]^n : \sum_{j=1}^{n} r_j(y_j+y'_j)=R, \sum_{j=1}^{n}(y_j-y'_j)=1, \\ a_j z_j \le y_j \le b_j z_j, c_j z'_j \le y'_j \le d_j z'_j, j=1,\dots,n \end{array} \right\}.
$$

Théorème 4.4 *Définir la fonction de pénalité* $p(\mathbf{y},\mathbf{y}',\mathbf{z},\mathbf{z}') : \mathbb{R}^n\times\mathbb{R}^n\times\mathbb{R}^n\times\mathbb{R}^n \longrightarrow \mathbb{R}$ *par*

$$
p(\mathbf{y},\mathbf{y}',\mathbf{z},\mathbf{z}') := \sum_{j=1}^{n}(z_j+z'_j) - \sum_{j=1}^{n}(z_j-z'_j)^2. \qquad (4.36)
$$

(i) *La fonction* $p(\mathbf{y},\mathbf{y}',\mathbf{z},\mathbf{z}')$ *est concave.*

(ii) *La fonction* $p(\mathbf{y},\mathbf{y}',\mathbf{z},\mathbf{z}')$ *est non-négative sur* $\mathcal{A}$ *et si*

$$
A_1 = \{(\mathbf{y},\mathbf{y}',\mathbf{z},\mathbf{z}') \in \mathcal{A} : z_i z'_i = 0, z_i, z'_i \in \{0,1\} \quad : i=1,\dots,n\}
$$

et

$$
A_2 = \{(\mathbf{y},\mathbf{y}',\mathbf{z},\mathbf{z}') \in \mathcal{A} : p(\mathbf{y},\mathbf{y}',\mathbf{z},\mathbf{z}') \le 0\},
$$

alors $A_1 = A_2$.

Preuve :

(i) Soient $p_j : \mathbb{R}^{4n} \longrightarrow \mathbb{R}$ les fonctions définies par

$$p_j(\mathbf{y}, \mathbf{y}', \mathbf{z}, \mathbf{z}') := (z_j + z'_j) - (z_j - z'_j)^2 \quad : \forall j = 1, \ldots, n$$

et $\varphi_j : \mathbb{R}^{4n} \longrightarrow \mathbb{R}$ les fonctions définies par

$$\varphi_j(\mathbf{y}, \mathbf{y}', \mathbf{z}, \mathbf{z}') := (z_j - z'_j)^2 \quad : \forall j = 1, \ldots, n.$$

Les fonctions $\varphi_j, j = 1, \ldots, n$ sont convexes car elles sont la composition de fonctions convexes avec des fonctions linéaires. Par conséquent p_j est concave car elle est la somme d'une fonction linéaire et d'une fonction concave. En outre, $p(\mathbf{y}, \mathbf{y}', \mathbf{z}, \mathbf{z}') := \sum_{j=1}^{n} p_j(\mathbf{y}, \mathbf{y}', \mathbf{z}, \mathbf{z}')$ et nous savons que la somme de fonctions concaves est concave, alors $p(\mathbf{y}, \mathbf{y}', \mathbf{z}, \mathbf{z}')$ est une fonction concave.

(ii) Après remis en ordre les éléments de (4.36), nous obtenons

$$p(\mathbf{y}, \mathbf{y}', \mathbf{z}, \mathbf{z}') := 2\sum_{j=1}^{n}(z_j z'_j) + \sum_{j=1}^{n} z_j(1 - z_j) + \sum_{j=1}^{n} z'_j(1 - z'_j).$$

Tous les éléments du côté droit sont non-négatifs sur $\mathcal{A}$ alors $p(\mathbf{y}, \mathbf{y}', \mathbf{z}, \mathbf{z}') \geq 0$ pour tout $(\mathbf{y}, \mathbf{y}', \mathbf{z}, \mathbf{z}') \in \mathcal{A}$. De plus

$$p(\mathbf{y}, \mathbf{y}', \mathbf{z}, \mathbf{z}') = 0 \Longleftrightarrow \begin{cases} z_j z'_j = 0 & : \quad \forall j = 1, \ldots, n, \\ z_j(1 - z_j) = 0 & : \quad \forall j = 1, \ldots, n, \\ z'_j(1 - z'_j) = 0 & : \quad \forall j = 1, \ldots, n, \end{cases} \tag{4.37}$$

ou

$$p(\mathbf{y}, \mathbf{y}', \mathbf{z}, \mathbf{z}') = 0 \Longleftrightarrow \begin{cases} z_j z'_j = 0 & : \quad \forall j = 1, \ldots, n, \\ z_j \in \{0, 1\} & : \quad \forall j = 1, \ldots, n, \\ z'_j \in \{0, 1\} & : \quad \forall j = 1, \ldots, n. \end{cases} \tag{4.38}$$

Cela veut dire

$$\{(\mathbf{y}, \mathbf{y}', \mathbf{z}, \mathbf{z}') \in \mathcal{A} : z_j z'_j = 0, z_j, z'_j \in \{0, 1\} : i = 1, \ldots, n\}$$

$$= \{(\mathbf{y}, \mathbf{y}', \mathbf{z}, \mathbf{z}') \in \mathcal{A} : p(\mathbf{y}, \mathbf{y}', \mathbf{z}, \mathbf{z}') = 0\}.$$

D'ailleurs $p(\mathbf{y}, \mathbf{y}', \mathbf{z}, \mathbf{z}')$ est non-négatif sur $\mathcal{A}$ donc

$$\{(\mathbf{y}, \mathbf{y}', \mathbf{z}, \mathbf{z}') \in \mathcal{A} : p(\mathbf{y}, \mathbf{y}', \mathbf{z}, \mathbf{z}') = 0\} = \{(\mathbf{y}, \mathbf{y}', \mathbf{z}, \mathbf{z}') \in \mathcal{A} : p(\mathbf{y}, \mathbf{y}', \mathbf{z}, \mathbf{z}') \leq 0\}.$$

En prenant en compte cette relation et (4.38), nous concluons $A_1 = A_2$.

□

Ce théorème nous permet de réécrire le modèle (P^{bin}_{seuil}) sous la forme suivante

$$\min \left\{ V(\mathbf{y}, \mathbf{y}') := (\mathbf{y} + \mathbf{y}')^t Q(\mathbf{y} + \mathbf{y}') : (\mathbf{y}, \mathbf{y}', \mathbf{z}, \mathbf{z}') \in \mathcal{A}, p(\mathbf{y}, \mathbf{y}', \mathbf{z}, \mathbf{z}') \leq 0 \right\}. \tag{4.39}$$

Puisque V est une fonction convexe et $\mathcal{A}$ est un polyèdre convexe borné et en outre p est concave et non-négatif sur $\mathcal{A}$, alors d'après le théorème (4.1), il existe $t_0 \geq 0$ tel que pour tout $t > t_0$, le modèle (4.39) est équivalent à

$$\min\left\{F(\mathbf{y},\mathbf{y}',\mathbf{z},\mathbf{z}') := (\mathbf{y}+\mathbf{y}')^t Q(\mathbf{y}+\mathbf{y}') + tp(\mathbf{y},\mathbf{y}',\mathbf{z},\mathbf{z}') : (\mathbf{y},\mathbf{y}',\mathbf{z},\mathbf{z}') \in \mathcal{A}\right\}.$$

La fonction F est convexe en variables $\mathbf{y}$ et $\mathbf{y}'$, mais F est concave en variables $\mathbf{z}$ et $\mathbf{z}'$. Par conséquent, F est une fonction DC. Une décomposition DC est la suivante

$$(P_{seuil}^{DC}) : \min\left\{F(\mathbf{y},\mathbf{y}',\mathbf{z},\mathbf{z}') := g(\mathbf{y},\mathbf{y}',\mathbf{z},\mathbf{z}') - h(\mathbf{y},\mathbf{y}',\mathbf{z},\mathbf{z}') : (\mathbf{y},\mathbf{y}',\mathbf{z},\mathbf{z}') \in \mathbb{R}^{4n}\right\},$$

telle que

$$g(\mathbf{y},\mathbf{y}',\mathbf{z},\mathbf{z}') := (\mathbf{y}+\mathbf{y}')^t Q(\mathbf{y}+\mathbf{y}') + \chi_{\mathcal{A}}(\mathbf{y},\mathbf{y}',\mathbf{z},\mathbf{z}'),$$

et

$$h(\mathbf{y},\mathbf{y}',\mathbf{z},\mathbf{z}') := t\left(\sum_{j=1}^{n}(z_j - z_j')^2 - \sum_{j=1}^{n}(z_j + z_j')\right).$$

$\chi_{\mathcal{A}}$ est la fonction indicatrice de $\mathcal{A}$, c'est-à-dire, $\chi_{\mathcal{A}}(\mathbf{y},\mathbf{y}',\mathbf{z},\mathbf{z}') = 0$ si $(\mathbf{y},\mathbf{y}',\mathbf{z},\mathbf{z}') \in \mathcal{A}$ et $+\infty$ sinon.

4.5.2 Résolution de (P_{seuil}^{DC}) par DCA

D'après le schéma générique de DCA, il nous faut calculer sous-gradient de la fonction h définie par $h(\mathbf{y},\mathbf{y}',\mathbf{z},\mathbf{z}') := t\left(\sum_{j=1}^{n}(z_j - z_j')^2 - \sum_{j=1}^{n}(z_j + z_j')\right)$. Le sous-gradient de h est de la forme suivante :

$$(\mathbf{u}^k,\mathbf{u}'^k,\mathbf{v}^k,\mathbf{v}'^k) \in \partial h(\mathbf{y}^k,\mathbf{y}'^k,\mathbf{z}^k,\mathbf{z}'^k) \Leftrightarrow \begin{cases} u_j^k = 0 & : \quad j = 1,\ldots,n, \\ u_j'^k = 0 & : \quad j = 1,\ldots,n, \\ v_j^k = t(2(z_j^k - z_j'^k) - 1) & : \quad j = 1,\ldots,n, \\ v_j'^k = t(2(z_j'^k - z_j^k) - 1) & : \quad j = 1,\ldots,n. \end{cases} \tag{4.40}$$

Afin de calculer $(\mathbf{y}^{k+1},\mathbf{y}'^{k+1},\mathbf{z}^{k+1},\mathbf{z}'^{k+1})$, le problème suivant doit être résolu :

$$\min\{(\mathbf{y}+\mathbf{y}')^t Q(\mathbf{y}+\mathbf{y}') - \langle(\mathbf{y},\mathbf{y}',\mathbf{z},\mathbf{z}'),(\mathbf{u}^k,\mathbf{u}'^k,\mathbf{v}^k,\mathbf{v}'^k)\rangle : (\mathbf{y},\mathbf{y}',\mathbf{z},\mathbf{z}') \in \mathcal{A}\}. \tag{4.41}$$

L'algorithme DCA qui résout de problème (P_{seuil}^{DC}) se résume ainsi :

Algorithme 4.3 *Schéma de DCA*

Initialisation :

– *Choisir* $(\mathbf{y}^0,\mathbf{y}'^0,\mathbf{z}^0,\mathbf{z}'^0) \in R^n \times R^n \times [0,1]^n \times [0,1]^n$ *et mettre* $k = 0$.

– *Choisir la tolérance ϵ positive suffisamment petite.*

Répéter :

– *Calculer* $(\mathbf{u}^k, \mathbf{u}'^k, \mathbf{v}^k, \mathbf{v}'^k) \in \partial h(\mathbf{y}^k, \mathbf{y}'^k, \mathbf{z}^k, \mathbf{z}'^k)$ *via (4.40).*

– *Calculer* $(\mathbf{y}^{k+1}, \mathbf{y}'^{k+1}, \mathbf{z}^{k+1}, \mathbf{z}'^{k+1}) \in \partial g^*(\mathbf{u}^k, \mathbf{u}'^k, \mathbf{v}^k, \mathbf{v}'^k)$ *en résolvant le programme quadratique (4.41).*

– $k + 1 \leftarrow k$.

Jusqu'à

$\left\|(\mathbf{y}^{k+1}, \mathbf{y}'^{k+1}, \mathbf{z}^{k+1}, \mathbf{z}'^{k+1}) - (\mathbf{y}^k, \mathbf{y}'^k, \mathbf{z}^k, \mathbf{z}'^k)\right\| \leq \epsilon$.

Le théorème de convergence de l'algorithme 4.3 se résume dans le théorème suivant ([91, 94, 139]).

Théorème 4.5 *(Propriétés de la convergence de l'algorithme (4.3))*

(i) L'algorithme 4.3 génère une suite $\{(\mathbf{y}^k, \mathbf{y}'^k, \mathbf{z}^k, \mathbf{z}'^k)\}$ *tel que la suite* $(\mathbf{y}^k + \mathbf{y}'^k)^t Q(\mathbf{y}^k + \mathbf{y}'^k) + tp(\mathbf{y}^k, \mathbf{y}'^k, \mathbf{z}^k, \mathbf{z}'^k)$ *est décroissante.*

(ii) Il existe un nombre non-négatif t tel que pour chaque $t \geq t_0$ *la suite* $\{p(\mathbf{y}^k, \mathbf{y}'^k, \mathbf{z}^k, \mathbf{z}'^k)\}$ *est décroissante. En particulier, si* $(\mathbf{y}^r, \mathbf{y}'^r, \mathbf{z}^r, \mathbf{z}'^r)$ *est une solution admissible de* (P^{bin}_{seuil}) *alors* $(\mathbf{y}^k, \mathbf{y}'^k, \mathbf{z}^k, \mathbf{z}'^k)$*, pour tout* $k \geq r$*, est admissible également.*

(iii) DCA a un taux de convergence linéaire pour le problème (P^{DC}_{seuil}).

(iv) La suite $\{(\mathbf{y}^k, \mathbf{y}'^k, \mathbf{z}^k, \mathbf{z}'^k)\}$ *converge à* $(\mathbf{y}^*, \mathbf{y}'^*, \mathbf{z}^*, \mathbf{z}'^*)$ *où le point* $(\mathbf{y}^*, \mathbf{y}'^*, \mathbf{z}^*, \mathbf{z}'^*)$ *est un point critique du problème* (P^{DC}_{seuil}).

4.5.2.1 Choix de point initial pour DCA

Afin de choisir un bon point initial, nous avons testé plusieurs choix. Le premier consiste d'abord à résoudre le problème relaxé du (P_{seuil}). Ce problème résulte de la suppression des contraintes de complémentarité et relaxation des contraintes de seuil. Soit $(\widetilde{\mathbf{y}}, \widetilde{\mathbf{y}'})$ la solution de ce problème. Le point initial de DCA est $(\mathbf{y}, \mathbf{y}', \mathbf{z}, \mathbf{z}')$, tel que $\mathbf{y} = \widetilde{\mathbf{y}}$, $\mathbf{y}' = \widetilde{\mathbf{y}'}$,$\mathbf{z}' = 0$ et

$$z_j := \begin{cases} 1, & \text{si} \quad R < r_j, \\ 0, & \text{sinon.} \end{cases}$$

La solution fournie par cette procédure n'est pas forcément une solution admissible pour (P^{DC}_{seuil}) mais après seulement une itération, DCA trouvera une solution admissible pour (P^{DC}_{seuil}).

Les autres points initiaux testés sont :

- la solution optimale du problème relaxé de (P^{bin}_{seuil}),
- la solution optimale du problème suivant

$$\min \left\{ p(\mathbf{y}, \mathbf{y}', \mathbf{z}, \mathbf{z}') := (\sum_{j=1}^{n}(z_j + z'_j) - \sum_{j=1}^{n}(z_j - z'_j)^2) : (\mathbf{y}, \mathbf{y}', \mathbf{z}, \mathbf{z}') \in \mathcal{A} \right\}.$$

4.5.2.2 Relancer DCA

Du point de vue théorique, il existe $t_0 > 0$ tel que pour chaque $t > t_0$, les problèmes (P^{bin}_{seuil}) et (P^{DC}_{seuil}) soient équivalents. Pourtant, le calcul de la valeur exacte de t_0 n'est pas facile. Nous avons testé plusieurs valeurs différentes et enfin nous avons choisi une valeur pour laquelle $(\mathbf{y}+\mathbf{y}')^t Q(\mathbf{y}+\mathbf{y}')$ soit aussi petit que possible tout en essayant d'obtenir les valeurs binaires pour $\mathbf{z}$ et $\mathbf{z}'$. Pour les valeurs de t que nous avons choisies, le premier objectif était atteint mais toutes les variables $\mathbf{z}$ et $\mathbf{z}'$ n'étaient pas binaires. Afin d'obtenir des solutions binaires nous avons relancé DCA. La procédure consiste d'abord à résoudre le problème (P^{DC}_{seuil}) par DCA. Soit $(\mathbf{y}, \mathbf{y}', \mathbf{z}, \mathbf{z}')^1$ la solution fournie par DCA. Si $(\mathbf{z}, \mathbf{z}') \in \{0,1\}^{2n}$, nous ne relançons pas l'algorithme DCA. Supposons qu'il existe un k $(1 \leq k \leq 1)$ tel que z_k soit très proche de 1 (ou respectivement de 0), alors nous ajoutons la contrainte $z_k = 1$ (ou respectivement $z_k = 0$) au problème (P^{DC}_{seuil}). Après avoir effectué ces changements, nous relançons DCA pour résoudre le problème modifié. Nous utilisons $(\mathbf{y}, \mathbf{y}', \mathbf{z}, \mathbf{z}')^1$ pour démarrer DCA. De cette manière, nous trouvons des solutions binaires.

4.5.3 Résolution globale de (P_{seuil})

4.5.3.1 Algorithme par séparation et évaluation (SE)

Afin d'évaluer les solutions fournies par DCA et de trouver la solution globale de (P_{seuil}), un algorithme par séparation et évaluation (SE) a été utilisé. Cet algorithme résout le problème relaxé de (P_{seuil}) pour trouver la borne inférieure. Plus précisément, la résolution du problème suivant donne la borne inférieure

$$
\begin{aligned}
(P^{relaxe}_{seuil}) : & \\
& \min\ (\mathbf{y}+\mathbf{y}')^t \mathbf{Q} (\mathbf{y}+\mathbf{y}') \\
& s.c. \\
& \qquad \sum_{j=1}^{n} r_j (y_j + y'_j) \;=\; R, \\
& \qquad \sum_{j=1}^{n} (y_j - y'_j) \;=\; 1, \\
& \qquad 0 \leq y_j \leq b_j \qquad\qquad : j = 1, \ldots, n, \\
& \qquad c_j \leq y'_j \leq 0 \qquad\qquad : j = 1, \ldots, n.
\end{aligned}
$$

Le problème (P^{relaxe}_{seuil}) s'obtient après que nous supprimons les contraintes de complémentarité, i.e.

$$y_j y'_j = 0 \qquad : j = 1, \ldots, n,$$

et remplaçons les contraintes suivantes

$$y_j \in \{0\} \cup [a_j, b_j], y'_j \in \{0\} \cup [c_j, d_j], \qquad j = 1, \ldots, n,$$

par

$$0 \leq y_j \leq b_j, c_j \leq y'_j \leq 1, \qquad j = 1, \ldots, n.$$

La borne supérieure est mise à jour lorsqu'une meilleure solution du problème (P_{seuil}) est trouvée. La séparation est effectuée sur les variables qui ne respectent pas soit les contraintes de complémentarité soit les contraintes de seuil, de façon que, soit

$$y_j = 0, y'_j = 0,$$

soit

$$y_j = 0, y'_j \in [c_j, 0],$$

soit

$$y'_j = 0, y_j \in [0, b_j].$$

4.5.3.2 Approche combinée pour résoudre (P_{seuil})

Pour trouver la solution optimale de (P_{seuil}) et améliorer la performance de l'algorithme par séparation et évaluation, nous l'avons combiné avec DCA. A chaque itération de l'algorithme par séparation et évaluation, si les conditions sont favorables, DCA est relancé. Le but est de trouver une meilleure solution. Cette solution sert à améliorer la borne supérieure. De cette façon, la convergence de SE sera plus rapide. DCA est relancé, soit juste après la première itération de SE, soit le moment où l'indice de la variable, sur laquelle la séparation va être effectuée, est suffisamment grand (par exemple supérieur ou égal à $3n/4$). Cette condition sert à éviter les cas de sur-utilisation de DCA. Également, nous avons plus de chances de trouver de solutions qui sont admissibles au problème (P^{bin}_{seuil}).

Le processus de la méthode combinée se résume ainsi :
Après chaque itération, si les conditions sont favorables (i.e., les conditions citées ci-dessus), alors

1. Construire le sous-problème DC et nommer le *sous-problème relaxé (SPR)*. Ce sous-problème correspond à celui de SE.
2. Résoudre (SPR) par DCA. Le point initial de DCA est la solution optimale du sous-problème de SE. Soit $(\mathbf{y}, \mathbf{y}', \mathbf{z}, \mathbf{z}')^1$ la solution fournie par DCA.
3. Pour chaque $i = 1, \ldots, n$,
   (a) si $z_i^1 \geq 0.5$ ajouter la contrainte $z_i = 1$ au problème (SPR) sinon ajouter la contrainte $z_i = 0$ au (SPR),
   (b) si $z_i'^1 \geq 0.5$ ajouter la contrainte $z'_i = 1$ au problème (SPR) sinon ajouter la contrainte $z'_i = 0$ au (SPR).
4. Relancer DCA pour résoudre le nouveau sous-problème en utilisant $(\mathbf{y}, \mathbf{y}', \mathbf{z}, \mathbf{z}')^1$ comme le point initial.
   (a) Si le nouveau sous-problème n'est pas réalisable, retourner à SE,

(b) sinon, soit $(\mathbf{y}, \mathbf{y}', \mathbf{z}, \mathbf{z}')^2$ la solution fournie par DCA. Si $(\mathbf{y}, \mathbf{y}', \mathbf{z}, \mathbf{z}')^2$ n'est pas une solution améliorante pour (P_{seuil}) alors aller à la prochaine étape, sinon faites une mise à jour de la borne supérieure et de la solution optimale.

5. Continuer l'approche SE.

4.5.3.3 Expérience numériques

Les algorithmes ont été codés avec le langage C++ et testés sur un ordinateur Pentium de 3GHz, 1Gb RAM. Pour résoudre les (sous-)problèmes quadratiques nous avons utilisé le logiciel CPLEX en version 9.1.

Les données que nous avons utilisées correspondent aux prix hebdomadaires des actifs financiers du mois de mars 1992 au mois de septembre 1997. Les indices utilisés sont $S\&P$ 100 aux états-unis et DAX 100 en Allemagne. Les nombres d'actifs sont 98 et 85, respectivement. Pour tous les tests réalisés, $a_j = 0.05, b_j = 1.0, c_j = -1.0, d_j = -0.0001$. La tolérance autorisée pour l'écart entre les bores supérieure et inférieure (i.e., ϵ) est 10^{-7}. Les valeurs choisies pour le paramètre de pénalité sont égales à 0.2×10^{-5} pour $S\&P$ 100 et à 0.2×10^{-4} pour DAX 100.

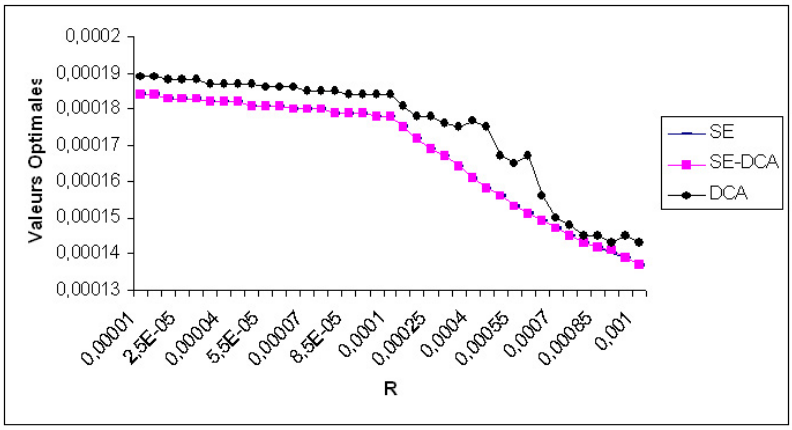
L'algorithme SE et l'approche combinée (SE-DCA) s'arrête si l'écart entre les bornes est inférieure à ϵ.

Les tests numériques ont été faits pour différentes valeurs de R. Nous avons choisi plus de 30 valeurs possibles de R. Les valeurs pour lesquelles les problèmes étaient réalisables appartenaient à l'intervalle suivant

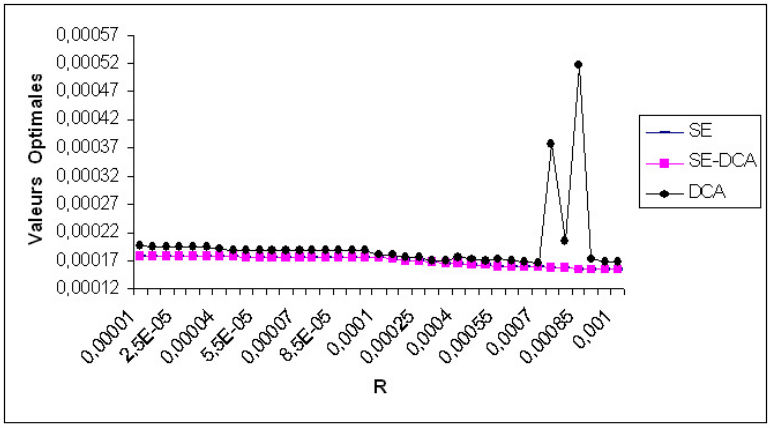
$$\min_{j=1,\ldots,n} r_j \leq R \leq \max_{j=1,\ldots,n} r_j.$$

Les figures (4.1-4.4) présentent les résultats sur deux jeux de données. Plus précisément, les figures 4.1 sont consacrées aux comparaisons des valeurs optimales fournies par DCA avec celles d'optimales globales. Les figures montrent que les solutions fournies par DCA sont très proches des solutions globales. L'efficacité de DCA est plus visible lorsque nous comparons les temps CPU des algorithmes. Les temps d'exécutions sont montrés sur les figures 4.2. En comparant les temps d'exécutions et les nombres d'itération de l'algorithme SE et l'approche combinée, nous constatons l'influence de DCA sur la convergence de l'approche combinée. Les figures (4.3,4.2) comparent le nombre d'itération et le temps d'exécution de chaque algorithme avec les autres. DCA s'arrête après moins de 10 itérations. Nous constatons que l'approche combinée est, en général, trois ou quatre fois plus rapide que SE. Les figures 4.4 présentent le nombre de relance de DCA en cours d'exécution de l'approche combinée.

Toutes les expérimentations étaient avec $\epsilon = 10^{-7}$. Si nous réduisons la précision, c'est-à-dire, si nous augmentons ϵ de 10^{-7} à $0.5 * 10^{-5}$, l'influence de DCA sur l'efficacité de l'approche combinée par rapport à SE devient plus visible. Les tableaux (4.4,4.5) présentent les résultats dans les mêmes conditions que les expériences précédentes sauf que ϵ est $0.5*10^{-5}$ et nous avons simplifié les conditions pour que DCA redémarre, de sorte que nous avons



(a) S&P 100



(b) Dax 100

FIG. 4.1 – Les optimums globaux et les valeurs optimales fournies par DCA

remplacé la condition sur l'indice de variable de séparation (i.e., $3n/4$) par $(n/2)$. D'après les tableaux, l'approche combinée trouve la solution optimale en une seule itération pour un nombre important de valeurs de R. La convergence rapide de l'approche combinée est liée à l'efficacité de DCA. En comparant les nombres d'itération et les temps CPU de l'algorithme par SE avec ceux de la méthode combinée, nous constatons l'efficacité de la méthode. Pour presque toutes les valeurs de R, la méthode combinée trouve la solution globale en moins d'une minute, tandis que SE a besoin de beaucoup plus de temps afin de trouver la solution globale.

4.6 Conclusion

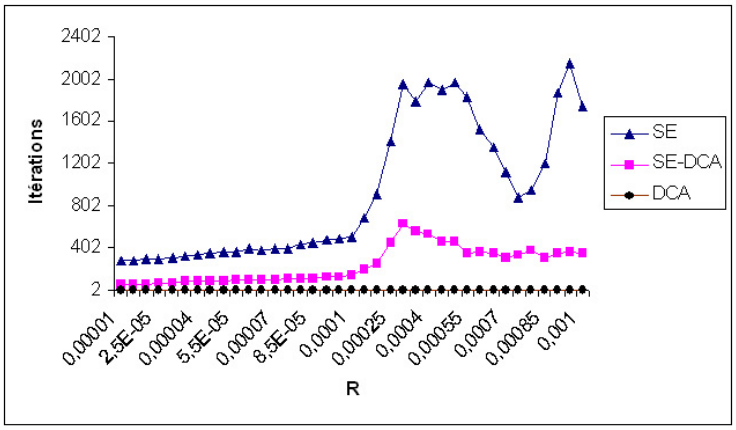
Dans ce chapitre, nous avons rappelé le modèle classique Moyenne-Variance (MV) de Markowitz. Nous remarquons que ce modèle ne prend pas en considération certains types de

	DCA			SE			SE-DCA			
R	iter	Val. Opt.	CPU	iter	Val. Opt.	CPU	iter	relance	Val. Opt.	CPU
0.00001	6	0.000189	1.078	689	0.000184	297.329	1	1	0.000184	1.094
0.00002	6	0.000188	1.094	723	0.000183	309.828	1	1	0.000184	1.093
0.00003	7	0.000188	1.250	775	0.000183	333.391	1	1	0.000183	1.094
0.00004	6	0.000187	1.079	842	0.000182	359.750	1	1	0.000182	1.110
0.00005	6	0.000187	1.078	882	0.000181	377.312	1	1	0.000182	1.094
0.00006	7	0.000186	1.219	966	0.000181	413.906	1	1	0.000181	1.109
0.00007	7	0.000185	1.250	963	0.000180	415.594	1	1	0.000180	1.094
0.00008	7	0.000185	1.250	1047	0.000179	457.250	46	3	0.000180	21.594
0.00009	7	0.000184	1.250	1182	0.000179	509.625	46	3	0.000179	21.703
0.0001	7	0.000184	1.234	1250	0.000178	568.469	46	3	0.000178	21.609
0.0002	7	0.000178	1.234	2254	0.000172	992.360	74	29	0.000174	57.891
0.0003	7	0.000176	1.235	4863	0.000167	2105.859	89	13	0.000167	48.297
0.0004	7	0.000177	1.234	4884	0.000161	2153.968	57	3	0.000162	26.078
0.0005	7	0.000167	1.391	4850	0.000156	2151.125	70	3	0.000157	31.578
0.0006	8	0.000167	1.438	3802	0.000151	1707.750	24	1	0.000152	10.531
0.0007	11	0.000150	1.359	2740	0.000147	1241.594	57	3	0.000148	25.750
0.0008	7	0.000145	1.219	2339	0.000143	1058.672	22	1	0.000145	10.703
0.0009	7	0.000143	1.422	4642	0.000140	2092.094	102	3	0.000142	43.969
0.001	8	0.000143	1.375	4305	0.000137	1936.046	165	3	0.000139	70.047

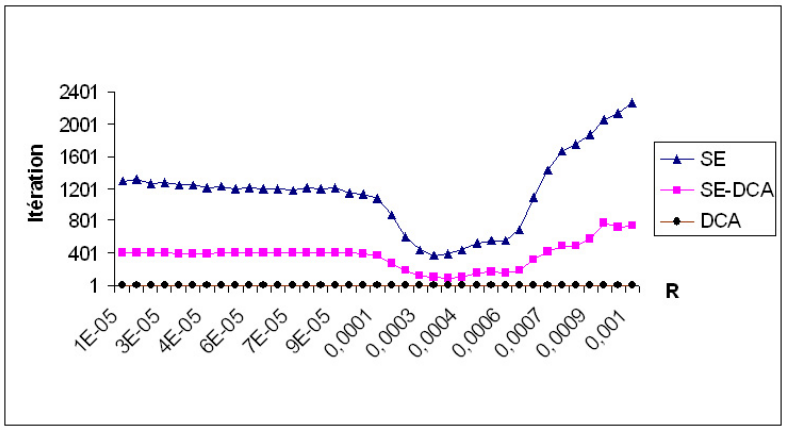
TAB. 4.4 – La performance des algorithmes pour le premier jeu de données (S&P 100) en utilisant l'algorithme par SE et l'approche combinée

	DCA			SE			SE-DCA			
R	iter	Val. Opt.	CPU	iter	Val. Opt.	CPU	iter	relance	Val. Opt.	CPU
0.00001	6	0.000195	0.235	4467	0.000178	1356.859	61	3	0.000178	19.844
0.00002	6	0.000194	0.235	4352	0.000177	1315.000	61	3	0.000178	19.641
0.00003	6	0.000194	0.235	4290	0.000177	1306.469	57	3	0.000177	18.687
0.00004	7	0.000190	0.235	4187	0.000177	1269.719	39	1	0.000177	12.031
0.00005	6	0.000189	0.235	4133	0.000176	1241.953	36	1	0.000177	11.047
0.00006	6	0.000188	0.235	4107	0.000176	1236.203	14	1	0.000176	4.641
0.00007	6	0.000188	0.235	4083	0.000176	1238.281	14	1	0.000176	4.672
0.00008	6	0.000187	0.235	4129	0.000175	1269.781	14	1	0.000175	4.610
0.00009	6	0.000187	0.235	3991	0.000175	1224.282	14	1	0.000175	4.719
0.0001	6	0.000181	0.235	3736	0.000174	1155.343	14	1	0.000175	4.672
0.0002	6	0.000176	0.235	2063	0.000170	653.485	1	1	0.000171	0.797
0.0003	6	0.000171	0.235	1305	0.000167	417.844	1	1	0.000167	0.781
0.0004	6	0.000175	0.235	1496	0.000164	470.157	1	1	0.000164	0.781
0.0005	6	0.000170	0.235	1900	0.000162	597.390	38	3	0.000163	13.265
0.0006	7	0.000170	0.235	2428	0.000159	763.063	42	7	0.000159	17.359
0.0007	6	0.000165	0.235	4924	0.000158	1573.516	42	7	0.000158	17.313
0.0008	6	0.000204	0.235	6023	0.000156	1926.531	42	7	0.000156	17.734
0.0009	7	0.000173	0.235	7212	0.000154	2299.563	103	13	0.000155	40.344
0.001	7	0.000168	0.235	7825	0.000153	2547.953	49	7	0.000153	19.329

TAB. 4.5 – La performance des algorithmes pour le deuxième jeu de données (Dax 100) en utilisant l'algorithme par SE et l'approche combinée



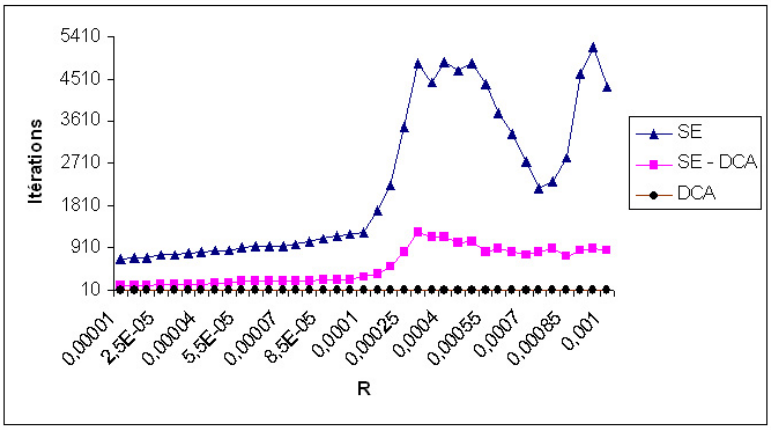
(a) S&P 100



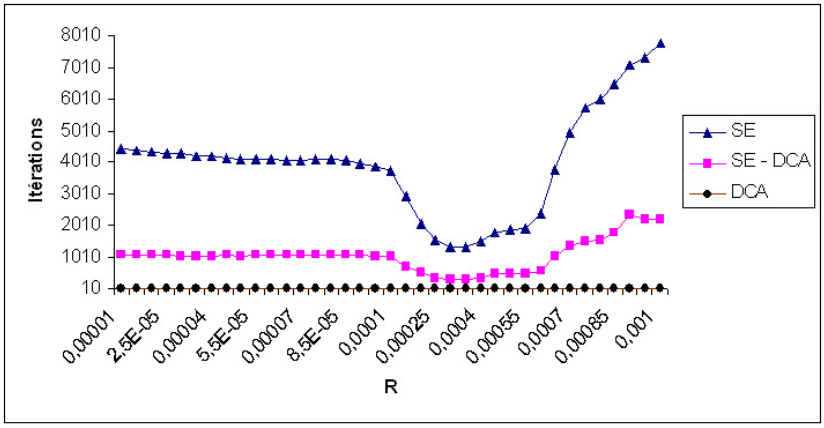
(b) Dax 100

FIG. 4.2 – CPU en seconde

contraintes du monde réel. Ce sont les contraintes de cardinalité, de seuil d'achat et de seuil. Après avoir ajouté ces contraintes, nous avons exprimé le modèle correspondant sous forme d'un programme mixte en variables binaires. En utilisant un résultat de pénalité exacte basée sur la programmation DC, nous avons reformulé le programme mixte en variables binaire sous forme d'un programme DC. Ensuite, DCA s'est appliqué afin de résoudre le problème DC. Les résultats présentés dans ce chapitre montrent l'efficacité et la performance de DCA pour résoudre les problèmes qui viennent d'être cités. Afin de trouver les solutions globales, une méthode combinée des algorithmes SE et DCA a été introduite. D'après les résultats numériques nous constatons la convergence rapide et la supériorité de la méthode combinée par rapport à l'algorithme SE.

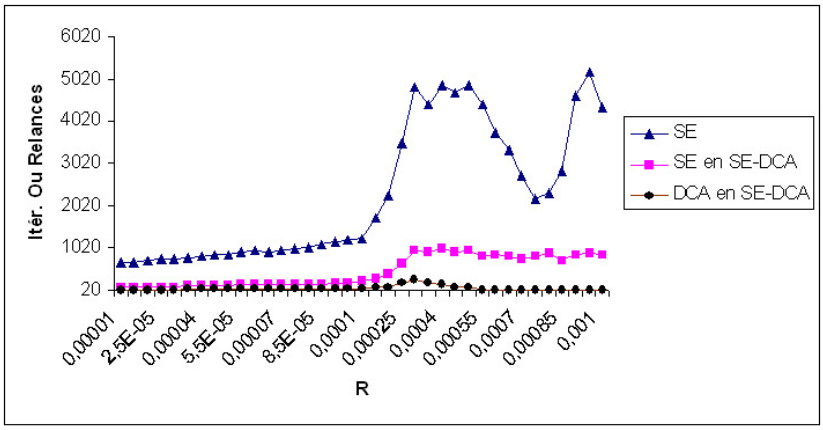


(a) S&P 100

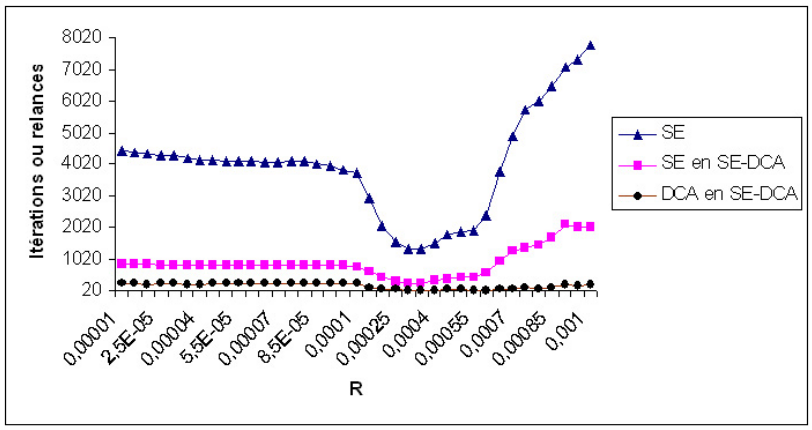


(b) Dax 100

FIG. 4.3 – Le nombre d'itération de chaque algorithme



(a) S&P 100



(b) Dax 100

FIG. 4.4 – Le nombre de relance de DCA pendant l'exécution de l'approche combinée (SE-DCA)

Chapitre 5

Gestion de portefeuille avec la mesure de risque de baisse sous les contraintes de cardinalité

Résumé *Ce chapitre concerne une nouvelle approche continue basée sur la programmation DC et DCA pour la résolution du problème de gestion de portefeuille avec la mesure de risque de baisse sous les contraintes de cardinalité. Le modèle sous les contraintes de cardinalité s'écrit sous la forme d'un problème d'une programmation mixte en variables binaires. Après avoir mis le modèle sous la forme DC, nous avons appliqué DCA pour le résoudre. Enfin, le problème a été résolu avec un algorithme par Séparation et Evaluation (SE), pour évaluer les résultats obtenus. Une méthode combinée basée sur DCA et SE a été développée pour obtenir les solutions globales tout en essayant d'établir une convergence rapide. Les résultats numériques confirment l'efficacité de notre approche.*

5.1 Introduction

La variance est une mesure de risque classique très utilisée. Elle représente les écarts des rendements par rapport à la moyenne.

Définition 5.1.1 *Soit f la fonction de densité d'une variable aléatoire comme r, on définit la variance de r par*

$$Var(r) = \sigma^2(r) := \sum_r (r - \mathbb{E}(r))^2 f(r) \tag{5.1}$$

où $\mathbb{E}$ est l'opérateur d'espérance mathématique et la somme est effectuée sur toutes les valeurs possibles de r. □

D'après la formule 5.1, la variance ne fait aucune différence entre les rendements inférieures à la moyenne et les rendements supérieures à la moyenne, car la variance prend le carrée des

écarts en compte. De plus, le poids de chaque écart est sa probabilité d'occurrence. Tandis que parmi les écarts avec les mêmes probabilités d'occurrence, les investisseurs préfèrent ceux positifs, i.e. les écarts qui ont des rendements supérieures à la moyenne. Au contraire, les investisseurs n'aiment pas les écarts négatifs. Alors les écarts négatifs sont considérés comme le risque. Markowitz a proposé la *semivariance* pour pallier le handicap de la variance [113]. La semivariance mesure les écarts négatifs et elle se place dans le cadre des *mesures de risque de baisse* ([27], [122]). Ces mesures de risque considèrent les écarts inférieures à un *objectif* comme le risque. La semivariance est la mesure de risque pour laquelle la moyenne est considérée comme l'objectif. Les mesures de baisse sont utilisées surtout pour les distributions non-normales [27].

Ce chapitre a pour le but d'étudier et de résoudre un cas général d'un modèle dont la mesure de risque est celle de baisse. Le modèle généralisé contient des contraintes de cardinalité. Le modèle a des avantages, le premier étant le choix de la mesure de risque de baisse. Le deuxième avantage est lié à la reformulation du modèle car contrairement au modèle MV il se formule de façon que nous n'ayons plus besoin de la matrice de variance-covariance. Finalement, il existe des contraintes de cardinalité qui limitent le nombre des actifs choisis dans le portefeuille optimal.

Le reste du chapitre est organisé de la façon suivante : Dans la section 5.2, la formulation du modèle de choix de portefeuille sera présenté. Le modèle sera généralisé en section 5.3, afin de prendre les contraintes de cardinalité en compte. En section 5.4, après avoir mis le problème sous la forme d'un modèle de programmation DC, nous utiliserons DCA pour le résoudre. Un algorithme combiné de DCA et l'algorithme par Séparation et Évaluation (SE) est présenté dans la section 5.5, tandis que les résultats numériques sont reportés dans la section 5.6. Nous terminons le chapitre par une conclusion.

5.2 Description et formulation

D'abord, nous considérons le modèle MV qui cherche à répartir le mieux possible le capital parmi n actifs financiers. Supposons qu'une liste des rendements historiques, pour une durée de m intervalles de temps, soit disponible. Le rendement moyen de l'actif i se calcule par

$$r_i := \frac{1}{m}\sum_{j=1}^{m} r_{ij}$$

où r_{ij} est le rendement de l'actif i dans l'intervalle $[j-1, j]$, tel que $i = 1, \ldots, n$ et $j = 1, \ldots, m$. Soient $\mathbf{r} = (r_1, r_2, \ldots, r_n)^t$ et $\mathbf{y} = (y_1, y_2, \ldots, y_n)^t$ le vecteur des variables de décision représentant les proportions du capital investi dans les actifs. Nous savons que le modèle MV peut s'écrire sous la forme suivante ([8]) :

$$min \left\{ V(\mathbf{y}) := \mathbf{y}^t\mathbf{Q}\mathbf{y} : \sum_{j=1}^{n} r_j y_j = R, \sum_{j=1}^{n} y_j = 1, y_j \geq 0,\ j = 1, \ldots, n \right\}. \tag{5.2}$$

La solution de ce modèle minimise le risque défini par $\mathbf{y}^t\mathbf{Q}\mathbf{y}$ pour un gain fixé auparavant, R. $\mathbf{Q}$ est la matrice de Variance-Covariance dont l'élément (i,j) est calculé par

$$\sigma_{i,j} := \frac{1}{m}\sum_{k=1}^{m}(r_{ik} - r_i)(r_{jk} - r_j). \tag{5.3}$$

Le programme 5.2 est un modèle de programmation quadratique pour lequel des algorithmes efficaces existent.

Théorème 5.1 *Le modèle (5.2) est équivalent à*

$$min\left\{ V(\mathbf{y}) := (1/m)\sum_{j=1}^{m}\left[\sum_{i=1}^{n} r_{ij}y_i - R\right]^2 : \sum_{i=1}^{n} r_i y_i = R, \sum_{i=1}^{n} y_i = 1, y_i \geq 0, : i = 1,\ldots,n \right\}. \tag{5.4}$$

Preuve : Soit A une matrice $m \times n$ dont l'élément (j,i) est a_{ji} et est défini par

$$a_{ji} := r_{ij} - r_i$$

ensuite,

$$\sum_{j=1}^{m} a_{ji}^2 = \sum_{j=1}^{m}(r_{ij} - r_i)^2 \quad : \quad \forall i = 1,\ldots,n, \tag{5.5}$$

qui sont les éléments sur la diagonale de la matrice A^tA et l'élément non-diagonal (i,k) de cette matrice se calcule par

$$\sum_{j=1}^{m} a_{ji}a_{jk} = \sum_{j=1}^{m}(r_{ij} - r_i)(r_{kj} - r_k). \tag{5.6}$$

Si nous comparons les relations (5.5) et (5.6) avec la définition des éléments de la matrice de variance-covariance, nous constatons que le i-ème élément sur la diagonale de A^tA est $m\sigma_i^2$ et l'élément (i,k) (non-diagonal) de A^tA est égal à $m\sigma_{ik}$. Le résultat est de $A^tA = m\mathbf{Q}$. Par ailleurs,

$$\mathbf{y}^t\mathbf{Q}\mathbf{y} = \frac{1}{m}\mathbf{y}^t(A^tA)\mathbf{y} = \frac{1}{m}(\mathbf{y}^tA^t)(A\mathbf{y}) = \frac{1}{m}(A\mathbf{y})^tA\mathbf{y}$$

et

$$A\mathbf{y} = \begin{pmatrix} a_{11} & \ldots & a_{1n} \\ a_{21} & \ldots & a_{2n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \ldots & a_{mn} \end{pmatrix}\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} a_{1i}y_i \\ \sum_{i=1}^{n} a_{2i}y_i \\ \vdots \\ \sum_{i=1}^{n} a_{mi}y_i \end{pmatrix}$$

alors,

$$\mathbf{y}^t\mathbf{Q}\mathbf{y} = \frac{1}{m}\left(\sum_{i=1}^n a_{1i}y_i, \quad \sum_{i=1}^n a_{2i}y_i, \quad \dots \quad \sum_{i=1}^n a_{mi}y_i \right) \begin{pmatrix} \sum_{i=1}^n a_{1i}y_i \\ \sum_{i=1}^n a_{2i}y_i \\ \vdots \\ \sum_{i=1}^n a_{mi}y_i \end{pmatrix}$$

ou

$$\mathbf{y}^t\mathbf{Q}\mathbf{y} = \frac{1}{m}\sum_{j=1}^m \left[\sum_{i=1}^n a_{ji}y_i\right]^2.$$

Puisque $a_{ji} := r_{ij} - r_i$ alors

$$\mathbf{y}^t\mathbf{Q}\mathbf{y} = \frac{1}{m}\sum_{j=1}^m \left[\sum_{i=1}^n (r_{ij} - r_i)y_i\right]^2 = \frac{1}{m}\sum_{j=1}^m \left[\sum_{i=1}^n r_{ij}y_i - \sum_{i=1}^n r_i y_i\right]^2.$$

D'ailleurs

$$\sum_{i=1}^n r_i y_i = R$$

ce qui donne le résultat suivant

$$\mathbf{y}^t\mathbf{Q}\mathbf{y} = \frac{1}{m}\sum_{j=1}^m \left[\sum_{i=1}^n r_{ij}y_i - R\right]^2.$$

Cette relation complète la démonstration du théorème. □

La représentation (5.4) montre que le calcule du risque $V(\mathbf{y})$ consiste à trouver le rendement de portefeuille à chaque intervalle de temps et à calculer son écart par rapport au gain fixé.

Soit

$$R_j := \sum_{i=1}^n r_{ij}y_i$$

le rendement du portefeuille $\mathbf{y} = (y_1, y_2, \dots, y_n)^t$ en intervalle j. Alors

$$V(\mathbf{y}) := \mathbf{y}^t\mathbf{Q}\mathbf{y} = \frac{1}{m}\sum_{j=1}^m [R_j - R]^2.$$

Le premier constat est l'indifférence entre les cas $R_j > R$ et $R_j < R$. Pourtant, un investisseur, en général, préfère le portefeuille pour lequel $R_j > R$. Car il signifie que le portefeuille a un rendement supérieur au gain fixé. Cet investisseur considère le cas $R_j < R$ comme le risque. Afin de distinguer les cas $R_j > R$ et $R_j < R$ et de considérer $R_j < R$ comme le risque, l'approche consiste à définir la mesure de risque par

$$V(\mathbf{y}) := \frac{1}{m}\sum_{j=1}^m [min(0, R_j - R)]^2.$$

Cette formule explique une *mesure risque de baisse* pour laquelle l'objectif est le rendement attendu du portefeuille (i.e., R). Pour le cas général, l'objectif n'est pas impérativement le rendement attendu (le gain fixé) du portefeuille ([27, 33, 65, 122]).

Avec la nouvelle mesure de risque, le modèle à traiter est

$$
\begin{aligned}
(P): \quad & \\
& \min\ V(y) := (1/m)\sum_{j=1}^{m}[min(0, R_j - R)]^2 \\
& s.c. \\
& \sum_{i=1}^{n} r_i y_i = R, \\
& \sum_{i=1}^{n} y_i = 1, \\
& y_i \geq 0 \qquad\qquad : i = 1, \ldots, n.
\end{aligned}
$$

Soient $\mathbf{x} = (x_1, \ldots, x_m)$ des variables réelles telles que

$$-x_j := min(0, R_j - R)$$

alors, pour chaque $j = 1, \ldots, m$ nous avons

$$x_j = \begin{cases} 0, & si \quad R_j - R \leq 0, \\ R_j - R, & sinon. \end{cases} \tag{5.7}$$

Les résultats immédiats de ces relations sont les contraintes $x_j \geq 0$ et $x_j + \sum_{i=1}^{n} r_{ij} y_i \geq R$ qui doivent être ajoutées au modèle. En remplaçant $R_j - R$ par x_j dans le modèle (P), le nouveau problème s'exprime ainsi

$$
\begin{aligned}
(P'): \quad & \\
& \min\ \theta(\mathbf{x}, \mathbf{y}) := (1/m)\sum_{j=1}^{m} x_j^2 \\
& s.c. \\
& \sum_{i=1}^{n} r_i y_i = R, \\
& \sum_{i=1}^{n} y_i = 1, \\
& x_j + \sum_{i=1}^{n} r_{ij} y_i \geq R \qquad : j = 1, \ldots, m, && (5.8) \\
& y_i \geq 0 \qquad\qquad : i = 1, \ldots, n, && (5.9) \\
& x_j \geq 0 \qquad\qquad : j = 1, \ldots, m. && (5.10)
\end{aligned}
$$

Théorème 5.2 *Les deux problèmes* (P) *et* (P') *sont équivalents dans le sens où*

- *si* $\mathbf{y}^*$ *est l'optimum de* (P)*, alors* $(\mathbf{x}^*, \mathbf{y}^*)$ *est celui de* (P')*, avec* $-x_j^* = min(0, R_j^* - R)$ *;*
- *si* $(\mathbf{x}^*, \mathbf{y}^*)$ *est l'optimum de* (P')*, alors* $\mathbf{y}^*$ *est celui de* (P)*.*

Preuve : Soient $(\mathbf{x}^*, \mathbf{y}^*)$ la solution optimale de (P'). Clairement

$$-x_j^* \leq R_j^* - R \quad \text{et} \quad -x_j^* \leq 0.$$

Cela veut dire $-x_j^* \leq min(0, R_j^* - R)$. Nous allons démontrer qu'il faut $-x_j^* = min(0, R_j^* - R)$. S'il existe $k (1 \leq k \leq m)$ tel que $-x_k^* < min(0, R_k^* - R)$, alors $(\mathbf{x}^*, \mathbf{y}^*)$ ne peut pas être la solution optimale de (P') car $\mathbf{x} = (x_1^*, \ldots, \widetilde{x}_k, \ldots, x_m^*, y_1, \ldots, y_n)$ où $\widetilde{x}_k := min(0, R_k^* - R)$ est admissible pour (P') et

$$\sum_{j=1}^{m} (x_j^*)^2 \geq \sum_{j \neq k, j=1}^{m} (x_j^*)^2 + (\widetilde{x}_k)^2$$

cette contradiction nous montre que :

$$-x_j^* = min(0, R_j^* - R) \quad : \quad \forall j.$$

Si la solution optimale de (P) n'est pas $\mathbf{y}^*$ alors il existe une solution admissible pour (P) comme $\widehat{\mathbf{y}}$ telle que

$$(1/m) \sum_{j=1}^{m} \left[min(0, R_j^* - R) \right]^2 > (1/m) \sum_{j=1}^{m} \left[min(0, \widehat{R}_j - R) \right]^2$$

où $\widehat{R}_j := \sum_{i=1}^{n} r_{ij} \widehat{y}_i$. $(\widehat{\mathbf{x}}, \widehat{\mathbf{y}})$ est une solution admissible de (P') et

$$-\widehat{\mathbf{x}} := min(0, \widehat{R}_j - R).$$

Puisque $(\mathbf{x}^*, \mathbf{y}^*)$ est optimal pour (P') alors

$$(1/m) \sum_{j=1}^{m} \widehat{x}_j^2 \geq (1/m) \sum_{j=1}^{m} (x_j^*)^2.$$

Mais

$$(1/m) \sum_{j=1}^{m} \widehat{x}_j^2 = (1/m) \sum_{j=1}^{m} \left[min(0, \widehat{R}_j - R) \right]^2$$

et

$$(1/m) \sum_{j=1}^{m} \left[min(0, R_j^* - R) \right]^2 = (1/m) \sum_{j=1}^{m} (x_j^*)^2.$$

Ce qui est en contradiction avec l'optimalité de $\widehat{\mathbf{y}}$. Nous pouvons alors conclure que si $(\mathbf{x}^*, \mathbf{y}^*)$ est une solution optimale de (P'), $\mathbf{y}^*$ est une solution optimale de (P).

D'une manière réciproque, soit $\mathbf{y}^*$ une solution optimale de (P) alors $(\mathbf{x}^*, \mathbf{y}^*)$ sera une solution admissible de (P'), où

$$-x_j^* := min(0, R_j^* - R) \quad j = 1, \ldots, m.$$

Si $(\mathbf{x}^*, \mathbf{y}^*)$ n'est pas l'optimum de (P') alors il existe $(\widehat{\mathbf{x}}, \widehat{\mathbf{y}})$ telle qu'elle est une solution optimale pour (P') et

$$-\widehat{x}_j := min(0, \widehat{R}_j - R) \quad j = 1, \ldots, m.$$

$\mathbf{y}^*$ est l'optimum de (P) et $\widehat{\mathbf{y}}$ est une solution admissible de (P) alors

$$(1/m) \sum_{j=1}^{m} \left[min(0, R_j^* - R)\right]^2 \leq (1/m) \sum_{j=1}^{m} \left[min(0, \widehat{R}_j - R)\right]^2$$

ou

$$(1/m) \sum_{j=1}^{m} (x_j^*)^2 \leq (1/m) \sum_{j=1}^{m} (\widehat{x}_j)^2$$

ce qui est en contradiction avec l'optimalité de $(\widehat{\mathbf{x}}, \widehat{\mathbf{y}})$ pour (P). Le résultat est que $(\mathbf{x}^*, \mathbf{y}^*)$ est l'optimum de (P').

En bref, nous avons démontré que les deux problèmes (P) et (P') sont équivalents tels que
• si $\mathbf{y}^*$ est l'optimum de (P), alors $(\mathbf{x}^*, \mathbf{y}^*)$ est celui de (P'), avec $-x_j^* = min(0, (R_j^* - R))$;
• si $(\mathbf{x}^*, \mathbf{y}^*)$ est l'optimum de (P'), alors $\mathbf{y}^*$ est celui de (P). □

L'ensemble des solutions admissibles de (P') est borné par rapport à $\mathbf{y}$, pour qu'il soit aussi borné par rapport à $\mathbf{x}$ nous allons imposer certaines contraintes tout en gardant l'équivalence entre (P) et (P').

Nous avons démontré que pour chaque solution optimale, la condition suivante doit être satisfaite pour tout $j = 1, ..., m$

$$x_j = -\min\{0, R_j - R\} = \max\{0, R - R_j\} = \max\{0, R - \sum_{i=1}^{n} r_{ij} y_i\}.$$

Ce qui montre que nous pouvons considérer les bornes supérieures suivantes pour les variables x_j

$$\beta_j := \max\{0, \alpha_j\} : j = 1, ..., m$$

où

$$\alpha_j := max \left\{ R - \sum_{i=1}^{n} r_{ij} y_i : (\mathbf{x}, \mathbf{y}) \in \mathbb{R}^n, \sum_{i=1}^{n} r_i y_i = R, \sum_{i=1}^{n} y_i = 1, x_j \geq 0, j = 1, \ldots, m \right\}.$$

Les bornes supérieures peuvent être calculées en résolvant les programmes linéaires, cités ci-dessus.

Après avoir ajouté les nouvelles contraintes, nous obtenons le problème suivant

$$
\begin{aligned}
(P'') : \\
&\min\ \theta(\mathbf{x},\mathbf{y}) := (1/m)\sum_{j=1}^{m} x_j^2 \\
&s.c. \\
&\sum_{i=1}^{n} r_i y_i = R, \\
&\sum_{i=1}^{n} y_i = 1, \\
&x_j + \sum_{i=1}^{n} r_{ij} y_i \geq R \qquad : j = 1,\ldots,m, \\
&y_i \geq 0 \qquad : i = 1,\ldots,n, \\
&0 \leq x_j \leq \beta_j \qquad : j = 1,\ldots,m.
\end{aligned}
$$

Le nouveau problème est équivalent aux (P') et (P). Soient $(\mathbf{x}^*,\mathbf{y}^*)$ et $(\widehat{\mathbf{x}},\widehat{\mathbf{y}})$ les optimums de (P') et (P''), respectivement. L'ensemble des solutions admissibles de (P'') est un sous-ensemble de celui de (P'), alors $\theta(\mathbf{x}^*,\mathbf{y}^*) \leq \theta(\widehat{\mathbf{x}},\widehat{\mathbf{y}})$. En outre, pour chaque $j = 1,\ldots,m$

$$x_j^* = \max\{0, R - \sum_{i=1}^{n} r_{ij} y_i^*\}.$$

Si $R - \sum_{i=1}^{n} r_{ij} y_i^* \leq 0$ alors $x_j^* = 0$ et ensuite $x_j^* \in [0,\beta_j]$. Si $R - \sum_{i=1}^{n} r_{ij} y_i^* > 0$ alors $x_j^* = R - \sum_{i=1}^{n} r_{ij} y_i^*$. D'après la définition de α_j nous avons $x_j^* \leq \alpha_j$. D'ailleurs, $\beta_j \geq \alpha_j$. Cela veut dire que $x_j^* \in [0,\beta_j]$. Nous en concluons l'equivalence entre les modèles (P') et (P'') car chaque solution optimale de (P') est une solution admissible de (P'').

5.3 Contraintes de cardinalité

Une généralisation du modèle (P'') consiste en l'introduction des contraintes de cardinalité. Ces contraintes limitent le nombre des actifs tenus dans le portefeuille. L'introduction de ces contraintes exige des contraintes de bornes. Ces contraintes limitent la proportion du capital investi dans chaque actif. Dans l'absence de vente-à-découvert, les bornes $0 \leq y_i \leq 1$ existent dans le modèle de façon naturelle. Soient a_i et b_i les paramètres associés aux bornes inférieure et supérieure sur l'investissement dans l'actif i. Il faut $0 \leq a_i \leq b_i \leq 1$. Soit '*card*'

le nombre des actifs que l'investisseur veut avoir dans le portefeuille. Si $a_i = 0$, alors *card* sera le nombre maximum d'actifs choisis dans lesquels une proportion positive du capital a été investi. Si $a_i \neq 0$ alors *card* donne le nombre exact des actifs choisis. Nous définissons z_i la variable de décision qui montre la présence ou l'absence de l'actif i dans le portefeuille. Si $z_i = 1$, l'actif i est inclus dans le portefeuille et $z_i = 0$ sinon. En utilisant ces notations supplémentaires, le modèle (P'') sous les contraintes de cardinalité s'exprime par

$$
\begin{aligned}
&(P_{card}) : \\
&\min\ \eta(\mathbf{x},\mathbf{y},\mathbf{z}) := (1/m)\sum_{j=1}^{m} x_j^2 \\
&s.c. \\
&\sum_{i=1}^{n} r_i y_i = R, \\
&\sum_{i=1}^{n} y_i = 1, \\
&x_j + \sum_{i=1}^{n} r_{ij} y_i \geq R \qquad : j = 1,\dots,m, \\
&0 \leq x_j \leq \beta_j \qquad : j = 1,\dots,m, \\
&\sum_{i=1}^{n} z_i = card, \qquad (5.11) \\
&a_i z_i \leq y_i \leq b_i z_i \qquad : i = 1\dots,n, \qquad (5.12) \\
&z_i \in \{0,1\} \qquad : i = 1,\dots,n. \qquad (5.13)
\end{aligned}
$$

A cause des contraintes 5.11,5.12 et 5.13 le problème (P) est difficile à résoudre. La plupart des méthodes utilisées pour résoudre un modèle de choix de portefeuille sous les contraintes de cardinalité s'appuient sur les méthodes heuristiques, comme algorithme génétique, recuit simulé et recherche tabou ([16, 29, 51, 154]). Notre approche pour résoudre le problème (P_{card}) consiste à utiliser la méthode DCA.

5.4 Programmation DC et DCA pour la résolution du problème

5.4.1 Reformulation

Afin de simplifier les formulations, nous définissons l'ensemble $\mathcal{A}$ comme la suite

$$
\mathcal{A} := \left\{ \begin{array}{l} (\mathbf{x},\mathbf{y},\mathbf{z}) \in \mathbb{R}^m \times \mathbb{R}^n \times [0,1]^n : \sum_{i=1}^{n} r_i y_i = R, \sum_{i=1}^{n} y_i = 1, \sum_{i=1}^{n} z_i = card, \\ x_j + \sum_{i=1}^{n} r_{ij} y_i \geq R, 0 \leq x_j \leq \beta_j, j = 1,\dots,m, a_i z_i \leq y_i \leq b_i z_i, i = 1,\dots,n \end{array} \right\}.
$$

Considérons la fonction de pénalité p définie par

$$p(\mathbf{x}, \mathbf{y}, \mathbf{z}) := \sum_{i=1}^{n} z_i(1 - z_i).$$

Clairement, la fonction p est concave, finie et non-négative sur $\mathcal{A}$. De plus, l'ensemble des solutions réalisables de (P_{card}) peut être écrit

$$\{(\mathbf{x}, \mathbf{y}, \mathbf{z}) \in \mathcal{A} : z_i \in \{0, 1\}\} = \{(\mathbf{x}, \mathbf{y}, \mathbf{z}) \in \mathcal{A} : p(\mathbf{x}, \mathbf{y}, \mathbf{z}) = 0\} = \{(\mathbf{x}, \mathbf{y}, \mathbf{z}) \in \mathcal{A} : p(\mathbf{x}, \mathbf{y}, \mathbf{z}) \leq 0\},$$

alors le problème (P_{card}) est simplifié sous la forme

$$\min \left\{ \eta(\mathbf{x}, \mathbf{y}, \mathbf{z}) := (1/m) \sum_{j=1}^{m} x_j^2 : (\mathbf{x}, \mathbf{y}, \mathbf{z}) \in \mathcal{A}, \quad p(\mathbf{x}, \mathbf{y}, \mathbf{z}) \leq 0 \right\}. \tag{5.14}$$

En utilisant le théorème 4.1 concernant la pénalité exacte en programmation DC (voir aussi [105]) sur le problème (5.14), nous allons le reformuler sous forme DC. Les conditions nécessaires pour pouvoir utiliser ce résultat est de l'existence d'une fonction de pénalité comme $p(\mathbf{x}, \mathbf{y}, \mathbf{z})$ qui est concave, finie et non-négative sur le polyèdre $\mathcal{A}$; en outre, la convexité de la fonction $\eta(\mathbf{x}, \mathbf{y}, \mathbf{z})$. D'après le théorème, il existe un $t_0 \geq 0$ tel que pour tout $t > t_0$ le problème (5.14) soit équivalent au problème suivant

$$\min \left\{ F(\mathbf{x}, \mathbf{y}, \mathbf{z}) := (1/m) \sum_{j=1}^{m} x_j^2 + tp(\mathbf{x}, \mathbf{y}, \mathbf{z}) : (\mathbf{x}, \mathbf{y}, \mathbf{z}) \in \mathcal{A} \right\}. \tag{5.15}$$

La fonction F est convexe par rapport aux $\mathbf{x}$ et $\mathbf{y}$ mais concave par rapport à $\mathbf{z}$, par conséquent F est une fonction DC avec la décomposition DC suivante

$$(P_{DC}) : \quad \min \left\{ g(\mathbf{x}, \mathbf{y}, \mathbf{z}) - h(\mathbf{x}, \mathbf{y}, \mathbf{z}) : (\mathbf{x}, \mathbf{y}, \mathbf{z}) \in \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n \right\},$$

où

$$g(\mathbf{x}, \mathbf{y}, \mathbf{z}) := (1/m) \sum_{j=1}^{m} x_j^2 + \chi_{\mathcal{A}}(\mathbf{x}, \mathbf{y}, \mathbf{z})$$

et

$$h(\mathbf{x}, \mathbf{y}, \mathbf{z}) := t \sum_{i=1}^{n} z_i(z_i - 1).$$

Ici, $\chi_{\mathcal{A}}$ est la fonction indicatrice sur $\mathcal{A}$, i.e., $\chi_{\mathcal{A}}(\mathbf{x}, \mathbf{y}, \mathbf{z}) = 0$ si $(\mathbf{x}, \mathbf{y}, \mathbf{z}) \in \mathcal{A}$ et $+\infty$ sinon.

5.4.2 Résolution de (P_{DC}) par DCA

Selon la description de DCA, la résolution de (P_{DC}) par DCA consiste en la détermination de deux suites $\{(\mathbf{x}^k, \mathbf{y}^k, \mathbf{z}^k)\}$ et $\{(\mathbf{u}^k, \mathbf{v}^k, \mathbf{w}^k)\}$. Afin de construire la suite $\{(\mathbf{u}^k, \mathbf{v}^k, \mathbf{w}^k)\}$, il

nous faut calculer le sous-gradient de la fonction h défini par $h(\mathbf{x},\mathbf{y},\mathbf{z}) := t\sum_{i=1}^{n} z_i(z_i-1)$. Le calcul est fait par

$$(\mathbf{u}^k,\mathbf{v}^k,\mathbf{w}^k) \in \nabla h(\mathbf{x}^k,\mathbf{y}^k,\mathbf{z}^k) \Leftrightarrow u_i^k = 0, v_j^k = 0, w_j^k = t(2z_j^k-1), \quad (5.16)$$
$$i = 1,\ldots,m, j = 1,\ldots,n.$$

La construction de la suite $\{(\mathbf{x}^k,\mathbf{y}^k,\mathbf{z}^k)\}$ exige la résolution du programme quadratique

$$\min\left\{(1/m)\sum_{j=1}^{m} x_j^2 - \langle(\mathbf{x},\mathbf{y},\mathbf{z}),(\mathbf{u}^k,\mathbf{v}^k,\mathbf{w}^k)\rangle : (\mathbf{x},\mathbf{y},\mathbf{z}) \in \mathcal{A}\right\}. \quad (5.17)$$

La solution du programme est $(\mathbf{x}^{k+1},\mathbf{y}^{k+1},\mathbf{z}^{k+1})$. L'algorithme DCA pour résoudre (P_{DC}) se résume

Algorithme 5.1 *Algorithme de DCA*

Initialisation :
- *Choisir* $(\mathbf{x}^0,\mathbf{y}^0,\mathbf{z}^0) \in \mathbb{R}^m \times \mathbb{R}^n \times [0,1]^n$ *et* $k = 0$.
- *Choisir la tolérance* ϵ *positive et suffisamment petite.*

Répéter
- *Calculer* $(\mathbf{u}^k,\mathbf{v}^k,\mathbf{w}^k) \in \partial h(\mathbf{x}^k,\mathbf{y}^k,\mathbf{z}^k)$ *via (5.16).*
- *Calculer* $(\mathbf{x}^{k+1},\mathbf{y}^{k+1},\mathbf{z}^{k+1}) \in \partial g^*(\mathbf{u}^k,\mathbf{v}^k,\mathbf{w}^k)$ *en résolvant le programme quadratique (5.17).*
- $k+1 \leftarrow k$.

Jusqu'à $\left\|(\mathbf{x}^{k+1},\mathbf{y}^{k+1},\mathbf{z}^{k+1}) - (\mathbf{x}^k,\mathbf{y}^k,\mathbf{z}^k)\right\| \leq \epsilon$.

Le théorème suivant concerne la convergence de DCA pour résoudre (P_{DC}). La démonstration ressemble à celle des algorithmes de DCA dans le chapitre précédent,

Théorème 5.3 *(Propriétés de la convergence de l'algorithme 5.1)*

(i) L'algorithme 5.1 génère une suite $\{(\mathbf{x}^k,\mathbf{y}^k,\mathbf{z}^k)\}$ *telle que la suite* $\{(1/m)\sum_{j=1}^{m}(x_j^k)^2 + tp(\mathbf{x}^k,\mathbf{y}^k,\mathbf{z}^k)\}$ *est décroissante.*

(ii) Il existe un nombre non négatif t *tel que pour chaque* $t \geq t_0$ *la suite* $\{p(\mathbf{x}^k,\mathbf{y}^k,\mathbf{z}^k)\}$ *est décroissante. En particulier, si* $(\mathbf{x}^r,\mathbf{y}^r,\mathbf{z}^r)$ *est une solution admissible de* (P''') *alors* $(\mathbf{x}^k,\mathbf{y}^k,\mathbf{z}^k)$*, pour tout* $k \geq r$*, est admissible également.*

(iii) DCA a un taux de convergence linéaire pour le problème (P_{DC}).

(iv) La suite $\{(\mathbf{x}^k,\mathbf{y}^k,\mathbf{z}^k)\}$ *converge à* $(\mathbf{x}^*,\mathbf{y}^*,\mathbf{z}^*)$ *où le point* $(\mathbf{x}^*,\mathbf{y}^*,\mathbf{z}^*)$ *est un point critique du problème* (P_{DC}).

Recherche d'un bon point initial pour DCA

Plusieurs choix de point initial ont été testés pour démarrer DCA. Le meilleur point est trouvé au sein de la procédure suivante

1. **Résoudre le problème relaxé :** Nous résolvons le problème relaxé du (P_{card}) afin de trouver $(\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}}, \widetilde{\mathbf{z}})$.
2. **Trouver une solution entière :** Nous mettons $\widetilde{z}_j = 1$ pour tout $j = 1, \ldots, n$. Le nouveau point $(\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}}, \widehat{\mathbf{z}})$ est utilisé afin de démarrer DCA. Ce point n'est pas, en général, un point admissible pour (P_{DC}), pourtant DCA trouve une solution admissible après une seule itération.

Les autres procédures de choix de point initial que nous avons testées sont les suivantes :

- la solution optimale du modèle relaxé de (P_{card}) ;
- après avoir résolu le modèle relaxé de (P_{card}), nous mettons les variables binaires non nulles égale à 1 ;
- la solution optimale du problème suivant

$$\min \left\{ \sum_{j=1}^{n} z_j(1 - z_j) : (\mathbf{x}, \mathbf{y}, \mathbf{z}) \in \mathcal{A} \right\}.$$

5.5 Résolution globale

Afin de vérifier la qualité des solutions fournies par DCA, un algorithme classique de Séparation et Évaluation (SE) a été utilisé. L'algorithme consiste à résoudre le problème (P_{card}). Les bornes inférieures sont calculées en relaxant les contraintes binaires, c'est-à-dire les contraintes $z_j \in \{0, 1\}$ sont remplacées par $0 \leq z_j \leq 1$. Les bornes supérieures sont mises à jour si une meilleure solution pour le modèle (P_{card}) est trouvée. La séparation est effectuée sur la variable fractionnelle z_j telle que soit $z_j = 0$, soit $z_j = 1$.

Une approche combinée de SE et DCA (SE-DCA) a été développée afin de trouver la solution globale de (P_{card}) tout en essayant d'accélérer la convergence de SE. A chaque itération de SE, si un nombre suffisant des variables $\mathbf{z}$ sont 0 ou 1, par exemple $(n/2)$, DCA est relancé. Le point initial est la solution optimale du sous-problème courant. Avant de l'utiliser, nous la traitons par la procédure citée ci-dessus. Si la solution fournie par DCA est améliorante alors nous l'utilisons pour mettre à jour la meilleure borne supérieure.

5.6 Expériences numériques

Nous avons codé les algorithmes en langage C++ et testé sur un ordinateur Pentium de 1.600 GHz et 512Mo RAM. A chaque itération de DCA, nous avons utilisé le logiciel CPLEX en version 9.1 pour trouver la solution des sous-problèmes quadratiques convexes. Nous avons fixé une limite pour le nombre d'itération de SE. La limite est de 40000 itérations.

Afin de réaliser les tests, nous avons généré les données à partir des prix des actifs financiers. Ce sont les actions de certaines compagnies américaines, du 27 septembre 2005 pendant 20

semaines. Les prix sont disponibles sur la page web de YAHOO ! FINANCE (http ://finance.yahoo.com). A partir des prix, nous avons calculé les rendements historiques, r_{ij}, de chaque actif. La formule de calcul est

$$r_{ij} = (p_{i,j+1} - p_{i,j})/p_{i,j},$$

où $p_{i,j}$ et $p_{i,j+1}$ sont les prix de l'actif i en deux instant successifs j et $j+1$ ($i = 1, \ldots, n$ et $j = 1, \ldots, m$). Dans nos expérience n est égal à 550 et m à 20.

Les expériences ont été faites pour deux ensembles de valeurs des paramètres. Les premières valeurs choisies sont :

- $R = 0.1$;
- $a_i = 0.0$: la borne inférieure sur la proportion du capital investi en actif i ;
- $b_i = 1.0$: la borne supérieure sur la proportion du capital investi en actif i ;
- $\epsilon = 10^{-6}$.

Les valeurs suivantes sont :

- $R = 0.05$;
- $a_i = 0.01$: la borne inférieure sur la proportion du capital investi en actif i ;
- $b_i = 1.0$: la borne supérieure sur la proportion du capital investi en actif i ;
- $\epsilon = 10^{-6}$.

Pour toutes les expériences, le paramètre de pénalité a été fixée à 0.05. Nous avons testé les algorithmes pour différentes valeurs de *card*. Les valeurs choisies sont $20, \ldots, 30$. Ce sont les valeurs pour lesquelles, comme nous pourrons le constater, le problème (P_{card}) est difficile à résoudre par l'algorithme classique SE.

Les tableaux 5.1 et 5.2 présentent les résultats. Dans ces tableaux, le nombre d'itération (iter) de chaque algorithme, les valeurs optimales fournies (Val. Opt.), le temps d'execution en secondes (CPU) et le nombre d'appel à DCA (relance) au cours de l'approche combinée (SE-DCA) ont été présentés.

Commentaires sur les résultats

Les résultats numériques dans le tableau 5.1 montrent que :

- Il y a une seule valeur pour tous les cas. La raison est due au fait que nous avons mis la borne inférieure a_j égale 0. a_j est la borne inférieure sur la proportion du capital investi dans l'actif j. En mettant $a_j = 0$, *card* est le nombre maximum des actifs composant le portefeuille. De cette façon, en changeant le paramètre *card*, le portefeuille optimal ne change pas. Et en effet, le portefeuille est le même pour toutes les différentes valeurs de *card*. Il nous faut choisir une valeur plus petite que le nombre d'actifs composant le portefeuille actuel pour que la solution change.
- DCA donne la solution globale, car l'approche combinée qui une méthode globale donne les mêmes valeurs optimales que DCA. Pour le cas $card = 22$, il nous faut attendre

	DCA			SE			SE-DCA			
card	iter	Val. Opt.	CPU	iter	Val. Opt.	CPU	iter	relance	Val. Opt.	CPU
20	4	0.003633	0.766	40000	0.004395	10502.000	8	2	0.003633	4.594
21	4	0.003633	0.766	40000	0.004395	10977.282	1	1	0.003633	1.531
22	4	0.003633	0.750	5470	0.003633	1414.156	5056	116	0.003633	1369.578
23	4	0.003633	0.750	40000	0.004395	10366.765	1	1	0.003633	1.531
24	4	0.003633	0.766	40000	0.004395	10804.219	1	1	0.003633	1.672
25	5	0.003633	0.922	40000	0.004395	10415.219	18	2	0.003633	18.359
26	5	0.003633	0.906	40000	0.004395	10642.328	1	1	0.003633	1.562
27	4	0.003633	0.891	40000	0.004497	10617.468	1	1	0.003633	1.531
28	4	0.003633	0.750	40000	0.004395	10611.860	1	1	0.003633	1.703
29	6	0.003633	1.062	40000	0.004395	10621.031	1	1	0.003633	1.687
30	4	0.003633	0.766	40000	0.004395	10953.375	1	1	0.003633	1.687

Tab. 5.1 – La performance des algorithmes pour le premier ensemble de paramètres

	DCA			SE			SE-DCA			
card	iter	Val. Opt.	CPU	iter	Val. Opt.	CPU	iter	relance	Val. Opt.	CPU
20	4	0.000097	0.907	40000	0.000009	15549.063	356	171	0.000001	277.406
21	4	0.000028	0.937	40000	0.000013	15411.813	44	6	0.000001	28.969
22	4	0.000028	1.000	40000	0.000016	15151.391	1688	805	0.000001	1273.860
23	4	0.000026	0.953	40000	0.000014	15255.046	53	12	0.000001	39.563
24	4	0.000022	0.938	40000	0.000012	16003.610	31	5	0.000001	21.172
25	4	0.000027	0.953	40000	0.000015	14916.516	50	18	0.000001	42.875
26	4	0.000025	0.953	40000	0.000012	14628.437	42	4	0.000000	26.391
27	4	0.000019	0.953	1564	0.000000	619.875	47	11	0.000000	34.750
28	4	0.000018	0.938	2884	0.000000	1070.047	26	6	0.000000	19.500
29	4	0.000020	0.937	6625	0.000000	2434.547	46	4	0.000000	28.890
30	4	0.000019	0.953	2783	0.000000	1018.953	30	4	0.000000	20.594

TAB. 5.2 – La performance des algorithmes pour le deuxième ensemble de paramètres

pour que la borne inférieure monte. C'est la raison pour laquelle un nombre important d'itération est nécessaire pour être sûr que la solution courante est l'optimale.

- Pour la plupart des valeurs de *card*, l'algorithme SE échoue et il n'arrive pas à trouver le portefeuille optimal dans la limite prévue. En revanche, comme le tableau montre, DCA trouve le portefeuille optimale dans un délai très court et en $4-6$ itérations. L'influence de DCA dans la convergence rapide de l'approche combinée (SE-DCA) est très remarquable. Pour la plupart des valeurs de *card*, SE-DCA s'arrête après une seule itération. Pour les autres cas, il nous faut attendre pour que l'écart entre les bornes supérieure et inférieure soit suffisamment serré.

Le tableau 5.2 montre que :

- Dans la plupart des cas, l'algorithme SE ne trouve pas la solution optimale dans la limite prévue, tandis que SE-DCA y arrive en un temps moyen très court. Le tableau montre aussi que les solutions fournies par SE ne sont pas très proches de celles trouvées par SE-DCA.
- DCA trouve la solution optimale en seulement 4 itérations. La précision des solutions fournies par DCA est 10^{-4}.
- Contrairement au cas $a_j = 0$, les bornes inférieures non-zéro nous permettent de produire des portefeuilles différents pour différentes valeurs de *card*.

5.7 Conclusion

Dans ce chapitre, nous avons présenté la méthode DCA pour un modèle de choix de portefeuille. Le modèle adopte une mesure de baisse comme la mesure de risque. De plus, il existe les contraintes de bornes et celles de cardinalité dans le modèle. Après avoir reformulé le modèle, nous avons utilisé DCA pour le résoudre. DCA trouve des solutions de très bonne qualité. Pour évaluer la qualité des solutions fournies par DCA, nous avons utilisé un algorithme classique de SE. Une approche combinée à la base de SE et DCA a été développée. Les résultats numériques confirment la convergence rapide de l'approche combinée (SE-DCA) et la bonne qualité des solutions obtenues par DCA.

Chapitre 6

Gestion de portefeuille sous les fonctions de coûts de transaction non convexes

Résumé *Ce chapitre concerne une nouvelle approche continue basée sur la programmation DC et DCA pour la résolution du problème de gestion de portefeuille sous les fonctions de coûts de transaction. Les fonctions étudiées sont non convexes et non-lisses. Un algorithme par séparation et évaluation basé sur la convexification du problème peut être utilisé pour résoudre le problème sous les fonctions de coûts non convexe et non-lisse. Notre approche consiste à estimer les fonctions non-lisses par des fonctions continues et DC polyédrales. Ensuite, DCA s'applique pour résoudre le nouveau modèle. Une approche combinée de DCA et un algorithme SE est proposée. Les résultats numériques confirment l'efficacité de notre approche pour résoudre le problème.*

6.1 Introduction

Une des formulations d'un problème de choix de portefeuille consiste à maximiser le rendement net du portefeuille pour un niveau toléré du risque. Le rendement net du portefeuille se définit par le rendement espéré du portefeuille dont les coûts de transactions ont été soustraits [60]. Les coûts de transactions sont les montants que l'investisseur engage pour ses transactions. L'inclusion des coûts de transaction est un sujet important qui rend le modèle plus réaliste. Si nous négligeons les coûts de transaction, nous courrons le risque de prendre de fausses décisions.

Dans les cas où les coûts de transactions sont négligés ou sont des fonctions linéaires, le modèle correspondant de choix de portefeuille se formalise par un programme linéaire ou quadratique [69]. Pour ces cas, le problème peut être résolu de façon efficace. Pourtant, si les coûts de transaction sont des fonctions non convexes ou non-lisses, le problème devient très difficile à résoudre. Les coûts de transaction que nous allons étudier dans ce chapitre concernent des fonctions non convexes et non-lisses. Plus précisément, nous considérons des

fonctions constantes par morceaux.

L'approche traditionnelle pour résoudre un tel problème consiste à reformuler les fonctions en introduisant des variables binaires ([60],[61]). Ensuite, un algorithme de séparation et évaluation (SE) peut être utilisé afin de résoudre le problème transformé. Cette approche n'est pas très pratique, car plus le nombre d'actifs ou de morceaux augmente, plus le nombre des variables binaires augmente. Par exemple, s'il y a sept morceaux dans les fonctions et que nous disposons de 1000 actifs parmi lesquels nous devons faire notre choix, alors le modèle binaire aura besoin de 7000 variables binaires. Dans ce cas le modèle, qu'il soit linéaire ou quadratique, ne se résout pas dans un délai raisonnable.

Une autre approche de SE consiste à sous-estimer les fonctions en escalier (les fonctions constantes par morceaux) par des enveloppes convexes et à résoudre le problème convexe [60]. Cette idée fonctionne pour les problèmes où le nombre de morceaux est petit, par exemple un ou deux. Lorsque les fonctions ont plus de morceaux constants, par exemple six ou sept, l'algorithme échoue.

Dans ce chapitre, notre étude se focalise sur l'utilisation d'une approche locale basée sur la programmation DC et DCA. L'approche consiste à estimer les fonctions en escalier par des différence de fonctions convexes polyédrales. Ensuite, DCA s'applique pour résoudre le problème.

Afin d'évaluer l'efficacité de l'approche proposée, nous avons comparé les résultats avec ceux fournis par le logiciel CPLEX en version 10.1 et ceux d'un algorithme SE proposé par Konno et al. [60]. Enfin une approche combinée de SE et DCA est proposée pour assurer une convergence plus rapide.

Dans ce chapitre, la mesure de risque adoptée est Mean-Absolute Deviation (MAD), proposé par Konno et al. Notre choix a deux raisons :

- MAD peut s'exprimer sous forme d'un modèle d'une programmation linéaire alors que MV est une programmation quadratique. Un programme linéaire se résout de manière plus efficace qu'un programme quadratique.

- MAD est compatible avec les critères de dominance stochastique. En général, ce n'est pas le cas pour le modèle MV [59].

Le reste du chapitre est organisé de façon suivante : la prochaine section concerne la description et la formulation du modèle de base ainsi qu'une introduction aux différentes formes des coûts de transaction. La section 6.3 est consacrée aux approximations des fonctions en escalier par des fonctions DC polyédrales et la résolution du modèle approché par DCA. Nous allons présenter les méthodes globales dans la section 6.5. Les résultats numériques sont présentés dans la section 6.6. Nous terminons le chapitre par une conclusion dans la dernière section.

6.2 Description et formulation

6.2.1 Modèle de Déviation Moyenne-Absolue (MAD)

Konno et Yamazaki [62] ont mis au point un modèle de choix de portefeuille exprimable sous forme d'un programme linéaire. Considérons R_u comme une variable aléatoire qui représente le taux d'intérêt de l'actif u. Nous supposons que $(R_1, R_2, \ldots, R_n)$ est distribué sur un ensemble fini de points $(r_{1t}, r_{2t}, \ldots, r_{nt})$ où $t = 1, \ldots, T$ et que

$$\wp_t = Pr\{(R_1, R_2, \ldots, R_n) = (r_{1t}, r_{2t}, \ldots, r_{nt})\} \tag{6.1}$$

soit connu auparavant pour $t = 1, \ldots, T$. Nous adoptons les notations suivantes :
– M = le capital ;
– x_u = la portion du capital investi en actif u ;
– β_u = la borne supérieure pour l'investissement en actif u ;
– ω = le niveau de risque autorisé.
La déviation absolue $W(x)$ du portefeuille $x = (x_1, x_2, \ldots, x_n)$ se définit par

$$W[R(x)] = E[\mid R(x) - E[R(x)] \mid] = \sum_{t=1}^{T} \wp_t \mid \sum_{u=1}^{n} (r_{ut} - r_u)x_j \mid$$

où $r_u = \sum_{t=1}^{T} \wp_t r_{ut}$ est la valeur espérée de R_u. Dans ce qui suit, nous mettons $\wp_t = 1/T$, alors $r_u = \sum_{t=1}^{T} r_{ut}/T$ et

$$W[R(x)] = \sum_{t=1}^{T} \wp_t \mid \sum_{u=1}^{n} (r_{ut} - r_u)x_j \mid = \sum_{t=1}^{T} \mid \sum_{u=1}^{n} (r_{ut} - r_u)x_u \mid /T.$$

Le modèle Mean-Absolute Deviation (MAD) est

$$\max \left\{ \sum_{u=1}^{n} r_u x_u : W[\sum_{u=1}^{n} R_u x_u] \leq \omega M, \sum_{u=1}^{n} x_u = M, 0 \leq x_u \leq \beta_u, u = 1, \ldots, n \right\}. \tag{6.2}$$

Nous pouvons mettre ce problème sous forme d'un modèle de programmation linéaire, pour cela nous introduisons les variables supplémentaires $\tilde{\psi}_t$ et $\tilde{\phi}_t$ telles que :

$$\tilde{\psi}_t - \tilde{\phi}_t = \wp_t \sum_{u=1}^{n} (r_{ut} - r_u)x_u$$

et

$$\tilde{\phi}_t \tilde{\psi}_t = 0, \tilde{\phi}_t \geq 0, \tilde{\psi}_t \geq 0, \quad t = 1, \ldots, T,$$

alors

$$W[R(x)] = \sum_{t=1}^{T} \mid \tilde{\psi}_t - \tilde{\phi}_t \mid .$$

Mais pour chaque $t = 1, \ldots, T$,

- si $\tilde{\phi}_t = 0$ et $\tilde{\psi}_t = 0$ alors $\tilde{\phi}_t + \tilde{\psi}_t =\mid \tilde{\psi}_t - \tilde{\phi}_t \mid$,
- si $\tilde{\psi}_t > 0$ alors $\tilde{\phi}_t = 0$ et $\tilde{\phi}_t + \tilde{\psi}_t = \tilde{\psi}_t =\mid \tilde{\psi}_t \mid=\mid \tilde{\psi}_t - \tilde{\phi}_t \mid$,
- si $\tilde{\phi}_t > 0$ alors $\tilde{\psi}_t = 0$ et $\tilde{\phi}_t + \tilde{\psi}_t = \tilde{\phi}_t =\mid \tilde{\phi}_t \mid=\mid 0 - \tilde{\phi}_t \mid=\mid \tilde{\psi}_t - \tilde{\phi}_t \mid$.

La déviation absolue s'écrit alors

$$W[R(x)] = \sum_{t=1}^{T} (\tilde{\phi}_t + \tilde{\psi}_t).$$

Par conséquent, le modèle MAD devient

$$\begin{cases} \max \sum_{u=1}^{n} r_u x_u \\ s.c. \begin{cases} \sum_{t=1}^{T} (\tilde{\phi}_t + \tilde{\psi}_t) \leq \omega M, & \\ \tilde{\psi}_t - \tilde{\phi}_t = (1/T) \sum_{u=1}^{n} (r_{ut} - r_u) x_u & t = 1, \ldots, T, \\ \sum_{u=1}^{n} x_u = M, & \\ \tilde{\phi}_t \tilde{\psi}_t = 0, & t = 1, \ldots, T, \\ \tilde{\phi}_t \geq 0, & t = 1, \ldots, T, \\ \tilde{\psi}_t \geq 0, & t = 1, \ldots, T, \\ 0 \leq x_u \leq \beta_u & u = 1, \ldots, n. \end{cases} \end{cases} \tag{6.3}$$

Si nous faisons le changement de variable $\phi_t := T\tilde{\phi}_t$ et $\psi_t := T\tilde{\psi}_t$, le modèle MAD s'exprime alors sous la forme suivante

$$\begin{cases} \max \sum_{u=1}^{n} r_u x_u \\ s.c. \begin{cases} \sum_{t=1}^{T} (\phi_t + \psi_t) \leq \omega M, & \\ \psi_t - \phi_t = (1/T) \sum_{u=1}^{n} (r_{ut} - r_u) x_u & t = 1, \ldots, T, \\ \sum_{u=1}^{n} x_u = M, & \\ \phi_t \psi_t = 0, & t = 1, \ldots, T, \\ \phi_t \geq 0, & t = 1, \ldots, T, \\ \psi_t \geq 0, & t = 1, \ldots, T, \\ 0 \leq x_u \leq \beta_u & u = 1, \ldots, n. \end{cases} \end{cases} \tag{6.4}$$

Les modèles 6.3 et 6.4 sont équivalents dans le sens où ils ont les mêmes valeurs optimales et il existe une relation réciproque entre les ensembles de solutions admissibles des modèles.

Dans le modèle 6.4, il existe des contraintes de complémentarité que nous pouvons les supprimer. Ceci est l'objet du théorème suivant :

Théorème 6.1 *Le modèle 6.4 est équivalent au modèle suivant qui est un modèle de programmation linéaire :*

$$\begin{cases} \max \sum_{u=1}^{n} r_u x_u \\ s.c. \begin{cases} \sum_{t=1}^{T} (\phi_t + \psi_t) \leq \omega M, \\ \psi_t - \phi_t = \wp_t \sum_{u=1}^{n} (r_{ut} - r_u) x_u & t = 1, \ldots, T, \\ \sum_{u=1}^{n} x_u = M, \\ \phi_t \geq 0, & t = 1, \ldots, T, \\ \psi_t \geq 0, & t = 1, \ldots, T, \\ 0 \leq x_u \leq \beta_u & u = 1, \ldots, n. \end{cases} \end{cases} \tag{6.5}$$

Preuve : Soit $(x_1^*, \ldots, x_n^*, \psi_1^*, \ldots, \psi_T^*, \phi_1^*, \ldots, \phi_T^*)$ la solution optimale du problème sans les contraintes de complémentarité. Nous définissons

$$\hat{\psi}_t = max(\psi_t^* - \phi_t^*, 0), \hat{\phi}_t = -min(0, \psi_t^* - \phi_t^*), t = 1, \ldots, T.$$

Clairement

$$\hat{\psi}_t \geq 0, \hat{\phi}_t \geq 0, t = 1, \ldots, T.$$

De plus,

$$\hat{\psi}_t - \hat{\phi}_t = max(\psi_t^* - \phi_t^*, 0) + min(0, \psi_t^* - \phi_t^*) = \psi_t^* - \phi_t^*, t = 1, \ldots, T.$$

Finalement,

$$\begin{aligned} \sum_{t=1}^{T} (\hat{\phi}_t + \hat{\psi}_t) &= \sum_{t=1}^{T} [max(\psi_t^* - \phi_t^*, 0) - min(0, \psi_t^* - \phi_t^*)] \\ &= \sum_{t=1}^{T} [max(\psi_t^* - \phi_t^*, 0) + max(0, \phi_t^* - \psi_t^*)] \leq \sum_{t=1}^{T} [\psi_t^* + \phi_t^*] \leq \omega M. \end{aligned}$$

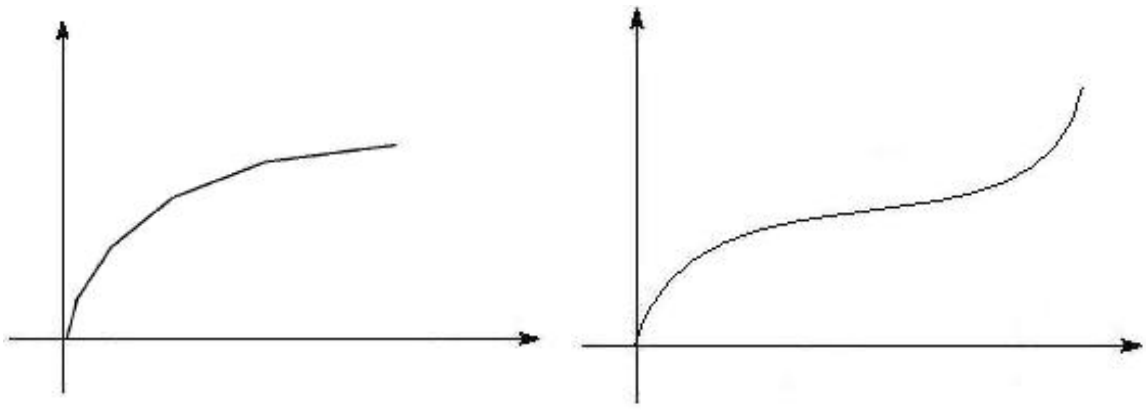
Alors $(x_1^*, \ldots, x_n^*, \hat{\psi}_1, \ldots, \hat{\psi}_T, \hat{\phi}_1, \ldots, \hat{\phi}_T)$ est aussi une solution admissible du modèle sans les contraintes de complémentarité. Puisqu'elle a la même valeur de fonction objectif que $(x_1^*, \ldots, x_n^*, \psi_1^*, \ldots, \psi_T^*, \phi_1^*, \ldots, \phi_T^*)$, alors $(x_1^*, \ldots, x_n^*, \hat{\psi}_1, \ldots, \hat{\psi}_T, \hat{\phi}_1, \ldots, \hat{\phi}_T)$ est aussi une solution optimale du modèle sans les contraintes de complémentarité. En même temps, cette

solution satisfait les contraintes de complémentarité. Cela veut dire que nous pouvons supprimer les conditions $\hat{\psi}_t\hat{\phi}_t = 0, t = 1, \ldots, T$ car pour chaque solution du modèle qui ne contient pas les contraintes de complémentarité nous pouvons toujours trouver une solution qui porte la même valeur pour la fonction objectif et de plus, elle satisfait les conditions de complémentarité. □

6.2.2 Coûts de transaction

En pratique, chaque fois qu'un investisseur achète ou vend un ou plusieurs actif(s) (autrement dit, lorsqu'il fait une transaction), il doit payer un montant en tant que coût de transaction ([114], [145]). Les coûts de transaction peuvent être constants, mais ce cas n'est pas très commun et en général les coûts sont proportionnels au volume de transaction (achat ou vente). La présence des coûts de transaction peut influencer significativement le rendement du portefeuille. Ils peuvent non seulement changer le nombre des actifs présents dans le portefeuille mais aussi les portions du capital investi dans chaque actif [114].

Il existe différentes formes des coûts de transaction. Des fois, les coûts de transaction sont relativement grands si les montants d'achat sont petits et ils augmentent au fur et à mesure que les montants d'achat continuent à augmenter. De cette façon, les coûts de transaction forment une fonction concave, linéaire ou constante par morceaux. Cette dernière est très utilisée dans les transactions sur internet ([54, 58, 60, 69]). La Figure 6.1 montre deux différentes formes des fonctions de coûts de transaction.



(a) Coût de transaction concave (b) Coût de transaction DC

FIG. 6.1 – Fonctions des coûts de transaction

Ici, nous considérons une forme particulière des coûts de transaction qui est utilisée en transactions sur internet [54]. Ce sont des coûts de transaction sous forme de fonctions en escalier (constantes par morceaux).

En général, les coûts de transaction associés au portefeuille $\mathbf{x} = (x_1, \ldots, x_n)$ se définissent par la somme des coûts sur chaque actif [145] :

$$f(x) := \sum_{u=1}^{n} f_u(x_u)$$

où $f_u(x_u)$ est la fonction des coûts de transaction sur l'actif u. Mathématiquement, les fonctions f_u $(u = 1, \ldots, n)$ sont définies sur $\mathbb{R}$ par

$$f_u(x_u) = \begin{cases} \gamma_u^0 \quad si \quad x_u \leq 0, & i = 0, \\ \gamma_u^i \quad si \quad v_u^{i-1} < x_u \leq v_u^i, & i = 1, \ldots, q(u) - 1, \\ \gamma_u^{q(u)} \quad si \quad v_u^{q(u)-1} < x_u \leq +\infty, & i = q(u). \end{cases} \tag{6.6}$$

où $V_u = \{0 = v_u^0 < v_u^1 < \cdots < v_u^{q(u)} = \beta_u\}$ est l'ensemble fini des points pour lesquels la fonction $f_u(x_u)$ est discontinue et $0 = \gamma_u^0 < \gamma_u^1 < \cdots < \gamma_u^{q(u)}$. Clairement, la fonction $f_u(x_u)$ est définie sur $[0, \beta_u]$ et elle possède $q(u)$ points de discontinuité (voir la Figure 6.2).

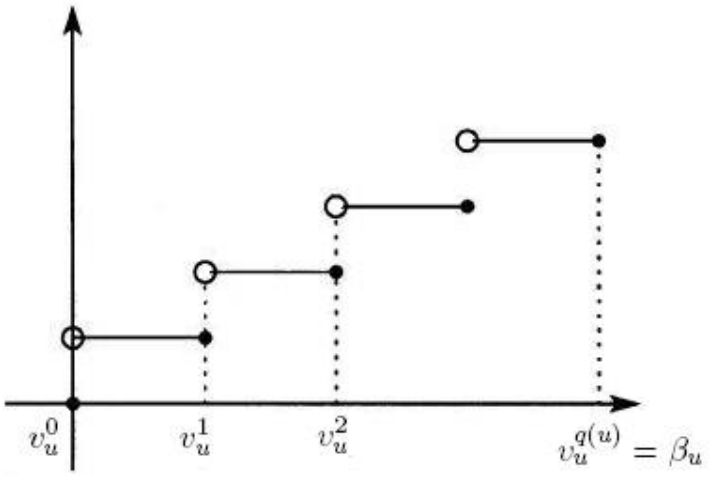


FIG. 6.2 – Fonction des coûts de transaction non convexe et constante par morceaux

6.2.3 Le modèle de choix de portefeuille sous les coûts de transaction

Le rendement net du portefeuille $x = (x_1, x_2, \ldots, x_n)$ est

$$\sum_{u=1}^{n} \{r_u x_u - f_u(x_u)\},$$

où $f_u(x_u)$ est une fonction en escalier définie par (6.6). Le modèle de choix de portefeuille sous les coûts de transaction s'explique par

$$\left\{\begin{array}{l} \max \sum\limits_{u=1}^{n}\{r_u x_u - f_u(x_u)\} \\ s.c. \left\{\begin{array}{ll} \sum\limits_{t=1}^{T}(\phi_t+\psi_t)/T \leq wM, & \\ \psi_t - \phi_t = \sum\limits_{u=1}^{n}(r_{ut}-r_u)x_u, & t=1,\ldots,T, \\ \phi_t \geq 0, \psi_t \geq 0, & t=1,\ldots,T, \\ \sum\limits_{u=1}^{n} x_u = M, & \\ 0 \leq x_u \leq \beta_u, & u=1,\ldots,n. \end{array}\right. \end{array}\right. \tag{6.7}$$

Afin de simplifier les notations, nous définissons

$$\mathcal{D} := \left\{\begin{array}{l} (x,\psi,\phi) \in \mathbb{R}^n \times \mathbb{R}^T \times \mathbb{R}^T : \\ \sum\limits_{u=1}^{n}(r_{ut}-r_u)x_u + \phi_t - \psi_t = 0, \qquad t=1,\ldots,T, \\ \sum\limits_{t=1}^{T}(\psi_t+\phi_t)/T \leq wM, \sum\limits_{u=1}^{n} x_u = M, \\ 0 \leq x_u \leq \beta_u, u=1,\ldots,n, \psi_t \geq 0, \phi_t \geq 0, \quad t=1,\ldots,T \end{array}\right\}. \tag{6.8}$$

De cette manière la résolution du modèle (6.7) est équivalent à la résolution du modèle suivant :

$$(P): \quad \min\left\{ f(x) := \sum_{u=1}^{n} f_u(x_u) - \sum_{u=1}^{n} r_u x_u : (x,\psi,\phi) \in \mathcal{D} \right\}.$$

Notre enjeu est de résoudre le problème (P).

6.2.4 Formulation $0-1$ du modèle de choix de portefeuille sous les coûts de transaction en escalier

Une formulation traditionnelle des problèmes avec des fonctions constantes (ou linéaires) par morceaux consiste en l'introduction de variables binaires. A chaque actif u et à chaque morceau i, une variable binaire, S_{ui}, est associée [60].

Soient $0 = v_u^0 < v_u^1 < \cdots < v_u^{q(u)} = \beta_u$ les points de discontinuité de la fonction $f_u(x_u)$. Soit $f_u(x_u) = \gamma_u^i$ pour $x_u \in (v_u^{i-1}, v_u^i] \quad : i = 1,\ldots,q(u)$ et posons $f_u(v_u^0) = \gamma_u^0 = 0$. De cette façon, la fonction des coûts de transaction en escalier se transforme en

$$f_u(x_u) = \sum_{i=1}^{q(u)} (\gamma_u^i - \gamma_u^{i-1}) S_{u,i-1}.$$

Pour obtenir $f_u(x_u) = \gamma_u^i$ tel que $x_u \in (v_u^{i-1}, v_u^i]$, il nous faut

$$S_{uj} = \begin{cases} 0, & \text{si} \quad j \geq i, \\ 1, & \text{si} \quad 0 \leq j < i. \end{cases} \tag{6.9}$$

Pour assurer ces conditions, il suffit d'ajouter les contraintes suivantes au modèle

$$[(x_u - v_u^i)/\beta_u] \leq S_{ui} \leq 1 + [(x_u - v_u^i)/\beta_u].$$

Ces relations assurent que pour tout u, i

$$x_u > v_u^i \Rightarrow S_{ui} = 1 \quad \text{et} \quad x_u \leq v_u^i \Rightarrow S_{ui} = 0.$$

De cette façon, la formulation binaire du problème (P) est :

$$(P_{bin}) : \begin{cases} min(\sum_{u=1}^{n} \sum_{i=1}^{q(u)} (\gamma_u^i - \gamma_u^{i-1}) S_{u,i-1} - \sum_{u=1}^{n} r_u x_u) \\ s.c : \begin{cases} (x, \psi, \phi) \in \mathcal{D}, \\ ((x_u - v_u^i)/\beta_u) \leq S_{ui} & : \forall i, \forall u, \\ S_{ui} \leq 1 + [(x_u - v_u^i)/\beta_u] & : \forall i, \forall u, \\ S_{ui} \in \{0, 1\} & : \forall i, \forall u. \end{cases} \end{cases}$$

Ce programme mixte $0-1$ se résout par un algorithme de séparation et évaluation (SE) ou une méthode de coupe. Clairement, pour un problème de 1000 actifs et six morceaux, il nous faut 6000 variables binaires. Plus le nombre d'actifs est grand ou le nombre de morceaux est important, plus le problème est difficile à résoudre. Si nous utilisons une mesure de risque quadratique, le problème deviendra encore plus difficile [60]. Notre approche consiste d'abord à estimer les fonctions en escalier par la différence de fonctions convexes polyédrales, ensuite nous appliquerons la méthode DCA pour résoudre le problème approché.

6.3 Approximation des fonctions de coûts de transaction

Les fonctions de coûts de transactions sont non convexes et non-lisses. Pour que nous puissions utiliser DCA, les fonctions de coûts seront estimées par des fonctions DC polyédrales.

6.3.1 Approximation de la fonction f_u par des fonctions DC polyédrales

Soient $0 = v_u^0 < v_u^1 < \cdots < v_u^{q(u)} = \beta_u$ les points de discontinuité de la fonction $f_u(x_u)$. Nous introduisons les points suivants

$$\tilde{v}_u^0, \tilde{v}_u^1, \ldots, \tilde{v}_u^{q(u)-1}$$

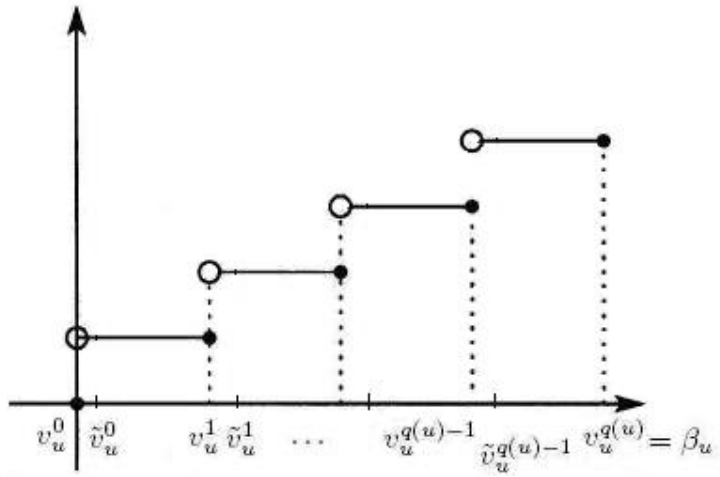


FIG. 6.3 – Les points supplémentaires

tels que (voir la Figure 6.3)

$$0 = v_u^0 < \tilde{v}_u^0 < v_u^1 < \tilde{v}_u^1 < \ldots, < v_u^{q(u)-1} < \tilde{v}_u^{q(u)-1} < v_u^{q(u)} = \beta_u.$$

Soit $\max\{\tilde{v}_u^i - v_u^i : i = 0, \ldots, q(u)-1\} \leq \eta$, un nombre réel, positif et suffisamment petit. Nous définissons les fonctions convexes polyédrales $L_u^i(x_u), i = 0, \ldots, q(u)$ sur $\mathbb{R}$ par (voir la figure 6.4)

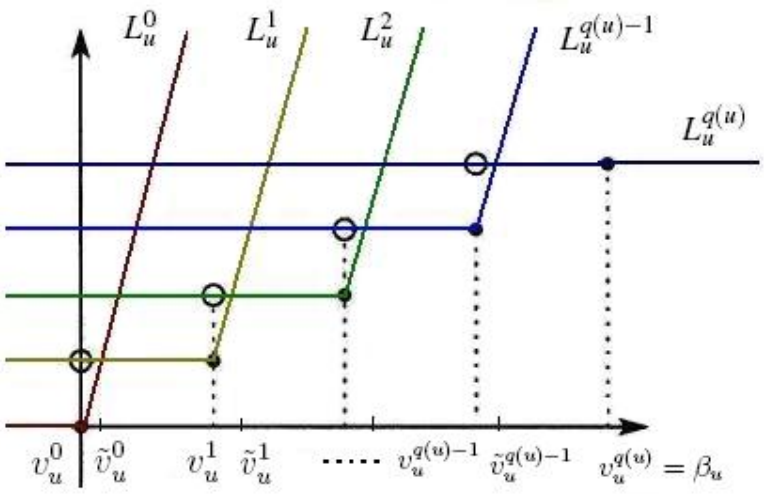


FIG. 6.4 – Les fonctions convexes polyédrales

– L_u^i est égale à γ_u^i sur l'intervalle $]-\infty, v_u^i]$, et elle est affine dans l'intervalle $[v_u^i, +\infty[$ telle que $L_u^i(v_u^i) = \gamma_u^i$ et $L_u^i(\tilde{v}_u^i) = \gamma_u^{i+1}$ (avec c_u^i comme pente) pour tout $i = 0, \ldots, q(u) - 1$,
– $L_u^{q(u)}(x_u) = \gamma_u^{q(u)}$ pour tout $x_u \in \mathbb{R}$.

Considérons la fonction $F_u(x_u)$ définie sur $\mathbb{R}$ par

$$F_u(x_u) = \begin{cases} L_u^i(x_u), & \text{si} \quad v_u^{i-1} \leq x_u \leq v_u^i, \quad i = 1, \ldots, q(u) - 1, \\ L_u^{q(u)}(x_u), & \text{si} \quad v_u^{q(u)-1} \leq x_u. \end{cases} \tag{6.10}$$

pour $u = 1, \ldots, n$. $F_u(x_u)$ est une fonction linéaire par morceaux et elle sera utilisée pour estimer la fonction $f_u(x_u)$. En effet, $F_u(x_u)$ est l'infimum des fonctions $L_u^i, i = 0, \ldots, q(u)$ (voir la figure 6.5), i.e.,

$$F_u = \min\{L_u^i : i = 0, \ldots, q(u)\}$$

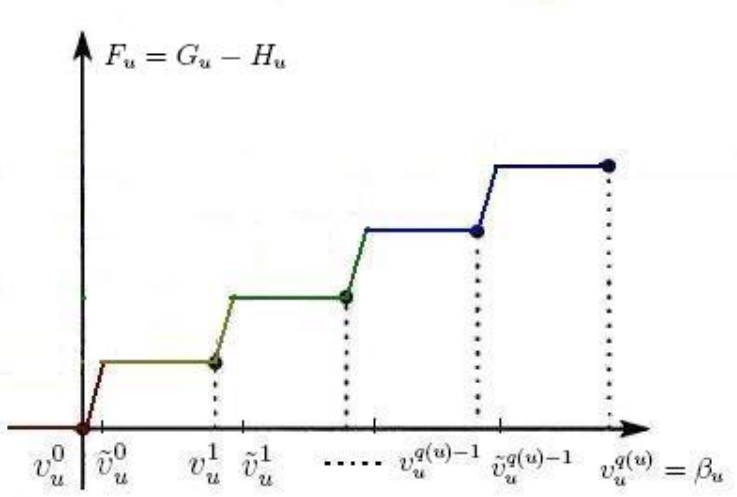


FIG. 6.5 – Les fonctions linéaires par morceaux $F_u(x_u)$

alors $F_u(x_u)$ est aussi une fonction polyédrale. D'ailleurs, nous pouvons écrire

$$F_u = \min_{i=0,\ldots,q(u)} \{L_u^i\} = \min_{i=0,\ldots,q(u)} \{\sum_{j=0}^{q(u)} L_u^j - \sum_{j=0, j\neq i}^{q(u)} L_u^j\} = \min_{i=0,\ldots,q(u)} \{\sum_{j=0}^{q(u)} L_u^j + \sum_{j=0, j\neq i}^{q(u)} (-L_u^j)\}$$

$$= \sum_{j=0}^{q(u)} L_u^j + \min_{i=0,\ldots,q(u)} \left(\sum_{j=0, j\neq i}^{q(u)} -L_u^j \right) = \sum_{j=0}^{q(u)} L_u^j - \max_{i=0,\ldots,q(u)} \left(\sum_{j=0, j\neq i}^{q(u)} L_u^j \right).$$

Soient

$$G_u := \sum_{i=0}^{q(u)} L_u^i \quad \text{et} \quad H_u := \max_{i=0,\ldots,q(u)} \left(\sum_{j=0, j\neq i}^{q(u)} L_u^j \right), \tag{6.11}$$

alors G_u et H_u sont des fonctions (finies) convexes polyédrales, de plus $F_u = G_u - H_u$. Cela veut dire que $F_u(x_u)$ est une fonction DC polyédrale et une décomposition DC de $F_u(x_u)$ est $F_u = G_u - H_u$.

Les définitions de G_u et H_u fournies par (6.11) se transforment aux formules suivantes par de simples calculs :

$$G_u(x_u) = \begin{cases} \sum\limits_{i=0}^{q(u)} \gamma_u^i, & si \quad x_u = v_u^0, \\ \sum\limits_{j=0}^{i-1} c_u^j(x_u - v_u^j) + \sum\limits_{i=0}^{q(u)} \gamma_u^i, & si \quad v_u^{i-1} \le x_u \le v_u^i, 1 \le i \le q(u)-1, \\ \sum\limits_{j=0}^{q(u)-1} c_u^j(x_u - v_u^j) + \sum\limits_{i=0}^{q(u)} \gamma_u^i, & si \quad v_u^{q(u)-1} \le x_u \le v_u^{q(u)}, \end{cases} \tag{6.12}$$

et

$$H_u(x_u) = \begin{cases} \sum\limits_{i=0}^{q(u)} \gamma_u^i, & si \quad x_u \le \tilde{v}_u^0, \\ \sum\limits_{j=0}^{i-1} c_u^j(x_u - \tilde{v}_u^j) + \sum\limits_{i=0}^{q(u)} \gamma_u^i, & si \quad \tilde{v}_u^{i-1} \le x_u \le \tilde{v}_u^i, 1 \le i \le q(u)-1, \\ \sum\limits_{j=0}^{q(u)-1} c_u^j(x_u - \tilde{v}_u^j) + \sum\limits_{i=0}^{q(u)} \gamma_u^i, & si \quad \tilde{v}_u^{q(u)-1} \le x_u \le v_u^{q(u)}, \end{cases} \tag{6.13}$$

ou de façon plus contractée,

$$G_u(x_u) = \max\{a_u^i x_u + b_u^i : i = 0, \ldots, q(u)\}, \quad \forall x_u \in \mathbb{R}, \tag{6.14}$$

où

$$a_u^i = \begin{cases} 0 & si \quad i = 0, \\ \sum\limits_{j=0}^{i-1} c_u^j & si \quad i = 1, \ldots, q(u), \end{cases}$$

et

$$b_u^i = \begin{cases} \sum\limits_{j=0}^{q(u)} \gamma_u^j & si \quad i = 0, \\ \sum\limits_{j=0}^{q(u)} \gamma_u^j - \sum\limits_{j=0}^{i-1} c_u^j v_u^j & si \quad i = 1, \ldots, q(u). \end{cases}$$

De la même manière,

$$H_u(x_u) = \max\{\tilde{a}_u^i x_u + \tilde{b}_u^i : i = 0, \ldots, q(u)\}, \quad \forall x_u \in \mathbb{R}, \tag{6.15}$$

où

$$\tilde{a}_u^i = \begin{cases} 0 & si \quad i = 0, \\ \sum\limits_{j=0}^{i-1} c_u^j & si \quad i = 1, \ldots, q(u), \end{cases}$$

alors $a_u^i = \tilde{a}_u^i$, $\forall i = 0, \ldots, q(u)$ et

$$\tilde{b}_u^i = \begin{cases} \sum_{j=0}^{q(u)} \gamma_u^j & si \quad i = 0, \\ \sum_{j=0}^{q(u)} \gamma_u^j - \sum_{j=0}^{i-1} c_u^j \tilde{v}_u^j & si \quad i = 1, \ldots, q(u). \end{cases}$$

6.3.2 Approximation de la fonction objectif de (P)

Après avoir estimé la fonction $f_u(x_u)$ par la fonction DC polyédrale, $F_u(x_u)$, nous avons l'estimation suivante de la fonction objectif du programme (P)

$$F(x, \psi, \phi) = \sum_{u=1}^{n} F_u(x_u) - \sum_{u=1}^{n} r_u x_u, \quad \forall (x_u, \psi_u, \phi_u) \in \mathbb{R}^{n+2T}$$

ou

$$F = (G - \sum_{u=1}^{n} r_u x_u) - H$$

avec les fonctions G et H qui sont deux fonctions (finies) convexes polyédrales sur $\mathbb{R}^{n+2T}$ telles que, pour tout $x = (x, \psi, \phi) \in \mathbb{R}^{n+2T}$,

$$G(x, \psi, \phi) := \sum_{u=1}^{n} G_u(x_u) \quad \text{et} \quad H(x, \psi, \phi) := \sum_{u=1}^{n} H_u(x_u).$$

Par la suite, F est une fonction DC polyédrale avec les composantes DC

$$(G(x, \psi, \phi) - \sum_{u=1}^{n} r_u x_u) \quad \text{et} \quad H(x, \psi, \phi).$$

Par conséquent, le programme suivant est une approximation du programme (P)

$$(P_{DC}): \quad \min \left\{ F(x, \psi, \phi) = \left(G(x, \psi, \phi) - \sum_{u=1}^{n} r_u x_u \right) - H(x, \psi, \phi) : (x, \psi, \phi) \in \mathcal{D} \right\}.$$

6.4 Résolution de (P_{DC}) par DCA

D'après le schéma général de DCA, la résolution de (P_{DC}) exige la détermination de deux suites $\{(x^k, \psi^k, \phi^k)\}$ et $\{(y^k, \xi^k, \eta^k)\}$ telles qu'à chaque itération $k = 0, 1, 2, \ldots$:

$$(x^k, \psi^k, \phi^k) \longrightarrow (y^k,\ \xi^k,\ \eta^k) \in \partial H(x^k, \psi^k, \phi^k)$$

$$\swarrow$$

$$(x^{k+1}, \psi^{k+1}, \phi^{k+1}) \in \partial \left((G - \sum_{u=1}^{n} r_u x_u) + \chi_{\mathcal{D}} \right)^* (y^k,\ \xi^k,\ \eta^k)$$

($\chi_{\mathcal{D}}$ est la fonction indicatrice de l'ensemble $\mathcal{D}$).

6.4.1 Calcul de $\partial H(x, \psi, \phi)$

Puisque

$$H(x, \psi, \phi) := \sum_{u=1}^{n} H_u(x_u), \quad \forall (x, \psi, \phi) \in \mathbb{R}^{n+2T},$$

nous avons [42, 68]

$$\partial H(x, \psi, \phi) = \prod_{u=1}^{n} \partial H_u(x_u), \quad \forall (x, \psi, \phi) \in \mathbb{R}^{n+2T}. \tag{6.16}$$

En prenant en compte la définition de $H_u(x_u)$ dans (6.15) nous aurons [42, 68]

$$\partial H_u(x_u) = co\{\tilde{a}_u^i : i = 0, \ldots, q(u), \tilde{a}_u^i x_u + \tilde{b}_u^i = H_u(x_u)\}. \tag{6.17}$$

(*co* représente *l'enveloppe convexe*).

6.4.2 Calcul de $\partial \left((G - \sum_{u=1}^{n} r_u x_u) + \chi_{\mathcal{D}} \right)^* (y^k,\ \xi^k,\ \eta^k)$

Contrairement au calcul de $\partial H(x, \psi, \phi)$, pour lequel nous disposons d'une formule explicite, le calcul de $\partial \left((G - \sum_{u=1}^{n} r_u x_u) + \chi_{\mathcal{D}} \right)^* (y^k, \xi^k, \eta^k)$ exige la minimisation de la fonction convexe polyédrale $(G(x, \psi, \phi) - \sum_{u=1}^{n} r_u x_u) - \langle (x, \psi, \phi), (y^k, \xi^k, \eta^k) \rangle$ sur l'ensemble convexe et compact $\mathcal{D}$:

$$\min \left\{ G(x, \psi, \phi) - \sum_{u=1}^{n} r_u x_u - \langle (x, \psi, \phi), (y^k,\ \xi^k,\ \eta^k) \rangle : (x, \psi, \phi) \in \mathcal{D} \right\}.$$

Plus précisément

$$\min_{(x,\psi,\phi) \in \mathcal{D}} \left\{ \left(\sum_{u=1}^{n} \max_{i=0,\ldots,q(u)} \{a_u^i x_u + b_u^i\} - r_u x_u \right) - \langle (x, \psi, \phi), (y^k,\ \xi^k,\ \eta^k) \rangle \right\}.$$

Puisque H_u depend seulement de x_u, alors $\xi_t^k = 0$, $\eta_t^k = 0$, $\forall k, \forall t$, et finalement il nous faut résoudre le problème suivant afin de calculer $(x^{k+1}, \psi^{k+1}, \phi^{k+1}) \in \partial \left((G - \sum_{u=1}^{n} r_u x_u) + \chi_{\mathcal{D}} \right)^* (y^k,$ $\xi^k, \eta^k)$:

$$\min \left\{ \sum_{u=1}^{n} \max_{i=0,\ldots,q(u)} \{a_u^i x_u + b_u^i\} - \sum_{u=1}^{n} (r_u + y_u^k) x_u : (x, \psi, \phi) \in \mathcal{D} \right\}. \tag{6.18}$$

Le problème (6.18) est équivalent au programme linéaire suivant

$$(LP^{k+1}) : \begin{cases} \min \left\{ \sum_{u=1}^{n} \tau_u - \sum_{u=1}^{n} (r_u + y_u^k) x_u \right\} \\ s.c. \quad \begin{cases} a_u^i x_u + b_u^i \leq \tau_u, & i = 0, \ldots, q(u), u = 1, \ldots, n, \\ (x, \psi, \phi) \in \mathcal{D}, \\ \tau = (\tau_1, \ldots, \tau_n) \in \mathbb{R}^n. \end{cases} \end{cases}$$

Dans le sens où :

- Soit (x^*, ψ^*, ϕ^*) une solution optimale de (6.18), alors $(x^*, \psi^*, \phi^*, \tau^*)$, où $\tau_u^* = \max_{i=0,\ldots,q(u)}\{a_u^i x_u + b_u^i\}$ pour tout $u = 1, \ldots, n$, est une solution optimale de $(LP)^{k+1}$.
- Soit $(x^*, \psi^*, \phi^*, \tau^*)$ une solution optimale de $(LP)^{k+1}$ alors $\tau_u^* = \max_{i=0,\ldots,q(u)}\{a_u^i x_u + b_u^i\}$ pour tout $u = 1, \ldots, n$ et (x^*, ψ^*, ϕ^*) est une solution optimale de (6.18).

6.4.3 Algorithme de DCA pour résoudre (P_{DC})

En utilisant les relations (6.16) et (6.18), l'algorithme DCA pour résoudre le problème (P_{DC}) se résume ainsi

Algorithme 6.1 *Algorithme de DCA*

Initialisation :

– *Soit* $(x^0, \psi^0, \phi^0) \in \mathbb{R}^{n+2T}$ *et* $k \geq 0$.
– *Choisir la tolérance* ϵ *positive suffisamment petite.*

Répéter :
– *Calculer* (y^k, ξ^k, η^k) *via (6.16) et (6.17).*
– *Résoudre le programme linéaire* $(LP)^{k+1}$ *afin de calculer* $(x^{k+1}, \psi^{k+1}, \phi^{k+1})$.
– $k + 1 \leftarrow k$.
Jusqu'à : $F(x^k, \psi^k, \phi^k) - F(x^{k+1}, \psi^{k+1}, \phi^{k+1}) \leq \epsilon$.

D'après le théorème de convergence de DCA, l'algorithme 6.1 s'arrête après un nombre fini d'itérations sur un point comme (x^*, ψ^*, ϕ^*). (x^*, ψ^*, ϕ^*) est un point critique de (P_{DC}). Ce point est un minimum local de (P_{DC}), si $\partial H(x^*, \psi^*, \phi^*)$ est un singleton. Une telle situation arrive presque tout le temps.

6.4.4 Trouver un bon point de départ pour DCA

Nous avons testé plusieurs points de départ (x^0, ψ^0, ϕ^0) pour démarrer l'algorithme DCA :

(i) $(x^0, \psi^0, \phi^0) = (\bar{x}, \bar{\psi}, \bar{\phi})$ où $(\bar{x}, \bar{\psi}, \bar{\phi})$ est la solution optimale du problème relaxé de (P). Le problème relaxé (P) se produit en sous-estimant les fonctions en escalier par leur enveloppe convexe.

(ii) Soient $(\bar{x}, \bar{\psi}, \bar{\phi})$ la solution optimale du problème relaxé de (P) et $(x^0, \psi^0, \phi^0) := (x^0, \bar{\psi}, \bar{\phi})$, où x_u^0 est un des points de discontinuité de la fonction $f_u(x_u)$, i.e. $\left\{v_u^0, v_u^1, \ldots, v_u^{q(u)}\right\}$ tel que pour chaque $u = 1, \ldots, n$:
   – Soit $\bar{x}_u \in [v_u^i, v_u^{i+1}]$, où $i \in \{0, \ldots, q(u)\}$.
   – Si $\bar{x}_u \geq (v_u^i + v_u^{i+1})/2$ alors $x_u^0 = v_u^{i+1}$ sinon $x_u^0 = v_u^i$.

(iii) (x^0, ψ^0, ϕ^0) est le vecteur zéro.

Le choix (ii) est choisi car les résultats correspondant sont les meilleurs.

6.5 Résolution globale du problème (P)

Pour évaluer la qualité des solutions fournies par DCA, nous avons utilisé des méthodes globales afin de résoudre le problème (P). Une des méthodes de résolution globale du problème (P), consiste à utiliser la formulation binaire de (P), i.e., (P_{bin}). Nous avons utilisé le logiciel CPLEX en version 10.1 pour le résoudre. Une autre méthode est d'utiliser l'algorithme par séparation et évaluation (SE) proposé par Konno et al. [60]. Cet algorithme utilise la résolution du problème relaxé de (P) pour trouver des bornes inférieures. Afin d'avoir le problème relaxé dans un intervalle comme $[\underline{a}, \overline{b}]$, nous remplaçons les fonctions en escalier par leur enveloppe convexe. Les enveloppes convexes s'obtiennent en faisant connecter les deux ou trois points comme c'est montré sur la figure 6.6. La borne supérieure est mise à jour si nous trouvons une meilleure solution au problème (P). La séparation est faite sur l'intervalle qui porte l'écart le plus important entre l'enveloppe convexe et la fonction en escalier (voir la Figure 6.7). L'algorithme s'arrête dès qu'il trouve une solution ε-optimale, c'est-à-dire une solution pour laquelle l'écart entre la borne inférieure et la borne supérieure est inférieure à ε.

Afin d'améliorer la convergence de l'algorithme SE, nous avons développé une approche combinée de SE et DCA (SE-DCA). L'approche combinée (SE-DCA) essaie d'utiliser la performance de DCA et sa vitesse afin d'accélérer la convergence vers la solution globale. A chaque itération de SE, si les conditions sont favorables (par exemple, si la solution du sous-problème est très proche de la meilleure solution actuelle), SE fait appel à DCA. La solution du sous-problème est utilisée pour démarrer DCA. Le rôle de DCA est de trouver une solution améliorante. Une telle solution nous sert à améliorer la meilleure borne supérieure. Si la solution fournie par DCA n'est pas une solution améliorante, l'algorithme SE continue sa procédure jusqu'au prochain moment où il peut appeler DCA. L'approche SE-DCA s'arrête si elle trouve une solution ε-optimale.

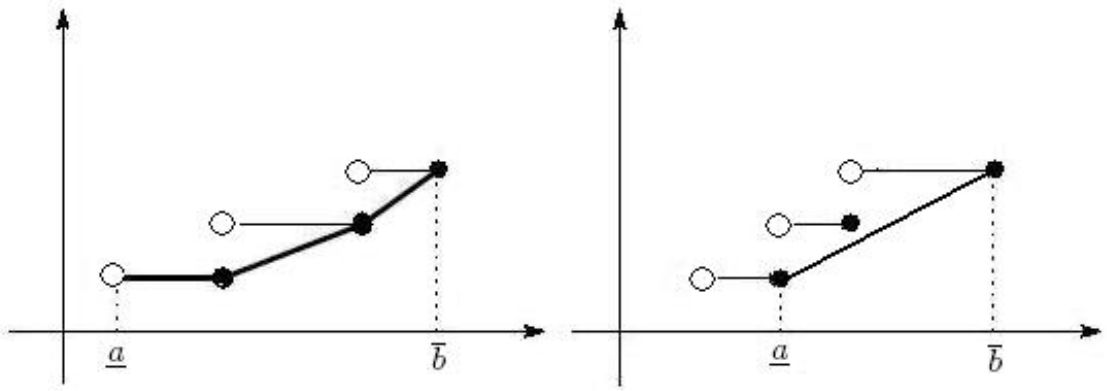


FIG. 6.6 – Enveloppe convexe de fonction de coûts de transaction en escalier

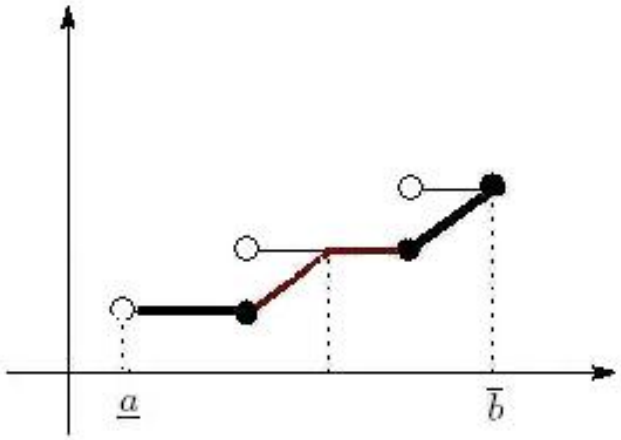


FIG. 6.7 – Séparation sur un intervalle

6.6 Expériences numériques

Nous avons testé les algorithmes et les modèles sur trois jeux de données.

Le tableau 6.1 montre les coûts de transaction que nous avons considérés. Dans ce tableau, le coût de chaque transaction (achat) représente 1% de la borne supérieure de chaque intervalle. C'est-à-dire, si $x_u \in (0.0, 2.0]$, alors le coût de transaction est de 0.02 et ainsi de suite pour les autres intervalles. En pratique, les coûts supérieures à 1% de l'investissement ne sont pas très commun, mais quand il s'agit d'investissements étrangers (sur les marchés étrangers) le coût peut être supérieur à 1% du capital investi [114].

-	v_u^0	$(v_u^0, v_u^1]$	$(v_u^1, v_u^2]$	$(v_u^2, v_u^3]$	$(v_u^3, v_u^4]$	$(v_u^4, v_u^5]$	$(v_u^5, v_u^6]$	$(v_u^6, v_u^7]$
-	0.0	(0.0,2.0]	(2.0,4.0]	(4.0,6.0]	(6.0,8.0]	(8.0,10.0]	(10.0,12.0]	(12.0,14.0]
γ_u^i :	0.0	0.02	0.04	0.06	0.08	0.10	0.12	0.14

TAB. 6.1 – Les coûts de transaction

Le niveau de risque autorisé est $w = 0.05$. La précision (ϵ) des solutions est 10^{-6} pour DCA. Pour tous les tests, les points $\tilde{v}_u^i : \quad i = 0, \ldots, q(u) - 1, u = 1, \ldots, n$ ont été définis par

$$\tilde{v}_u^i := v_u^i + 0.0001, \quad i = 0, \ldots, q(u) - 1.$$

Nous avons testé les algorithmes pour les modèles à six et sept morceaux. Pour le premier jeu de données, la borne supérieure sur la proportion du capital investi dans l'actif u est de $\beta_u = 14.0$ et le nombre de morceaux pour les fonctions de coût est de sept. Pour les autres jeux de données, β_u vaut 12.0 et le nombre de morceaux est de six.

Les tests ont été faits pour différentes valeurs du capital (M). La motivation est de vérifier le comportement des algorithmes face aux différentes valeurs de M. En augmentant la valeur de M, nous nous attendons à ce que le problème devienne de plus en plus difficile. La raison est qu'il y a une limite sur les proportions du capital que nous pouvons investir dans chaque actif. Alors, Il faut que l'algorithme distribue le capital entre plus d'actifs et de façon *efficace*.

Nous avons utilisé le logiciel CPLEX en version 10.1 pour résoudre le problème (P_{bin}). Nous avons considéré deux conditions d'arrêt pour CPLEX. La première concerne l'écart entre les meilleures bornes supérieure et inférieure calculées par CPLEX, nous l'avons fixée à $\epsilon := 10^{-3}$. La seconde concerne le temps d'exécution du logiciel. La limite sur le temps est fixé à 1200 secondes pour le premier ensemble de données et 1000 secondes pour les autres.

Tous les algorithmes ont été codés en C++ et ils ont été testés sur un ordinateur Pentium IV CPU 3 GHz et 1 Go de RAM. A chaque itération de DCA, SE et SE-DCA nous avons utilisé le logiciel CPLEX en version 10.1 pour résoudre les sous-problèmes linéaires.

Le premier jeu de données correspond aux prix hebdomadaires des actifs en indice *S&P* 500. L'horizon débute à Mars 1992 et elle continue pendant 289 semaines. Le nombre d'actifs est 457. Ici, n vaut 457 et T est égale à 289. Les données sont disponibles sur le site "http ://people.brunel.ac.uk/˜mastjjb/jeb/orlib/indtrackinfo.html".

Pour ces données, les algorithmes DCA et SE ont été utilisés. Nous avons mis deux critères d'arrêt pour SE. Le premier concerne l'écart entre les meilleures bornes inférieure et supérieure. Nous avons autorisé une marge de $\epsilon := 10^{-3}$. De plus, une limite de 900 itérations a été fixée sur le nombre maximum d'itérations. Nous avons choisi ce nombre, car chaque itération de SE exige 1.5 secondes pour résoudre un sous-problème. De cette façon, nous pouvons équilibrer les temps d'exécution de SE et CPLEX.

Le tableau 6.2 présente les résultats. Dans ce tableau, le nombre de sous-problèmes sondés par CPLEX (nodes), le nombre d'itération de SE et DCA (iter.), le temps de CPU en secondes

	CPLEX			Séparation et Evaluation			DCA		
M	nodes	Opt. val.	CPU	iter.	Val. opt.	CPU	iter.	Val. opt.	CPU
50	43300	0.017818	1208.840	88	0.016910	113.250	3	0.116416	3.297
60	41501	0.035766	1201.810	187	0.035137	244.734	2	0.055518	2.813
70	37301	0.055426	1200.940	900	0.058425	1139.719	2	0.074998	2.203
80	38301	0.074550	1201.300	31	0.074055	39.328	3	0.093639	3.422
90	37000	0.096650	1213.220	339	0.095012	433.468	3	0.154088	3.422
100	36501	0.116044	1207.390	900	0.135331	1151.859	2	0.155331	1.953

TAB. 6.2 – La performance des algorithmes pour le premier jeu de données ($S\&P$)

(CPU) et les valeurs ϵ-optimales (Val. opt.) pour chacun des algorithmes ont été présentés.

Le deuxième jeu de données correspond aux prix hebdomadaires des actifs de certaines compagnies américaines. Le nombre d'actifs est de 500. L'horizon débute le 27 décembre 2005 et elle continue pendant 20 semaines. Alors, n vaut 500 et T vaut 20. Les prix des actifs sont disponibles sur le site *YAHOO ! FINANCE page* (http ://finance.yahoo.com). Les rendements des actifs, r_{ut}, sont calculés par

$$r_{ut} = (p_{u,t+1} - p_{u,t})/p_{u,t},$$

où $p_{u,t}$ et $p_{u,t+1}$ sont le prix de l'actif u en deux instant successifs t et $t+1$ ($u = 1, \ldots, n$ et $t = 1, \ldots, T$).

Le tableau 6.3 montre le nombre de sous-problèmes sondés (nodes) par CPLEX pendant son exécution, le nombre d'itérations de SE et DCA (iter.), le temps d'exécution (CPU) en secondes et finalement les meilleures valeurs optimales (Val. opt.) fournies par les algorithmes et le logiciel CPLEX. Les valeurs optimales présentées dans le tableau ont été multipliées par (-1).

L'algorithme SE s'arrête soit si son nombre d'itération dépasse la limite de 5000 itérations, soit lorsque l'écart entre les meilleures bornes inférieure et supérieure est plus petit que $\epsilon := 10^{-3}$.

Nous avons choisi le troisième jeu de données d'une grande dimension. Il correspond aux prix hebdomadaires des actifs en indice *Russell* 3000. Le nombre d'actifs est de 2151 et l'horizon considéré commence en Mars 1992 et dure 30 semaines. Les données sont disponibles sur le site "http ://people.brunel.ac.uk/˜mastjjb/jeb/orlib/indtrackinfo.html". Ici, n vaux 2151 et T vaut 30.

Les algorithmes SE, DCA et SE-DCA ont été utilisés pour résoudre les problèmes. Également, le logiciel CPLEX a été utilisé. Le tableau 6.4 montre les résultats. A l'instar des autres tableaux, celui-ci présente le nombre de sous-problèmes sondés (nodes) par CPLEX pendant son exécution, le nombre d'itérations de SE et DCA (iter.), le temps d'exécution (CPU) en

	CPLEX			Séparation et Evaluation			DCA		
M	nodes	Opt. val.	CPU	iter.	Val. opt.	CPU	iter.	Val. opt.	CPU
10	146	0.723192	2.670	1370	0.721726	273.375	3	0.691447	0.750
20	510	1.509187	5.550	1252	1.509182	253.969	3	1.462120	0.609
30	1024	2.282747	17.240	5000	2.281142	990.844	3	2.239324	0.625
40	913	3.061181	14.480	1216	3.061173	247.828	3	2.979999	0.594
50	847	3.843308	15.170	3871	3.843130	756.719	3	3.830677	0.968
60	1483	4.610707	21.880	5000	4.610616	970.922	3	4.566408	0.593
70	869	5.367585	11.590	5000	5.367490	938.437	3	5.334487	0.594
80	338	6.092228	9.140	1589	6.090536	308.531	2	6.062314	0.453
90	201	6.806528	4.480	409	6.806510	79.250	2	6.806510	0.328
100	669	7.483239	18.630	5000	7.481416	841.282	3	7.478477	0.609
110	98	8.154372	4.310	467	8.152946	89.812	2	8.095418	0.344
120	901	8.785032	21.880	2139	8.784860	413.484	2	8.761774	0.360
130	541	9.403340	10.890	506	9.403320	99.047	3	9.363180	0.593
140	828	9.976866	17.580	4543	9.976844	871.765	3	9.964637	0.562
150	674	10.534156	15.950	1713	10.534136	333.281	2	10.526964	0.344
160	661	11.056510	16.860	976	11.056491	189.594	2	11.051380	0.344
170	1643	11.548996	36.020	2329	11.551624	440.672	3	11.551624	0.640
180	3362	12.000752	65.300	1079	12.000734	211.656	2	11.991416	0.329
190	27146	12.411932	497.050	2690	12.411772	504.266	3	12.393400	0.594
200	47483	12.820010	841.910	874	12.819989	166.843	2	12.813254	0.343
210	51601	13.219184	1003.200	2042	13.219160	388.860	2	13.205253	0.344
220	49801	13.599266	1000.030	635	13.599238	121.391	2	13.597422	0.343
230	48141	13.968100	1000.437	859	13.969617	164.140	3	13.939492	0.579
240	46301	14.332768	1000.594	2927	14.336082	565.375	3	14.297366	0.578
250	46901	14.701980	1000.469	1881	14.701948	361.203	3	14.674778	0.594

TAB. 6.3 – La performance des algorithmes pour le deuxième jeu de données

secondes et finalement les meilleures valeurs optimales (Val. opt.) fournies par les algorithmes et le logiciel CPLEX. De plus, le nombre des relances de DCA pendant l'exécution de SE-DCA (relance) y est inscrit.

L'algorithme SE s'arrête soit si son nombre d'itération est supérieur à 1000, soit lorsque l'écart entre les meilleures bornes inférieure et supérieure est plus petit que $\epsilon := 10^{-3}$. Puisque la taille des problèmes à résoudre est grande alors nous avons choisi 1000 comme limite au nombre d'itérations de SE pour que le temps maximum d'exécution de SE soit compatible avec celui de CPLEX.

Les valeurs optimales présentées ont été multipliées par (-1).

Afin de vérifier l'influence des fonctions de coûts de transaction en escalier sur les portefeuilles optimaux et sur la complexité des modèles, nous avons résolu le modèle (P) sans tenir en compte des coûts de transaction. Le logiciel CPLEX a été utilisé pour résoudre les problèmes respectifs. Nous avons utilisé les mêmes jeux de données présentés ci-dessus.

6.6.1 Commentaires sur les résultats

Aux vues des résultats présentés aux tableaux, nous constatons que

- DCA fournit des solutions de très bonne qualité dans un temps très court. Le nombre d'itération de DCA est 2 ou 3 pour tout les cas. Son temps d'exécution est inférieure à une seconde pour les deux premiers jeux de données. DCA donne les mêmes valeurs optimales que SE pour $M = 90$ et $M = 170$. Le nombre d'itération de DCA est 2 et 3, respectivement.
- Plus M est grand, plus les problèmes sont difficiles à résoudre pour CPLEX. Au contraire, DCA est très stable en face des différentes valeurs de M.
- Pour le troisième jeu de données, les problèmes sont plus difficiles car le nombre d'actifs est très grand. CPLEX exige entre 70.890 et 1002.800 secondes pour résoudre les problèmes. Tandis que, DCA résout les problèmes en moins de 23 secondes pour toutes les différentes valeurs de M.
- Nous constatons que l'approximation *polyèdrale* joue un rôle important dans l'efficacité de l'approche. Bien que l'algorithme SE et le logiciel CPLEX aient besoin de résoudre un nombre important des sous-problèmes pour atteindre les solutions, le nombre d'itération de DCA reste très faible. Pour tous les cas, nous avons besoin au maximum de 3 itérations pour résoudre le problème. La performance de DCA est peut-être plus visible dans l'approche combinée (SE-DCA). Pour les cas $M = 200, 230, 250$, les valeurs optimales fournies par l'approche combinée sont meilleurs que celles de CPLEX. De plus, dans la plupart des cas, SE-DCA donne des solutions de qualité supérieure par rapport aux solutions fournies par SE. D'ailleurs, pour $M = 40, 200$, SE-DCA s'arrête après avoir trouvé les solutions globales tandis que SE n'arrive pas à trouver les solutions dans la limite prévue.

	CPLEX			Séparation et Evaluation			SE-DCA				DCA		
M	nodes	Opt. val.	CPU	iter.	Val. opt.	CPU	iter.	relance	Val. opt.	CPU	iter.	Val. opt.	CPU
10	1673	0.484134	70.890	1000	0.472324	1098.047	1000	13	0.472324	1181.563	3	0.444994	4.891
20	2902	1.045465	153.090	1000	1.044500	1108.750	1000	15	1.044500	1243.329	3	1.012359	16.359
30	1899	1.599846	210.440	1000	1.599750	1103.594	1000	20	1.599844	1263.985	3	1.486622	18.406
40	2939	2.152991	377.310	1000	2.149407	1109.656	748	11	2.152989	852.547	2	2.053807	11.875
50	7541	2.679210	1001.500	1000	2.672287	1111.171	1000	7	2.672287	1104.109	3	2.668101	7.672
60	7573	3.234320	933.330	1000	3.217419	1104.453	1000	10	3.229966	1183.828	3	3.144015	22.297
70	4851	3.763378	1001.030	1000	3.739001	1104.859	1000	5	3.739001	1142.141	3	3.700731	14.859
80	4301	4.286734	1002.800	1000	4.284365	1170.062	1000	8	4.284365	1140.172	3	4.228966	14.000
90	5111	4.822331	1001.770	1000	4.812887	1110.359	1000	13	4.813162	1204.875	3	4.672412	14.375
100	4001	5.351168	1002.300	1000	5.349740	1113.265	1000	6	5.351164	1114.219	2	5.313198	2.625
110	5741	5.858839	1000.050	1000	5.858831	1117.219	1000	20	5.858831	1185.531	3	5.706908	19.359
120	5441	6.331496	1001.300	1000	6.330285	1106.265	1000	5	6.331485	1096.328	3	6.245026	13.078
130	3337	6.782229	328.360	546	6.782217	623.062	546	11	6.782217	602.500	3	6.780472	3.890
140	4001	7.180542	1001.312	1000	7.176099	1098.563	1000	7	7.179158	1106.563	3	7.125709	4.094
150	3601	7.576792	1000.969	1000	7.571015	1101.610	1000	19	7.571015	1137.781	3	7.483173	15.688
160	4601	8.968668	1000.938	1000	7.967381	1090.890	1000	8	7.967381	1082.860	2	7.959605	2.078
170	2651	8.360978	1000.937	1000	8.352708	1084.703	1000	18	8.353377	1258.031	3	8.290695	18.016
180	1871	8.731364	1001.109	1000	8.730721	1086.047	1000	30	8.730808	1120.312	2	8.700640	8.000
190	4261	9.094636	1001.578	1000	9.085002	1087.828	1000	10	9.085002	1105.375	3	9.063879	14.578
200	2001	9.441358	1001.031	1000	9.431421	1072.047	346	4	9.450112	389.562	3	9.373741	3.812
210	2311	9.786790	1001.000	1000	9.778142	1083.344	1000	17	9.778174	1159.328	3	9.766231	12.234
220	2061	10.121934	1001.078	1000	10.114978	1092.594	1000	4	10.121141	1075.891	3	10.088907	4.906
230	3101	10.447622	1001.140	823	10.457530	889.812	823	4	10.457530	891.625	3	10.431347	4.843
240	1891	10.772840	1001.125	1000	10.762068	1083.859	1000	10	10.762720	1082.375	2	10.762068	1.984
250	1761	11.087680	1001.015	273	11.102833	295.375	273	3	11.102833	298.500	3	11.078073	4.828

TAB. 6.4 – La performance des algorithmes pour le troisième jeu de données (*Russell*)

6.6.2 Influence des fonctions de coût de transaction

Après avoir éliminé les fonctions de coût, le modèle (P) devient très facile à résoudre. CPLEX résout le problème en moins d'une seconde. Ceci est le cas pour toutes les valeurs de M et pour les trois jeux de données testés. Si nous comparons les temps dont CPLEX a besoin pour résoudre le problème en présence des fonctions des coûts en escalier, nous constatons l'influence de ce type de fonctions sur la complexité du modèle.

En ce qui concerne les portefeuilles optimaux, l'inclusion des fonctions de coût les change. Le nombre d'actifs présents dans les portefeuilles optimaux sans les fonctions de coût est différent du nombre d'actifs en présence des fonctions de coût. La différence est plus remarquable pour le premier jeu de données. La figure 6.8 montre le nombre d'actifs dans les portefeuilles optimaux sans et avec les fonctions des coûts de transaction.

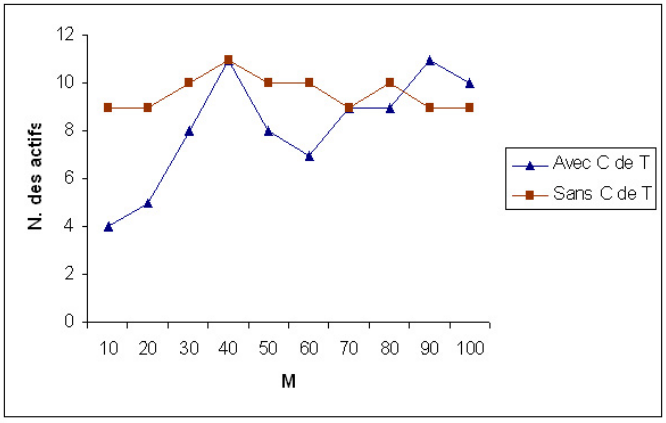
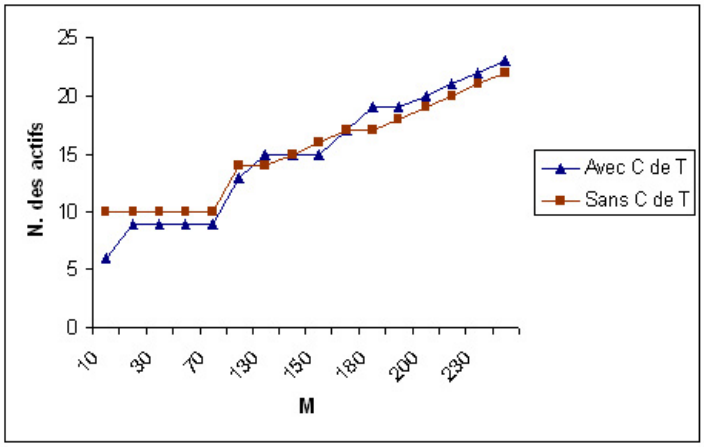


FIG. 6.8 – Le nombre d'actifs composant le portefeuille, sans et avec les coûts de transaction

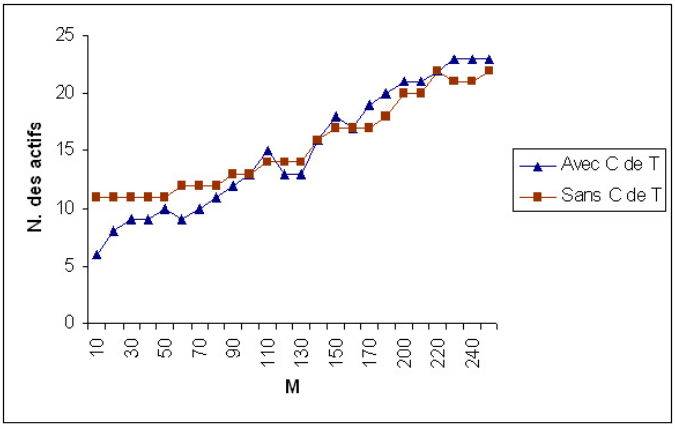
Pour le deuxième jeu de données, pour plus de la moitié des cas, les actifs composants les portefeuilles changent après que nous ajoutons les fonctions de coût. Dans tous les cas étudiés, les proportions du capital investi dans certains actifs changent. Les changement sont plus remarquables pour les cas où M est petit. Pour les tests avec de grandes valeurs de M, le nombre d'actifs composants les portefeuilles ne change pas considérablement. Pourtant, certains actifs sont remplacés par d'autres. La figure 6.9 montre les nombres d'actifs composants les portefeuilles pour les deuxième et troisième jeux de données. La figure montre seulement les cas où il y a un nombre différent d'actifs dans le portefeuille optimal. Par rapport au deuxième jeu de données, les tests sur le troisième jeu de données montre une l'influence plus visible des fonctions de coût. Pour toutes valeurs de M sauf $M = 190$, les compositions des portefeuilles optimaux changent.

6.7 Conclusion

Dans ce chapitre nous avons abordé une approche locale et continue pour résoudre le problème de choix de portefeuille en présence des fonctions de coût de transaction en escalier. Les résultats numériques montrent que les solutions fournies par DCA sont de bonne qualité. La convergence finie de l'algorithme DCA est liée à l'approximation des fonctions de coûts par des fonctions DC *polyédrales*. Notre approche pourrait être utilisée dans les autres domaines où les fonctions en escalier rendent le problème difficile à résoudre.



(a) Le nombre d'actifs pour le deuxième jeu d'essai



(b) Le nombre d'actifs pour le troisième jeu d'essai

FIG. 6.9 – Le nombre d'actifs composant le portefeuille, sans et avec les coûts de transaction

Chapitre 7

Une analyse au pire des cas pour l'investissement robuste en gestion de portefeuille en présence de contraintes de cardinalité

Résumé *La plupart des modèles utilisés en gestion de portefeuille sont des modèles sensibles par rapport aux erreurs des entrées et des paramètres. Pour cela, plusieurs modèles basés sur la stratégie min-max ont été proposés. Les modèles min-max fournissent des solutions robustes par rapport à ces erreurs. Dans ce chapitre, nous présentons un modèle min-max permettant la prise en compte des contraintes de cardinalité. La méthode DCA est utilisée pour résoudre le modèle. Les résultats sont comparés avec ceux fournis par CPLEX. Les résultats montrent l'efficacité de notre approche par rapport à CPLEX.*

7.1 Introduction

L'erreur d'estimation a toujours été reconnue comme un problème important dans la construction de portefeuille [10]. Une méthode pour attaquer ce problème consiste à utiliser les techniques d'optimisation robuste. Dans ce contexte, l'optimisation robuste essaie de minimiser le rendement du portefeuille et/ou maximiser le risque du portefeuille sous certaines contraintes tel que le résultat soit valable pour un niveau de confiance. Le modèle que nous avons étudié est une généralisation du modèle de Markowitz (voir [36] et [149]). La généralisation consiste à étudier l'analyse au pire des cas en ayant plusieurs matrices de covariance et plusieurs vecteurs de rendement moyen au lieu d'une seule matrice et un seul vecteur. De plus, le modèle offre la possibilité d'avoir des portefeuilles de benchmark. Ce sont des portefeuilles auxquelles l'investisseur essaie d'arriver. Enfin, le modèle contient certaines contraintes qui rendent le modèle plus réaliste. Il s'agit des contraintes de borne et de cardinalité ([16, 29, 51]). Ces contraintes limitent la proportion du capital investi dans chaque actif et le nombre d'actifs

composant le portefeuille. Le modèle est robuste, dans le sens où nous étudions un modèle *min-max* et les solutions d'un modèle min-max retournent les plus mauvaises stratégies d'investissement possibles, dans le sens où si une autre stratégie que celle du modèle min-max se réalise, l'investissement sera plus rentable. Pour cela, min-max parie sur plusieurs *scénarios* possibles ([36], [149]), où un scénario est une réalisation possible de chacun(e) de matrices de covariance ou de vecteurs de rendement. Plus précisément, une matrice de covariance est appelée un *scénario de risque* et chaque vecteur de rendement est nommé un *scénario de rendement*.

Le modèle min-max nous permet d'évaluer plusieurs cas simultanément et enfin de choisir celui qui est le pire. De cette façon, le portefeuille choisi est plus robuste, car nous sommes sûr que le portefeuille réalisé (dans le future) ne sera pas pire que celui prévu.

Notre modèle contient des contraintes de cardinalité. Ces contraintes avaient déjà été étudiées dans de nombreux articles et sur plusieurs modèles, comme le modèle MV, le modèle Déviation-Moyenne Absolue (MAD), etc. ([16, 29, 51, 61]). Pour la plupart des cas, les méthodes utilisées étaient des heuristiques. Ici, notre enjeu est d'utiliser la méthode DCA.

Le problème étudié s'exprime sous forme d'un programme avec la fonction objectif linéaire sous des contraintes linéaires et également des contraintes quadratiques. De plus, certaines variables de décision sont binaires. Après avoir utilisé un résultat de pénalité exacte (voir [105]), nous reformulons le modèle comme un modèle de programmation DC. Ensuite, nous utilisons DCA pour le résoudre.

Le reste du chapitre est organisé de façon suivante : la prochaine section concerne la description et la formulation du modèle. Le modèle sera généralisé en section 7.3, afin de prendre les contraintes de cardinalité en compte. En section 7.4, après avoir mis le problème sous forme d'un programme DC, nous utilisons DCA pour le résoudre. Les résultats numériques sont reportés dans la section 7.5. Nous terminons le chapitre par une conclusion.

7.2 Description et formulation

Dans cette section, nous allons d'abord présenter le modèle MV de Markowitz ([36], [112]). Les notations dont nous avons besoin pour le reste du chapitre sont présentées ci-dessous :

- T : l'horizon de planning de portefeuille ;
- n : le nombre d'actifs ;
- $\mathbf{r}_j \in \mathcal{R}^N, j = 1, \ldots, J$: les vecteurs de rendement ;
- $\Lambda_i \in \mathcal{R}^{n\times n}, i = 1, \ldots, I$: les matrices de variance-covariance ;
- $\mathbf{x}$: le vecteur des variables de décision ;
- $\overline{\mathbf{x}}_k \in \mathbb{R}^n : k = 1, \ldots, K$: les vecteurs de portefeuilles de benchmark ;
- $\mathbf{p} \in \mathbb{R}^n$: la position actuelle de l'investisseur ;
- $\mathbf{c}_b, \mathbf{c}_s \in \mathbb{R}^n$: les vecteurs qui représentent les coûts de transaction pour les achats et les ventes, respectivement ;

- $\mathbf{x}_b, \mathbf{x}_s$: les variables de décision d'achat et de vente ;
- $\mathbf{z}$: variables de décision (binaires), $z_i = 1$ si l'actif i est inclus dans le portefeuille et 0 sinon ;
- τ : le coût total des transactions (achat et vente) ;
- μ, ν : les pires valeurs de rendement et de risque ;
- l_i, u_i : les bornes inférieure et supérieure sur la proportion du capital investi dans l'actif i ;
- $card$: le paramètre de cardinalité qui représente *le nombre* souhaité d'actifs composant le portefeuille ;
- α : le paramètre d'aversion au risque ;
- $\mathbf{1} \equiv (1, \ldots, 1)^t$.

7.2.1 Modèle MV

Le modèle MV de Markowitz considère un portefeuille de n actifs défini en terme d'un ensemble des poids x_i tels que $i = 1, \ldots, n$. Ce sont des proportions du capital qui doit être distribué parmi les actifs. La somme de toutes les proportions doit être égale à 1, i.e.,

$$\mathbf{1}^t\mathbf{x} = 1.$$

Si l'investisseur détient actuellement des actifs $1, \ldots, n$, alors le vecteur $\mathbf{p}$ (tel que $\mathbf{1p} = 1$) représente sa position actuelle. Par contre, s'il ne possède aucun actif (mais, souhaitent acheter), alors $\mathbf{p} = \mathbf{0}$. La répartition du budget initial (de 1) peut être représentée par les contraintes suivantes :

$$\mathbf{p} + \mathbf{x}_b - \mathbf{x}_s = \mathbf{x}.$$

Soit τ les coûts totaux de transaction que l'investisseur engage pour son investissement [114]. Ceci considère les coûts totaux que l'investisseur doit payer pour que sa position actuelle $\mathbf{p}$ se transforme à $\mathbf{x}$. Les coûts correspondant aux $\mathbf{x}_b$ et $\mathbf{x}_s$ sont $\mathbf{c}_b$ et $\mathbf{c}_s$, autrement dit

$$\mathbf{c}_b^t\mathbf{x}_b + \mathbf{c}_s^t\mathbf{x}_s = \tau.$$

Le rendement espéré (attendu) du portefeuille est $(\mathbf{x} - \overline{\mathbf{x}})^t\mathbf{r}$ et le risque est mesuré par $(\mathbf{x} - \overline{\mathbf{x}})^t\Lambda(\mathbf{x} - \overline{\mathbf{x}})^t$. Il n'y a qu'une seule matrice de variance-covariance. Par conséquent, le modèle MV se formule comme un problème de programmation quadratique

$$\begin{aligned}
(MV) : \quad & \\
& \min\ \alpha(\mathbf{x} - \overline{\mathbf{x}})^t\Lambda(\mathbf{x} - \overline{\mathbf{x}}) - (1 - \alpha)[(\mathbf{x} - \overline{\mathbf{x}})^t\mathbf{r} - \tau] \\
& s.c. \\
& \mathbf{p} + \mathbf{x}_b - \mathbf{x}_s = \mathbf{x}, \\
& \mathbf{c}_b^t\mathbf{x}_b + \mathbf{c}_s^t\mathbf{x}_s = \tau, \\
& \mathbf{1}^t\mathbf{x} = 1, \\
& \mathbf{x}, \mathbf{x}_b, \mathbf{x}_s \geq \mathbf{0}.
\end{aligned}$$

où le paramètre α $(0 \leq \alpha \leq 1)$ détermine le niveau d'aversion au risque. En faisant varier α entre 0 et 1, nous obtenons l'ensemble de toutes les stratégies d'investissement efficace. $\alpha = 0$ correspond à la position riscophile et $\alpha = 1$ est la position totalement riscophobe.

La solution optimale du modèle (MV) est très sensible par rapport aux paramètres d'entrée du modèle. La moindre erreur d'estimation dans la matrice de covariance et/ou le vecteur de rendement se traduirait par une fausse solution. Plus particulièrement s'il s'agit d'une re-balancement de portefeuille, les erreurs peuvent faire engager l'investisseur à grandes valeurs de coûts de transaction [18]. Ces arguments nous encouragent à étudier les modèles pour lesquels les solutions optimales sont relativement moins sensibles à l'égard des erreurs éventuelles.

7.2.2 Modèle min-max pour les décisions robustes

Supposons qu'un investisseur est indécis entre plusieurs matrices de variance-covariance et/ou entre plusieurs vecteurs de rendements. Autrement dit, il existe un ensemble de matrices de covariance comme $\Lambda_i \in \mathcal{R}^{n\times n}, i = 1, \ldots, I$ et un ensemble de vecteurs de rendement moyen comme $\mathbf{r}_j \in \mathcal{R}^n, j = 1, \ldots, J$. De plus, soit K le nombre de portefeuilles de benchmark. Nous supposons qu'aucun(e) vecteur (matrice) n'a plus d'importance que les autres, même s'il y a des différences entre eux (elles). De cette façon, l'investisseur n'a aucun moyen de distinguer ces choix. Une représentation contractée du modèle min-max est donnée par

$$\min_{\mathbf{1}^t\mathbf{x}=1, \mathbf{x}\geq\mathbf{0}} \left\{ \alpha \cdot \max_{i,k}\{(\mathbf{x}-\overline{\mathbf{x}}_k)^t \Lambda_i (\mathbf{x}-\overline{\mathbf{x}}_k)\} - (1-\alpha) \min_{j,k}\{(\mathbf{x}-\overline{\mathbf{x}}_k)^t \mathbf{r}_j - t(\mathbf{x})\} \right\} \tag{7.1}$$

avec $i = 1, \ldots, I$,$j = 1, \ldots, J$ et $k = 1, \ldots, K$. La fonction $t(\mathbf{x})$ représente les coûts de transaction que l'investisseur doit payer s'il veut changer sa stratégie de $\mathbf{p}$ à $\mathbf{x}$, en effet

$$t(\mathbf{x}) = \mathbf{c}_b^t \mathbf{x}_b + \mathbf{c}_s^t \mathbf{x}_s.$$

Afin de résoudre (7.1), nous introduisons des variables μ et ν telles que :

$$\mu := \min_{j,k}[(\mathbf{x}-\overline{\mathbf{x}}_k)^t \mathbf{r}_j - t(\mathbf{x})] \quad \text{et} \quad \nu := \max_{i,k}(\mathbf{x}-\overline{\mathbf{x}}_k)^t \Lambda_i (\mathbf{x}-\overline{\mathbf{x}}_k).$$

De cette manière, μ représente le pire rendement possible et ν correspond à la pire valeur possible du risque. En utilisant les nouvelles variables, le modèle (7.1) se transforme à

$$
\begin{aligned}
(MMX):& \\
&\min\ \alpha\nu-(1-\alpha)\mu \\
&s.c. \\
&\qquad\qquad \mathbf{1}^t\mathbf{x}=1, \\
&\qquad\quad \mathbf{p}+\mathbf{x}_b-\mathbf{x}_s=\mathbf{x}, \\
&\qquad\quad \mathbf{c}_b^t\mathbf{x}_b+\mathbf{c}_s^t\mathbf{x}_s=\tau, \\
&(\mathbf{x}-\overline{\mathbf{x}}_k)^t\Lambda_i(\mathbf{x}-\overline{\mathbf{x}}_k)\le\ \nu \qquad : i=1,\dots,I, k=1,\dots,K, \\
&\qquad (\mathbf{x}-\overline{\mathbf{x}}_k)^t\mathbf{r}_j-\tau\ge\ \mu \qquad : j=1,\dots,J, k=1,\dots,K, \\
&\qquad\qquad \mathbf{x},\mathbf{x}_b,\mathbf{x}_s\ge\mathbf{0}.
\end{aligned}
$$

Au total, il y a IK contraintes quadratiques, $JK+n+2$ contraintes linéaires et les contraintes de non-négativité.

7.2.3 Robustesse du modèle min-max

La robustesse est une propriété de base des conditions d'optimalité pour les modèles min-max [36]. Soit

$$
\mathbf{x}^* := argmin_{\mathbf{1}^t\mathbf{x}=1,\mathbf{x}\ge\mathbf{0}}\left\{\alpha\cdot\max_{i,k}\{(\mathbf{x}-\overline{\mathbf{x}}_k)^t\Lambda_i(\mathbf{x}-\overline{\mathbf{x}}_k)\}-(1-\alpha)\min_{j,k}\{(\mathbf{x}-\overline{\mathbf{x}}_k)^t\mathbf{r}_j-t(\mathbf{x})\}\right\}
$$

et

$$
\Phi(\mathbf{x}^*) := min_{\mathbf{1}^t\mathbf{x}=1,\mathbf{x}\ge\mathbf{0}}\left\{\alpha\cdot\max_{i,k}\{(\mathbf{x}-\overline{\mathbf{x}}_k)^t\Lambda_i(\mathbf{x}-\overline{\mathbf{x}}_k)\}-(1-\alpha)\min_{j,k}\{(\mathbf{x}-\overline{\mathbf{x}}_k)^t\mathbf{r}_j-t(\mathbf{x})\}\right\},
$$

alors

$$
\begin{aligned}
\Phi(\mathbf{x}^*)\equiv&\min_{\mathbf{1}^t\mathbf{x}=1,\mathbf{x}\ge\mathbf{0}}\left\{\alpha\cdot\max_{i,k}\{(\mathbf{x}-\overline{\mathbf{x}}_k)^t\Lambda_i(\mathbf{x}-\overline{\mathbf{x}}_k)\}\right\} \\
&\qquad\qquad -\min_{\mathbf{1}^t\mathbf{x}=1,\mathbf{x}\ge\mathbf{0}}\left\{(1-\alpha)\min_{j,k}\{(\mathbf{x}-\overline{\mathbf{x}}_k)^t\mathbf{r}_j-t(\mathbf{x})\}\right\} \\
&=\alpha\cdot\max_{i,k}\{(\mathbf{x}^*-\overline{\mathbf{x}}_k)^t\Lambda_i(\mathbf{x}^*-\overline{\mathbf{x}}_k)\}-(1-\alpha)\min_{j,k}\{(\mathbf{x}^*-\overline{\mathbf{x}}_k)^t\mathbf{r}_j-t(\mathbf{x}^*)\} \\
&\ge\alpha\cdot\{(\mathbf{x}^*-\overline{\mathbf{x}}_k)^t\Lambda_i(\mathbf{x}^*-\overline{\mathbf{x}}_k)\}-(1-\alpha)\{(\mathbf{x}^*-\overline{\mathbf{x}}_k)^t\mathbf{r}_j-t(\mathbf{x}^*)\},\forall i,j,k.
\end{aligned}
$$

L'inégalité indique la non-infériorité de la stratégie min-max. Autrement dit, la performance attendue est garantie et elle satisfait les conditions au pire des cas et nous aurions une meilleure solution si un scénario différent du au de pire des cas soit réalisé.

7.3 Contraintes de cardinalité

Les contraintes de cardinalité ont pour objet de contrôler le nombre d'actifs composant le portefeuille optimal. Ces contraintes exigent les contraintes de bornes sur les actifs. Ce sont les contraintes qui limitent les proportions du capital investi dans chaque actif. Soient l_i et u_i les bornes inférieure et supérieure sur l'actif i. Les contraintes de bornes sont :

$$l_i z_i \leq w_i \leq u_i z_i, \quad z_i \in \{0,1\}, \;\; i = 1, 2, \cdots, n,$$

où z_i est une variable binaire et elle est égale à 1 si l'actif i est inclus dans le portefeuille et 0 sinon. La contrainte de cardinalité est

$$\sum_{i=1}^{n} z_i = card,$$

où *card* est le paramètre de cardinalité qui représente le nombre d'actifs qui composent le portefeuille.

Après avoir ajouté les contraintes de cardinalité et de bornes au modèle (MMX), nous obtenons le modèle suivant

$$
\begin{aligned}
(P_{card}) : \\
&\min \; \alpha\nu - (1-\alpha)\mu \\
&s.c. \\
&\mathbf{1}^t\mathbf{x} = 1, && && (7.2)\\
&\mathbf{p} + \mathbf{x}_b - \mathbf{x}_s = \mathbf{x}, && && (7.3)\\
&\mathbf{c}_b^t\mathbf{x}_b + \mathbf{c}_s^t\mathbf{x}_s = \tau, && && (7.4)\\
&(\mathbf{x} - \overline{\mathbf{x}}_k)^t \Lambda_i (\mathbf{x} - \overline{\mathbf{x}}_k) \leq && \nu \qquad : i = 1, \ldots, I, k = 1, \ldots, K, && (7.5)\\
&(\mathbf{x} - \overline{\mathbf{x}}_k)^t \mathbf{r}_j - \tau \geq && \mu \qquad : j = 1, \ldots, J, k = 1, \ldots, K, && (7.6)\\
&\sum_{i=1}^{n} z_i = card, && && (7.7)\\
&l_i z_i \leq x_i \leq u_i z_i && \qquad : i = 1, \ldots, n, && (7.8)\\
&z_i \in \{0,1\} && \qquad : i = 1, \ldots, n, && (7.9)\\
&\mathbf{x}_b, \mathbf{x}_s \geq \mathbf{0}. && && (7.10)
\end{aligned}
$$

Dans le modèle (P_{card}), il y a au total IK contraintes quadratiques, $JK + 3n + 3$ contraintes linéaires et les contraintes de non-négativité, ainsi que n variables binaires. Notons que (P_{card}) est un modèle général pour lequel toutes les informations ne sont pas forcément nécessaires ou disponibles. Par exemple, si $K = 0$, il n'y aura plus de portefeuille de benchmark et $\overline{\mathbf{x}}_k$ peut être supprimé. Si $I = 0$, les contraintes quadratiques seront éliminées. De la même manière, si $J = 0$, les contraintes (7.6) seront supprimées du modèle. De plus, si nous ne disposons d'aucune information sur les coûts de transaction, nous pouvons négliger les contraintes (7.3) et (7.4), ainsi que les variables $\tau, \mathbf{x}_b, \mathbf{x}_s$.

7.4 Programmation DC et DCA pour la résolution du problème

7.4.1 Reformulation

Soit $\mathcal{A} \subset \mathbb{R}^{4n+3}$ l'ensemble de tous les points $(\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}, \mu, \nu, \tau) \in \mathbb{R}^{4n+3}$ satisfaisant toutes les contraintes du modèle (P_{card}) à l'exception des contraintes (7.9).

Nous définissons

$$f(\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}, \mu, \nu, \tau) := \sum_{i=1}^{n} z_i(1 - z_i).$$

Clairement, f est une fonction concave et non-négative sur l'ensemble $\mathcal{A}$. D'ailleurs, le problème (P_{card}) s'écrit

$$\begin{aligned}
&\{(\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}, \mu, \nu, \tau) \in \mathcal{A} : z_i \in \{0, 1\} : i = 1, \ldots, n\} \\
&= \{(\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}, \mu, \nu, \tau) \in \mathcal{A} : f((\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}, \mu, \nu, \tau) = 0\} \\
&= \{(\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}, \mu, \nu, \tau) \in \mathcal{A} : f(\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}, \mu, \nu, \tau) \leq 0\}.
\end{aligned}$$

Alors nous pouvons exprimer le modèle (P_{card}) sous forme suivante

$$\min\{V(\mu, \nu) := \alpha\nu - (1-\alpha)\mu : (\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}, \mu, \nu, \tau) \in \mathcal{A}, \quad f(\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}, \mu, \nu, \tau) \leq 0\}. \quad (7.11)$$

La fonction objectif du modèle (7.11) est convexe. De plus, $\mathcal{A}$ est un polyèdre convexe et borné alors d'après [105], il existe $\sigma_0 \geq 0$ tel que pour tout $\sigma > \sigma_0$, le programme (7.11) est équivalent au

$$\min\{F(\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}, \mu, \nu, \tau) := \alpha\nu - (1 - \alpha)\mu + \sigma \sum_{i=1}^{n} z_i(1 - z_i) : (\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}, \mu, \nu, \tau) \in \mathcal{A}\}. \quad (7.12)$$

La fonction F est convexe (linéaire) par rapport aux variables $\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mu, \nu, \tau$ et concave par rapport aux variables $\mathbf{z}$. Par conséquent, F est une fonction DC polyédrale. Une formulation naturelle de (7.12) est

$$(P_{DC}) : \quad \min\{g(\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}, \mu, \nu, \tau) - h(\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}, \mu, \nu, \tau) : (\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}, \mu, \nu, \tau) \in \mathbb{R}^{4n+3}\},$$

où

$$g(\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}, \mu, \nu, \tau) := \chi_{\mathcal{A}}(\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}, \mu, \nu, \tau)$$

et

$$h(\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}, \mu, \nu, \tau) := (1 - \alpha)\mu - \alpha\nu + \sigma \sum_{i=1}^{n} z_i(z_i - 1).$$

Ici, χ_A est la fonction indicatrice sur $\mathcal{A}$, i.e.

$$\chi_{\mathcal{A}}(\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}, \mu, \nu, \tau) = \begin{cases} 0 & si \quad (\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}, \mu, \nu, \tau) \in \mathcal{A}, \\ \infty & sinon. \end{cases}$$

7.4.2 Résolution de (P_{DC}) par DCA

Selon le schéma général de DCA, il nous faut d'abord calculer le sous-gradient de la fonction h définie par $h(\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}, \mu, \nu, \tau) := (1-\alpha)\mu - \alpha\nu + \sigma \sum_{i=1}^{n} z_i(1-z_i)$. Le sous-gradient de h est

$$\mathbf{u}^l \in \partial h(\mathbf{x}^l, \mathbf{x}_b^l, \mathbf{x}_s^l, \mathbf{z}^l, \mu^l, \nu^l, \tau^l) \Leftrightarrow \; u_i^l = \begin{cases} \sigma(2z_{i-3n}^l - 1) & : si \quad i = 3n+1, \ldots, 4n, \\ (1-\alpha) & : si \quad i = 4n+1, \\ -\alpha & : si \quad i = 4n+2, \\ 0 & : sinon. \end{cases} \tag{7.13}$$

Ensuite, il nous faut résoudre le programme suivant :

$$\min\{-\langle(\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}, \mu, \nu, \tau), \mathbf{u}^l\rangle : (\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}, \mu, \nu, \tau) \in \mathcal{A}\}. \tag{7.14}$$

La solution optimale de ce programme linéaire sera $(\mathbf{x}^{l+1}, \mathbf{x}_b^{l+1}, \mathbf{x}_s^{l+1}, \mathbf{z}^{l+1}, \mu^{l+1}, \nu^{l+1}, \tau^{l+1})$. L'algorithme DCA pour résoudre (P_{DC}) se résume ainsi

Algorithme 7.1 *Algorithme de DCA*

Initialisation :
- *Choisir* $(\mathbf{x}^0, \mathbf{x}_b^0, \mathbf{x}_s^0, \mathbf{z}^0, \mu^0, \nu^0, \tau^0) \in R^{4n+3}$ *et* $l = 0$.
- *Choisir la tolérance* ε *positive et suffisamment petite.*
- *Choisir le paramètre de pénalité* σ *positive suffisamment grand.*

Répéter
- *Calculer* $\mathbf{u}^l \in \partial h(\mathbf{x}^l, \mathbf{x}_b^l, \mathbf{x}_s^l, \mathbf{z}^l, \mu^l, \nu^l, \tau^l)$ *via (7.13).*
- *Calculer* $(\mathbf{x}^{l+1}, \mathbf{x}_b^{l+1}, \mathbf{x}_s^{l+1}, \mathbf{z}^{l+1}, \mu^{l+1}, \nu^{l+1}, \tau^{l+1}) \in \partial g^*(\mathbf{u}^l)$ *en résolvant le programme linéaire (7.14).*
- $l + 1 \leftarrow l$.

Jusqu'à $\left\|(\mathbf{x}^{l+1}, \mathbf{x}_b^{l+1}, \mathbf{x}_s^{l+1}, \mathbf{z}^{l+1}, \mu^{l+1}, \nu^{l+1}, \tau^{l+1}) - (\mathbf{x}^l, \mathbf{x}_b^l, \mathbf{x}_s^l, \mathbf{z}^l, \mu^l, \nu^l, \tau^l)\right\| \leq \varepsilon$.

D'après le théorème de convergence de DCA, l'algorithme 7.1 a une convergence finie pour résoudre le programme (P_{DC}).

7.5 Expériences numériques

Afin d'évaluer la qualité des solutions fournies par DCA, nous les avons comparées avec les solutions optimales de (P_{card}). Nous avons utilisé le logiciel CPLEX en version 10.1 pour résoudre le programme (P_{card}).

L'algorithme DCA a été codé en C++ et a été testé sur un ordinateur Pentium IV 3 GHz et 1 Go RAM. Le logiciel CPLEX en version 10.1 a été utilisé pour résoudre les sous-problèmes linéaires (sous des contraintes quadratiques).

Le choix du paramètre de pénalité, i.e. σ, est important pour que nous puissions établir l'équivalence entre les problèmes (P_{card}) et (P_{DC}). Le paramètre doit être choisi de façon que la partie de la fonction de pénalité i.e.,

$$f(\mathbf{x}, \mathbf{x}_b, \mathbf{x}_s, \mathbf{z}, \mu, \nu, \tau) := \sum_{i=1}^{n} z_i(1 - z_i)$$

au programme (P_{DC}) ne soit pas trop favorisée, car dans le cas contraire les solutions ne seront pas de bonne qualité. Pour nos expériences, σ est égal à 1.0. Pour cette valeur, DCA trouve toujours des solutions qui sont admissibles au modèle (P_{card}).

Recherche d'un bon point initial pour DCA

Nous avons testé plusieurs choix de point initial, comme

- La solution optimale du problème relaxé de (P_{card}), pour obtenir le problème relaxé, il suffit de remplacer les contraintes (7.9) par $0 \leq z_i \leq 1$ pour tout $1 \leq i \leq n$.
- Après avoir résolu le modèle relaxé de (P_{card}), nous fixons les variables binaires non nulles à 1.
- Après avoir résolu le modèle relaxé de (P_{card}), nous fixons toutes les variables binaires à 1.

Les résultats obtenus par le premier choix sont mieux que ceux obtenus par les autres choix.

Générer les scénarios de risque

Les scénarios de risque peuvent être générés de différentes façons. Nous avons d'abord considéré un horizon de temps, et après nous l'avons divisée en plusieurs sous-intervalles de temps. Ensuite, nous avons calculé les matrices de covariance sur chacun des intervalles. Ce sont les scénarios de risque que nous avons adoptés.

Il existe des méthodes plus sophistiquées pour générer les scénarios de risque telles que les modèles ARCH-GARCH et bootstrap ([36], [37]).

Générer les scénarios de rendement

Les scénarios de rendement ont été générés par une approche basée sur la simulation probabilistique. Cette approche construit un arbre dont chaque nœud contient un cluster (groupe) des scénarios (qui sont des vecteurs dans $\mathbb{R}^n$). Parmi les scénarios, il y en a un qui est le scénario central (centroid). L'arbre final consiste en l'ensemble des scénarios centraux (centroid) de chaque nœud ainsi que l'ensemble de certaines arrêtes qui relient les nœuds.

Le schéma de base de cette approche se résume (voir aussi [38]) :

Algorithme 7.2 *(Algorithme pour générer les scénarios de rendement)*

Initialisation : *Créer un nœud de racine à N scénarios. Initialiser chaque scénario par des prix (au temps 0) d'actifs. Construire une queue et ajouter le nœud racine à la queue.*

Simulation : *Prendre un nœud de la queue et l'en éliminer. Faire une simulation sur une seule période ($[t, t+1] : t = 0, \ldots, T$) pour chaque scénario du nœud. Il existe deux sortes de simulation : parallèle et séquentielle.*

Choix des graines randomisées : *Nous choisissons certains scénarios comme des graines de façon aléatoire. Ce sont les scénarios autour desquels les autres scénarios seront regroupés.*

Regroupement des scénarios : *Nous regroupons les scénarios autour des graines, de façon que les scénarios qui se trouvent dans le même groupe soient les scénarios les plus proches d'une graine précise.*

Sélection des scénarios : *Pour chaque groupe, nous définissons un centre et nous choisissons le scénario qui est le plus proche du centre du groupe.*

Mise en queue : *Nous attribuons un enfant à chaque groupe (cluster) avec une probabilité proportionnelle au nombre de scénarios dans le groupe (cluster). Si les nœuds enfants ne sont pas des nœuds finaux, alors ajoutez-les à la queue. Si la queue n'est pas vide, aller à l'étape* Simulation *; sinon arrêter l'algorithme.*

Les autres paramètres

D'après le modèle (MMX), il existe d'autres paramètres que les scénarios de rendement et de risque, comme les coûts de transaction, et les portefeuilles de benchmark, etc. Nous avons fixé les coûts de transaction à 0.1% de chaque transaction que ce soit l'achat ou la vente. Les bornes inférieure et supérieure sur la proportion du capital investi dans chaque actif sont fixées à 0.01 et 1.0, respectivement. Nous avons considéré un seul portefeuille de benchmark, i.e. $K = 1$ et de plus nous avons mis $\mathbf{b}_k = 1.0/n$ pour tout $k = 1, \ldots, K$. Nous n'avons considéré aucun portefeuille initial, c'est-à-dire $\mathbf{p} = \mathbf{0}$. Le nombre de scénarios de rendement est égal à 10 et de risque est égal à 1. Les informations (les prix) sur les actifs ont été recueillies sur le site de Yahoo ! Finance. Elles correspondent aux prix mensuels de 98 actifs financiers depuis 121 mois. Plus précisément, le début de l'horizon est le mois d'août 1997 et dure 10 ans.

L'algorithme a été codé en C++ et il a été testé sur un ordinateur Pentium IV CPU 3 GHz et 1 Go de RAM. A chaque itération de DCA, nous avons utilisé le logiciel CPLEX

en version 10.1 pour résoudre les sous-problèmes linéaires (sous les contraintes linéaires et quadratiques).

7.5.1 Résultats numériques

L'importance du travail est liée à la présence des contraintes de cardinalité et au caractère min-max du modèle qui garantit la robustesse des résultats. Afin de pouvoir étudier ces deux aspects du modèle, les tests ont été effectués par rapport aux différentes valeurs de deux paramètres. Il s'agit du paramètre de cardinalité ($card$) et du paramètre d'aversion au risque (α). En fixant un de ces paramètres et en faisant varier l'autre, nous avons d'abord étudié l'efficacité de l'approche DCA pour résoudre le problème ; ensuite nous avons pu étudier l'influence de plusieurs scénarios au lieu d'un seul.

Afin de vérifier l'efficacité de DCA et la qualité des solutions fournies par DCA, nous avons résolu le problème (P_{card}) par le logiciel CPLEX en version 10.1. CPLEX trouve des solutions raisonnables au bout de vingt minutes c'est la raison pour laquelle nous avons fixé une limite de 1200 secondes sur le temps d'exécution de CPLEX.

D'abord, nous avons fixé le paramètre α à 1.0 et nous avons testé l'algorithme pour $card = 5, \ldots, 20$. Ce sont des valeurs pour lesquelles le problème (P_{card}) est difficile à résoudre. Nous avons choisi trois scénarios de rendement, un scénario de risque et un portefeuille de benchmark.

Le tableau 7.1 montre les résultats. Dans ce tableau, le paramètre de cardinalité (card), le rendement du portefeuille (Rend.), le risque attribué au portefeuille (Risque), le temps d'exécution (CPU) en secondes, pour l'algorithme DCA ainsi que le logiciel CPLEX ont été présentés.

Ensuite, nous avons choisi la valeur 10 pour le paramètre $card$ et nous avons fait varier α entre 0.0 et 1.0. $card$ a été fixé à 10 car c'est une des valeurs pour lesquelles le problème est difficile à résoudre.

Nous avons réalisé les tests pour un scénario de risque et un portefeuille de benchmark. Nous avons choisi trois scénarios de rendement. Les tests ont été effectués d'abord sur le modèle avec un des trois scénarios de rendement et ensuite sur le modèle en présence des trois scénarios. Le tableau 7.2 montre les résultats sur le modèle avec un seul scénario et le tableau 7.3 correspond aux résultats du modèle à plusieurs scénarios. Dans ces tableaux, le paramètre d'aversion au risque (α), les valeurs optimales (Val. opt.) et le temps d'exécution (CPU) en secondes, pour l'algorithme DCA ainsi que pour le logiciel CPLEX ont été présentés.

Enfin, la performance du modèle min-max a été étudiée en terme de pire des cas par des frontières de risque-rendement.

Les figures 7.1(a) et 7.1(b) comparent les stratégies d'investissement au pire des cas avec un seul scénario et avec multiples scénarios. Dans chaque figure, la frontière la plus basse correspond aux modèles à plusieurs (i.e., trois) scénarios. Les autres courbes correspondent

	CPLEX			DCA		
card	CPU	Risque	Rend.	CPU	Risque	Rend.
5	1200.250	0.001105	-1.000	0.765	0.002725	0.023861
6	1200.062	0.000867	-1.000	0.781	0.004050	0.013534
7	1200.079	0.000725	0.000	0.985	0.003968	0.025722
8	1200.062	0.000590	-1.000	0.797	0.003824	0.040129
9	1200.062	0.000578	-1.000	0.891	0.003704	0.030346
10	1200.063	0.000532	0.000	0.796	0.003279	0.000029
11	1205.141	0.000455	-1.000	0.984	0.001906	0.014492
12	1206.547	0.000409	-1.000	0.875	0.001894	0.011593
13	1200.469	0.000348	0.000	0.922	0.001764	0.003485
14	1203.516	0.000244	-1.000	0.672	0.001585	0.003696
15	1204.359	0.000268	0.000	0.750	0.001551	-0.006994
16	1200.046	0.000200	-1.000	0.890	0.001052	0.016135
17	1201.062	0.000194	-1.000	0.765	0.000786	0.011584
18	1200.047	0.000225	-1.000	0.625	0.000448	-0.008982
19	1203.843	0.000200	-1.000	0.687	0.000395	-0.029241
20	1200.750	0.000191	-1.000	0.688	0.000619	-0.045734

TAB. 7.1 – Les résultats pour différentes valeurs de *card*. Puisque $\alpha = 1.0$ alors les valeurs de *Risque* correspondent aux valeurs optimales de fonction objectif.

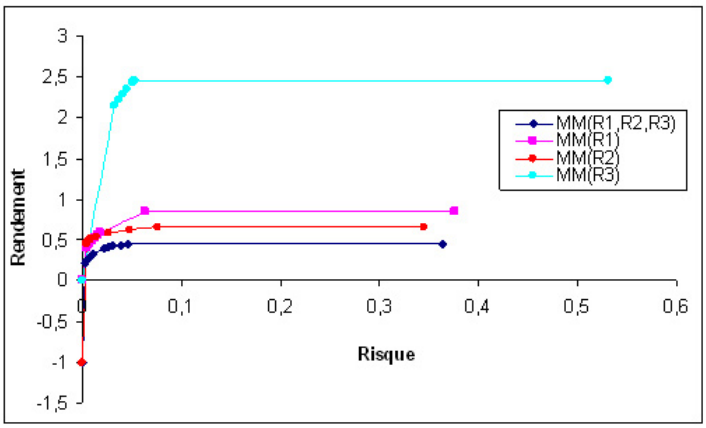
	CPLEX		DCA	
α	Val. opt.	CPU	Val. opt.	CPU
1.0	0.000586	1200.063	0.003279	0.781
0.96	-0.011740	126.297	-0.011813	0.625
0.95	-0.015946	0.500	-0.015946	0.532
0.94	-0.020428	0.438	-0.020428	0.625
0.93	-0.025249	0.859	-0.025249	0.609
0.92	-0.030411	0.453	-0.030411	0.625
0.91	-0.035917	0.844	-0.035917	0.453
0.9	-0.041749	0.906	-0.041749	0.484
0.8	-0.118714	0.203	-0.118714	0.422
0.7	-0.209630	0.172	-0.209630	0.375
0.6	-0.300572	0.172	-0.300572	0.343
0.5	-0.391536	0.485	-0.391536	0.282
0.4	-0.482551	0.188	-0.482551	0.344
0.3	-0.573571	0.469	-0.573571	0.297
0.2	-0.664618	0.172	-0.664618	0.281
0.1	-0.755666	0.187	-0.755666	0.281
0.0	-0.846713	0.187	-0.846713	0.297
Intervalle		0.18 - 1200		0.2 - 0.78

TAB. 7.2 – Comparaisons des résultats pour un seul scénario

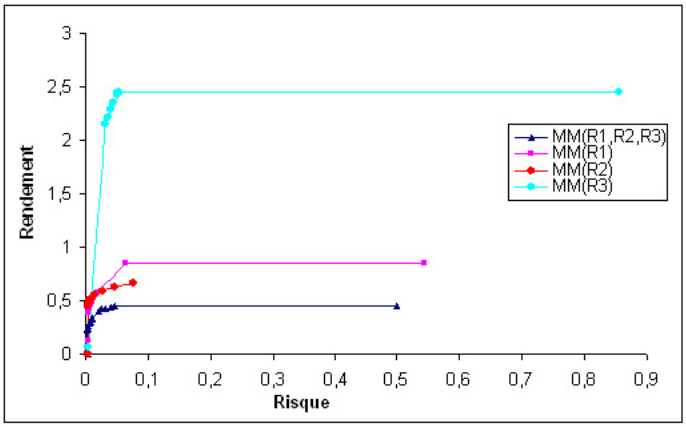
	CPLEX		DCA	
α	Val. opt.	CPU	Val. opt.	CPU
1.0	0.000532	1200.079	0.003279	0.782
0.96	-0.006169	71.954	-0.006112	0.703
0.95	-0.008485	3.859	-0.008479	0.516
0.94	-0.011030	0.485	-0.011069	0.547
0.93	-0.013859	0.922	-0.013859	0.656
0.92	-0.016877	0.531	-0.016877	0.562
0.91	-0.020105	0.593	-0.020105	0.703
0.9	-0.023393	0.625	-0.023478	0.594
0.8	-0.063321	0.563	-0.063321	0.375
0.7	-0.107344	0.672	-0.107344	0.391
0.6	-0.152804	0.203	-0.152804	0.453
0.5	-0.199755	0.234	-0.199755	0.328
0.4	-0.248653	0.203	-0.248653	0.343
0.3	-0.297840	0.203	-0.297840	0.344
0.2	-0.347028	0.203	-0.347028	0.359
0.1	-0.396215	0.203	-0.396215	0.313
0.0	-0.445402	0.203	-0.445402	0.328
Intervalle		0.2 - 1200		0.3 - 0.7

TAB. 7.3 – Comparaisons des résultats pour trois scénarios

aux modèles avec un seul scénario de rendement. Ces figures montrent que la stratégie de min-max à multiples scénarios présente une borne inférieure. Autrement dit, si un scénario différent de celui de min-max se réalise, la performance sera améliorée.



(a) CPLEX



(b) DCA

FIG. 7.1 – Les strategies min-max avec un seul scénario de rendement et avec multiples scénarios de rendement

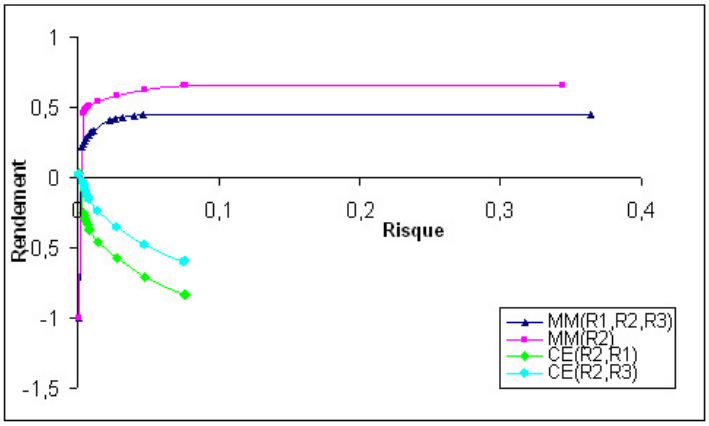
Nous avons aussi réalisé une évaluation croisée. Dans l'évaluation croisée, nous considérons un scénario et résolvons le problème qui lui correspond. Soit $\mathbf{x}^*$ la solution optimale du problème. Ensuite nous examinons la performance du portefeuille si un autre scénario était réalisé. Pour cela, il nous suffit de calculer le risque et le rendement en utilisant les scénarios et la solution du problème résolu. Les formules suivantes ont été utilisées afin de calculer les risques et les rendements :

$$(\mathbf{x}^* - \overline{\mathbf{x}}_k)^t \Lambda_i (\mathbf{x}^* - \overline{\mathbf{x}}_k) : i = 1, \ldots, I, k = 1, \ldots, K$$

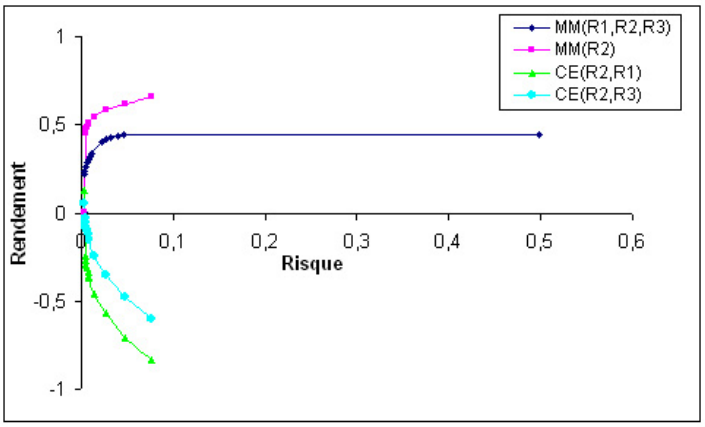
et

$$(\mathbf{x}^* - \overline{\mathbf{x}}_k)^t \mathbf{r}_j - \tau : j = 1, \ldots, J, k = 1, \ldots, K.$$

Pour nos tests, il existe trois scénarios R1, R2 et R3. Nous avons choisi le scénario R2. Les figures 7.2(a) et 7.2(b) montrent l'évaluation croisée du scénario R2. Ces figures montrent que si l'investisseur prend sa décision selon un seul scénario, la condition de son investissement dans le futur pourrait être pire que la stratégie min-max de tous les scénarios.



(a) CPLEX

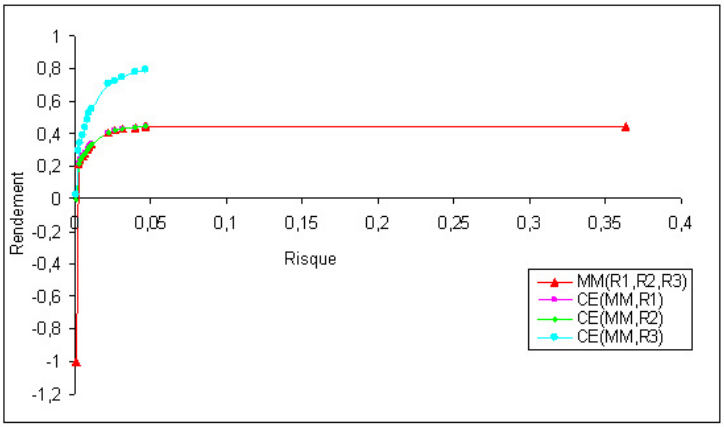


(b) DCA

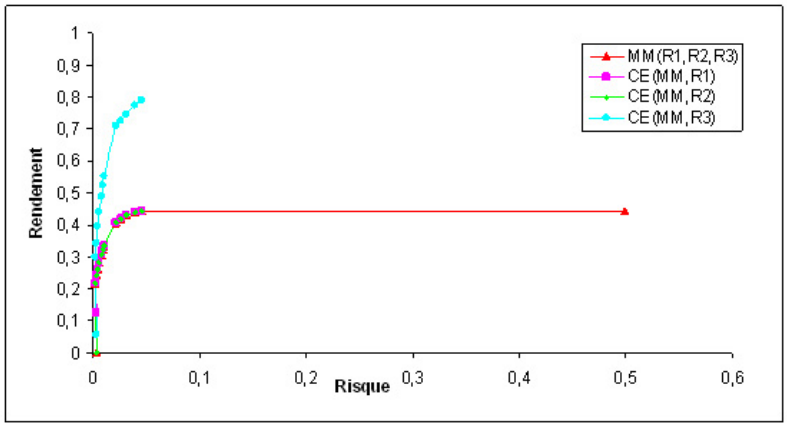
FIG. 7.2 – Evaluation croisée avec un seul scénario de rendement

Le modèle min-max fournit un portefeuille optimal qui est simultanément le pire des cas possibles. Par conséquent, la stratégie optimale obtenue de cette façon a une performance qui est en effet une borne inférieure. Autrement dit, la performance de la stratégie d'investissement s'améliore si un autre scénario se réalise. La non-infériorité de stratégie min-max est présentée dans les figures 7.3(a) et 7.3(b). La frontière efficiente de min-max est celle qui est la plus basse. Les autres frontières s'obtiennent si on considère un seul scénario dans le modèle. Ceci confirme l'aspect important de min-max : si le pire des cas ne se réalise pas, la performance du portefeuille pourra s'améliorer.

Les figures 7.4(a) et 7.4(b) présente l'avantage d'utiliser le min-max. Les trois courbes les plus hausses placées représentent les frontières efficientes tracées par les solutions optimales des



(a) CPLEX



(b) DCA

FIG. 7.3 – Non-infériorité de min-max

modèles avec un seul scénario. La courbe qui se trouve dans le centre est celle qui correspond à la stratégie fournie par le modèle à plusieurs scénarios. Les courbes en bas montrent le pire des cas pour les scénarios de rendement à l'égard du portefeuille optimal. Le modèle min-max construit un portefeuille optimal par le scénario le plus défavorable (pire des cas). Par conséquent, la performance de la stratégie de pire des cas est la meilleure borne inférieure pour toutes les stratégies possibles. Cette performance peut s'améliorer si un autre scénario que le pire des cas se réalise. De cette façon, la non-infériorité du modèle min-max garantit la robustesse des stratégies d'investissement.

Commentaires sur les résultats

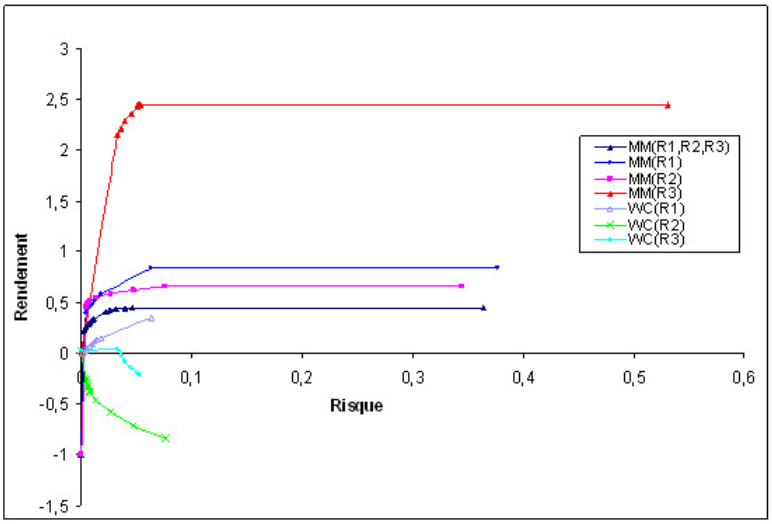
- Les expériences numériques montrent que le modèle min-max n'est pas stable à l'égard du paramètre d'aversion au risque, i.e., α. Car il est facile de résoudre pour les petites valeurs de α et la résolution du modèle devient de plus en plus difficile quand α tend

vers 1.0.

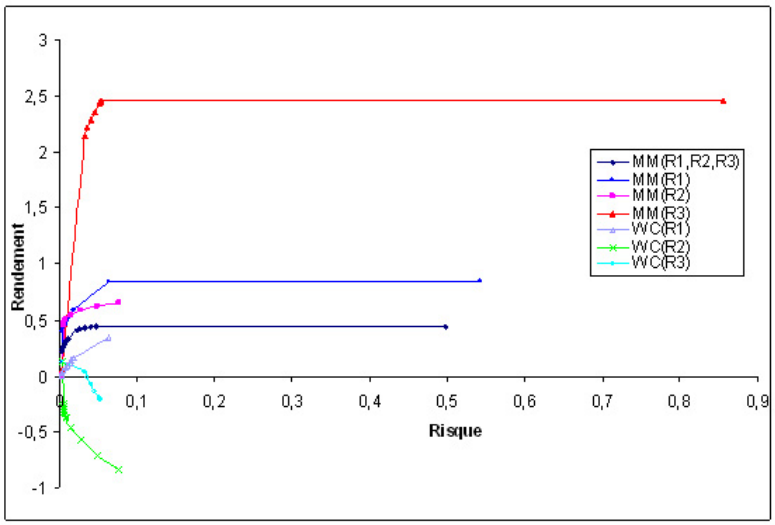
- Selon le tableau 7.1 les résultats fournis par DCA sont prometteurs, surtout au niveau du temps de CPU. Plus le paramètre de cardinalité est grand plus les solutions sont de meilleures qualités.
- Selon les tableaux 7.2 et 7.3 les solutions fournies par DCA sont de très bonnes qualités. Particulièrement, pour $\alpha = 0.96$ dans le tableau 7.2 et pour $\alpha = 0.9, 0.94$ dans le tableau 7.3. Ce sont les cas pour lesquels DCA donne des solutions de meilleurs qualités. Le temps de CPU est très court pour DCA, mais le temps de calcul de CPLEX augmente lorsque α augmente.
- En ce qui concerne les figures, nous constatons que les courbes tracées en aide des solutions fournies par DCA sont presque identiques aux courbes des solutions de CPLEX. Les différences majeures correspondent aux cas où α est égal à 1.0.

7.6 Conclusion

Dans ce chapitre, nous nous sommes adressés à la robustesse en gestion de portefeuille. Nous avons présenté un modèle min-max en présence de contraintes de cardinalité. Le modèle peut accepter plusieurs scénarios, qu'ils soient de rendement ou de risque. Le modèle s'exprime sous forme d'un programme avec des contraintes quadratiques et des variables mixtes. Nous l'avons reformulé sous forme d'un programme DC, ensuite nous avons utilisé la méthode DCA pour le résoudre. Les résultats numériques sont très encourageants car les comparaisons que nous avons effectuées avec les solutions fournies par le logiciel CPLEX, confirment la performance de DCA pour résoudre des problèmes de type min-max en présence de contraintes de cardinalité.



(a) CPLEX



(b) DCA

FIG. 7.4 – Analyse au pire des cas

Conclusion

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

Par un souci constant d'exploiter l'efficacité des décompositions DC et des points initiaux de DCA, nous avons étudié avec soin la structure spéciale de chaque problème pour trouver des algorithmes bien adaptés, robustes et peu coûteux. Les résultats montrent que notre objectif est atteint.

Pour le modèle de choix de portefeuille sous les contraintes de seuil d'achat, nous avons comparé les résultats avec des algorithmes par Séparation et Évaluation (SE).

Quant aux différents modèles de choix de portefeuille sous les contraintes de cardinalité, nous avons proposé une combinaison de l'algorithme par Séparation et Évaluation et DCA. Cette combinaison permet de trouver des bons points initiaux pour DCA et/ou de prouver la globalité de DCA. La performance de l'approche combinée est largement supérieure au schéma SE classique.

L'étude sur l'investissement robuste avait plus d'aspect financier et nous avons comparé les effets de plusieurs scénarios, que ce soit de risque ou de rendement, sur la stratégie optimale d'investissement. Sur le plan numérique, nous avons comparé les résultats fournis par DCA avec ceux de CPLEX.

Concernant le problème sous les contraintes de seuil, nous avons proposé une nouvelle fonction de pénalité qui satisfait non seulement la condition sur la valeur binaire des variables mais aussi les conditions de complémentarité.

L'art de modélisation occupe une place importante dans nos travaux. En effet, grâce aux techniques de formulation/reformulation nous avons pu mettre en évidence la forme DC des problèmes étudiés. Quant au problème d'optimisation de portefeuille avec les fonctions

des coûts de transaction en escalier, cet art est plus visible. Grâce à la formulation DC polyédrale, nous avons réussi à estimer et résoudre un problème non convexe et non lisse de façon efficace. La convergence rapide et finie de DCA s'explique par son efficacité et de plus par la formulation DC polyédrale de la fonction approchée.

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

Perspectives

Une question pertinente lors d'utilisation de DCA est : *comment trouver le bon point initial ?* Pour la plupart des problèmes les choix répondent parfaitement car nous avons une convergence très rapide vers des solutions de très bonnes qualités. Pourtant, la question reste ouverte pour des problèmes d'optimisation de portefeuille avec les fonctions des coûts de transaction ou l'Investissement robuste sous les contraintes de cardinalité car les résultats peuvent encore s'améliorer.

Pour la résolution des programmes linéaires et/ou quadratiques en variables mixte 0-1 dans le chapitre 4, le paramètre de pénalité influence considérablement la qualité des solutions obtenues. Il serait donc très intéressant d'étudier plus en détails le choix d'un tel paramètre.

Quant au modèle d'investissement robuste, nous avons étudier un modèle min-max *discret* qui était une généralisation du modèle MV de Markowitz sous les contraintes de cardinalité, la même généralisation peut s'effectuer sur un autre modèle où nous pouvons garder la même structure mais changer du modèle min-max de son cadre discret à un modèle continue.

Il existe des modèles en finance qui sont sous la forme d'une programmation mixte en variables entières. Il est toujours très coûteux de remplacer les variables entières par les variables binaires car le nombre de variables augmente de façon exponentielle. Pourrons-nous trouver des formulations DC afin de résoudre ces problèmes sans avoir besoin de remplacer les variables entières par celles binaires ?

Enfin, pour reformuler les programmes mixtes binaires sous forme DC, nous avons utilisé une fonction concave qui remplace les conditions binaires sur les variables, en outre nous avons proposé une autre fonction concave qui remplace les mêmes conditions ainsi que les contraintes de complémentarité. Ceci est fait grâce à la structure particulière du problème. Il existe des modèles qui contiennent seulement les contraintes de complémentarité. La question est : *comment peut-on reformuler le problème sous forme d'un programme DC ?*

Tous ces questions feront l'objet de nos travaux dans un futur proche.

ANNEXE

Computational Management Science manuscript No.
(will be inserted by the editor)

Hoai An LE THI[1] · **Mahdi MOEINI** ·
Tao PHAM DINH

Portfolio Selection under Downside Risk Measures and Cardinality Constraints based on DC Programming and DCA

Received: date / Accepted: date
[1] Corresponding author

Abstract In this paper, we consider the case of downside risk measures with cardinality and bounding constraints in portfolio selection. These constraints limit the amount of capital to be invested in each asset as well as the number of assets composing the portfolio. While the standard Markowitz's model is a convex quadratic program, this new model is a NP-hard mixed integer quadratic program. Realizing the computational intractability for this class of problems, especially large-scale problems, we first reformulate it as a DC program with the help of exact penalty techniques in DC (Difference of Convex functions) programming and then solve it by DCA. To check globality of computed solutions, a global method combining the local algorithm DCA with a Branch-and-Bound algorithm is investigated. Numerical simulations show that DCA is an efficient and promising approach for the considered problem.

Keyword. Portfolio selection, Downside risk, DC programming, DCA, Branch-and-Bound.
AMS 90C11, 90C26, 91B28

1 Introduction

In portfolio selection problem, given a set of available securities or assets, we want to find the optimum way of investing a particular amount of money in these assets. Each of the different ways to diversify this money among the several assets is called a portfolio [3]. For solving this portfolio problem, Markowitz ([13, 14]) has set up a quantitative framework. The Markowitz's model, or Mean-Variance model assumes that the return on a portfolio of assets can be completely described by the expected return and the variance of returns (risk) among these assets. For a particular universe of assets, the set of efficient portfolios of assets offers the minimum risk for a given level of return. These portfolios can be found by convex quadratic programs (QP). However, the Markowitz's standard model does not contain some practical constraints. For example, the standard Mean-Variance model has not got

Hoai An LE THI and Mahdi MOEINI
Equipe Algorithmique et Optimisation
Laboratoire Informatique Théorique et Appliquée (LITA), EA 3097
UFR MIM, Université Paul Verlaine - Metz,
Ile du Saulcy - 57045 Metz Cedex, France
Tel. : +33 (0)3 87 31 54 41,
Fax : +33 (0)3 87 31 53 09
E-mail: lethi@univ-metz.fr, moeini@univ-metz.fr

Tao PHAM DINH
Laboratory of Modelling, Optimization & Operations Research,
NationalInstitute for Applied Sciences - Rouen, BP 08, Place Emile Blondel F 76131
Mont Saint Aignan Cedex, France
E-mail: pham@insa-rouen.fr

any bounding constraints limiting the amount of money to be invested in each asset neither limits the number of assets composing the portfolio. This kind of constraints is very useful in practice and is called *cardinality and bounding constraints*. Fortunately, the standard model can be generalized to include these constraints.

There have been numerous techniques developed over the years in order to implement the theory of portfolio selection. Among them, the downside risk measures initialized by Roy [19] are intensively studied. The downside risk measures consider the risk so as to reflect the possibility of return being "too low" rather than "too high". Intuitively, the later is a "good" situation while the former is "bad".

In this paper, we focus on solving the problem of portfolio selection under downside risk measures with cardinality and bounding constraints. We investigate a local deterministic approach based on DC (Difference of Convex functions) programming and DCA (DC Algorithms) that were introduced by Pham Dinh Tao in their preliminary form in 1985. They have been afterwards extensively developed since 1994 by Le Thi Hoai An and Pham Dinh Tao and become classic and more popular (see e.g. [4], [7] - [11], [16,17], [20] and references therein). DCA has been successfully applied to many large-scale (smooth or nonsmooth) nonconvex programs in various domains of applied sciences, for which it provided quite often a global solution and proved to be more robust and efficient than standard methods (see e.g. [4], [7] - [11], [16,17], [20] and reference therein).

The existence of cardinality and bounding constraints makes the portfolio selection problem non-convex, and thus very difficult to solve by existing algorithms. By introducing the binary variables, we first express the aforementioned constraints as mixed zero-one linear constraints. Afterwards, using an exact penalty result, we reformulate the considered portfolio problem in terms of a DC program (a so-called DC program is that of minimizing a DC function over a convex set). DC programming approach and DCA are then suggested the resulting DC program. The efficiency of DCA is compared it with that of a branch-and-bound algorithm.

The rest of the paper is organized as follows. Section 2 presents the model of the portfolio selection problem under cardinality and bounding constraints, and its reformulation in terms of a DC program. Section 3 deals with DC programming and a special realization of DCA to the underlying portfolio problem. Section 4 is devoted to numerical simulations and some conclusions are provided in Section 5.

2 Portfolio selection problem under cardinality and bounding constraints

2.1 Problem formulation

First of all, let us remind the well known Markowitz's ([13,14]) Mean-Variance model for the portfolio selection problem. Let n be the number of available assets, r_i be the mean return of asset i, Q be an $n \times n$ Variance-Covariance (positive semidefinite) matrix such that its (i,j)-th element $\sigma_{i,j}$ is the covariance between returns of assets i and j and can be calculated using the following formula:

$$\sigma_{ij} = (1/m) \sum_{k=1}^{m} (r_{ik} - r_i)(r_{jk} - r_j) \,.$$

Here r_{ik} is the (i,k)-th historical data and m is the number of periods considered. Let R be the desired expected return and the decision variables y_i represent the proportion ($0 \leq y_i \leq 1$) of capital to be invested in asset i and $y^T = (y_1, ..., y_n)$. Using this notation, the standard Markowitz's Mean-Variance model can be mathematically formulated as ([1])

$$min \left\{ V(y) := y^T Q y : \sum_{i=1}^{n} r_i y_i = R, \sum_{i=1}^{n} y_i = 1, y_i \geq 0, : i = 1, \ldots, n \right\}. \tag{1}$$

Solving problem (1) minimizes the total variance (risk) associated with the portfolio by ensuring that the portfolio has an expected return R. Note that, in this paper, no short-sale is allowed.
This formulation is a simple convex quadratic program for which efficient algorithms are available. By solving the above QP for varying values of R, we can trace out the efficient frontier, a smooth non-decreasing curve that gives the best possible tradeoff of risk against return.

One can show that the following model and the standard Markowitz model have the same set of optimal solutions ([1]):

$$min\left\{V(y) := (1/m)\sum_{j=1}^{m}\left[\sum_{i=1}^{n}(r_{ij}y_i - R)\right]^2 : \sum_{i=1}^{n} r_i y_i = R, \sum_{i=1}^{n} y_i = 1, y_i \geq 0, : i = 1, \ldots, n\right\}. \quad (2)$$

This representation shows that the calculation of risk $V(y)$ involves finding the portfolio return for each of the m time periods in the asset history, and hence, obtaining the total squared deviation from the target return. Let us define

$$R_j := \sum_{i=1}^{n} r_{ij}y_i$$

as the portfolio return for the j-th time period. Then it is clear that $V(y)$ in (2) makes no distinction between the two cases $R_j > R$ and $R_j < R$. However, one probably prefers the former one. In this study, only deviations below the target return are considered as the risk, because the other returns are desirable. Usually, the only returns that disturb an investor are those below the target return. This measure is called a semi-variance. Semi-variance measures downside risk relative to benchmark given by expected return. It is just one of a number of possible measures of downside risk. It is worth noting that several definitions for downside risk have been proposed ([1], [15]). One approach mentioned in [1] and [15] is to write

$$V(y) := (1/m)\sum_{j=1}^{m}\left[min(0, (R_j - R))\right]^2.$$

So downside risk $V(y)$ only includes the squared deviations when portfolio expected return falls below the target.

To include the cardinality and bounding constraints in the above model, some additional notations are necessary. Let a_i and b_i be the lower and upper bounds, respectively, for the proportion of capital to be invested in asset i, with $0 \leq a_i \leq b_i \leq 1$. Let $card$ denote the desired maximum number of different assets that compose the portfolio with a positive value of investment. When $a_i > 0$, $card$ will be exactly the desired number of those different assets. Define the additional decision variables z_i equal to 1 if asset i is included in the portfolio and 0 otherwise. The generalized downside model for the portfolio selection problem under cardinality and bounding constraints can be written as

$$\begin{cases} \min V(y) := (1/m)\sum\limits_{j=1}^{m}\left[min(0, (R_j - R))\right]^2 \\ s.t: \begin{cases} \sum\limits_{i=1}^{n} r_i y_i = R, \\ \sum\limits_{i=1}^{n} y_i = 1, \\ \sum\limits_{i=1}^{n} z_i = card, \\ a_i z_i \leq y_i \leq b_i z_i \ , i = 1, \ldots, n \\ z_i \in \{0, 1\} \qquad , i = 1, \ldots, n. \end{cases} \end{cases} \quad (3)$$

Due to the cardinality and bounding constraints, this is a hard problem for which efficient algorithms are not available.

2.2 Reformulation

By introducing the new vector of variables $x^T = (x_1, ...x_m)$, Problem (3) can be equivalently reformulated as the following mixed integer quadratic program

$$\left\{\begin{array}{l}\min \theta(x,y,z) := (1/m)\sum\limits_{j=1}^{m} x_j^2 \\ s.t: \left\{\begin{array}{ll}\sum\limits_{i=1}^{n} r_i y_i = R, & \\ \sum\limits_{i=1}^{n} y_i = 1, & \\ \sum\limits_{i=1}^{n} z_i = card, & \\ x_j + \sum\limits_{i=1}^{n} r_{ij} y_i \geq R & , j = 1,\ldots,m \\ a_i z_i \leq y_i \leq b_i z_i & , i = 1,\ldots,n \\ z_i \in \{0,1\} & , i = 1,\ldots,n \\ x_j \geq 0 & , j = 1,\ldots,m. \end{array}\right. \end{array}\right. \tag{4}$$

Using the exact penalty techniques in DC programming developed in [12], we will formulate Problem (4) in the form of a convex-concave quadratic minimization problem with linear constraints (or nonconvex quadratic program), which is naturally a DC program. We define the set $\mathcal{A}$ as

$$\mathcal{A} := \left\{\begin{array}{l}(x,y,z) \in \mathbb{R}^m \times \mathbb{R}^n \times [0,1]^n : \sum\limits_{i=1}^{n} r_i y_i = R, \sum\limits_{i=1}^{n} y_i = 1, \sum\limits_{i=1}^{n} z_i = card, \\ x_j + \sum\limits_{i=1}^{n} r_{ij} y_i \geq R, j = 1,\ldots,m, : a_i z_i \leq y_i \leq b_i z_i, i = 1,\ldots,n, x_j \geq 0, j = 1,\ldots,m \end{array}\right\}.$$

It can be seen that the polyhedral convex set $\mathcal{A}$ is bounded with respect to the variables y and z but unbounded above with respect to the variable x. However, we can replace $\mathcal{A}$ with another bounded polyhedral convex set $\mathcal{A}'$ while maintaining the equivalence between Problem (3) and the new problem obtained from Problem (4) by only changing $\mathcal{A}$ for $\mathcal{A}'$. Indeed, it is clear, from the definition of x, that at the optimality we must have:

$$x_j = -\min\{0, R_j - R\} = \max\{0, R - R_j\} = \max\{0, R - \sum_{i=1}^{n} r_{ij} y_i\}, \text{ for } j = 1, ..., m.$$

It follows that the variables x_j, $j = 1, ..., m$, can be bounded above by

$$\beta_j := \max\{0, \alpha_j\},$$

where

$$\begin{array}{l}\alpha_j \ := \max\{R - \sum\limits_{i=1}^{n} r_{ij} y_i : (y,z) \in \mathbb{R}^n \times [0,1]^n : \sum\limits_{i=1}^{n} r_i y_i = R, \sum\limits_{i=1}^{n} y_i = 1, \sum\limits_{i=1}^{n} z_i = card, \\ a_i z_i \leq y_i \leq b_i z_i, i = 1,\ldots,n, x_j \geq 0, j = 1,\ldots,m\}. \end{array}$$

The scalars α_j, $j = 1, ..., m$, can be computed by solving the corresponding linear programs. The bounded polyhedral convex set $\mathcal{A}'$ then is defined by

$$\mathcal{A}' := \left\{\begin{array}{l}(x,y,z) \in \mathbb{R}^m \times \mathbb{R}^n \times [0,1]^n : \sum\limits_{i=1}^{n} r_i y_i = R, \sum\limits_{i=1}^{n} y_i = 1, \sum\limits_{i=1}^{n} z_i = card, \\ x_j + \sum\limits_{i=1}^{n} r_{ij} y_i \geq R, \ j = 1,\ldots,m, a_i z_i \leq y_i \leq b_i z_i, \ 0 \leq x_j \leq \beta_j, \ i = 1,\ldots,n, \ j = 1,\ldots,m \end{array}\right\}.$$

Hence, in what follows, we are dealing with the next nonconvex program

$$\begin{cases} \min \theta(x,y,z) := (1/m) \sum\limits_{j=1}^{m} x_j^2 \\ s.t: \begin{cases} \sum\limits_{i=1}^{n} r_i y_i = R, \\ \sum\limits_{i=1}^{n} y_i = 1, \\ \sum\limits_{i=1}^{n} z_i = card, \\ x_j + \sum\limits_{i=1}^{n} r_{ij} y_i \geq R \ , j = 1, \ldots, m \\ a_i z_i \leq y_i \leq b_i z_i \quad , i = 1, \ldots, n \\ z_i \in \{0,1\} \quad , i = 1, \ldots, n \\ 0 \leq x_j \leq \beta_j \quad , j = 1, \ldots, m, \end{cases} \end{cases} \tag{5}$$

which is equivalent to Problem (4).

Next, we define the function $p(x,y,z)$ as follows:

$$p(x,y,z) := \sum_{i=1}^{n} z_i(1 - z_i).$$

Clearly, p is a concave function which is nonnegative on $\mathcal{A}'$ and the feasible region of Problem (4) can be written as

$$\{(x,y,z) \in \mathcal{A}' : z \in \{0,1\}^n\} = \{(x,y,z) \in \mathcal{A}' : p(x,y,z) = 0\} = \{(x,y,z) \in \mathcal{A} : p(x,y,z) \leq 0\}.$$

So, Problem (5) can be expressed as:

$$\min\{\theta(x,y,z) := (1/m) \sum_{j=1}^{m} x_j^2 : (x,y,z) \in \mathcal{A}', \quad p(x,y,z) \leq 0\}. \tag{6}$$

Since the convex function θ is necessarily Lipschitz on the bounded polyhedral convex set $\mathcal{A}'$, according to exact penalty techniques in DC programming [8], [12], there is $t_0 \geq 0$ such that for any $t > t_0$, Problem (6) is equivalent to

$$\min \left\{ F_t(x,y,z) := (1/m) \sum_{j=1}^{m} x_j^2 + tp(x,y,z) : (x,y,z) \in \mathcal{A}' \right\}. \tag{7}$$

The function F_t is convex in the variables x and y and concave in the variable z. Consequently, it is a DC function. A standard DC formulation of Problem (7) is as follows:

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

where

$$g(x,y,z) := (1/m) \sum_{j=1}^{m} x_j^2 + \chi_{\mathcal{A}'}(x,y,z),$$

and

$$h(x,y,z) := t \sum_{i=1}^{n} z_i(z_i - 1).$$

Here $\chi_{\mathcal{A}'}$ is the indicator function of $\mathcal{A}'$, i.e. $\chi_{\mathcal{A}'}(x,y,z) = 0$ if $(x,y,z) \in \mathcal{A}'$, $+\infty$ otherwise.

3 Solution method via DC programming and DCA

3.1 DCA for general DC programs

Let $\Gamma_0(\mathbb{R}^n)$ denote the convex cone of all lower semicontinuous proper convex functions on $\mathbb{R}^n$, and consider the general DC program

$$(P_{dc}) \quad \alpha = \inf\{f(x) := g(x) - h(x) : x \in \mathbb{R}^n\} \tag{8}$$

where $g, h \in \Gamma_0(\mathbb{R}^n)$. Such a function f is called a DC function, and $g - h$, a DC decomposition of f while the convex functions g and h are DC components of f.
Let C be a convex set. The problem

$$\inf\{f(x) := k(x) - h(x) : x \in C\} \tag{9}$$

can be transformed to an unconstrained DC program by using the indicator function of C, i.e.,

$$\inf\{f(x) := g(x) - h(x) : x \in \mathbb{R}^n\} \tag{10}$$

where $g := k + \chi_C$.
Let $g^*(y) := \sup\{\langle x, y\rangle - g(x) : x \in \mathbb{R}^n\}$ be the conjugate function of g. Then, the following program is called the dual program of (P_{dc}):

$$(D_{dc}) \quad \alpha_D = \inf\{h^*(y) - g^*(y) : y \in \mathbb{R}^n\}. \tag{11}$$

Under the natural convention in DC programming, that is $+\infty - (+\infty) = +\infty$, and by using the fact that every function $h \in \Gamma_0(\mathbb{R}^n)$ is characterized as a pointwise supremum of a collection of affine functions, say

$$h(x) := \sup\{\langle x, y\rangle - h^*(y) : y \in \mathbb{R}^n\},$$

one can prove that $\alpha = \alpha_D$. We observe the perfect symmetry between primal and dual DC programs: the dual to (D_{dc}) is exactly (P_{dc}).
If g or h are polyhedral convex functions then (P_{dc}) is called a polyhedral DC program, which plays a main role in nonconvex programming (see [16], [17], [10], [11], and references therein), and enjoys interesting properties (from both theoretical and practical viewpoints) concerning the local optimality and the convergence of the DCA. Recall that, for $\theta \in \Gamma_0(\mathbb{R}^n)$ and $x_0 \in dom\ \theta := \{x \in \mathbb{R}^n : \theta(x_0) < +\infty\}$, $\partial\theta(x_0)$ denotes the subdifferential of θ at x_0, i.e., ([18],[5])

$$\partial\theta(x_0) := \{y \in \mathbb{R}^n : \theta(x) \geq \theta(x_0) + \langle x - x_0, y\rangle, \forall x \in \mathbb{R}^n\}. \tag{12}$$

The subdifferential $\partial\theta(x_0)$ is a closed convex set in $\mathbb{R}^n$. It generalizes the derivative in the sense that θ is differentiable at x_0 if and only if $\partial\theta(x_0)$ is reduced to a singleton which is exactly$\{\nabla\theta(x_0)\}$. The necessary local optimality condition for the primal DC program (P_{dc}) is:

$$\partial h(x^*) \subset \partial g(x^*). \tag{13}$$

A point x^* verifies the condition $\partial h(x^*) \cap \partial g(x^*) \neq \emptyset$ is called a critical point of $g - h$. The condition (13) is also sufficient for many important classes of DC programs, for example, in polyhedral DC programs with the function h being polyhedral, or in case the function f is locally convex at x^* ([9, 11, 16]).

The transportation of global solutions between (P_{dc}) and (D_{dc}) is expressed by:

$$[\cup_{y^* \in \mathcal{D}} \partial g^*(y^*)] \subset \mathcal{P}, \quad [\cup_{x^* \in \mathcal{P}} \partial h(x^*)] \subset \mathcal{D} \tag{14}$$

where $\mathcal{P}$ and $\mathcal{D}$ denote the solution sets of (P_{dc}) and (D_{dc}) respectively. Under technical conditions, this transportation holds also for local solutions of (P_{dc}) and (D_{dc}) ([7, 11, 16, 17, 10]).

Based on local optimality conditions and duality in DC programming, the DCA consists in the construction of two sequences $\{x^k\}$ and $\{y^k\}$, candidates to be optimal solutions of primal and dual programs respectively, such that the sequences $\{g(x^k) - h(x^k)\}$ and $\{h^*(y^k) - g^*(y^k)\}$ are decreasing,

and $\{x^k\}$ (resp. $\{y^k\}$) converges to a primal feasible solution $\widetilde{x}$ (resp. a dual feasible solution $\widetilde{y}$) verifying local optimality conditions and

$$\widetilde{x} \in \partial g^*(\widetilde{y}), \quad \widetilde{y} \in \partial h(\widetilde{x}). \tag{15}$$

The DCA then yields the next scheme:

$$y^k \in \partial h(x^k); \quad x^{k+1} \in \partial g^*(y^k). \tag{16}$$

In other words, these two sequences $\{x^k\}$ and $\{y^k\}$ are determined in the way that x^{k+1} (resp. y^k) is a solution to the convex program (P_k) (resp. (D_k)) defined by

$$\inf\{g(x) - [h(x^k) + \langle x - x^k, y^k\rangle] : x \in \mathbb{R}^n\}, \quad (P_k)$$

$$\inf\{h^*(y) - [g^*(y^k) + \langle y - y^k, x^{k+1}\rangle] : y \in \mathbb{R}^n\} \quad (D_{k+1}).$$

In fact, at each iteration one replaces in the primal DC program (P_{dc}) the second component h by its affine minorization $h_k(x) := h(x^k) + \langle x - x^k, y^k\rangle$ at a neighbourhood of x^k to give birth to the convex program (P_k) whose the solution set is nothing but $\partial g^*(y^k)$. Likewise, the second DC component g^* of the dual DC program (D_{dc}) is replaced by its affine minorization $(g^*)_k(y) := g^*(y^k) + \langle y - y^k, x^{k+1}\rangle$ at a neighbourhood of y^k to obtain the convex program (D_{k+1}) whose $\partial h(x^{k+1})$ is the solution set. DCA performs so a double linearization with the help of the subgradients of h and g^*.

It is worth noting that ([7,10,11,16,17]) DCA works with the convex DC components g and h but not the DC function f itself. Moreover, a DC function f *has infinitely many DC decompositions which have crucial impacts on the qualities* (speed of convergence, robustness, efficiency, globality of computed solutions,...) of DCA.

We mention now the main convergence properties of DCA ([7,10,11,16,17]). In this paragraph, denote by C (resp. D) a convex set containing the sequence $\{x^k\}$ (resp. $\{y^k\}$) and $\rho(g, C)$ (or $\rho(g)$ if $C = I\!R^n$) the modulus of strong convexity of g on C given by:

$$\rho(g, C) = \sup\{\rho \geq 0 : g - (\rho/2)\|\cdot\|^2 \text{ be convex on } C\}.$$

DCA's convergence properties: ([7,10,11,16,17])

DCA is a descent method without line search which enjoys the following properties:

i) The sequences $\{g(x^k) - h(x^k)\}$ and $\{h^*(y^k) - g^*(y^k)\}$ are decreasing and
   - $g(x^{k+1}) - h(x^{k+1}) = g(x^k) - h(x^k)$ iff $y^k \in \partial g(x^k) \cap \partial h(x^k), y^k \in \partial g(x^{k+1}) \cap \partial h(x^{k+1})$ and $[\rho(g, C) + \rho(h, C)]\|x^{k+1} - x^k\| = 0$. Moreover if g or h are strictly convex on C then $x^k = x^{k+1}$. In such a case DCA terminates at the kth iteration (finite convergence of DCA)
   - $h^*(y^{k+1}) - g^*(y^{k+1}) = h^*(y^k) - g^*(y^k)$ iff $x^{k+1} \in \partial g^*(y^k) \cap \partial h^*(y^k), x^{k+1} \in \partial g^*(y^{k+1}) \cap \partial h^*(y^{k+1})$ and $[\rho(g^*, D) + \rho(h^*, D)]\|y^{k+1} - y^k\| = 0$. Moreover if g^* or h^* are strictly convex on D, then $y^{k+1} = y^k$.
     In such a case DCA terminates at the kth iteration (finite convergence of DCA).

ii) If $\rho(g, C) + \rho(h, C) > 0$ (resp. $\rho(g^*, D) + \rho(h^*, D) > 0$)) then the series $\{\|x^{k+1} - x^k\|^2$ (resp. $\{\|y^{k+1} - y^k\|^2\}$ converges.

iii) If the optimal value α of problem (P_{dc}) is finite and the infinite sequences $\{x^k\}$ and $\{y^k\}$ are bounded then every limit point $\widetilde{x}$ (resp. $\widetilde{y}$) of the sequence $\{x^k\}$ (resp. $\{y^k\}$) is a critical point of $g - h$ (resp. $h^* - g^*$).

iv) DCA has a linear convergence for general DC programs.

v) DCA has a finite convergence for polyhedral DC programs.

Before closing this outline of DCA, it is crucial to keep in mind the second interpretation of DCA:

Let x^* be an optimal solution of primal DC program (P_{dc}) and $y^* \in \partial h(x^*)$. In virtue of (14) y^* is an optimal solution of the dual DC program (D_{dc}). Let h_* be the affine minorization of h defined by

$$h_*(x) := h(x^*) + \langle x - x^*, y^*\rangle$$

and consider the next convex program

$$\alpha_* := \inf\{g(x) - h_*(x) : \mathbb{R}^n\} = \inf\{f(x) + h(x) - h_*(x) : x \in \mathbb{R}^n\}. \tag{17}$$

Since the function $f_*(x) = f(x) + h(x) - h_*(x)$ is a convex majorization of $f, \alpha_* \geq \alpha$. But $f_*(x^*) = f(x^*) = \alpha$. Hence $\alpha_* = \alpha$. On the other hand, the optimal solution set of (17) is $\partial g^*(y^*)$ that is contained in the optimal solution set P of (P_{dc}), following (14). Taking into account of (14) and the decrease of the sequence $\{g(x^k) - h(x^k)\}$, one can understand better the role played by the linearized programs (P_k) and (17) and explain partially the reason why DCA converges to an optimal solution of (P_{dc}) from a good initial point.

3.2 DCA for solving (7)

According to the description of DCA in the previous subsection, we have to compute $\partial h(x, y, z)$ and $\partial g^*(u, v, w)$. As usual, the first computation is explicit

$$\partial h(x, y, z) = \{\nabla h(x, y, z)\} = \{0, 0, t(2z - e^{(n)})\} \tag{18}$$

where $e^{(n)} \in \mathbb{R}^n$ is the vector of ones. On the other hand, the computation of $\partial g^*(u, v, w)$ needs to solve the following convex quadratic program

$$\min \left\{ (1/m) \sum_{j=1}^{m} x_j^2 - \langle (x, y, z), (u, v, w) \rangle : (x, y, z) \in \mathcal{A}' \right\} \tag{19}$$

because its solution set, reduced to a singleton, is exactly $\partial g^*(u, v, w) = \{\nabla g^*(u, v, w)\}$. Finally, DCA applied to Problem (7) can be described as follows.

Algorithm DCA

1. **Initialization**: Let ε be sufficiently small positive number, let $(x^0, y^0, z^0) \in \mathbb{R}^m \times \mathbb{R}^n \times [0, 1]^n$, and set $k = 0$;
2. **Iteration**: $k = 0, 1, 2, ...$
   set $u^k = 0$ and $v^k = 0$ and $w^k = t(2z^k - e^{(n)})$.
3. Solve the next convex quadratic program

$$\min\{(1/m) \sum_{j=1}^{m} x_j^2 - \langle (x, y, z), (u^k, v^k, w^k) \rangle : (x, y, z) \in \mathcal{A}'\} \tag{20}$$

   to obtain $(x^{k+1}, y^{k+1}, z^{k+1})$.

4. If $\| (x^{k+1}, y^{k+1}, z^{k+1}) - (x^k, y^k, z^k) \| \leq \varepsilon$, then STOP and $(x^{k+1}, y^{k+1}, z^{k+1})$ is the computed solution, otherwise set $k = k + 1$ and go to Step 2.

4 Implementation and computational experiments

For evaluating the quality of the solutions computed by DCA we solve Problem (5) by a standard branch-and-bound algorithm for mixed zero-one programs. More precisely, the lower bound is computed by solving the relaxed problem of (4) (the binary constraints $z_i \in \{0, 1\}$ are replaced by $0 \leq z_i \leq 1$) which is a convex quadratic program, and the upper bound is updated when a better feasible solution (i.e., with smaller objective value) to Problem (5) is found. The subdivision is performed by setting $z_i = 0$ or $z_i = 1$.

Furthermore, in order to globally solving Problem (5) we develop a combination of DCA and the branch-and-bound algorithm (denoted B&B-DCA). The combined procedure aims at checking globality of solutions computed by DCA applied to the equivalent DC program (7) and possibly providing better solutions for restarting DCA. By this way it improves the upper bounds and the convergence of the branch-and-bound scheme.

The combined B&B-DCA scheme can be summarized as follows:

Algorithm B&B-DCA

Initialization.

Solve the relaxed problem of (5) to obtain an optimal solution $(\bar{x}, \bar{y}, \bar{z})$. If $\bar{z}_i \in \{0,1\}$ for all $i = 1, \ldots, n$, then STOP, $(\bar{x}, \bar{y}, \bar{z})$ is an optimal solution of (5), else apply DCA to Problem (7) for getting a solution denoted $(\tilde{x}, \tilde{y}, \tilde{z})$. If $\tilde{z}_i \in \{0,1\}$ for all $i = 1, \ldots, n$, then set $\gamma^0 := (1/m)\sum_{i=1}^{m} \tilde{x}_j^2$, else $\gamma^0 := +\infty$.

Set $\mathcal{M} \leftarrow \{M_0\} := \mathcal{A}', \ell \leftarrow 0$.

Set $\beta^0 := \beta(M_0) := (1/m)\sum_{i=1}^{m} \bar{x}_j^2$.

Iteration $\ell \geq 0$

Select M_ℓ such that $\beta^\ell = \beta(M_\ell) = \min\{\beta(M) : M \in \mathcal{M}\}$.

If $(\gamma^\ell - \beta^\ell) < \varepsilon$ then STOP, $(\tilde{x}, \tilde{y}, \tilde{z})$ is an $\varepsilon-$ solution of Problem (5).

Divide M_ℓ into $M_{\ell_1} = \{(x,y,z) \in M_\ell : z_{j*} = 1\}$ and $M_{\ell_2} = \{(x,y,z) \in M_\ell : z_{j*} = 0\}$ via an index $j*$ such that $\bar{z}_{j*}^{\ell} \notin \{0,1\}$.

For each M_{ℓ_i} $(i = 1, 2)$

(i) Solve the corresponding relaxed problem to obtain an optimal solution $(\bar{x}^{\ell i}, \bar{y}^{\ell i}, \bar{z}^{\ell i})$ and the optimal value $\beta(M_{\ell_i})$.

(ii) If $\bar{x}^{\ell i}, \bar{y}^{\ell i}, \bar{z}^{\ell i})$ is a better feasible solution of Problem (5) then update $\gamma^\ell := \beta(R_{\ell_i})$ and the current best feasible solution $(\tilde{x}, \tilde{y}, \tilde{z})$.

(iii) If the condition of restarting DCA (see Subsection 4.2) is satisfied then apply DCA to problem (7) to obtain $(x_t^{\ell i}, y_t^{\ell i}, z_t^{\ell i})$.

If $(x_t^{\ell i}, y_t^{\ell i}, z_t^{\ell i})$ is a better feasible solution of Problem (5), then update $\gamma^\ell = (1/m)\sum_{i=1}^{m} {x_t^{\ell i}}_j^2$ and the current best feasible solution $(\tilde{x}, \tilde{y}, \tilde{z}) := (x_t^{\ell i}, y_t^{\ell i}, z_t^{\ell i})$.

Set $\mathcal{M} \leftarrow \mathcal{M} \setminus \{M_i : \mu(M_i) > \gamma^\ell\}$ and go to iteration $\ell + 1$.

4.1 Extension to general mixed 0-1 quadratic programming

It is easy to verify that the algorithms DCA, B&B and B&B-DCA developed above for solving (4) remain valid for general mixed 0-1 convex quadratic programs of the form

$$\begin{cases} \min \left(\frac{x}{y}\right)^T C \left(\frac{x}{y}\right) + a^T x + b^T y \\ s.t.\ \ A_1 x + B_1 y + C_1 z \leq d_1,\ \ A_2 x + B_2 y + C_2 z \leq d_2,\ \ z \in \{0,1\}^q,\ \ x \geq 0, y \geq 0, \end{cases} \tag{21}$$

where C is a symmetric positive semidefinite, $A_i, B_i, C_i, (i = 1, 2)$,are matrices of appropriate orders and d_1, d_2 are the corresponding vectors. This class of nonconvex programs - which plays a crucial role in Optimization and Operations Research - is known as that NP hard problems whose finding global solutions is a very important challenging work, especially for large-size problems. Of course their complexity depends on their specific structure. In other words, an algorithm could be very efficient for some ones and less performant for others. In practice, only computational experiments allow to assess the effectiveness and the efficiency of the proposed algorithms.

For our portfolio selection under downside risk measures and cardinality constraints, the numerical simulation is carried out on the data that are generated using weekly stock prices of 550 American healthcare companies over 20 weeks (from 27 December 2005 to 15 May 2006). The data are available through YAHOO FINANCE page (http://finance.yahoo.com). The returns, r_{ij}, have been calculated using the formula:

$$r_{ij} = (p_{i,j+1} - p_{i,j})/p_{i,j},$$

where $p_{i,j}$ is the closing price of the share i at the week j ($i = 1, \ldots, n$ and $j = 1, \ldots, m$). The number n of different assets is equal to 550 and the number m of the considered periods is 20.

The algorithms are coded in C++ and run on a Pentium 1.600 GHz of 512 DDRAM. To solve the resulting convex quadratic programs, we use the software CPLEX version 9.1.

4.2 Finding a good initial point for DCA

According to Subsection 3.1, one of the key questions in DCA is finding a good initial solution to start. For this problem we have tested DCA with the following choices of initial points:

- The optimal solution of the relaxed problem of (5), denoted $(\bar{x}, \bar{y}, \bar{z})$;
- The point obtained from $(\bar{x}, \bar{y}, \bar{z})$ by changing each noninteger values $\bar{z}_i$ to one or zero.
- The point obtained from $(\bar{x}, \bar{y}, \bar{z})$ by rounding each nonzero value $\bar{z}_i$ to one;
- The optimal solution of the next DC program with known optimal value

$$0 := \min \left\{ p(x, y, z) := \sum_{i=1}^{n} z_i(1 - z_i) : (x, y, z) \in \mathcal{A}' \right\}.$$

In our numerical simulations, the second choice is the best.

4.3 When DCA is restarted?

During the branch-and-bound process we restart DCA when the number of the $0-1$ components of the binary variables (z_i) of the optimal solution to the current relaxed problem is sufficiently large, namely greater than or equal to $(n/2)$. Nevertheless, in some cases, the above criterion for restarting DCA may induce overusing of DCA inside the branch-and-bound scheme, and so increase the computational cost of the combined algorithm B&B-DCA. To avoid this inconvenience we limit the maximum number of restarting DCA (in our experiment this number is equal to 1000).

4.4 Computational results

In this paper, we consider the two sets of data. While the first one has the values $R = 0.1$, $a_i = 0.0$, and $b_i = 1.0$ in (4), the second one has $R = 0.05$, $a_i = 0.01$, and $b_i = 1.0$ $(i = 1, \ldots, n)$.

We have tested DCA and the classical branch-and-bound algorithm without DCA (denoted B&B) and with DCA (the combined B&B-DCA)) for different values of the cardinality parameter (*card*). The value of the penalty parameter t is 0.05 for all the experiments. The tolerance ε is equal to 10^{-7}.

The stopping criteria of the branch-and-bound algorithm is either the difference between the best upper bound and the best lower bound is smaller than ε or its number of iterations is greater than 40000. In the first case we say that the algorithm gives an ε-optimal solution.

In tables 1 and 2, we give the results for the two considered sets of data. In these tables, the number of iterations (iter.), the number of using DCA inside branch-and-bound algorithm (use), the computational time in seconds (CPU), and the objective value (Opt. val.) of the solution obtained by each of the algorithms are shown.

In the used data sets, the problems corresponding to $card = 1$ have no solution. The experiments in this paper correspond to the values of *card* in the interval $[20, 30]$. We note that, as will be seen later, with these values the classical branch-and-bound algorithm do not solve efficiently the corresponding problems.

Comments on the numerical results. From the numerical results we observe that

- DCA gives a very good approximation of the optimal solution within a very short time. More precisely, DCA and the combined B&B-DCA found the same optimal value with the precision, respectively, 10^{-6} and 10^{-4} in the first and the second data set. The number of iterations of DCA is equal to 4, 5, and 6 for 19, 2 and 1 problems, respectively. The running time is less than one second, except for one problem (1.062 second). Moreover, it is worth noting that all solutions given by DCA are integral in z, i.e. feasible to Problem (4).
- Due to the performance of DCA, the combined algorithm B&B-DCA is very efficient, and it is much better than the B&B algorithm: B&B-DCA works well (furnish an ε- optimal solution) to all 22 problems while B&B works only on 5 problems. For these five problems, the maximum ratio of the running time of the two algorithm is 87. The average of iterations and CPU of the B&B-DCA algorithm (for all 22 problems) is 341 and 145.84 seconds respectively. On the other hand, more the cardinality number is large, more important the influence of DCA on the combined algorithm B&B-DCA is.

Table 1 Numerical results for the first set of parameters

card	B&B			B&B-DCA				DCA		
	iter.	Opt. val.	CPU	iter.	use	Opt. val.	CPU	iter.	Opt. val.	CPU
20	40000	0.004395	10502.000	8	2	0.003633	4.594	4	0.003633	0.766
21	40000	0.004395	10977.282	1	1	0.003633	1.531	4	0.003633	0.766
22	5470	0.003633	1414.156	5056	116	0.003633	1369.578	4	0.003633	0.750
23	40000	0.004395	10366.765	1	1	0.003633	1.531	4	0.003633	0.750
24	40000	0.004395	10804.219	1	1	0.003633	1.672	4	0.003633	0.766
25	40000	0.005395	10415.219	18	2	0.003633	8.359	5	0.003633	0.922
26	40000	0.004395	10642.328	1	1	0.003633	1.562	5	0.003633	0.906
27	40000	0.004497	10617.468	1	1	0.003633	1.531	4	0.003633	0.891
28	40000	0.004395	10611.860	1	1	0.003633	1.703	4	0.003633	0.750
29	40000	0.004395	10621.031	1	1	0.003633	1.687	6	0.003633	1.062
30	40000	0.004395	10953.375	1	1	0.003633	1.687	4	0.003633	0.766

Table 2 Numerical results for the second set of parameters

card	B&B			B&B-DCA				DCA		
	iter.	Opt. val.	CPU	iter.	use	Opt. val.	CPU	iter.	Opt. val.	CPU
20	40000	0.000009	15549.063	356	171	0.000001	277.406	4	0.000097	0.907
21	40000	0.000013	15411.813	44	6	0.000001	28.969	4	0.000028	0.937
22	40000	0.000016	15151.391	1688	805	0.000001	1273.860	4	0.000028	1.000
23	40000	0.000014	15255.046	53	12	0.000001	39.563	4	0.000026	0.953
24	40000	0.000012	16003.610	31	5	0.000001	21.172	4	0.000022	0.953
25	40000	0.000015	14916.516	50	18	0.000001	42.875	4	0.000027	0.938
26	40000	0.000012	14628.437	42	4	0.000000	26.391	4	0.000025	0.953
27	1564	0.000000	619.875	47	11	0.000000	34.750	4	0.000019	0.953
28	2884	0.000000	1070.047	26	6	0.000000	19.500	4	0.000018	0.938
29	6625	0.000000	2434.547	46	4	0.000000	28.890	4	0.000020	0.937
30	2783	0.000000	1018.953	30	4	0.000000	20.594	4	0.000019	0.953

- The B&B algorithm do not solve efficiently these problems. In fact, in most the cases (17/22) the B&B algorithm is stopped when the number of iterations exceeds the limit 40000 while the objective value of the best known solution is not yet close to the optimal value given by the B&B-DCA algorithm. One reason may be that the dimension of these problems, especially the number of binary variables is large (550). It is so recommended to use the combined B&B-DCA algorithm for globally solving these problems.

5 Conclusion

In this paper, we have presented a new approach for solving the portfolio selection problem. An extension of the standard Markowitz mean-variance model including the cardinality and bounding constraints has been considered. These constraints make the corresponding portfolio selection problem nonconvex and very difficult to solve by existing algorithms. Replacing the Markowitz risk function by a semivariance and downside risk measure, we reformulated the problem as a DC program which can be solved efficiently using a deterministic approach based on DC programming and DCA. A combination of DCA and B&B has also been proposed to check globality of solutions computed by DCA and to improve the convergence of B&B. Numerical simulations show the globality of DCA, its inexpensiveness, and its superiority with respect to the standard branch-and-bound techniques. They also indicate the positive influence of DCA on B&B from the two points of view : CPU time and the quality of computed solutions.

References

1. M. Bartholomew-Biggs, Nonlinear Optimization with Financial Applications, 261 pp. Kluwer Academic Publishers, First edition, United States of America (2005).
2. Chang T.J., Meade N., Beasley J.E. and, Sharaiha Y.M., Heuristics for cardinality constrained portfolio optimization, Computers & Operations Research, Volume 27, pp. 1271-1302 (2000).

3. Fernandez A., Gomez S., Portfolio selection using neural networks, Computers & Operations Research, To appear.
4. J.E. Harrington, B.F. Hobbs, J.S. Pang, A. Liu, G. Roch, Collusive game solutions via optimisation, Math. Program. Ser. B, Volume 104, No. 1-2, pp. 407-435 (2005).
5. Hiriart-Urruty J-B. , Lemarechal C. , *Convex Analysis and Minimization Algorithms, Parts I&II*, Springer Verlag (1991)
6. Jobst N., Horniman M., Lucas C., Mitra G., Computational aspects of alternative portfolio selection models in the presence of discrete asset choice constraints, Quantitative Finance, Volume 1, pp. 1-13 (2001).
7. Le Thi, H.A., Contribution à l'optimisation non convexe et l'optimisation globale: Théorie, Algorithmes et Applications, Habilitation à Diriger des Recherches, Université de Rouen, (1997).
8. Le Thi H. A., Pham Dinh T., Le Dung M., *Exact penalty in DC programming*, Vietnam Journal of Mathematics, 27:2 (1999), pp. 169-178
9. Le Thi H.A. and Pham Dinh T., A continuous approach for globally solving linearly constrained quadratic zero-one programming problems, Optimization, Volume 50, No. 1-2, pp. 93-120 (2001).
10. Le Thi Hoai An and Pham Dinh Tao, *Large Scale Molecular Optimization from distances matrices by a DC optimization approach,* SIAM Journal of Optimization,Volume 14, Number 1, 2003, pp.77-116.
11. Le Thi H.A. and Pham Dinh T., The DC (difference of convex functions) Programming and DCA revisited with DC models of real world non convex optimization problems, Annals of Operations Research, Volume 133, pp. 23-46 (2005).
12. Le Thi H.A., Pham Dinh T., Huynh V.N., Exact Penalty Techniques in DC Programming, Research Report, LMI, National Institute for Applied Sciences - Rouen, France, (2005), submitted.
13. Markowitz, Harry M., Portfolio Selection, Journal of Finance, Volume 7, No.1, pp. 77-91 (1952).
14. Markowitz, Harry M., Portfolio Selection, John Wiley and Sons, New York, First Edition, (1959).
15. Nawrocki Q., A brief history of Downside Risk Measures, Technical Report, Villanova, Arcola, PA.
16. Pham Dinh T. and Le Thi H.A, Convex analysis approach to d.c. programming: Theory, Algorithms and Applications, Acta Mathematica Vietnamica, dedicated to Professor Hoang Tuy on the occasion of his 70th birthday, Volume 22,No. 1 pp. 289-355 (1997).
17. Pham Dinh T. and Le Thi H.A, DC optimization algorithms for solving the trust region subproblem, SIAM J. Optimization, Volume 8, pp. 476-505 (1998).
18. Rockafellar R.T., Convex Analysis, Princeton University Press, Princeton, First edition, (1970).
19. Roy, A. D., Safety First And The Holding Of Assets, Econometrica, 1952, Vol. 20(3), 431- 449.
20. Weber S., Schnörr C., Schüle Th., Hornegger J., Binary Tomography by Iterating Linear Programs, R. Klette, R. Kozera, L. Noakes and J. Weickert (Eds.), Computational Imaging and Vision - Geometric Properties from Incomplete Data, Kluwer Academic Publishers, (2005).

Optimization
Vol. 00, No. 00, Month-Month 200x, 1–26

DC Programming Approach for Portfolio Optimization under Step Increasing Transaction Costs

Hoai An LE THI[†], Mahdi MOEINI[†] and Tao PHAM DINH[‡,*],
†LITA, UFR-MIM, Université Paul Verlaine-Metz, Ile du Saulcy, 57045, Metz, France
‡Laboratory of Modelling, Optimization & Operations Research,
National Institute for Applied Sciences - Rouen, BP 08, Place Emile Blondel F 76131
Mont Saint Aignan Cedex, France.
(Received 00 Month 200x; In final form 00 Month 200x)

We address a class of particularly hard-to-solve portfolio optimization problems, namely the portfolio optimization under step increasing transaction costs. The step increasing functions are approximated, as closely as desired, by a difference of polyhedral convex functions. Then we apply the DCA (Difference of Convex functions Algorithm) to the resulting polyhedral DC program. For testing the efficiency of the DCA we compare it with CPLEX and the branch and bound algorithm proposed by Konno et al.

Keywords: Portfolio selection, Step increasing transaction cost function, DC programming, DCA, Branch and Bound.

Mathematics Subject Classification 2000: 90C26; 90C57; 90C90; 91B28

1 Introduction

One of the standard formulations of the portfolio optimization problem is to maximize the net return subject to the constraint of magnitude of risk. The net return is the expected rate of return of the portfolio subtracted by the transaction cost. The inclusion of transaction costs is an essential element of any realistic portfolio optimization, and the portfolio selection with the transaction costs has received a considerable research attention in recent years (see e.g. [1–7]).

If we ignore transaction costs or if the transaction cost is a linear function of the amount of the transaction, then the concerned portfolio optimization problem can be

*Corresponding author. Email: pham@insa-rouen.fr
Tel: 0033 235528335
Fax: 0033 235528332

Optimization ISSN 0233-1934 print/ ISSN 1029-4945 online ©200x Taylor & Francis Ltd
http://www.tandf.co.uk/journals
DOI: 10.1080/02331930xxxxxxxxxxxx

formulated as a linear or quadratic programming problem if we use Mean-Absolute Deviation (MAD) model [8] or Mean-Variance (MV) model [9] respectively. For both of the cases the corresponding problem can be solved without difficulty. On the other hand, if the transaction cost is not negligible and particularly if it is a nonconvex function then the considered portfolio optimization problem will be a difficult nonconvex programming problem.

The transaction cost function is usually nonconvex, typically concave or DC (Difference of Convex functions) ([2, 3, 7]), increasing piecewise linear concave or increasing piecewise constant (step increasing) ([4, 5]). The cases of concave or more generally DC transaction cost have been efficiently handled in several works ([2, 3, 7]) based on new techniques of global optimization, for example the robust methods for minimizing a concave objective function under linear constraints ([12, 13]). However the portfolio optimization with piecewise linear concave and/or step increasing transaction costs is still difficult([4]).

The traditional approach for solving these problems is reformulating them as 0-1 integer programming problems and then using branch and bound or branch and cut algorithms. But this approach fails when the number of assets is large and the number of pieces/steps is relatively large. For instance, if the number of linear pieces is seven and the number of assets is more than 1000, then we need more than 7000 binary variables, which is still out of the scope of the state-of-the-art integer programming software ([4]). Recently, branch and bound methods using new strategies of global optimization have been developed in [4] to portfolio optimization problems under piecewise linear concave transaction cost and step increasing transaction cost. The methods have been successfully applied to the former problems with at most seven linear pieces but for the later case the best results remain just for a small number of steps (one or two).

In this paper we focus on solving the portfolio optimization problems under step increasing cost with a larger number of steps, i.e. six or seven steps, for which the corresponding problem is difficult to solve by classical approaches. This type of transaction costs functions are used in internet transactions [4]. We investigate a local deterministic approach based on DC (Difference of Convex functions) programming and DCA (DC Algorithms) that were introduced by Pham Dinh Tao in their preliminary form in 1985. They have been afterwards extensively developed since 1994 by Le Thi Hoai An and Pham Dinh Tao (see e.g. [14–17] and references therein), and successfully applied to many (smooth or nonsmooth) nonconvex programs in various domains of applied sciences (see e.g. [15–20] and references therein). To our knowledge, DCA is among very few algorithms of nonconvex programming approach that can solve efficiently large-scale optimization problems. It is our main motivation for using DCA in this work.

In our approach based on DC programming and DCA, the step increasing transaction cost function will be approximated by the difference of polyhedral convex functions and then the DCA will be applied to the resulting DC program. For testing

the efficiency of the algorithm, we compare it with the CPLEX applied to an equivalent mixed 0-1 programming problem, and also with the branch and bound algorithm proposed by Konno et al. [4].

In this study, we used the single period Mean-Absolute Deviation (MAD) model proposed by Konno et al. [8] instead of the Mean-Variance (MV) model employed by Markowitz [9]. The reasons are twofold :

- It is well known that MAD model can be casted into a linear programming problem, which can be solved much faster than the corresponding MV model which is a quadratic programming problem. Linear programming problems have computational advantages over quadratic programming problems, especially when the program contains integer variables.
- According to W. Ogryczak and A. Ruszczynski ([10, 11]) the portfolios that rely on efficient frontier generated by MAD model are efficient in the sense of second degree stochastic dominance *regardless of the distribution* of the asset return. Note that, MV is, in general, not consistent with stochastic dominance rules.

The paper is organized as follows. In Section 2 we present the statement and the formulation of the portfolio optimization problem under step increasing transaction costs. Section 3 is devoted to DC programming and DCA for general DC programs. The DCA for solving portfolio optimization under step increasing transaction cost functions is described in Section 4. The branch and bound (B&B) algorithm proposed by Konno et al. and the combined B&B and DCA are outlined in Section 5. Computational results are reported in the last section.

2 Problem statements and formulation

In this paper we adopt the MAD model introduced firstly in [8] and extensively developed in a series of the articles ([2–4,8]), taking into account the step increasing transaction cost function.

2.1 *Mean-Absolute Deviation model*

First of all, let us recall the well known Mean-Absolute Deviation (MAD) model studied in [2–4, 8].

Let n be the number of available assets in the market, M be the total amount of investments, and let R_u be the random variable representing the rate of return of the u-th asset. Consider a portfolio $x = (x_1, x_2, \ldots, x_n)$ whose u-th component x_u represents the amount of investment into u-th asset. Assume that $(R_1, R_2, \ldots, R_n)$ is distributed over a finite set of points $(r_{1t}, r_{2t}, \ldots, r_{nt})$ with $t = 1, ..., T$ and that the probability

$$\wp_t = Pr\{(R_1, R_2, \ldots, R_n) = (r_{1t}, r_{2t}, \ldots, r_{nt})\} \quad (1)$$

is known in advance. Then the expected rate of return r_u of u-th asset is given by $r_u = \sum_{t=1}^{T} \wp_t r_{ut}$, and the expected rate of return of the portfolio x is $\sum_{u=1}^{n} r_u x_u$.

Denote by $R(x)$ the rate of return of the portfolio x (that is $\sum_{u=1}^{n} R_u x_u$). The absolute deviation $W[R(x)]$ of $R(x)$ is defined by

$$W[R(x)] = E[\mid R(x) - E[R(x)] \mid] = \sum_{t=1}^{T} \wp_t \mid \sum_{u=1}^{n} (r_{ut} - r_u)x_u \mid . \tag{2}$$

In what follows, we will assume for simplicity that $\wp_t = 1/T$ for all t. So $r_u = \sum_{t=1}^{T} r_{ut}/T$ and, consequently,

$$W[R(x)] = \sum_{t=1}^{T} \mid \sum_{u=1}^{n} (r_{ut} - r_u)x_u \mid /T. \tag{3}$$

Let w be a constant specifying the allowable level of risk. The Mean-Absolute Deviation (MAD) model is defined as follows

$$\max \left\{ \sum_{u=1}^{n} r_u x_u : W[\sum_{u=1}^{n} R_u x_u] \leq wM, \sum_{u=1}^{n} x_u = M, 0 \leq x_u \leq \beta_u, u = 1, \ldots, n \right\}, \tag{4}$$

where β_u is a constant specifying the upper bound of the amount of investment into the u-th asset.

The MAD model can be reformulated as a linear programming problem ([2]) by introducing a set of nonnegative variables $\tilde{\phi}_t$ and $\tilde{\psi}_t$ satisfying the conditions

$$\tilde{\psi}_t - \tilde{\phi}_t = \wp_t \sum_{u=1}^{n} (r_{ut} - r_u)x_u = (1/T) \sum_{u=1}^{n} (r_{ut} - r_u)x_u, \quad t = 1, \ldots, T$$

and

$$\tilde{\phi}_t \tilde{\psi}_t = 0, \tilde{\phi}_t \geq 0, \tilde{\psi}_t \geq 0, \quad t = 1, \ldots, T.$$

In fact, with the new variables the absolute deviation $W[R(x)]$ is represented as

$$W[R(x)] = \sum_{t=1}^{T} (\tilde{\phi}_t + \tilde{\psi}_t).$$

Hence the MAD model (4) becomes

$$\left\{\begin{array}{l}\max \sum\limits_{u=1}^{n} r_u x_u \\ s.t. \left\{\begin{array}{ll}\sum\limits_{t=1}^{T}(\tilde{\phi}_t+\tilde{\psi}_t)\le wM, & \\ \tilde{\psi}_t-\tilde{\phi}_t=(1/T)\sum\limits_{u=1}^{n}(r_{ut}-r_u)x_u & t=1,\ldots,T, \\ \sum\limits_{u=1}^{n} x_u = M, & \\ \tilde{\phi}_t\tilde{\psi}_t=0, & t=1,\ldots,T, \\ \tilde{\phi}_t\ge 0, & t=1,\ldots,T, \\ \tilde{\psi}_t\ge 0, & t=1,\ldots,T, \\ 0\le x_u\le \beta_u & u=1,\ldots,n. \end{array}\right. \end{array}\right. \tag{5}$$

By changing of variables, namely $\phi_t := T\tilde{\phi}_t$ and $\psi_t := T\tilde{\psi}_t$, the MAD model (without transaction cost) can be written in the form

$$\left\{\begin{array}{l}\max \sum\limits_{u=1}^{n} r_u x_u \\ s.t. \left\{\begin{array}{ll}\sum\limits_{t=1}^{T}(\phi_t+\psi_t)/T\le wM, & \\ \psi_t-\phi_t=\sum\limits_{u=1}^{n}(r_{ut}-r_u)x_u & t=1,\ldots,T, \\ \sum\limits_{u=1}^{n} x_u = M, & \\ \phi_t\psi_t=0, & t=1,\ldots,T, \\ \phi_t\ge 0, \psi_t\ge 0, & t=1,\ldots,T, \\ 0\le x_u\le \beta_u & u=1,\ldots,n. \end{array}\right. \end{array}\right. \tag{6}$$

According to [2] one can eliminate the complementary constraints, the proof can be found in [2], but for the sake of completeness we repeat the proof here. Let $(x_1^*,\ldots,x_n^*,\psi_1^*,\ldots,\psi_T^*,\phi_1^*,\ldots,\phi_T^*)$ be an optimal solution of the problem (6) without complementarity constraints. Let us define

$$\hat{\psi}_t = max(\psi_t^*-\phi_t^*,0), \hat{\phi}_t = -min(0,\psi_t^*-\phi_t^*), t=1,\ldots,T.$$

We have, for $t=1,\ldots,T$

$$\hat{\psi}_t,\hat{\phi}_t\ge 0, \hat{\psi}_t\hat{\phi}_t=0, \hat{\psi}_t-\hat{\phi}_t=\psi_t^*-\phi_t^* \text{ and } \hat{\psi}_t+\hat{\phi}_t\le\psi_t^*+\phi_t^*.$$

Hence $(x_1^*,\ldots,x_n^*,\hat{\psi}_1,\ldots,\hat{\psi}_T,\hat{\phi}_1,\ldots,\hat{\phi}_T)$ is a feasible solution of (6) at which the objective function, depending only on the variables $x_u, u = 1,\ldots,n$, takes the same value as at $(x_1^*,\ldots,x_n^*,\psi_1^*,\ldots,\psi_T^*,\phi_1^*,\ldots,\phi_T^*)$. Consequently, $(x_1^*,\ldots,x_n^*,\hat{\psi}_1,\ldots,\hat{\psi}_T,\hat{\phi}_1,\ldots,\hat{\phi}_T)$ is an optimal solution of (6) and the complementarity conditions $\phi_t\psi_t = 0, t = 1,\ldots,T$ can be removed from (6). Further, since the transaction costs functions appear on the objective function and they depend only on variables $x_u, u = 1,\ldots,n$ and not on the variables $\phi_t,\psi_t, t = 1,\ldots,T$, the proof remains true after adding the transaction costs functions into the model.

2.2 *Transaction costs*

In practice, every time an investor buys or sells some of his/her holdings, he/she must pay an extra amount that is called *transaction cost*. It can be a fixed cost but this case is not so common and in general the transaction costs are proportional to the volume of the purchases or sales. Transaction costs are usually relatively large when the amount of purchase is smaller and they continue to increase gradually with small rate, so transaction costs function becomes concave (see Figure 1(a) and Figure 2). The step increasing transaction costs function depicted in Figure 2 is very popular in e-trade system [21].

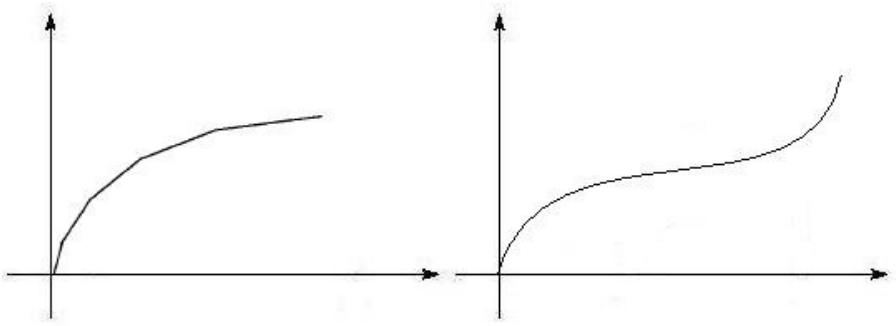


Figure 1. (a): Piecewise linear concave transaction costs function and (b): Cost subject to market impact

The shape of transaction costs function can be changed if we take the *market impact cost* into account. The market impact costs are the changes in the price of the assets due to large amount of transactions. If someone buys large quantities of an asset then its price will go up, similarly its price will go down if there are a lot of shares of this asset for sale. One of the typical forms of cost functions is depicted in (Figure 1(b)) which is a DC function. In this study we do not take the market impact into account.

There are other forms of transaction costs functions, for example they may have a lower limit i.e. minimum transaction costs or They can be nonsmooth functions. For more detailed study of different forms of the transaction costs functions we refer the reader to ([22] and [6] and references therein).

The transaction cost associated with a portfolio $x = (x_1, x_2, \ldots, x_n)$ is usually defined as the sum of individual transaction cost on each asset:

$$f(x) := \sum_{u=1}^{n} f_u(x_u),$$

where $f_u(x_u)$ is the individual cost on the u-th asset. Mathematically, the functions f_u, for $u = 1, \ldots, n$, are defined on $\mathbb{R}$ by

$$f_u(x_u) = \begin{cases} \gamma_u^0 \quad if \quad x_u \leq 0, & i = 0, \\ \gamma_u^i \quad if \quad v_u^{i-1} < x_u \leq v_u^i, & i = 1, \ldots, q(u)-1, \\ \gamma_u^{q(u)} \quad if \quad v_u^{q(u)-1} < x_u \leq +\infty, & i = q(u). \end{cases} \tag{7}$$

where $V_u = \{0 = v_u^0 < v_u^1 < \ldots < v_u^{q(u)} = \beta_u\}$ is a finite set of values representing the points of discontinuity of the function $f_u(x_u)$ and $0 = \gamma_u^0 < \gamma_u^1 < \ldots < \gamma_u^{q(u)}$. The individual transaction cost function is in fact $f_u(x_u)$ reduced on $[0, \beta_u]$ and has exactly $q(u)$ steps increasing (Figure 2).

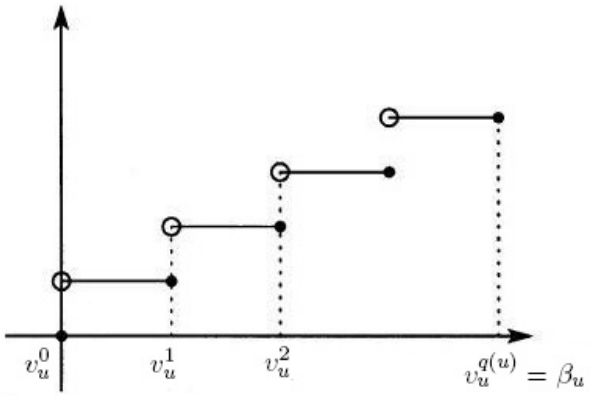


Figure 2. Step increasing transaction cost function

2.3 *Portfolio optimization problem under step increasing transaction cost*

For the portfolio $x = (x_1, x_2, \ldots, x_n)$ the net return is given by

$$\sum_{u=1}^{n} \{r_u x_u - f_u(x_u)\}, \tag{8}$$

where $f_u(x_u)$ is the step increasing function given in (7), and the portfolio optimization problem under step increasing transaction costs can be written in the form

$$\left\{ \begin{array}{l} \max \sum\limits_{u=1}^{n} \{r_u x_u - f_u(x_u)\} \\ s.t. \left\{ \begin{array}{ll} \sum\limits_{t=1}^{T} (\phi_t + \psi_t)/T \leq wM, & \\ \psi_t - \phi_t = \sum\limits_{u=1}^{n} (r_{ut} - r_u) x_u, & t = 1, \ldots, T, \\ \phi_t \geq 0, \psi_t \geq 0, & t = 1, \ldots, T, \\ \sum\limits_{u=1}^{n} x_u = M, & \\ 0 \leq x_u \leq \beta_u, & u = 1, \ldots, n. \end{array} \right. \end{array} \right. \tag{9}$$

Defining

$$\mathscr{D} := \left\{ \begin{array}{l} (x, \psi, \phi) \in \mathbb{R}^n \times \mathbb{R}^T \times \mathbb{R}^T : \\ \sum\limits_{u=1}^{n} (r_{ut} - r_u) x_u + \phi_t - \psi_t = 0, \qquad t = 1, \ldots, T, \\ \sum\limits_{t=1}^{T} (\psi_t + \phi_t)/T \leq wM, \sum\limits_{u=1}^{n} x_u = M, \\ 0 \leq x_u \leq \beta_u, u = 1, \ldots, n, \psi_t \geq 0, \phi_t \geq 0, t = 1, \ldots, T \end{array} \right\}, \tag{10}$$

we can express the problem (9) as

$$(P): \quad \min \left\{ f(x) := \sum_{u=1}^{n} f_u(x_u) - \sum_{u=1}^{n} r_u x_u : (x, \psi, \phi) \in \mathscr{D} \right\}.$$

This paper deals with solving problem (P). Our approach consists on approximating the step increasing transaction cost functions in (P) by the difference of polyhedral convex functions and then applying the DCA to the resulting DC program.

3 DC programming and DCA

DC programming and DCA constitute the backbone of smooth/nonsmooth nonconvex programming and global optimization. They address the problem of minimizing a function f which is the difference of two convex functions on the whole space $\mathbb{R}^n$ or on a convex set $C \subset \mathbb{R}^n$. Generally speaking, a DC program is an optimization problem of the form :

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

where g, h are lower semi-continuous proper convex functions on $\mathbb{R}^n$. Such a function f is called a DC function, and $g - h$ a DC decomposition of f while g and h are the DC components of f. The convex constraint $x \in C$ can be incorporated in the objective function of (P_{dc}) by using the indicator function on C denoted by χ_C which is defined by $\chi_C(x) = 0$ if $x \in C$, and $+\infty$ otherwise :

$$\inf\{f(x) := g(x) - h(x) : x \in C\ \} = \inf\{\chi_C(x) + g(x) - h(x) : x \in \mathbb{R}^n\}.$$

Let

$$g^*(y) := \sup\{\langle x, y\rangle - g(x) : x \in \mathbb{R}^n\}$$

be the conjugate function of a convex function g. The dual program of (P_{dc}) is given by

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

One can prove that ([16]) $\alpha = \alpha_D$, and there is the perfect symmetry between primal and dual DC programs: the dual to (D_{dc}) is exactly (P_{dc}). For a convex function θ, the subdifferential of θ at $x_0 \in \text{dom}\ \theta := \{x \in \mathbb{R}^n : \theta(x_0) < +\infty\}$, denoted by $\partial\theta(x_0)$, is defined by ([23])

$$\partial\theta(x_0) := \{y \in \mathbb{R}^n : \theta(x) \geq \theta(x_0) + \langle x - x_0, y\rangle, \forall x \in \mathbb{R}^n\}. \qquad (11)$$

The subdifferential $\partial\theta(x_0)$ generalizes the derivative in the sense that θ is differentiable at x_0 if and only if $\partial\theta(x_0) \equiv \{\bigtriangledown\theta(x_0)\}$.

If at least one of the convex DC components is polyhedral convex, we say that (P_{dc}) is a *polyhedral DC program*. In this case (D_{dc}) is also a polyhedral DC program. The special class of polyhedral DC programs, which plays a key role in nonconvex optimization, possesses worthy properties, from both theoretical and computational viewpoints, *as necessary and sufficient local optimality conditions, and finite convergence for DCA* (see, e.g., [14, 16, 17]).

DCA is based on the local optimality conditions of (P_{dc}), namely

$$\partial h(x^*) \cap \partial g(x^*) \neq \emptyset \tag{12}$$

(such a point x^* is called a *critical point* of $g-h$), and

$$\emptyset \neq \partial h(x^*) \subset \partial g(x^*). \tag{13}$$

Note that (13) is a necessary local optimality condition for (P_{dc}). For many classes of DC programs, it is also a sufficient optimality condition (see [15, 16]). As an example, we would like to mention the class of *polyhedral DC programs in which h is a polyhedral convex function.*

The idea of DCA is simple: each iteration k of DCA approximates the concave part $-h$ by its affine majorization (that corresponds to taking $y^k \in \partial h(x^k)$) and minimizes the resulting convex function (that is equivalent to determining a point $x^{k+1} \in \partial g^*(y^k)$).

DCA scheme

Initialization: Let $x^0 \in \mathbb{R}^n$ be an initial guess, $0 \leftarrow k$.

Repeat

- Calculate $y^k \in \partial h(x^k)$
- Calculate $x^{k+1} \in \arg\min\{g(x) - h(x^k) - \langle x - x^k, y^k\rangle : x \in \mathbb{R}^n\} \quad (P_k)$
- $k+1 \leftarrow k$

Until convergence of $\{x^k\}$.

Note that (P_k) is a convex optimization problem and in so far "easy" to solve.

Convergence properties of DCA and its theoretical basis can be found in [14–17]. It is worth mentioning that

- DCA is a descent method (*without linesearch*) : the sequences $\{g(x^k) - h(x^k)\}$ and $\{h^*(y^k) - g^*(y^k)\}$ are decreasing such that $g(x^{k+1}) - h(x^{k+1}) \leq h^*(y^k) - g^*(y^k) \leq g(x^k) - h(x^k)$.
- If $g(x^{k+1}) - h(x^{k+1}) = g(x^k) - h(x^k)$ then x^k is a critical point of $g-h$ and y^k is a critical point of $h^* - g^*$. In such a case, DCA terminates at k^{th} iteration.
- If the optimal value α of (P_{dc}) is finite and the infinite sequences $\{x^k\}$ and $\{y^k\}$ are bounded, then every limit point x^* (resp. y^*) of the sequence $\{x^k\}$ (resp. $\{y^k\}$) is a critical point of $g-h$ (resp. h^*-g^*), i.e., $\partial h(x^*) \cap \partial g(x^*) \neq \emptyset$ (resp. $\partial h^*(y^*) \cap \partial g^*(y^*) \neq \emptyset$). In particular, if h (resp. g^*) is polyhedral function and h (resp. g^*) is differentiable at x^* (resp. y^*) then x^* (resp. y^*) is a local minimizer of (P_{dc}) (resp. (D_{dc})).
- DCA has a *linear convergence* for DC programs. Especially, for polyhedral DC programs the sequences $\{x^k\}$ and $\{y^k\}$ contain finitely many elements and DCA has a finite convergence.

For a complete study of DC programming and DCA the reader is referred to [14–17] and the references therein. The solution of a nonconvex program (P_{dc}) by DCA must be composed of two stages: the search for an *appropriate* DC decomposition of f and that for a *good* initial point.

We shall apply *all these DC enhancement features* to solve the problem of portfolio optimization under step increasing transaction cost functions which is reformulated as a DC program.

4 Approximate polyhedral DC programs to (P) and solutions by DCA

Since the individual transaction cost functions $f_u(x_u)$ are discontinuous on $[0,\beta_u]$, the objective function $f(x)$ seems not to be a DC function on $X = \mathbb{R}^n$. There are some ways to approximate the function f by DC functions on X to build approximate DC programs. Due to the special structure of $f_u(x_u)$ and good DCA's behaviour in polyhedral DC programs [14, 16, 17], we have chosen *polyhedral DC functions* to approximate f.

4.1 *Approximate polyhedral DC functions to the step increasing cost function* f_u

Beside the discontinuity points $0 = v_u^0 < v_u^1 < \ldots, < v_u^{q(u)} = \beta_u$ we introduce the following ones

$$\tilde{v}_u^0, \tilde{v}_u^1, \ldots, \tilde{v}_u^{q(u)-1}$$

such that (see Figure 3)

$$0 = v_u^0 < \tilde{v}_u^0 < v_u^1 < \tilde{v}_u^1 < \ldots, < v_u^{q(u)-1} < \tilde{v}_u^{q(u)-1} < v_u^{q(u)} = \beta_u.$$

and $\max\{\tilde{v}_u^i - v_u^i : i = 0,\ldots,q(u)-1\} \leq \eta$, a positive number sufficiently small, and the associated (finite) polyhedral convex functions on $\mathbb{R}$, $L_u^i(x_u), i = 0,\ldots,q(u)$ (see Figure 4):

- L_u^i is constantly equal to γ_u^i on $]-\infty, v_u^i]$, affine on $[v_u^i, +\infty[$ such that $L_u^i(v_u^i) = \gamma_u^i$ and $L_u^i(\tilde{v}_u^i) = \gamma_u^{i+1}$(with slope c_u^i) for $i = 0,\ldots,q(u)-1$,
- $L_u^{q(u)}(x_u) = \gamma_u^{q(u)}$ for every $x_u \in \mathbb{R}$.

Consider now the piecewise linear function $F_u(x_u)$, for $u = 1,\ldots,n$, defined on $\mathbb{R}$ by

$$F_u(x_u) = \begin{cases} L_u^i(x_u) & \text{if} \quad v_u^{i-1} \leq x_u \leq v_u^i, \quad i = 1,\ldots,q(u)-1, \\ L_u^{q(u)}(x_u) & \text{if} \quad v_u^{q(u)-1} \leq x_u \end{cases} \tag{14}$$

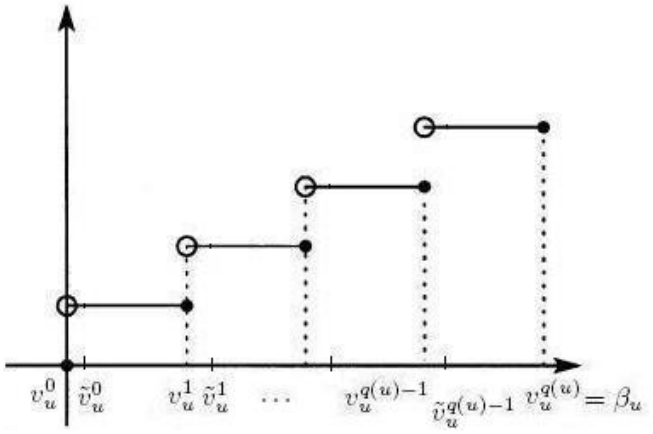


Figure 3. Introducing some auxiliary points beside the discontinuity points

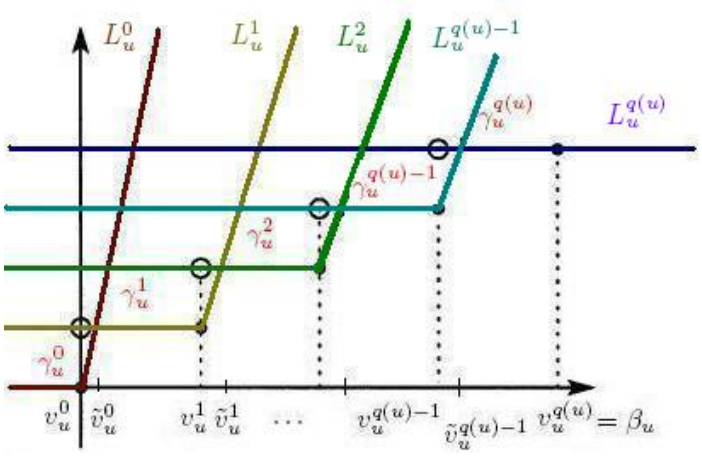


Figure 4. Defining polyhedral convex functions

which will be used to approximate the function $f_u(x_u)$.
By the very definition of F_u we can write:

$$F_u = \min\{L_u^i : i = 0, \ldots, q(u)\},$$

i.e. F_u is the pointwise infimum of the finite collection of polyhedral convex functions $L_u^i, i = 0, \ldots, q(u)$ (see Figure 5). Hence F_u is a polyhedral DC function with the following DC decomposition: $F_u = G_u - H_u$, where G_u and H_u are (finite) polyhedral

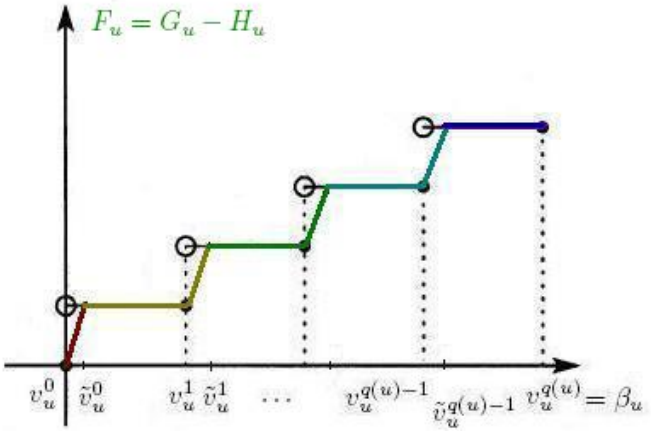


Figure 5. Piecewise linear function $F_u(x_u)$

convex functions on $\mathbb{R}$ defined by

$$G_u := \sum_{i=0}^{q(u)} L_u^i,\ H_u := \max_{i=0,\ldots,q(u)} \left(\sum_{j=0, j\neq i}^{q(u)} L_u^j \right). \tag{15}$$

In order to simplify computations in DCA for solving the approximate polyhedral DC program (P_{dc}) we will express each of the functions G_u and H_u as a pointwise supremum of a finite collection of affine functions. A simple calculation gives the following results:

$$G_u(x_u) = \begin{cases} \sum_{i=0}^{q(u)} \gamma_u^i, & if \quad x_u = v_u^0, \\ \sum_{j=0}^{i-1} c_u^j(x_u - v_u^j) + \sum_{i=0}^{q(u)} \gamma_u^i, & if \quad v_u^{i-1} \leq x_u \leq v_u^i, 1 \leq i \leq q(u)-1, \\ \sum_{j=0}^{q(u)-1} c_u^j(x_u - v_u^j) + \sum_{i=0}^{q(u)} \gamma_u^i, & if \quad v_u^{q(u)-1} \leq x_u \leq v_u^{q(u)}, \end{cases} \tag{16}$$

and,

$$H_u(x_u) = \begin{cases} \sum_{i=0}^{q(u)} \gamma_u^i, & if \quad x_u \leq \tilde{v}_u^0, \\ \sum_{j=0}^{i-1} c_u^j (x_u - \tilde{v}_u^j) + \sum_{i=0}^{q(u)} \gamma_u^i, & if \quad \tilde{v}_u^{i-1} \leq x_u \leq \tilde{v}_u^i, 1 \leq i \leq q(u)-1, \\ \sum_{j=0}^{q(u)-1} c_u^j (x_u - \tilde{v}_u^j) + \sum_{i=0}^{q(u)} \gamma_u^i, & if \quad \tilde{v}_u^{q(u)-1} \leq x_u \leq v_u^{q(u)}. \end{cases} \tag{17}$$

It implies that

$$G_u(x_u) = \max\{a_u^i x_u + b_u^i : i = 0, \ldots, q(u)\} \quad \forall x_u \in \mathbb{R}, \tag{18}$$

where

$$a_u^i = \begin{cases} 0 & if \quad i = 0, \\ \sum_{j=0}^{i-1} c_u^j & if \quad i = 1, \ldots, q(u), \end{cases} \tag{19}$$

and

$$b_u^i = \begin{cases} \sum_{j=0}^{q(u)} \gamma_u^j & if \quad i = 0, \\ \sum_{j=0}^{q(u)} \gamma_u^j - \sum_{j=0}^{i-1} c_u^j v_u^j & if \quad i = 1, \ldots, q(u). \end{cases} \tag{20}$$

Likewise

$$H_u(x_u) = \max\{\tilde{a}_u^i x_u + \tilde{b}_u^i : i = 0, \ldots, q(u)\}, \quad \forall x_u \in \mathbb{R}, \tag{21}$$

where

$$\tilde{a}_u^i = \begin{cases} 0 & if \quad i = 0, \\ \sum_{j=0}^{i-1} c_u^j & if \quad i = 1, \ldots, q(u). \end{cases} \tag{22}$$

So $a_u^i = \tilde{a}_u^i$, $\forall i = 0,\ldots,q(u)$, and

$$\tilde{b}_u^i = \begin{cases} \sum\limits_{j=0}^{q(u)} \gamma_u^j & if \quad i = 0, \\ \sum\limits_{j=0}^{q(u)} \gamma_u^j - \sum\limits_{j=0}^{i-1} c_u^j \tilde{v}_u^j & if \quad i = 1,\ldots,q(u). \end{cases} \tag{23}$$

4.2 *Approximate polyhedral DC functions to the objective function* $f(x)$ *of (P)*

The approximate polyhedral DC function $F_u(x_u)$ to the transaction cost functions $f_u(x_u)$ leads naturally to the following approximate function $F(x,\psi,\phi)$ to the objective function $f(x)$ of (P)

$$F(x,\psi,\phi) = \sum_{u=1}^{n} F_u(x_u) - \sum_{u=1}^{n} r_u x_u, \quad \forall (x_u,\psi_u,\phi_u) \in \mathbb{R}^{n+2T}.$$

In other words,

$$F = (G - \sum_{u=1}^{n} r_u x_u) - H,$$

where the functions G and H are (finite) polyhedral convex on $\mathbb{R}^{n+2T}$ such that, for all $x = (x,\psi,\phi) \in \mathbb{R}^{n+2T}$,

$$G(x,\psi,\phi) := \sum_{u=1}^{n} G_u(x_u), H(x,\psi,\phi) := \sum_{u=1}^{n} H_u(x_u).$$

It follows that F is a polyhedral DC function with the following DC components

$$(G(x,\psi,\phi) - \sum_{u=1}^{n} r_u x_u) \text{ and } \quad H(x,\psi,\phi),$$

and the resulting approximate polyhedral DC program to the problem (P) of portfolio optimization under step increasing transaction cost functions is of the form

$$\min \left\{ F(x,\psi,\phi) = \left(G(x,\psi,\phi) - \sum_{u=1}^{n} r_u x_u \right) - H(x,\psi,\phi) : (x,\psi,\phi) \in \mathscr{D} \right\}. \tag{24}$$

4.3 *DCA for solving the approximate polyhedral DC program* (24)

According to section (3) the DCA applied to (24) consists of constructing two sequences $\{(x^k, \psi^k, \phi^k)\}$ and $\{(y^k, \xi^k, \eta^k)\}$ such that, at each iteration $k = 0, 1, 2, \ldots$:

$$(x^k, \psi^k, \phi^k) \longrightarrow (y^k,\ \xi^k,\ \eta^k) \in \partial H(x^k, \psi^k, \phi^k)$$

$$\longrightarrow (x^{k+1}, \psi^{k+1}, \phi^{k+1}) \in \partial \left((G - \sum_{u=1}^{n} r_u x_u) + \chi_{\mathscr{D}} \right)^* (y^k,\ \xi^k,\ \eta^k)$$

($\chi_{\mathscr{D}}$ denotes, as indicated in section (3), the indicator function of the closed convex set $\mathscr{D}$ in $\mathbb{R}^{n+2T}$).

4.3.1 *Computing* $\partial H(x, \psi, \phi)$.

Since

$$H(x, \psi, \phi) := \sum_{u=1}^{n} H_u(x_u), \quad \forall (x, \psi, \phi) \in \mathbb{R}^{n+2T},$$

we have [24, 25]

$$\partial H(x, \psi, \phi) = \prod_{u=1}^{n} \partial H_u(x_u), \quad \forall (x, \psi, \phi) \in \mathbb{R}^{n+2T}. \tag{25}$$

Using the definition of $H_u(x_u)$ in (21) we have [24, 25]

$$\partial H_u(x_u) = co\{\tilde{a}_u^i : i = 0, \ldots, q(u), \tilde{a}_u^i x_u + \tilde{b}_u^i = H_u(x_u)\} \tag{26}$$

(*co* stands for the *convex hull*).

4.3.2 *Computing* $\partial \left((G - \sum_{u=1}^{n} r_u x_u) + \chi_{\mathscr{D}} \right)^* (y^k,\ \xi^k,\ \eta^k)$.

Unlike the computation of $\partial H(x, \psi, \phi)$ that is explicit, the computation of a subgradient $\partial \left((G - \sum_{u=1}^{n} r_u x_u) + \chi_{\mathscr{D}} \right)^* (y^k, \xi^k, \eta^k)$ amounts to minimizing the polyhedral convex function $(G(x, \psi, \phi) - \sum_{u=1}^{n} r_u x_u) - \langle (x, \psi, \phi), (y^k,\ \xi^k,\ \eta^k) \rangle$ over the compact convex

$\mathscr{D}$:

$$\min\left\{G(x,\psi,\phi)-\sum_{u=1}^{n}r_u x_u-\langle(x,\psi,\phi),(y^k,\ \xi^k,\ \eta^k)\rangle : (x,\psi,\phi)\in\mathscr{D}\right\},$$

or more precisely, by using the definition of G in (18)

$$\min_{(x,\psi,\phi)\in\mathscr{D}}\left\{\left(\sum_{u=1}^{n}\max_{i=0,\ldots,q(u)}\{a_u^i x_u+b_u^i\}-r_u x_u\right)-\langle(x,\psi,\phi),(y^k,\ \xi^k,\ \eta^k)\rangle\right\}.$$

Since H_u depends only on x_u, $\xi_t^k=0$, $\eta_t^k=0$, $\forall k,\forall t$, and the last problem becomes

$$\min\left\{\sum_{u=1}^{n}\max_{i=0,\ldots,q(u)}\{a_u^i x_u+b_u^i\}-\sum_{u=1}^{n}(r_u+y_u^k)x_u : (x,\psi,\phi)\in\mathscr{D}\right\}. \tag{27}$$

It is equivalent to the following linear program

$$(LP^{k+1})\quad\begin{cases}\min\left\{\sum\limits_{u=1}^{n}\tau_u-\sum\limits_{u=1}^{n}(r_u+y_u^k)x_u\right\}\\ s.t.\quad\begin{cases}a_u^i x_u+b_u^i\le\tau_u, & i=0,\ldots,q(u),u=1,\ldots,n,\\ (x,\psi,\phi)\in\mathscr{D},\\ \tau=(\tau_1,\ldots,\tau_n)\in\mathbb{R}^n\end{cases}\end{cases}$$

in the sense that:

- If (x^*,ψ^*,ϕ^*) is an optimal solution to (27), then $(x^*,\psi^*,\phi^*,\tau^*)$, with $\tau_u^*=\max_{i=0,\ldots,q(u)}\{a_u^i x_u+b_u^i\}$ for $u=1,\ldots,n$, is an optimal solution to $(LP)^{k+1}$.
- If $(x^*,\psi^*,\phi^*,\tau^*)$ is an optimal solution to $(LP)^{k+1}$ then $\tau_u^*=\max_{i=0,\ldots,q(u)}\{a_u^i x_u+b_u^i\}$ for $u=1,\ldots,n$ and (x^*,ψ^*,ϕ^*) is an optimal solution to (27).

4.4 *Description of DCA for solving the approximate problem* (24)

We are now in a position to summarize the DCA for solving (24).

DCA for solving (24)

(i) *Initialization :* Let (x^0,ψ^0,ϕ^0) be given and $k\ge 0$.

(ii) *Iterations :* Compute $(y^k,\ \xi^k,\ \eta^k)$ by using (25), (26), and compute $(x^{k+1},\psi^{k+1},\phi^{k+1})$ by solving $(LP)^{k+1}$.

(iii) *Stopping rule :* If $F(x^k,\psi^k,\phi^k)-F(x^{k+1},\psi^{k+1},\phi^{k+1})\le\varepsilon$, terminate. Otherwise increase k by 1 and return to step (ii).

According to the properties of convergence of DCA mentioned in Section 3, this algorithm converges after a finite number of iterations to a point (x^*, ψ^*, ϕ^*) that is a critical point of (24). This point is a local minimizer of (24) if $\partial H(x^*, \psi^*, \phi^*)$ is reduced to a singleton. Such situation occurs almost always (see the computation of ∂H in (25) and (26)).

4.5 *Finding an initial point of DCA*

We have started DCA with different choices of initial points (x^0, ψ^0, ϕ^0) :

(i) Let $(\bar{x}, \bar{\psi}, \bar{\phi})$ be the optimal solution of the relaxed problem of (P) which is obtained by convexifying the objective function (see [4]);

(ii) Set $(x^0, \psi^0, \phi^0) := (x^0, \bar{\psi}, \bar{\phi})$, where x_u^0 is chosen among the set of discontinuity points $\left\{v_u^0, v_u^1, \ldots, v_u^{q(u)}\right\}$ by: for each $u = 1, \ldots, n$
- Find $i \in \{0, \ldots, q(u)\}$ such that $\bar{x}_u \in [v_u^i, v_u^{i+1}]$;
- If $\bar{x}_u \geq (v_u^i + v_u^{i+1})/2$ set $x_u^0 = v_u^{i+1}$ else $x_u^0 = v_u^i$.

(iii) (x^0, ψ^0, ϕ^0) is the zero vector.

In the last two procedures, the initial solution may be infeasible, but after one iteration we obtain a feasible one. In all tests, the procedure (ii) gives the best results; so we have chosen it to compute initial points for DCA.

5 Global methods for solving (P)

For evaluating the quality of the solutions computed by DCA, we solve the problem (P) by a branch and bound algorithm (see [4]). More precisely, consider the step increasing costs function defined in the interval $[\underline{a}, \overline{b}]$. This function is underestimated by its convex envelope over $[\underline{a}, \overline{b}]$, which can be obtained by connecting two or three end points (Figure 6). Subdivision is made on the interval for which the deviation between the step increasing costs function and the under estimator is the maximum (Figure 7). The upper bound is updated when a better solution is discovered. The algorithm iterates until an ε-optimal solution is found or the infeasibility is proved. (For more details on the description of the algorithm see [4]).

Furthermore, in order to globally solving (P) we develop a combination of DCA and the branch and bound (B&B) algorithm (denoted BB-DCA). The combined procedure aims at checking (resp. computing) the globality of the solution obtained by DCA (resp. better solutions to restart DCA). By this way, DCA could improve upper bounds and the convergence of the B&B scheme. More precisely, we start the B&B scheme by applying DCA to get the best current solution, and at each iteration of B&B, if the optimal solution $(\bar{x}, \bar{\psi}, \bar{\phi})$of the relaxed problem is better than the best current solution, then we restart DCA from (x^0, ψ^0, ϕ^0) given by procedure (i) or (ii)

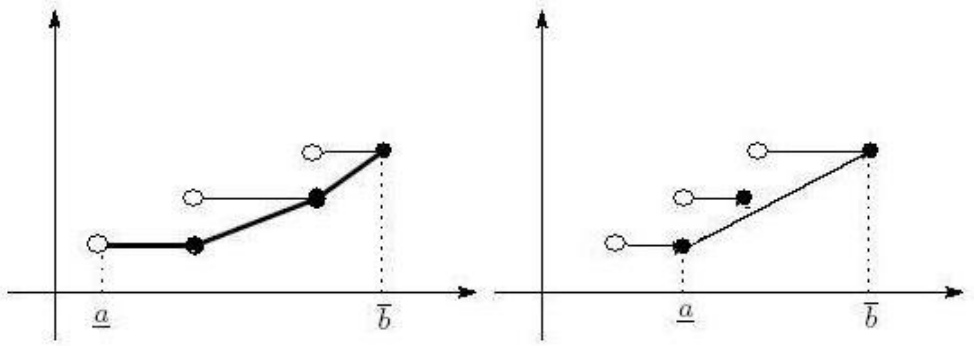


Figure 6. Convex envelope of step increasing transaction costs function

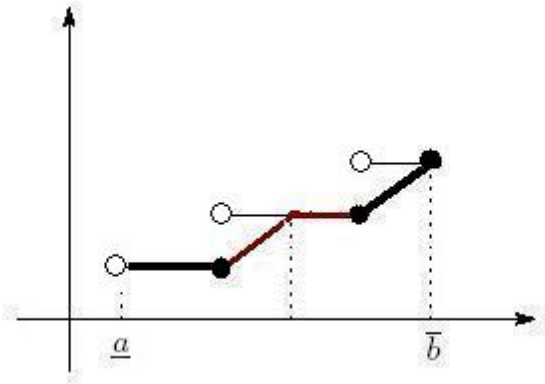


Figure 7. Subdivision of a chosen interval

in Subsection 4.5.

6 Computational experiments and conclusions

The DCA described in Section 4 has been implemented and tested on three data sets.

Table 1 shows step increasing cost table associated with a typical transaction. On each interval, the amount of the transaction cost is about 1.0% of the maximum possible value of the investment, i.e. if $x_u \in (0.0, 2.0]$ then the transaction cost is 0.02. As it is shown in Table 1, we have considered six steps for the transaction costs functions.

In all of the experiments we supposed that $w = 0.05$. For the first data set, the upper bound β_u is equal to 14.0 for $u = 1, \ldots, n$, and the number of steps is equal to

seven. For the two other data sets, we have considered six steps and $\beta_u = 12$. The experiments were done for different values of M. By testing different values for M, we are interested in the behavior of the problem from computational view point. By increasing the value of M we expect to have more difficult problems. There is an upper bound on accepted values of investments in each asset so the amount of the investment must be distributed among different assets in the best possible way, we think this can make problem more difficult to be solved, at least for some relatively large values of M.

The tolerance ε for DCA is equal to 10^{-6}. In all of our test problems the points $\tilde{v}_u^i : \quad i = 0, \ldots, q(u) - 1, u = 1, \ldots, n$ are defined as follows

$$\tilde{v}_u^i := v_u^i + 0.0001, \quad i = 0, \ldots, q(u) - 1.$$

Also, we solved the equivalent mixed 0-1 linear programming problem of (P) (see [5]) by using CPLEX version 9.1. CPLEX uses a branch & cut algorithm. In this algorithm, CPLEX solves a series of continuous subproblems. To manage those subproblems efficiently, CPLEX builds a tree in which each subproblem is a node. The root of the tree is the continuous relaxation of the original mixed 0-1 linear programming problem.

We considered $\varepsilon := 10^{-3}$ as the gap between the best mixed integer feasible solution and the lower bound in CPLEX . We have considered a time limit for CPLEX. The time limit is 1200 seconds for the first data set and 1000 seconds for the others.

The algorithms are coded in C++ and run on a Pentium 3.000 GHz of 1.0 DDRAM.

To solve the resulted linear programs, we used the software CPLEX version 9.1.

The first data set corresponds to weekly prices from March 1992 during 289 weeks. These prices come from the index *S&P* 500. The number n of different assets is 457. The set of prices is publicly available at "http://people.brunel.ac.uk/˜mastjjb/jeb/orlib/indtrackinfo.html". Here, n is 457 and T is 289.

For this set of data we solved the problem by DCA, B&B algorithm, and CPLEX. The results are presented in Table 2 which indicates the number of explored nodes by CPLEX (nodes), the number of iterations of branch and bound and DCA (iter), CPU time in seconds (CPU), and the ε-optimal value (Opt. val.) for each of the methods (i.e. branch and bound (B&B) and DCA) and the software CPLEX. They show that the standard software CPLEX cannot solve the problem efficiently.

The stopping criterion of the B&B algorithm is either the difference of the best upper bound and the best lower bound (gap) is smaller than $\varepsilon := 10^{-3}$ or its number of iterations is greater than 900. We have chosen 900 as the limit for the number of iterations because for this data set the B&B needs almost one second and a half for each iteration and so, with the limit of 900 iterations, the maximum time needed corresponds, approximately, to the that of CPLEX.

The second data set is generated by using weekly stock prices of 500 healthcare

companies over 20 weeks (from 27 December 2005 to 15 May 2006). The stock prices are collected from *YAHOO FINANCE page* (http://finance.yahoo.com). The returns, r_{ut}, have been calculated using the formula:

$$r_{ut} = (p_{u,t+1} - p_{u,t})/p_{u,t},$$

where $p_{u,t}$ is the closing price of the u-th asset at the t-th week ($u = 1,\ldots,n$ and $t = 1,\ldots,T$). The number n of different assets is equal to 500 and T is 20. In the Table 3 the number of explored nodes by CPLEX (nodes), the number of iterations for branch and bound and DCA (iter.), CPU time in second (CPU), and the ε-optimal value (Opt. val.) for each of the algorithms as well as for CPLEX are shown. In this table the optimal values were multiplied by (-1) (so they correspond to the *maximization* form of the problem (P).)
The stopping criteria of the branch and bound algorithm is either the difference of the best upper bound and the best lower bound (gap) is smaller than $\varepsilon := 10^{-3}$ or its number of iterations is greater than 5000.

We have also used a third data set that corresponds to weekly prices from March 1992 during 30 weeks. These prices come from the index *Russell* 3000. The number n of different assets is 2151. As a similar manner of the first data set, the prices are publicly available at "http://people.brunel.ac.uk/~mastjjb/jeb/orlib/indtrackinfo.html". Here, n is 2151 and T is 30.

For this set of data we solved the problem by DCA, B&B algorithm, the combined B&B-DCA and CPLEX. The results are presented in Table 4 which indicates the number of explored nodes by CPLEX (nodes), the number of iterations for branch and bound and DCA (iter.), the number of restarting DCA in B&B-DCA (rest), CPU time in seconds (CPU), and the ε-optimal value (Opt. val.) for different values of M. Similarly, as in the case of the previous data set, the optimal values, presented in the table, were multiplied by (-1). The stopping criterion of the B&B algorithm is either the difference of the best upper bound and the best lower bound (gap) is smaller than $\varepsilon := 10^{-3}$ or its number of iterations is greater than 1000. We have chosen 1000 as the limit for the number of iterations because for this data set the B&B needs almost one second for each iteration and so, with the limit of 1000 iterations, the maximum time needed corresponds to the time limit of CPLEX.

Finally, we solved the model (P) without taking into account the transaction costs functions. We used CPLEX to solve the linear programming problems resulting from eliminating transaction costs functions from problem (P). Computational experiments have been done with the same three data sets and the same conditions as the cases we use the transaction costs functions.

6.1 *Comments on the numerical results*

From the numerical results we observe that

- DCA gives a very good approximation of the optimal solution within a short time. The number of iterations of DCA is equal to 2 or 3 for all the problems. The running time is less than one second for the first set of data and, larger the value of M is, the more expensive CPLEX is. However DCA seems to be insensitive to values of M. As for the B&B, it has difficulties in solving the problem, even for small values of M. DCA obtains the same optimal values as the B&B for $M = 90$ and $M = 170$ in, respectively, 2 and 3 iterations.
- For the second set of data, since the number of assets is large (2151) the problem is more difficult. For all values of M CPLEX needs between 70.890 seconds and 1002.800 seconds to solve the problem, while DCA gives a very good approximation of the optimal solution in less than 22.297 seconds for all values of M. CPLEX did not find an optimal solution within the allowed time limit (i.e. 1000 seconds) for more than half of the cases. Since the dimension of the problem is large, B&B needs more time to solve the problem. For $M = 240$, the optimal solution provided by DCA is the same as the solution obtained by the B&B after 1000 iterations.
- As we can observe, polyhedral approximation of the step increasing cost function and polyhedral decomposition of the approximated objective function play a crucial role in the efficiency (the rate of convergence and the quality of solutions) of DCA. The differences between the numbers of iterations of the three algorithms DCA, B&B and CPLEX are very significant. In all tests DCA needs at most 3 iterations to reach very good upper bounds (resp. lower bounds) for the minimization (resp. maximization) problem. The performance of DCA diminishes its implication for the efficiency of B&B-DCA with respect to B&B: in Table 3, BB-DCA terminated with global solutions for $M = 40$, 200 for which B&B could not find the global solution within the allowed number of iterations and the former returned better solutions than the latter in more than half the cases. Note also the difference of computational times between the two algorithms is not significant.

6.2 *Influence of the transaction costs functions*

After eliminating the transaction costs functions from the model (P), the resulting model becomes so easy to solve. CPLEX solves it in less than one second for all different values of M and for the three data sets. One of the reasons comes back to the linear character of the model. By comparing the CPU time for solving these problem with and without step increasing transaction costs functions, we observe an important influence of these functions on the computational complexity of the nonconvex program (P).

On the other hand, the transaction costs functions modify both solutions and optimal values. For the first data set, there is a great difference between the results obtained with and without transaction costs functions. Figure 8 shows the number of assets with positive weights with (*with TC*) and without (*without TC*) transaction costs functions. For the second data set, the results are not very different. For more

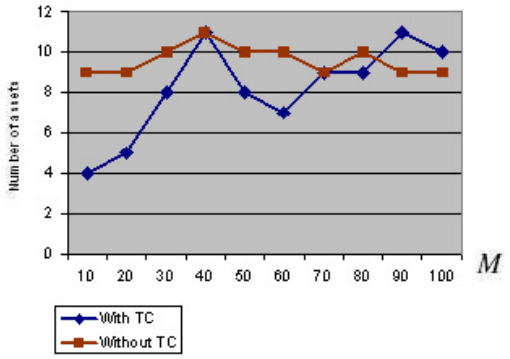


Figure 8. Number of assets for the first data set with and without taking transaction costs into account

than half of the cases, the locations differ after adding the transaction costs, for the others only the weights change. The changes in locations are more important for small values of investment, i.e. small values of M. For large values of M, the number of assets included in the portfolio does not change considerably but some of the asset are replaced by some other ones. Figure 9 shows the number of the assets included in the optimal portfolios for the second and the third data sets and for different values of M. The figure presents only the results for which optimal solution vector contains different assets. The results in presence of transaction costs functions are denoted by (*with TC*) versus (*without TC*) which indicates that the transaction costs are ignored. Regarding the influence of the transaction costs functions on the third data set, we observe that for all values of $M \neq 190$, the locations are different with and without transaction costs. For $M = 190$, only the weights in the optimal portfolio change.

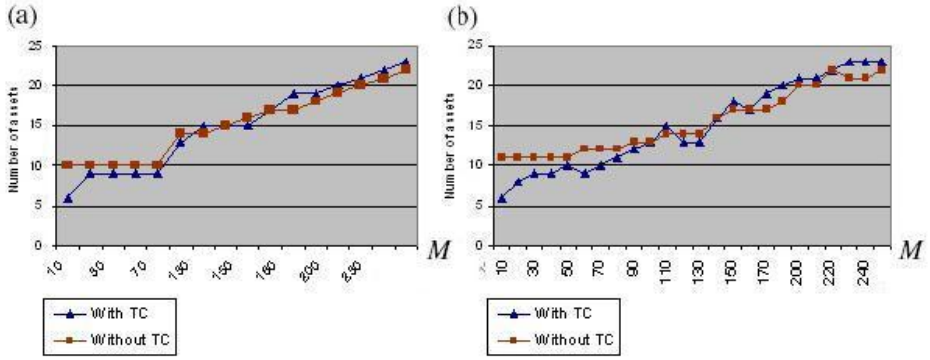


Figure 9. (a): Number of assets for the second data set and (b): Number of assets for the third data set with and without taking transaction costs into account

As the figures show there is no huge decline on the number of different assets that are present in the optimal portfolio, even for relatively small investments, because the

transaction costs increase gradually as the investments increase. Hence, eliminating some of the assets and investing their corresponding weights in the remaining assets will not necessarily lead to reduction in the overall transaction costs. If we impose a larger minimum transaction costs [22] then it may make more reduction in the number of assets.

6.3 *Conclusion*

In this paper we have presented a new approach based on DC programming and DCA to solve the portfolio selection problem under step increasing transaction cost functions. Preliminary computational experiments (Tables 2, 3, and 4) have been carried on a series of test problems with structure similar to those encountered in practice. The numerical results are promising. As we can observe, DCA is quite simple and inexpensive, especially for approximate polyhedral DC programs for which it has finite convergence. The combined BB-DCA allows to check the good quality of solutions computed by DCA. Our approach might then be applied to large-scale portfolio optimization problems.

References

[1] Davis M. H. A. and Norman A. R., 1990, Portfolio selection with transaction costs, *Mathematics of Operations Research,* **15**, No. 4, 676–713.
[2] Konno, H. and Wijayanayake, A., 2001, Portfolio optimization problem under concave transaction costs and minimal transaction unit constraints, *Mathematical Programming Ser. B,* **89**, 233–250.
[3] Konno, H. and Wijayanayake, A., 2002, Portfolio optimization under DC transaction costs and minimal transaction unit constraints, *Journal of Global Optimization,* **22**, 137–154.
[4] Konno, H. and Yamamoto, R., 2005, Global optimization versus integer programming in portfolio optimization under nonconvex transaction costs, *Journal of Global Optimization,* **32**, 207–219.
[5] Konno, H. and Yamamoto, R., 2005, Integer programming approaches in mean-risk models, *Computational Management Science,* **2**, 339–351.
[6] Lobo M.S., Fazel M., Boyd S., 2007, Portfolio optimization with linear and fixed transaction costs, *Annals of Operations Research,* **152**(1), 341–365.
[7] Xue H.G., Xu C.X., Feng Z.X., 2006, Mean–variance portfolio optimal problem under concave transaction cost, *Applied Mathematics and Computation,* **174**, 1–12.
[8] Konno, H. and Yamazaki, H., 1991, Mean-absolute deviation portfolio optimization model and its applications to Tokyo stock market, *Management Science,* **37**, 519–531.
[9] Markowitz, H., 1959, *Portfolio Selection; Efficient Diversification of Investment,* (First edition) (New York: John Wiley and Sons).
[10] Ogryczak, W. and Ruszczynski, A., 1999, From stochastic dominance to mean-risk model: Semideviation as the risk measures, *European Journal of Operational Research,* **116**, 33–50.
[11] Ogryczak, W. and Ruszczynski, A., 2001, On consistency of stochastic dominance and mean- semideviation models, *Mathematical Programming,* **89**, 217–232.
[12] Phong T.Q., Le Thi H.A., Pham Dinh T., 1995, Decomposition branch and bound method for globally solving linearly constrained indefinite quadratic minimization problems, *Operations Research Letters,* **17**, 215–220.
[13] Tuy H., 1998, *Convex Analysis and Global Optimization,* (Dortrecht: Kluwer Academic Publishers).
[14] Le Thi, H.A., 1997, Contribution à l'optimisation non convexe et l'optimisation globale: Théorie, Algorithmes et Applications, Habilitation à Diriger des Recherches, Université de Rouen.
[15] Le Thi H.A., Pham Dinh T., 2005, The DC (difference of convex functions) Programming and DCA revisited with DC models of real world non convex optimization problems, *Annals of Operations Research,* **133**, 23–46.
[16] Pham Dinh T., Le Thi H.A, 1997, Convex analysis approach to d.c. programming: Theory, Algorithms and Applications, *Acta Mathematica Vietnamica,* dedicated to Professor Hoang Tuy on the occasion of his 70th birthday, **22**(1), 289–355.

[17] Pham Dinh T., Le Thi H.A, 1998, DC optimization algorithms for solving the trust region subproblem, *SIAM J. Optimization,* **8**, 476–505.
[18] Harrington, J.E., Hobbs, B.F., Pang, J.S., Liu, A., Roch, G., 2005, Collusive game solutions via optimisation, *Mathematical Programming Ser. B,* **104**(1-2), 407–435.
[19] Schüle T., Schnörr C., Weber S., and Hornegger J., 2005, Discrete tomography by convex-concave regularization and d.c. programming. *Discrete Applied Mathematics*, **151**, 229-243.
[20] Schnörr C., Schüle T., Weber S., 2007, Variational reconstruction with DC-programming. In: G.T. Herman and A. Kuba (Eds) *Advances in Discrete Tomography and Its Applications,* (Birkhäuser, Boston).
[21] Konno, H., 2005, Applications of Global Optimization to Portfolio Analysis. In: C. Audet, P. Hansen and G. Savard (Eds) *Essays and Surveys in Global Optimization* (First edn) (USA: Springer), pp. 195–210.
[22] Maringer, D., 2005, Portfolio Management With Heuristic Optimization, Springer, Dordrecht.
[23] Rockafellar R.T., 1970, *Convex Analysis,* (First edition), (Princeton: Princeton University Press).
[24] Hiriart Urruty J.B., Lemarechal C., 1993, Convex Analysis and Minimization Algorithms, Springer, Berlin.
[25] Laurent P.J., 1995, Approximation et Optimisation, Kluwer Academic Publishers, Dordrecht.

Table 1. Transaction costs

$(v_u^i, v_u^{i+1}]$:	0.0	(0.0,2.0]	(2.0,4.0]	(4.0,6.0]	(6.0,8.0]	(8.0,10.0]	(10.0,12.0]	(12.0,14.0]
γ_u^i :	0.0	0.02	0.04	0.06	0.08	0.10	0.12	0.14

Table 2. Numerical results for the first data set

	CPLEX			B&B			DCA		
M	nodes	Opt. val.	CPU	iter.	Opt. val.	CPU	iter.	Opt. val.	CPU
50	43300	0.017818	1208.840	88	0.016910	113.250	3	0.116416	3.297
60	41501	0.035766	1201.810	187	0.035137	244.734	2	0.055518	2.813
70	37301	0.055426	1200.940	900	0.058425	1139.719	2	0.074998	2.203
80	38301	0.074550	1201.300	31	0.074055	39.328	3	0.093639	3.422
90	37000	0.096650	1213.220	339	0.095012	433.468	3	0.154088	3.422
100	36501	0.116044	1207.390	900	0.135331	1151.859	2	0.155331	1.953

Table 3. Numerical results for the second data set

	CPLEX			B&B			DCA		
M	nodes	Opt. val.	CPU	iter.	Opt. val.	CPU	iter.	Opt. val.	CPU
10	146	0.723192	2.670	1370	0.721726	273.375	3	0.691447	0.750
20	510	1.509187	5.550	1252	1.509182	253.969	3	1.462120	0.609
30	1024	2.282747	17.240	5000	2.281142	990.844	3	2.239324	0.625
40	913	3.061181	14.480	1216	3.061173	247.828	3	2.979999	0.594
50	847	3.843308	15.170	3871	3.843130	756.719	3	3.830677	0.968
60	1483	4.610707	21.880	5000	4.610616	970.922	3	4.566408	0.593
70	869	5.367585	11.590	5000	5.367490	938.437	3	5.334487	0.594
80	338	6.092228	9.140	1589	6.090536	308.531	2	6.062314	0.453
90	201	6.806528	4.480	409	6.806510	79.250	2	6.806510	0.328
100	669	7.483239	18.630	5000	7.481416	841.282	3	7.478477	0.609
110	98	8.154372	4.310	467	8.152946	89.812	2	8.095418	0.344
120	901	8.785032	21.880	2139	8.784860	413.484	2	8.761774	0.360
130	541	9.403340	10.890	506	9.403320	99.047	3	9.363180	0.593
140	828	9.976866	17.580	4543	9.976844	871.765	3	9.964637	0.562
150	674	10.534156	15.950	1713	10.534136	333.281	2	10.526964	0.344
160	661	11.056510	16.860	976	11.056491	189.594	2	11.051380	0.344
170	1643	11.548996	36.020	2329	11.551624	440.672	3	11.551624	0.640
180	3362	12.000752	65.300	1079	12.000734	211.656	2	11.991416	0.329
190	27146	12.411932	497.050	2690	12.411772	504.266	3	12.393400	0.594
200	47483	12.820010	841.910	874	12.819989	166.843	2	12.813254	0.343
210	51601	13.219184	1003.200	2042	13.219160	388.860	2	13.205253	0.344
220	49801	13.599266	1000.030	635	13.599238	121.391	2	13.597422	0.343
230	48141	13.968100	1000.437	859	13.969617	164.140	3	13.939492	0.579
240	46301	14.332768	1000.594	2927	14.336082	565.375	3	14.297366	0.578
250	46901	14.701980	1000.469	1881	14.701948	361.203	3	14.674778	0.594

Table 4. Numerical results for the third data set (Russell 3000)

		CPLEX			B&B				BB-DCA			DCA	
M	nodes	Opt. val.	CPU	iter.	Opt. val.	CPU	iter.	rest	Opt. val.	CPU	iter.	Opt. val.	CPU
10	1673	0.484134	70.890	1000	0.472324	1098.047	1000	13	0.472324	1181.563	3	0.444994	4.891
20	2902	1.045465	153.090	1000	1.044500	1108.750	1000	15	1.044500	1243.329	3	1.012359	16.359
30	1899	1.599846	210.440	1000	1.599750	1103.594	1000	20	1.599844	1263.985	3	1.486622	18.406
40	2939	2.152991	377.310	1000	2.149407	1109.656	748	11	2.152989	852.547	2	2.053807	11.875
50	7541	2.679210	1001.500	1000	2.672287	1111.171	1000	7	2.672287	1104.109	3	2.668101	7.672
60	7573	3.234320	933.330	1000	3.217419	1104.453	1000	10	3.229966	1183.828	3	3.144015	22.297
70	4851	3.763378	1001.030	1000	3.739001	1104.859	1000	5	3.739001	1142.141	3	3.700731	14.859
80	4301	4.286734	1002.800	1000	4.284365	1170.062	1000	8	4.284365	1140.172	3	4.228966	14.000
90	5111	4.822331	1001.770	1000	4.812887	1110.359	1000	13	4.813162	1204.875	3	4.672412	14.375
100	4001	5.351168	1002.300	1000	5.349740	1113.265	1000	6	5.351164	1114.219	2	5.313198	2.625
110	5741	5.858839	1000.050	1000	5.858831	1117.219	1000	20	5.858831	1185.531	3	5.706908	19.359
120	5441	6.331496	1001.300	1000	6.330285	1106.265	1000	5	6.331485	1096.328	3	6.245026	13.078
130	3337	6.782229	328.360	546	6.782217	623.062	546	11	6.782217	602.500	3	6.780472	3.890
140	4001	7.180542	1001.312	1000	7.176099	1098.563	1000	7	7.179158	1106.563	3	7.125709	4.094
150	3601	7.576792	1000.969	1000	7.571015	1101.610	1000	19	7.571015	1137.781	3	7.483173	15.688
160	4601	8.968668	1000.938	1000	7.967381	1090.890	1000	8	7.967381	1082.860	2	7.959605	2.078
170	2651	8.360978	1000.937	1000	8.352708	1084.703	1000	18	8.353377	1258.031	3	8.290695	18.016
180	1871	8.731364	1001.109	1000	8.730721	1086.047	1000	30	8.730808	1120.312	2	8.700640	8.000
190	4261	9.094636	1001.578	1000	9.085002	1087.828	1000	10	9.085002	1105.375	3	9.063879	14.578
200	2001	9.441358	1001.031	1000	9.431421	1072.047	346	4	9.450112	389.562	3	9.373741	3.812
210	2311	9.786790	1001.000	1000	9.778142	1083.344	1000	17	9.778174	1159.328	3	9.766231	12.234
220	2061	10.121934	1001.078	1000	10.114978	1092.594	1000	4	10.121141	1075.891	3	10.088907	4.906
230	3101	10.447622	1001.140	823	10.457530	889.812	823	4	10.457530	891.625	3	10.431347	4.843
240	1891	10.772840	1001.125	1000	10.762068	1083.859	1000	10	10.762720	1082.375	2	10.762068	1.984
250	1761	11.087680	1001.015	273	11.102833	295.375	273	3	11.102833	298.500	3	11.078073	4.828

Submitted to *Operations Research*
manuscript (Please, provide the mansucript number!)

Optimization of a Long-Short Portfolio Under Threshold Constraints Using DC Programming and DCA

Hoai An LE THI
Laboratory of Theoretical and Applied Computer Science, Paul Verlaine Metz University,
Ile du Saulcy, 57045, Metz, France, lethi@univ-metz.fr

Mahdi MOEINI
Laboratory of Theoretical and Applied Computer Science, Paul Verlaine Metz University,
Ile du Saulcy, 57045, Metz, France, moeini@univ-metz.fr

For Peer Review

In matter of Portfolio selection, we consider a generalization of the Markowitz Mean-Variance model which includes threshold constraints and one can sell assets short if it leads to a better risk-return structure of the portfolio. Threshold constraints limit the amount of capital to be invested in (or sold short from) each asset and prevent very small investments in (or short selling from) any asset. The resulting problem is no longer a convex quadratic problem as the Markowitz model, since it contains some non convex constraints. After changing of variables, we formulate the problem as a mixed 0-1 quadratic programming problem with linear and complementarity constraints. Then, using an exact penalty result, we reformulate the last problem in terms of a DC (Difference of Convex functions) program. A so-called DC program is that of minimizing a DC function over a convex set. We then develop DC programming approach and DCA (DC Algorithms) to solve this new model. Some preliminary comparative results of DCA and a branch and bound algorithm are presented which show that DCA is an efficient and promising approach for this problem. Finally, a combination of DCA and the branch and bound algorithm is proposed for globally solving the problem.

Key words: Portfolio selection; DC programming; DCA; branch and bound.

1. Introduction

In the portfolio selection problem, given a set of available securities or assets, we want to find out the optimum way of investing a particular amount of money in these assets. Each one of the different ways to diversify this money between the several assets is called a portfolio (4). For solving this portfolio problem, Markowitz (13, 14) has set up a quantitative framework. Markowitz's model which is called Mean-Variance model assumes that the return on a portfolio of assets can

be completely described by the expected return and the variance of returns (risk) between these assets. For a particular universe of assets, the set of portfolios of assets that offers the minimum risk for a given level of return is the set of efficient portfolios. These portfolios can be found by convex quadratic programs (QP). However the Markowitz's standard model does not contain some practical constraints. For example, the standard Mean-Variance model has not got any bounding constraints limiting the amount of money to be invested in each asset neither prevents very small amounts of investments in each asset. This kind of constraints is very useful in practice and is called *threshold constraints* (1). In order to overcome these inconveniences, the standard model can be generalized to include these constraints.

In a recent article (7), Konno et al. have considered a long-short portfolio optimization problem where one can sell assets short if it leads to a better risk-return structure of the portfolio. Long-short strategy is attracting more attention of practitioners. However, there exists almost no literature which handled this problem in a rigorous portfolio optimization framework (8).

In this paper, we focus on solving a generalization of the Markowitz Mean-Variance model which includes threshold constraints. Moreover, in our model one can sell assets short if it leads to a better risk-return structure of the portfolio (1, 8). Threshold constraints limit the amount of capital to be invested in (or sold short from) each asset and prevent very small investments in (or short selling from) any asset (1). The model was proposed in (1). In order to solve this model, we investigate a local deterministic approach based on DC (Difference of Convex functions) programming and DCA (DC Algorithms) that were introduced by Pham Dinh Tao in their preliminary forms in 1985. They have been extensively developed since 1994 by Le Thi Hoai An and Pham Dinh Tao and become now classic (see e.g. (6, 9, 11, 15, 16, 18) and references therein). DCA has been successfully applied to many large-scale (smooth or nonsmooth) nonconvex programs in various domains of applied sciences, for which it provided quite often a global solution and proved to be more robust and efficient than standard methods (see e.g. (6, 9, 11, 15, 16, 18) and reference therein). The existence of threshold constraints makes the corresponding portfolio selection problem nonconvex and so very difficult to solve by existing algorithms. After changing of variables, we formulate the

For Peer Review

problem as a mixed 0-1 quadratic programming problem with linear constraints as well as some complementary constraints. The existence of complementary constraints makes the problem even more difficult. By using an exact penalty result, we reformulate the last problem in terms of a DC (Difference of Convex functions) program. A so-called DC program is that of minimizing a DC function over a convex set. We then suggest using DC programming approach and DCA to solve the resulting problem. For testing the efficiency of DCA, we compare it with a branch and bound algorithm. Finally, we propose a combination of DCA and branch and bound for globally solving the problem.

The paper is organized as follows. After the introduction, we present in section 2 the model of the long-short portfolio selection problem under threshold constraints, and the reformulation in term of a DC program. Section 3 deals with DC programming and a special realization of DCA to the underlying portfolio problem. Section 4 is devoted to the combined DCA - Branch and Bound algorithm. The numerical results are reported in section 5 and finally some conclusions are given in section 6.

2. Portfolio selection problem under threshold constraints

2.1. Problem formulation

First of all, as we introduce the notations that we are going to use in this paper, let us remind the well known Markowitz's Mean-Variance model (1, 4, 13, 14) for the portfolio selection problem. Let n be the number of available assets, r_i be the mean return of asset i, Q be an $n \times n$ Variance-Covariance (positive semidefinite) matrix such that its (i,j)-th element, that is $\sigma_{i,j}$ is the covariance between returns of assets i and j and its value is calculated by using the following formula:

$$\sigma_{ij} = (1/m)\sum_{k=1}^{m}((r_{ik} - r_i)(r_{jk} - r_j)). \tag{1}$$

Here r_{ik} is the (i,k)-th historical data and m is the number of periods that we have considered. Let R be the desired expected return and the decision variables x_i represent the proportion ($0 \leq x_i \leq 1$) of capital to be invested in asset i and $x^T = (x_1, ..., x_n)$. Using this notation, the standard Markowitz's Mean-Variance model is (see (1))

For Peer Review

$$\min\left\{V(x) := x^T Q x : \sum_{i=1}^{n} r_i x_i = R, \sum_{i=1}^{n} x_i = 1, x_i \geq 0,\ i = 1, \ldots, n\right\}. \tag{2}$$

By solving this problem, one minimizes the total variance (risk) associated with the portfolio by ensuring that the portfolio has an expected return R.

This formulation is a simple convex quadratic program for which efficient algorithms are available. By solving the above quadratic programming for varying values of R, we can trace out the efficient frontier, a smooth non-decreasing curve that gives the best possible tradeoff of risk against return (4).

For generalizing the standard Markowitz model into a long-short form with the inclusion of *threshold* constraints, we will use some additional notations. Let a_i and b_i be the lower and upper bounds, respectively, for the proportion of capital to be invested in asset i, with $0 < a_i \leq b_i \leq 1$. Similarly, let c_i and d_i be the lower and upper bounds, respectively, for the proportion of capital to be sold short in asset i, with $-1 \leq c_i \leq d_i < 0$. The generalized Mean-Variance model for the portfolio selection problem under threshold constraints can be written as

$$\min\left\{V(x) := x^T Q x : \sum_{i=1}^{n} r_i x_i = R, \sum_{i=1}^{n} |x_i| = 1, x_i \in \{0\} \cup [a_i, b_i] \cup [c_i, d_i],\ i = 1, \ldots, n\right\}. \tag{3}$$

Due to the last constraints ($x_i \in \{0\} \cup [a_i, b_i] \cup [c_i, d_i]$) and the absolute valued constraints ($\sum_{i=1}^{n} |x_i| = 1$), this is a hard problem for which efficient algorithms are not available.

In the model (3), we assumed that the investor has a fixed sum of money to invest and selling short involves putting up an amount of money which equals to the short sale. So the total funds invested short (i.e., when $x_i < 0$), plus the funds invested long (i.e., when $x_i \geq 0$) must be equal to the original investment (i.e., $\sum_{i=1}^{n} |x_i| = 1$) (see (3)).

In this paper, we have considered no transaction costs and no riskless assets, but it is worth noting that if we add the transaction costs and/or riskless assets, one gets a similar model of (3) (see (3, 7)).

For Peer Review

2.2. Reformulation

First of all, by introducing a set of auxiliary variables y_i, y_i' such that

$$x_i = y_i - y_i', y_i y_i' = 0, y_i \in \{0\} \cup [a_i, b_i], y_i' \in \{0\} \cup [c_i, d_i], \tag{4}$$

we can express the problem (3) as follows

$$(P): \begin{cases} \min V(y,y') := (y+y')^T Q (y+y') \\ s.t: \begin{cases} \sum_{i=1}^{n} r_i (y_i + y_i') = R, \\ \sum_{i=1}^{n} (y_i - y_i') = 1, \\ y_i \in \{0\} \cup [a_i, b_i] \quad : i = 1, \ldots, n, \\ y_i' \in \{0\} \cup [c_i, d_i] \quad : i = 1, \ldots, n, \\ y_i y_i' = 0 \quad : i = 1, \ldots, n. \end{cases} \end{cases}$$

Due to the constraints $y_i \in \{0\} \cup [a_i, b_i], y_i' \in \{0\} \cup [c_i, d_i], y_i y_i' = 0$, it is difficult to solve the problem (P). By introducing the additional variables z_i and z_i' such that

$$z_i = 1 \text{ iff } y_i \in [a_i, b_i], \ 0 \text{ otherwise},$$

and

$$z_i' = 1 \text{ iff } y_i' \in [c_i, d_i], \ 0 \text{ otherwise},$$

we can reformulate the last problem as a mixed $0-1$ quadratic problem, namely:

$$(P'): \begin{cases} \min V(y,y') := (y+y')^T Q (y+y') \\ s.t: \begin{cases} \sum_{i=1}^{n} r_i (y_i + y_i') = R, \\ \sum_{i=1}^{n} (y_i - y_i') = 1, \\ a_i z_i \le y_i \le b_i z_i \quad : i = 1, \ldots, n, \\ c_i z_i' \le y_i' \le d_i z_i' \quad : i = 1, \ldots, n, \\ z_i z_i' = 0 \quad : i = 1, \ldots, n, \\ z_i, z_i' \in \{0,1\} \quad : i = 1, \ldots, n. \end{cases} \end{cases}$$

Furthermore, using the exact penalty result presented in (12), we will formulate (P') in the form of a convex-concave minimization problem with linear constraints which is consequently a DC program. Let

$$\mathcal{A} := \left\{ \begin{array}{l} (y, y', z, z') \in \mathbb{R}^n \times \mathbb{R}^n \times [0,1]^n \times [0,1]^n : \sum_{i=1}^{n} r_i (y_i + y_i') = R, \sum_{i=1}^{n} (y_i - y_i') = 1, \\ a_i z_i \le y_i \le b_i z_i, c_i z_i' \le y_i' \le d_i z_i', i = 1, \ldots, n. \end{array} \right\}.$$

For Peer Review

THEOREM 1. *Let* $p(y,y',z,z'):\mathbb{R}^n\times\mathbb{R}^n\times\mathbb{R}^n\times\mathbb{R}^n\longrightarrow\mathbb{R}$ *be the penalty function defined by*

$$p(y,y',z,z'):=\sum_{i=1}^{n}(z_i+z_i')-\sum_{i=1}^{n}(z_i-z_i')^2. \tag{5}$$

(i) *The function* $p(y,y',z,z')$ *is concave.*

(ii) *The function* $p(y,y',z,z')$ *is nonnegative on* $\mathcal{A}$ *and*

$$\{(y,y',z,z')\in\mathcal{A}:z_iz_i'=0,z_i,z_i'\in\{0,1\}\quad:i=1,\ldots,n\}=\{(y,y',z,z')\in\mathcal{A}:p(y,y',z,z')\leq 0\}.$$

Proof : **(i)** Define the functions $p_i:\mathbb{R}^{4n}\longrightarrow\mathbb{R}$ by

$$p_i(y,y',z,z'):=(z_i+z_i')-(z_i-z_i')^2\quad:\forall i=1,\ldots,n$$

define also $\varphi_i:\mathbb{R}^{4n}\longrightarrow\mathbb{R}$ by

$$\varphi_i(y,y',z,z'):=(z_i-z_i')^2\quad:\forall i=1,\ldots,n.$$

The functions φ_i, $i=1,\ldots,n$ are convex functions since they are compositions of a linear function with a convex one. Consequently p_i, $i=1,\ldots,n$ are a concave function, and thereby $p(y,y',z,z'):=\sum_{i=1}^{n}p_i(y,y',z,z')$ is a concave function.

(ii) After rearranging the terms in (5) one can see that

$$p(y,y',z,z'):=2\sum_{i=1}^{n}(z_iz_i')+\sum_{i=1}^{n}z_i(1-z_i)+\sum_{i=1}^{n}z_i'(1-z_i').$$

The terms on the right hand side are nonnegative on $\mathcal{A}$, so is $p(y,y',z,z')$. Furthermore, we have

$$p(y,y',z,z')=0\Longleftrightarrow\begin{cases}z_iz_i'=0 & :\quad\forall i=1,\ldots,n,\\ z_i(1-z_i)=0 & :\quad\forall i=1,\ldots,n,\\ z_i'(1-z_i')=0 & :\quad\forall i=1,\ldots,n,\end{cases} \tag{6}$$

or again

$$p(y,y',z,z')=0\Longleftrightarrow\begin{cases}z_iz_i'=0 & :\quad\forall i=1,\ldots,n,\\ z_i\in\{0,1\} & :\quad\forall i=1,\ldots,n,\\ z_i'\in\{0,1\} & :\quad\forall i=1,\ldots,n.\end{cases} \tag{7}$$

Hence

$$\{(y,y',z,z')\in\mathcal{A}:z_iz_i'=0,z_i,z_i'\in\{0,1\}\quad:i=1,\ldots,n\}=\{(y,y',z,z')\in\mathcal{A}:p(y,y',z,z')=0\}.$$

On the other hand, $p(y,y',z,z')$ is nonnegative on $\mathcal{A}$. Therefore

$$\{(y,y',z,z')\in\mathcal{A}:p(y,y',z,z')=0\}=\{(y,y',z,z')\in\mathcal{A}:p(y,y',z,z')\leq 0\}.$$

For Peer Review

The above theorem shows that the problem (P') can be expressed as

$$\min\left\{V(y,y') := (y+y')^T Q(y+y') : (y,y',z,z') \in \mathcal{A}, p(y,y',z,z') \leq 0\right\}. \tag{8}$$

In (8), the objective function V is convex, $\mathcal{A}$ is a bounded polyhedral convex set, and p is concave and nonnegative on $\mathcal{A}$. Then according to (12), there is $t_0 \geq 0$ such that for any $t > t_0$, the program (8) is equivalent to

$$\min\left\{F(y,y',z,z') := (y+y')^T Q(y+y') + tp(y,y',z,z') : (y,y',z,z') \in \mathcal{A}\right\}. \tag{9}$$

The function F is convex in variables y and y' and concave in variables z and z'. Consequently, it is a DC function (Difference of two Convex functions). A natural DC formulation of the problem (9) is

$$\min\left\{g(y,y',z,z') - h(y,y',z,z') : (y,y',z,z') \in \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n\right\}, \tag{10}$$

where

$$g(y,y',z,z') := (y+y')^T Q(y+y') + \chi_{\mathcal{A}}(y,y',z,z'),$$

and

$$h(y,y',z,z') := t(\sum_{i=1}^{n}(z_i - z_i')^2 - \sum_{i=1}^{n}(z_i + z_i')).$$

Here $\chi_{\mathcal{A}}$ is the indicator function on $\mathcal{A}$, i.e. $\chi_{\mathcal{A}}(y,y',z,z') = 0$ if $(y,y',z,z') \in \mathcal{A}$ and $+\infty$ otherwise.

3. Solution method via DC programming and DCA

3.1. DCA for general DC programs

Let $\Gamma_0(\mathbb{R}^n)$ denote the convex cone of all lower semicontinuous proper convex functions on $\mathbb{R}^n$, and consider the general DC program

$$(P_{dc}) \quad \alpha = \inf\{f(x) := g(x) - h(x) : x \in \mathbb{R}^n\} \tag{11}$$

where $g, h \in \Gamma_0(\mathbb{R}^n)$. Such a function f is called DC function, and $g - h$, DC decomposition of f while the convex functions g and h are DC components of f.

Let C be a convex set. The problem

$$\inf\{f(x) := k(x) - h(x) : x \in C\} \tag{12}$$

For Peer Review

can be transformed to an unconstrained DC program by using the indicator function on C, i.e.,

$$\inf\{f(x) := g(x) - h(x) : x \in \mathbb{R}^n\} \tag{13}$$

where $g := k + \chi_C$.

Let $g^*(y) := \sup\{\langle x, y\rangle - g(x) : x \in \mathbb{R}^n\}$ be the conjugate function of g. Then, the following program is called the dual program of (P_{dc}):

$$(D_{dc}) \quad \alpha_D = \inf\{h^*(y) - g^*(y) : y \in \mathbb{R}^n\}. \tag{14}$$

Under the natural convention in DC programming that is $+\infty - (+\infty) = +\infty$, and by using the fact that every function $h \in \Gamma_0(\mathbb{R}^n)$ is characterized as a pointwise supremum of a collection of affine functions, say

$$h(x) := \sup\{\langle x, y\rangle - h^*(y) : y \in \mathbb{R}^n\},$$

one can prove that $\alpha = \alpha_D$. We observe the perfect symmetry between primal and dual DC programs: the dual to (D_{dc}) is exactly (P_{dc}).

Recall that, for $\theta \in \Gamma_0(\mathbb{R}^n)$ and $x_0 \in dom\ \theta := \{x \in \mathbb{R}^n : \theta(x_0) < +\infty\}$, $\partial\theta(x_0)$ denotes the subdifferential of θ at x_0, i.e., (17)

$$\partial\theta(x_0) := \{y \in \mathbb{R}^n : \theta(x) \geq \theta(x_0) + \langle x - x_0, y\rangle, \forall x \in \mathbb{R}^n\}. \tag{15}$$

The subdifferential $\partial\theta(x_0)$ is a closed convex set in $\mathbb{R}^n$. It generalizes the derivative in the sense that θ is differentiable at x_0 if and only if $\partial\theta(x_0)$ is reduced to a singleton which is exactly$\{\nabla\theta(x_0)\}$. The necessary local optimality condition for the primal DC program (P_{dc}) is:

$$\partial g(x^*) \supset \partial h(x^*). \tag{16}$$

A point x^* verifies the condition $\partial h(x^*) \cap \partial g(x^*) \neq \emptyset$ is called a critical point of $g - h$. The condition (16) is also sufficient for many important classes of DC programs, for example, in case of the function f is locally convex at x^* ((10, 11, 15)).

For Peer Review

The transportation of global solutions between (P_{dc}) and (D_{dc}) is expressed by:

$$[\cup_{y^* \in \mathcal{D}} \partial g^*(y^*)] \subset \mathcal{P}, \quad [\cup_{x^* \in \mathcal{P}} \partial h(x^*)] \subset \mathcal{D} \tag{17}$$

where $\mathcal{P}$ and $\mathcal{D}$ denote the solution sets of (P_{dc}) and (D_{dc}), respectively. Under technical conditions, this transportation holds also for local solutions of (P_{dc}) and (D_{dc}) ((9, 11, 15, 16)).

Based on local optimality conditions and duality in DC programming, the DCA consists of the construction of two sequences $\{x^k\}$ and $\{y^k\}$, candidates to be optimal solutions of primal and dual programs, respectively, such that the sequences $\{g(x^k) - h(x^k)\}$ and $\{h^*(y^k) - g^*(y^k)\}$ are decreasing, and $\{x^k\}$ (resp. $\{y^k\}$) converges to a primal feasible solution $\widetilde{x}$ (resp. a dual feasible solution $\widetilde{y}$) verifying local optimality conditions and

$$\widetilde{x} \in \partial g^*(\widetilde{y}), \quad \widetilde{y} \in \partial h(\widetilde{x}). \tag{18}$$

The DCA then yields the next scheme:

$$y^k \in \partial h(x^k); \quad x^{k+1} \in \partial g^*(y^k). \tag{19}$$

In other words, these two sequences $\{x^k\}$ and $\{y^k\}$ are determined in the way that x^{k+1} (resp. y^k) is a solution to the convex program (P_k) (resp. (D_k)) defined by

$$\inf\{g(x) - h(x^k) - \langle x - x^k, y^k\rangle : x \in \mathbb{R}^n\}, \quad (P_k)$$

$$\inf\{h^*(y) - g^*(y^{k-1}) - \langle y - y^{k-1}, x^k\rangle : y \in \mathbb{R}^n\} \quad (D_k).$$

In fact, at each iteration one replaces in the primal DC program (P_{dc}) the second component h by its affine minorization $h_k(x) := h(x^k) + \langle x - x^k, y^k\rangle$ at a neighborhood of x^k to give birth to the convex program (P_k) whose the solution set is nothing but $\partial g^*(y^k)$. Likewise, the second DC component g^* of the dual DC program (D_{dc}) is replaced by its affine minorization $(g^*)_k(y) := g^*(y^k) + \langle y - y^k, x^{k+1}\rangle$ at a neighborhood of y^k to obtain the convex program (D_k) whose $\partial h(x^{k+1})$ is the solution set. DCA performs so a double linearization with the help of the subgradients of h and g^*.

For Peer Review

It is worth noting that DCA works with the convex DC components g and h but not the DC function f itself (9, 11, 15, 16). Moreover, a DC function f *has infinitely many DC decompositions which have crucial impacts on the qualities* (speed of convergence, robustness, efficiency, globality of computed solutions,...) of DCA.

Convergence properties of DCA and its theoretical basis can be found in (9, 11, 15), for instance it is important to mention that:

- DCA is a descent method (the sequences $\{g(x^k)-h(x^k)\}$ and $\{h^*(y^k)-g^*(y^k)\}$ are decreasing) without linesearch.
- If the optimal value α of problem (P_{dc}) is finite and the infinite sequences $\{x^k\}$ and $\{y^k\}$ are bounded then every limit point $\widetilde{x}$ (resp. $\widetilde{y}$) of the sequence $\{x^k\}$ (resp. $\{y^k\}$) is a critical point of $g-h$ (resp. h^*-g^*).
- DCA has a linear convergence for general DC programs.

3.2. DCA for solving (9)

According to the general framework of DCA, we first need to compute a sub-gradient of the function h defined by $h(y,y',z,z') := t(\sum_{i=1}^{n}(z_i-z'_i)^2 - \sum_{i=1}^{n}(z_i+z'_i))$. From the definition of h, we have

$$(u^k,u'^k,v^k,v'^k)\in\partial h(y^k,y'^k,z^k,z'^k) \Leftrightarrow u_i^k=0, u_i'^k=0, v_j^k=t(2(z_j^k-z_j'^k)-1), v_j'^k=t(2(z_j'^k-z_j^k)-1), \tag{20}$$

for all $i,j=1,\ldots,n$. Secondly, we have to compute an optimal solution of the following convex quadratic program

$$\min\{(y+y')^TQ(y+y')-\langle(y,y',z,z'),(u^k,u'^k,v^k,v'^k)\rangle : (y,y',z,z')\in\mathcal{A}\} \tag{21}$$

that will be $(y^{k+1},y'^{k+1},z^{k+1},z'^{k+1})$. To sum up, the DCA applied to (9) can be described as follows.

Algorithm DCA

- **Initialization**: Let ϵ be a sufficiently small positive number, let $(y^0,y'^0,z^0,z'^0)\in R^n\times R^n\times[0,1]^n\times[0,1]^n$, and set $k=0$.

For Peer Review

- **Iteration**: For $k = 0, 1, 2, ...$

Set $u_i^k = 0$, $u_i'^k = 0$, $v_j^k = t(2(z_j^k - z_j'^k) - 1)$, and $v_j'^k = t(2(z_j'^k - z_j^k) - 1)$ for $i = 1, ..., n$.

and solve the following quadratic program

$$\min\{(y+y')^T Q(y+y') - \langle (y, y', z, z'), (u^k, u'^k, v^k, v'^k)\rangle : (y, y', z, z') \in \mathcal{A}\}. \tag{22}$$

to obtain $(y^{k+1}, y'^{k+1}, z^{k+1}, z'^{k+1})$.

- **Termination criterion**: If $\| y^{k+1} - y^k \| + \| y'^{k+1} - y'^k \| + \| z^{k+1} - z^k \| + \| z'^{k+1} - z'^k \| \leq \epsilon$, then stop, $(y^{k+1}, y'^{k+1}, z^{k+1}, z'^{k+1})$ is a solution, otherwise set $k = k+1$ and go to step 2.

THEOREM 2 **(Convergence Theorem of DCA for solving (9))**. **(i)** *Algorithm DCA generates a sequence* $\{(y^k, y'^k, z^k, z'^k)\}$ *such that the sequence* $\{F(y^k, y'^k, z^k, z'^k)\}$ *is decreasing.*

(ii) *There is a nonnegative number* t_0 *such that for* $t > t_0$, *the sequence* $\{p(y^k, y'^k, z^k, z'^k)\}$ *is decreasing. In particular if* (y^k, y'^k, z^k, z'^k) *is a feasible solution of* (P') *then* (y^k, y'^k, z^k, z'^k), *for* $k \geq r$, *is feasible too.*

(iii) *DCA has a linear convergence for (9).*

(iv) *The sequence* $\{(y^k, y'^k, z^k, z'^k)\}$ *converges to* $(\tilde{y}, \tilde{y'}, \tilde{z}, \tilde{z'})$, *where the point* $(\tilde{y}, \tilde{y'}, \tilde{z}, \tilde{z'})$ *is a critical point of the problem (9).*

Proof : **(i)**, **(iii)**, and **(iv)** are immediate consequences of DCA's convergence theorem for a general DC program (see (9, 10, 15, 16)).

(ii) Let $\mathcal{V}(\mathcal{A})$ be the vertex set of $\mathcal{A}$. If $\mathcal{V}(\mathcal{A})$ is contained in the feasible set of (P') then $t_0 = 0$ (12). Otherwise, let

$$\xi := min\{p(\hat{y}, \hat{y'}, \hat{z}, \hat{z'}) - p(y, y', z, z') : ((\hat{y}, \hat{y'}, \hat{z}, \hat{z'}), (y, y', z, z')) \in \mathcal{V}(\mathcal{A}) \times \mathcal{V}(\mathcal{A}), p(\hat{y}, \hat{y'}, \hat{z}, \hat{z'}) > p(y, y', z, z')\}$$

$$\eta := max\{[(\hat{y}+\hat{y'})^T Q(\hat{y}+\hat{y'}) - (y+y')^T Q(y+y')] : ((\hat{y}, \hat{y'}, \hat{z}, \hat{z'}), (y, y', z, z')) \in \mathcal{V}(\mathcal{A}) \times \mathcal{V}(\mathcal{A})\},$$

then $0 < \xi < \infty$ and $0 \leq \eta < \infty$, since $\mathcal{V}(A)$ is a finite set. Consider now the nonnegative number t_0 defined by

$$t_0 := \frac{\eta}{\xi}$$

For Peer Review

and set $t > t_0$. Let $\{(y^k, y'^k, z^k, z'^k)\}$ be the sequence generated by algorithm DCA applied to (9), with this value of $\mathbf{t}$. Assume by contradiction that there is $r \geq 1$ such that $p(y^{r+1}, y'^{r+1}, z^{r+1}, z'^{r+1}) > p(y^r, y'^r, z^r, z'^r)$. Since $t > t_0$, we have

$$t[p(y^{r+1}, y'^{r+1}, z^{r+1}, z'^{r+1}) - p(y^r, y'^r, z^r, z'^r)] > t_0[p(y^{r+1}, y'^{r+1}, z^{r+1}, z'^{r+1}) - p(y^r, y'^r, z^r, z'^r)],$$

or

$$t[p(y^{r+1}, y'^{r+1}, z^{r+1}, z'^{r+1}) - p(y^r, y'^r, z^r, z'^r)] > \frac{\eta}{\xi}[p(y^{r+1}, y'^{r+1}, z^{r+1}, z'^{r+1}) - p(y^r, y'^r, z^r, z'^r)].$$

By the very definition of ξ:

$$\xi \leq [p(y^{r+1}, y'^{r+1}, z^{r+1}, z'^{r+1}) - p(y^r, y'^r, z^r, z'^r)],$$

so

$$t[p(y^{r+1}, y'^{r+1}, z^{r+1}, z'^{r+1}) - p(y^r, y'^r, z^r, z'^r)] > \eta.$$

Hence by considering the definition of η, we have

$$t[p(y^{r+1}, y'^{r+1}, z^{r+1}, z'^{r+1}) - p(y^r, y'^r, z^r, z'^r)] > (y^r + y'^r)^T Q(y^r + y'^r) - (y^{r+1} + y'^{r+1})^T Q(y^{r+1} + y'^{r+1})$$

i.e.,

$$[(y^{r+1} + y'^{r+1})^T Q(y^{r+1} + y'^{r+1}) + tp(y^{r+1}, y'^{r+1}, z^{r+1}, z'^{r+1})] > [(y^r + y'^r)^T Q(y^r + y'^r) + tp(y^r, y'^r, z^r, z'^r)].$$

This is not possible because $\left\{F(y^k, y'^k, z^k, z'^k) := ((y^k + y'^k)^T Q(y^k + y'^k) + tp(y^k, y'^k, z^k, z'^k))\right\}$ is a decreasing sequence.

In particular, if $(y^r, y'^r, z^r), z'^r)$ is a feasible solution of (P'), then (y^k, y'^k, z^k, z'^k) is feasible too, for $k \geq r$.

3.3. Finding a good initial point for DCA

In fact, one of the key questions in DCA is how to find a good initial solution for it. The question is still open. In this work, in order to find a good initial solution we first solve the relaxed problem

For Peer Review

of (P) where the constraints $y_i \in \{0\} \cup [a_i, b_i]$ and $y'_i \in \{0\} \cup [c_i, d_i]$ are replaced by $0 \leq y_i \leq b_i$ and $c_i \leq y'_i \leq 0$, respectively to obtain an optimal solution $(\bar{y}_i, \overline{y'}_i)_{i=1,\ldots,n}$. The initial point of DCA is then taken as $(\bar{y}_i, \overline{y'}_i, \bar{z}_i, \overline{z'}_i)_{i=1,\ldots,n}$ where $\overline{z'}_i = 0$ for $i = 1, \ldots, n$, $\bar{z}_i = 1$ if $R < r_i$ and $\bar{z}_i = 0$ otherwise $(i = 1, \ldots, n)$. Notice that this point may not be feasible to (9), but we need just one iteration of DCA to obtain a feasible solution of (9), and all the other iterations of DCA will improve the solution.

We have tested DCA from different initial points:

- The point obtained by the above procedure;
- The optimal solution of the relaxed problem of (P');
- The optimal solution of the next problem

$$\min \left\{ p(y, y', z, z') := (\sum_{i=1}^{n} (z_i + z'_i) - \sum_{i=1}^{n} (z_i - z'_i)^2) : (y, y', z, z') \in \mathcal{A} \right\}.$$

In our experiments the initial point of DCA given by the first procedure is the best.

3.4. Restarting DCA

From theoretical point of view (see (12)) one can always find a $t_0 \geq 0$ such that for any $t > t_0$, the program (8) is equivalent to program (9), but calculating the exact value of t_0 is not so simple. In this study, we have tested several values for the penalty parameter t and finally we have chosen a suitable value. For the chosen value of t, the solution provided by DCA may not satisfy all binary constraints i.e., $z_i, z'_i \in \{0, 1\}$. We investigate a procedure for restarting DCA in such a case. After using DCA to solve (9), we obtain a solution like $(y, y', z, z')^1$. If $z_i, z'_i \in \{0, 1\}$ for all $i = 1, \ldots, n$, we do not need to restart DCA. If there is k $(1 \leq k \leq n)$ such that z_k is very close to 1 (or respectively 0), we add the constraint $z_k = 1$ (respectively $z_k = 0$) to program (9). If there is l $(1 \leq l \leq n)$ such that z_l is neither close to 0 nor close to 1, we add no constraint concerning this variable. We apply DCA from the starting point $(y, y', z, z')^1$ to the new program obtained by adding these new constraints to program (9). In all of the experiments that we have done, the solutions provided by the restarting DCA always satisfy the constraints $z_i, z'_i \in \{0, 1\}$.

For Peer Review

4. A combined DCA - Branch and Bound algorithm for solving (P)

For evaluating the quality of solutions computed by DCA and their globality, we solve the problem (P) using a branch and bound algorithm. More precisely, the lower bound is computed by solving the following convex quadratic program

$$\begin{cases} \min V(y,y') := (y+y')^T Q(y+y') \\ s.t: \begin{cases} \sum_{i=1}^{n} r_i(y_i+y_i') = R, \\ \sum_{i=1}^{n} (y_i - y_i') = 1, \\ 0 \le y_i \le b_i \qquad : i=1,\ldots,n, \\ c_i \le y_i' \le 0 \qquad : i=1,\ldots,n \end{cases} \end{cases} \tag{23}$$

which is obtained from (P) by eliminating the following constraints

$$y_i y_i' = 0 \quad : \quad i=1,\ldots,n,$$

and relaxation of the constraints $y_i \in \{0\} \cup [a_i, b_i]$ and $y_i' \in \{0\} \cup [c_i, d_i]$, $(i=1,\ldots,n)$ by $0 \le y_i \le b_i$ and $c_i \le y_i' \le 0$, respectively. The upper bound is updated when a better feasible solution to (P) is discovered. The subdivision is performed in the way that, either

$$y_i = 0, y_i' = 0,$$

or

$$y_i = 0, y_i' \in [c_i, 0],$$

or

$$y_i' = 0, y_i \in [0, b_i].$$

Hybrid approach for solving (P)

In order to globally solving (P), we apply DCA inside the branch and bound algorithm described in the previous section. We denote this hybrid approach by BB-DCA. The combined procedure aims at checking the globality of solutions computed by DCA applied to the equivalent DC program (9) and restarting DCA when its solution is not global.

We use DCA in the first time at the end of the first iteration of the branch and bound algorithm. Then during the branch and bound process, we restart DCA when the optimal solution to the

For Peer Review

current relaxed problem (23) has a sufficiently large number of $0-1$ components, namely greater than or equal to $(3n/4)$. This criterion is used in order to prevent overusing of DCA inside the branch and bound scheme.

The restarting DCA inside the branch and bound algorithm is proceeded as follows:

1. Construct the subproblem in the form of (9) which corresponds to the current relaxed program and name it Relaxed Program (RP).

2. Solve (RP) by DCA with the initial point being the solution to the current relaxed problem to obtain the solution $(y, y', z, z')^1$.

3. For each $i = 1, \ldots, n$,

   - if $z_i^1 \geq 0.5$, then add the constraint $z_i = 1$ to the problem (RP), else add the constraint $z_i = 0$ to (RP);

   - if $z_i'^1 \geq 0.5$, then add the constraint $z_i' = 1$ to the problem (RP), else add the constraint $z_i' = 0$ to (RP).

4. Applying DCA to the new problem from the initial point $(y, y', z, z')^1$

   - If the new problem is infeasible, return to the branch and bound algorithm,

   - else let $(y, y', z, z')^2$ be the solution provided by DCA for this problem.

5. If $(y, y', z, z')^2$ is a better solution for (P), then update the upper bound and the best current solution of the problem.

6. Continue the branch and bound algorithm.

Numerical experiments in the next section show the efficiency of the combined procedure in comparison with the branch and bound algorithm.

5. The numerical results

The algorithms are coded in C++ and run on a Pentium IV $3.000GHz$ of $1.00GB$ RAM. To solve the resulting quadratic programs, we used the software CPLEX version 9.1.

We have tested the algorithms on two sets of data that have been already used in (2, 4, 5). These data correspond to weekly prices from March 1992 to September 1997 and they come from

For Peer Review

the indices : S&P 100 in United States and Dax 100 in Germany. The number n of different assets considered for each one of the test problems is 98 and 85, respectively. The mean returns and covariances between these returns have been calculated for the data. All the results presented here have been computed using the values $a_i = 0.05$, $b_i = 1.0$, $c_i = -1.0$, and $d_i = -0.0001$ in (9) and (P). We have tested DCA and the branch and bound algorithm (as described above) without DCA (denoted BB) and with DCA (denoted BB-DCA)) for more than thirty different values of desired expected return R. The tolerance ϵ is equal to 10^{-7}. The parameter t is taken the value 0.2×10^{-5} for the first set of data and 0.2×10^{-4} for the second set of data when only DCA is applied. In the combined algorithm (BB-DCA), t is equal to 0.2×10^{-5} for both sets of data.

The stopping criteria of the branch and bound algorithm is either the difference of the best upper bound and the best lower bound is smaller than $\epsilon := 10^{-6}$ or its number of iterations is greater than 10000. In the first case, we say that the algorithm gives an ϵ-optimal solution. In figures (1-8), we give the results for the two considered sets of data. In figures (1, 2), the global optimum values and the optimal values obtained by DCA are shown. The CPU time (in seconds) for each of the algorithms is presented in figures (3, 4). The number of iterations for each of the algorithms (that is, branch and bound (BB), the combined approach (BB-DCA), and (DCA) are presented in figures (5, 6). Finally, the number of iterations of the contributed algorithms (branch and bound as well as DCA) in the combined approach (denoted respectively by "BB in BB-DCA" and "DCA in BB-DCA") is presented in figures (7, 8).

Since the problems are feasible for those values of R for which the following condition is satisfied

$$\min_{i=1,\ldots,n} r_i \leq R \leq \max_{i=1,\ldots,n} r_i,$$

we have chosen more than thirty different values of R in the interval $[10^{-5}, 10^{-2}]$. We notice that, as will be seen later, with these values the branch and bound algorithm does not solve efficiently the corresponding problems.

We also reduce the tolerance level (increased ϵ) from 10^{-7} to $0.5 * 10^{-5}$ in order to study the influence of this change on the numerical results. The results are presented in the tables (1) and

For Peer Review

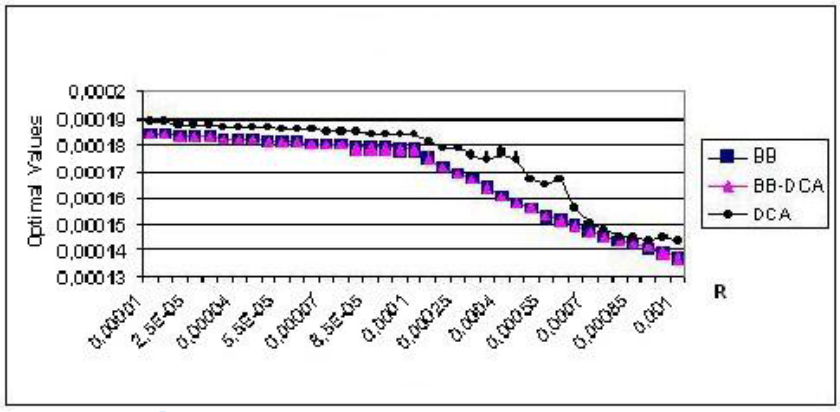


Figure 1 Global optimal values and the optimal values provided by DCA for the first set of data (S&P 100).

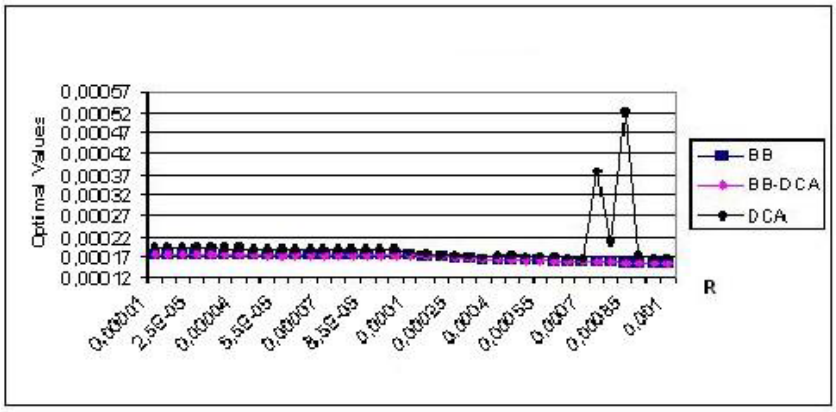


Figure 2 Global optimal values and the optimal values provided by DCA for the second set of data (Dax 100).

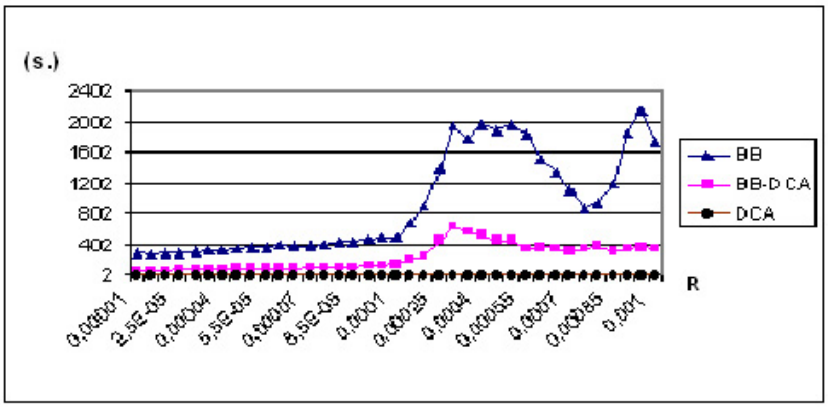


Figure 3 CPU time (in second) of each algorithm, for the first set of data (S&P 100).

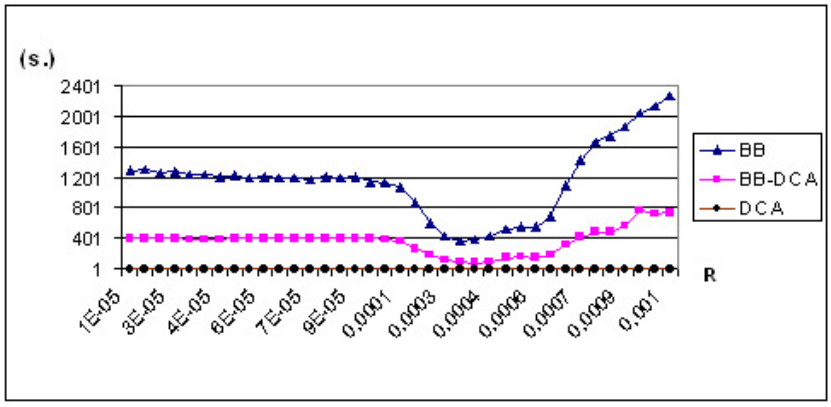


Figure 4 CPU time (in second) of each algorithm, for the second set of data (Dax 100).

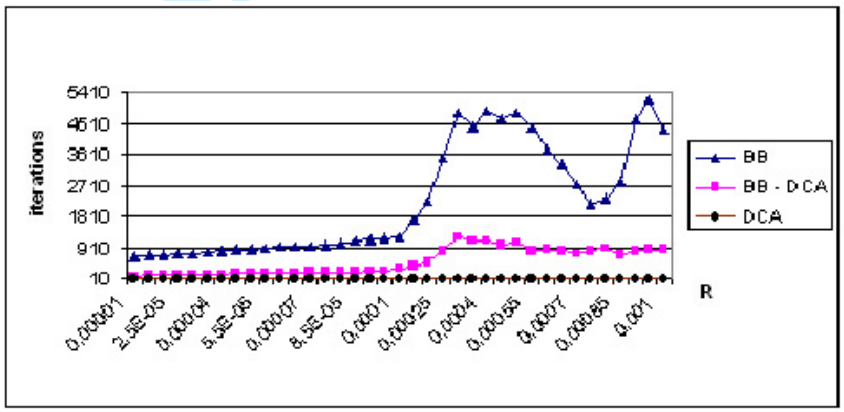


Figure 5 Number of iterations of each algorithm, for the first set of data (S&P 100).

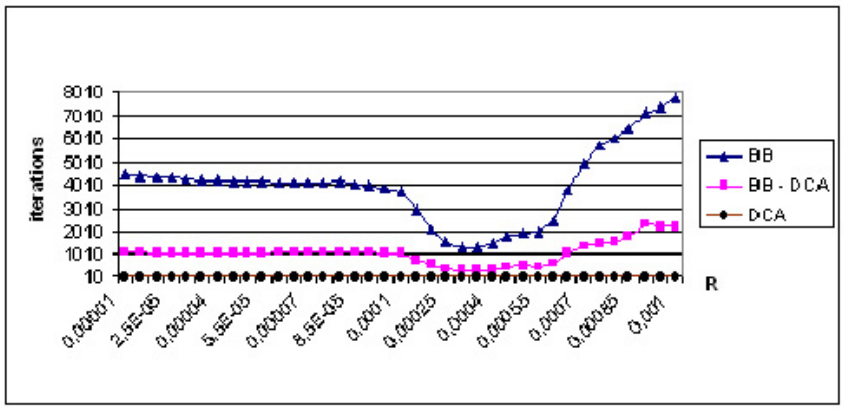


Figure 6 Number of iterations of each algorithm, for the second set of data (Dax 100).

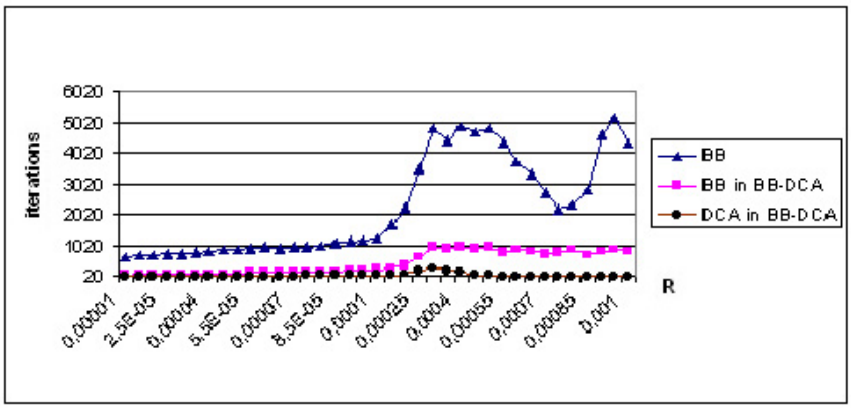


Figure 7 Number of branch and bound iterations in the combined approach and the number of calling DCA by branch and bound algorithm in the combined approach, for the first set of data (S&P 100).

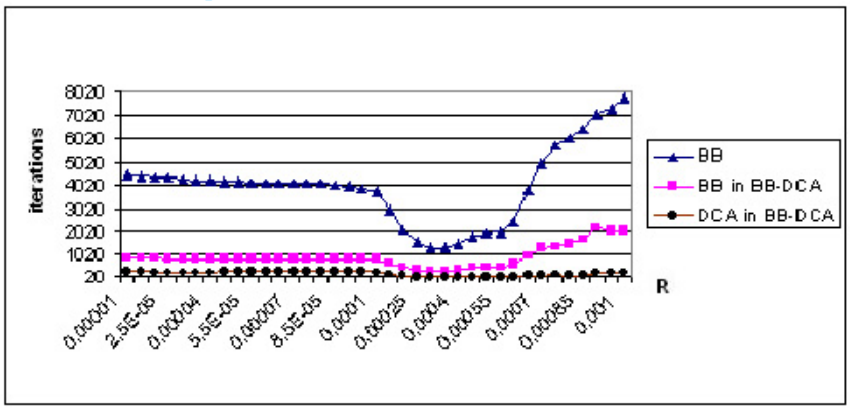


Figure 8 Number of branch and bound iterations in the combined approach and the number of calling DCA by branch and bound algorithm in the combined approach, for the second set of data (Dax 100).

(2).

With the increase of ϵ, we weaken the criterion on restarting DCA in branch and bound algorithm: $(3n/4)$ is now replaced by $(n/2)$ (see Section 4).

Comments on the numerical results.

From the numerical results we observe that

• DCA gives a very good approximation of the optimal solution within a very short time. More precisely, DCA and the combined BB-DCA found, most of the time, the same optimal value with the precision 10^{-5} in the first and the second data set. The number of iterations of DCA is equal

Table 1 Numerical results for the first data set (S&P 100).

-	BB			BB-DCA			
R	iter.	Opt. val.	CPU	iter.	use	Opt. val.	CPU
0.00001	689	0.000184	297.329	1	1	0.000184	1.094
0.00002	723	0.000183	309.828	1	1	0.000184	1.093
0.00003	775	0.000183	333.391	1	1	0.000183	1.094
0.00004	842	0.000182	359.750	1	1	0.000182	1.110
0.00005	882	0.000181	377.312	1	1	0.000182	1.094
0.00006	966	0.000181	413.906	1	1	0.000181	1.109
0.00007	963	0.000180	415.594	1	1	0.000180	1.094
0.00008	1047	0.000179	457.250	46	3	0.000180	21.594
0.00009	1182	0.000179	509.625	46	3	0.000179	21.703
0.0001	1250	0.000178	568.469	46	3	0.000178	21.609
0.0002	2254	0.000172	992.360	74	29	0.000174	57.891
0.0003	4863	0.000167	2105.859	89	13	0.000167	48.297
0.0004	4884	0.000161	2153.968	57	3	0.000162	26.078
0.0005	4850	0.000156	2151.125	70	3	0.000157	31.578
0.0006	3802	0.000151	1707.750	24	1	0.000152	10.531
0.0007	2740	0.000147	1241.594	57	3	0.000148	25.750
0.0008	2339	0.000143	1058.672	22	1	0.000145	10.703
0.0009	4642	0.000140	2092.094	102	3	0.000142	43.969
0.001	4305	0.000137	1936.046	165	3	0.000139	70.047

to 6 or 7 or 8 for the first data set and 6 or 7 for the second data set. Note that for only one case, the number of DCA iterations is more than 10, this case corresponds to $R = 0.00065$ in the first set of data. That is why we have limited the maximum number of iterations in DCA algorithm by 10. On the other hand, the running time of DCA is less than one second for the second data set and less than two seconds for the first data set, for all of values of R.

• The classical branch and bound algorithm (without DCA) does not solve efficiently these problems. In fact, in all cases, it can not find even a feasible solution until the last iterations. Even, when we increase ϵ, the situation does not change.

• Due to the performance of DCA, the combined algorithm BB-DCA is very efficient, and it is much better than the branch and bound algorithm, especially when we increase ϵ to $0.5 * 10^{-5}$:

For Peer Review

Table 2 Numerical results for the second data set (Dax 100).

-	BB			BB-DCA			
R	iter.	Opt. val.	CPU	iter.	use	Opt. val.	CPU
0.00001	4467	0.000178	1356.859	61	3	0.000178	19.844
0.00002	4352	0.000177	1315.000	61	3	0.000178	19.641
0.00003	4290	0.000177	1306.469	57	3	0.000177	18.687
0.00004	4187	0.000177	1269.719	39	1	0.000177	12.031
0.00005	4133	0.000176	1241.953	36	1	0.000177	11.047
0.00006	4107	0.000176	1236.203	14	1	0.000176	4.641
0.00007	4083	0.000176	1238.281	14	1	0.000176	4.672
0.00008	4129	0.000175	1269.781	14	1	0.000175	4.610
0.00009	3991	0.000175	1224.282	14	1	0.000175	4.719
0.0001	3736	0.000174	1155.343	14	1	0.000175	4.672
0.0002	2063	0.000170	653.485	1	1	0.000171	0.797
0.0003	1305	0.000167	417.844	1	1	0.000167	0.781
0.0004	1496	0.000164	470.157	1	1	0.000164	0.781
0.0005	1900	0.000162	597.390	38	3	0.000163	13.265
0.0006	2428	0.000159	763.063	42	7	0.000159	17.359
0.0007	4924	0.000158	1573.516	42	7	0.000158	17.313
0.0008	6023	0.000156	1926.531	42	7	0.000156	17.734
0.0009	7212	0.000154	2299.563	103	13	0.000155	40.344
0.001	7825	0.000153	2547.953	49	7	0.000153	19.329

For Peer Review

the tables (1) and (2) show that the combined approach finds quite often an $\epsilon - optimal$ solution after just one iteration. In fact, in the combined approach, DCA improves the performance of the branch and bound procedure from several aspects: first of all, DCA finds good feasible solutions so it provides very good upper bounds for the optimal value. By the way, a considered number of rectangle are deleted during the branch and bound algorithm. Secondly, DCA is very fast, so using DCA in the branch and bound procedure is not really more expensive.

6. Conclusion

In this paper, we present a new approach for solving the portfolio selection problem. Instead of the standard Markowitz mean-variance model, we have used an extension including threshold and bounding constraints ((1)). Furthermore, we have accepted the selling short in the model if it

provides a better risk-return tradeoff. These constraints make the corresponding portfolio selection problem nonconvex and so very difficult to solve by existing algorithms. We have transformed this problem into a mixed integer quadratic program with complementary constraints and developed a deterministic approach based on DC programming and DCA. Numerical simulations show the efficiency of DCA, its inexpensiveness and its superiority with respect to the branch and bound techniques. A combined approach, based on DCA and branch and bound, has been proposed. The corresponding results show the positive influence of DCA on branch and bound from two points of view : CPU time and the quality of computed solutions.

References

[1] Bartholomew-Biggs M. 2005. *Nonlinear Optimization with Financial Applications*. Kluwer Academic Publishers, United States of America.

[2] Chang T.J., N. Meade, J.E. Beasley, and, Y.M. Sharaiha. 2000. Heuristics for cardinality constrained portfolio optimization. *Computers & Operations Research* **27** 1271–1302.

[3] E.J. Elton, M.J. Grubner, S.J. Brown, W.N. Goetzmann. 2003. *Modern Portfolio Theory and Investment Analysis*. John Wiley & Sons, Inc., United States of America.

[4] Fernandez A., S. Gomez. 2007. Portfolio selection using neural networks. *Computers & Operations Research* **34** 1177–1191.

[5] Jobst N., M. Horniman, C. Lucas, G. Mitra. 2001. Computational aspects of alternative portfolio selection models in the presence of discrete asset choice constraints. *Quantitative Finance* **1** 1–13.

[6] Harrington J.E., B.F. Hobbs, J.S. Pang, A. Liu, G. Roch. 2005. Collusive game solutions via optimisation. *Math. Program. Ser. B* **104**(1-2), 407–435.

[7] Konno H., T. Koshizuka, and R. Yamamoto, Mean-variance portfolio optimization problems under short sale opportunity. to appear in *Dynamics of Continuous, Discrete and Inpulsive System*

[8] Konno H., K. Akishino, R. Yamamoto. 2005. Optimization of a Long-Short Portfolio under Nonconvex Transaction Cost. *Computational Optimization and Applications* **32** 115–132.

[9] Le Thi, H.A. 1997. Contribution à l'optimisation non convexe et l'optimisation globale: Théorie, Algorithmes et Applications, Habilitation à Diriger des Recherches, Université de Rouen.

For Peer Review

[10] Le Thi H.A. and T. Pham Dinh. 2001. A continuous approach for globally solving linearly constrained quadratic zero-one programming problems. *Optimization* **50**(1-2) 93–120.

[11] Le Thi H.A. and T. Pham Dinh. 2005. The DC (difference of convex functions) Programming and DCA revisited with DC models of real world non convex optimization problems. *Annals of Operations Research* **133** 23–46.

[12] Le Thi H.A., T. Pham Dinh, V.N. Huynh. 2005. Exact Penalty Techniques in DC Programming, Research Report, LMI, National Institute for Applied Sciences - Rouen, France.

[13] Markowitz, Harry M. 1952. Portfolio Selection. *Journal of Finance* **7**(1) 77–91.

[14] Markowitz, Harry M. 1959. *Portfolio Selection*. John Wiley and Sons, New York.

[15] Pham Dinh T. and H.A Le Thi. 1997. Convex analysis approach to d.c. programming: Theory, Algorithms and Applications. *Acta Mathematica Vietnamica* dedicated to Professor Hoang Tuy on the occasion of his 70th birthday, **22**(1) 289–355.

[16] Pham Dinh T. and H.A. Le Thi. 1998. DC optimization algorithms for solving the trust region subproblem. *SIAM J. Optimization* **8** 476–505.

[17] Rockafellar R.T. 1970. *Convex Analysis*. Princeton University Press, Princeton, NJ. First edition.

[18] Stefan Weber, Christoph Schnörr, Thomas Schüle, Joachim Hornegger. 2005. Binary Tomography by Iterating Linear Programs, R. Klette, R. Kozera, L. Noakes and J. Weickert (Eds.), *Computational Imaging and Vision - Geometric Properties from Incomplete Data*. Kluwer Academic Publishers.

For Peer Review

Journal of Global Optimization manuscript No.
(will be inserted by the editor)

Robust Investment Strategies with Discrete Asset Choice Constraints Using DC Programming and DCA

Nalan GULPINAR · Hoai An LE THI · Mahdi MOEINI

Received: date / Accepted: date

Abstract In this paper, we consider worst-case (min-max) investment strategies for robust optimal portfolio problems with discrete asset choice constraints. We extend the classical Markowitz framework with discrete asset choice constraints to worst-case portfolio selection with rival scenario specifications. Robustness is ensured by considering the optimal strategy in view of multiple rival scenarios generated and evaluating the portfolio corresponding to the best performance, simultaneously with the worst-case scenario. Min-max optimization is performed over various rival scenarios of risk and return, relative to various benchmarks, and transaction costs. Discrete constraints, such as buy-in thresholds and cardinality, represent the investor's choice on the assets. The existence of discrete constraints makes the min-max problem a non-convex and NP-hard. A local deterministic optimization approach based on DC (Difference of Convex functions) programming is introduced and a DC Algorithm is developed to solve min-max mean-variance portfolio optimization problem. DCA is proved to be robust and often provides a global solution. The computational results using historical data show that DC algorithm is efficient than standard methods.

Keywords DC programming · DCA · worst-case analysis · mean-variance portfolios.

Mathematics Subject Classification (2000) 90C11 · 90C26 · 91B28

Nalan Gulpinar
Warwick Business School, University of Warwick, Coventry, CV4 7AL, UK.
E-mail: Nalan.Gulpinar@wbs.ac.uk

Hoai An LE THI
Equipe Algorithmique et Optimisation
Laboratoire Informatique Théorique et Appliquée (LITA), EA 3097
UFR MIM, Université Paul Verlaine - Metz,
Ile du Saulcy - 57045 Metz Cedex, France
Tel. : +33 (0)3 87 31 54 41,
Fax : +33 (0)3 87 31 53 09
E-mail: lethi@univ-metz.fr

Mahdi MOEINI
E-mail: moeini@univ-metz.fr

1 Introduction

A stochastic decision model under uncertainty consists of a problem of estimation and managing risk associated with random parameter. Risk management procedures are developed to reduce or prevent high losses incurred from making an incorrect decision. In particular, in finance, risk management techniques combine both rigorous and elegant theoretical results and practical effectiveness [3]. This paper is concerned with robust optimal investment strategies arising in financial portfolio management. The maximization of portfolio return for a given risk level can be formulated as a decision making problem under uncertainty.

A classical example is the single-period mean-variance optimization model in which expected portfolio return is maximized and risk measured by the variance of portfolio return is minimized. This model was first introduced by Markowitz [11], in which future uncertainty on asset returns is represented by a single forecast of return and a covariance matrix. In reality, however, it is often difficult or impossible to rely on a single forecast. There are different rival risk and return estimates and the corresponding probabilities that need to be taken into account during investment decision-making process. In addition, it is well known that return forecasts and risk estimations are inherently inaccurate. Although this type of inaccuracy can be addressed through the specification of rival scenarios, the imprecise nature of the moment forecasts needs to be tackled to reduce the risk of decision-making on the wrong scenario [5].

In this paper we extend the classical Markowitz mean-variance framework with discrete asset choice constraints to worst-case portfolio selection with rival scenario specifications. The general min-max model integrates all rival scenario issues for risk (covariance), return, alternative benchmarks and the effect of transaction costs. The robustness arises from the non-inferiority of the worst-case optimal (min-max) strategy.

In min-max optimisation, competing data sets as rival scenarios are considered simultaneously. Worst-case optimal portfolio risk and return efficient frontiers are determined in view of all rival scenarios, rather than any single scenario. Thus, min-max optimisation is more robust to the realization of worst-case scenarios than if only a single scenario is included in the optimisation model, or if multiple scenarios are used in a mean-variance framework. In this way, the investor is guarded against the risk of adopting an investment strategy based on the wrong scenario.

The investor's asset choice preferences are presented by buy-in threshold, cardinality and transaction round-lots constraints [1]. The classical Markowitz framework with discrete asset choice constraints can be formulated as a quadratic mixed integer (binary) programming problem. Therefore, the existence of discrete asset choice constraints makes the underlying problem NP-hard and non-convex. It is fact that the number of assets and rival return and risk scenarios considered is the main factor defining the size of the problem and the level of difficulty to solve the problem by the standard existing approaches.

In this paper, we introduce a local deterministic approach based on Difference of Convex functions. DC programming constitutes the backbone of non-convex programming. It was first introduced, in their preliminary form, by Pham Dinh Tao in 1985. DC programming and DCA have been extensively developed since 1994 by Le Thi Hoai An and Pham Dinh Tao and become now classic and more and more popular (see e.g. [6], [7,9], [13,14], [17] and references therein).

The DC formulation of the underlying optimization model, is obtained in terms of a difference of convex functions by an exact penalty function. This provides a general DC program in which a DC function is minimized over a closed convex set. DCA employs an iterative algorithm (called DC Algorithm) for finding a local optimal solution for the DC

program formulation of the underlying problem. The DC algorithm is based on a descent method without line search. More precisely, it is primal-dual subgradient method based on local optimality and DC duality.

In fact, it has a linear convergence for global DC programs and finite convergence for polyhedral DC programs. The special class of polyhedral DC programs, which plays a key role in non-convex optimization, possesses specific properties, from both theoretical and computational viewpoints (see, e.g.,[7,13,14]).

Although a global optimal solution is essential, in some applications like investment problems, finding a local optimal solution (close enough to the optimal) in real-time also plays an important role. Despite the local character of DCA, it provides quite often a global solution in a very short time. It has been successfully applied to many large-scale (smooth or nonsmooth) non-convex programs in various domains of applied sciences, for example *tomography* [17] and *machine learning* [12]. The numerical experiments show that DCA is more robust and efficient than standard methods (see e.g. [6], [7,9], [13,14], [17] and reference therein).

The problem under study in this paper can be casted into a quadratically constrained mixed integer (binary) programming problem. These mathematical programming problems are difficult to solve due to the big search space to prove optimality in Branch and Bound algorithms. In this paper, we use DCA to find the optimal solution for the corresponding DC program formulation of the original problem The current state-of-art optimization solver, CPLEX, is used to solve the corresponding mixed integer programming problems Our computational results show that in almost all cases DCA achieves the global optimal solution in real-time comparing to the standard approach of Branch and Bound in CPLEX solver.

The rest of the paper is organized as follows. In section 2, we present robust portfolio optimization model with discrete asset choice constraints. Section 3 is concerned with the reformulation of the underlying problem in term of DC programming and the special DCA to solve the general worst-case portfolio allocation problem. Section 4 is devoted to our computational experiments and computational results are presented. Conclusions are reported in section 5.

2 Problem Statement

In this section we first introduce general mean-variance portfolio allocation problem with discrete asset choice constraints, then present worst-case optimization model in view of rival risk and return scenarios. Single period Markowitz mean-variance optimization model with continuous asset choice constraints are described in more details in [11].

A full description of our notation is given in Table 1. All quantities in boldface represent vectors in $\mathbb{R}^n$ unless otherwise noted. The transpose of a vector or matrix will be denoted with the symbol $'$.

2.1 Portfolio Allocation Model with Discrete Constraints

Consider N number of assets to construct a portfolio. Let w_i denote weights of assets $i = 1,\cdots,N$ which sum to unity:

$$\sum_{i=1}^{N} w_i = 1.$$

N	number of investment assets;
$\mathbf{r}$	stochastic vector of return values for all assets;
$\mathbf{w}$	decision vector indicating asset balances;
$\mathbf{b}$	market benchmark;
$\Lambda_i \in \mathbb{R}^{n\times n}$	covariance matrices associated with return scenarios, for $i = 1,\ldots,I$;
$\mathbf{p}$	current portfolio position;
$\mathbf{c}_b, \mathbf{c}_s$	vector of unit transaction costs for buying and selling, respectively;
$\mathbf{x}_b, \mathbf{x}_s$	transaction variables for buying and selling;
τ	total transaction cost;
$\mathbf{z}$	the binary decision variable;
α	risk aversion parameter;
μ, ν	worst-case return and worst-case risk;
l_i, u_i	the lower and upper bounds for asset i;
C	the cardinality number.

Table 1 Notation

The basic model requires data regarding with the current position, and forecast behavior of a number of financial instruments. The current portfolio position is defined by investor's holdings. If the investor has no holdings, then $\mathbf{p} = \mathbf{0}$. Initial budget is normalised to one and allocation of the capital among the assets is represented by the following constraint.

$$\mathbf{p} + \mathbf{x}_b - \mathbf{x}_s = \mathbf{w}. \tag{1}$$

Let τ denote the transaction costs incurred by purchasing or selling assets subject to costs $\mathbf{c}_b$, $\mathbf{c}_s$. Therefore, total transaction cost of the purchase or sale is formulated as

$$\mathbf{c}_s^{'}\mathbf{x}_b + \mathbf{c}_s^{'}\mathbf{x}_s = \tau. \tag{2}$$

Let z_i denote binary decision variable which takes 1 if asset i is invested and zero, otherwise. Finite lower and upper bounds, l_i and u_i, respectively, are associated with each asset i.

The buy-in thresholds are represented by the sets of constraints

$$l_i z_i \leq w_i \leq u_i z_i, \text{ and } z_i \in \{0,1\}, \ i = 1,2,\cdots,N.$$

Cardinality constraints require buy-in thresholds to be applied. They are simply modelled by constraining the sum of the binary variables to be equal to C :

$$\sum_{i=1}^{N} z_i = C.$$

where C represents the number of assets to be in the portfolio.

In mean-variance portfolio optimization framework, there are two conflicting objectives. While the expected portfolio return $\mathbf{p} = \mathbf{0}$ on investment $E[R_p] = (\mathbf{w}-\mathbf{b})^{'}\mathbf{r}$ is maximized, the expected portfolio risk is minimized. Expected portfolio risk is measured as the variance of the portfolio return relative to the benchmark $\overline{\mathbf{w}}$ and formulated as $(\mathbf{w}-\mathbf{b})^{'}\Lambda_i(\mathbf{w}-\mathbf{b})$.

Trade-off between the two objectives depends on the level of the risk that the investor prefers. Therefore, single period mean-variance optimization problem can be formulated as the following mixed integer (binary) quadratic programming problem, (P_s).

$$\begin{aligned} (P_s): \quad & \min \ \alpha(\mathbf{w}-\mathbf{b})^{'}\Lambda(\mathbf{w}-\mathbf{b}) \ - \ (1-\alpha)(\mathbf{w}-\mathbf{b})^{'}\mathbf{r} + \tau \\ & \text{subject to} \end{aligned}$$

$$
\begin{aligned}
\sum_{i=1}^{N} w_i &= 1, \\
\mathbf{p}+\mathbf{x}_b-\mathbf{x}_s &= \mathbf{w}, \\
\mathbf{c}_b^{'}\mathbf{x}_b+\mathbf{c}_s^{'}\mathbf{x}_s &= \tau, \\
\sum_{i=1}^{N} z_i &= C, \\
l_i z_i \le w_i &\le u_i z_i, \qquad i=1,2,\cdots,N \\
z_i &\in \{0,1\}, \qquad i=1,2,\cdots,N \\
\mathbf{x}_b,\mathbf{x}_s &\ge \mathbf{0}.
\end{aligned}
$$

where $\alpha \in [0,1]$ represents risk aversion. Notice that this model only requires single benchmark, rival risk and return scenarios as input data $\mathbf{b},\mathbf{r},\Lambda$. This model will be extended to worst-case design model with multiple rival scenarios in the next section.

2.2 Worst-case Portfolio Optimization Model

The mean-variance portfolio allocation model under discrete asset choice constraints requires a single return and covariance matrix forecasts and the current and benchmark portfolio positions. In reality, however, it is impossible to make a decision in view of single return and risk scenario. Moreover, future uncertainty must be represented by a number of scenarios so that the investor can guard himself against making a decision on a wrong scenario.

Let I, J, and K denote total number of covariance matrices, return vectors and benchmark portfolios. We assume that uncertainty on asset returns represented by J number of return scenarios and the corresponding probabilities at time period one.

A compact representation of worst-case portfolio optimization problem is formulated as follows:

$$
\min_{\mathbf{w}\in X}\left\{\alpha\cdot\max_{i,k}\{(\mathbf{w}-\mathbf{b}_k)^T\Lambda_i(\mathbf{w}-\mathbf{b}_k)\}-(1-\alpha)\min_{j,k}\{(\mathbf{w}-\mathbf{b}_k)^T\mathbf{r}_j-t(\mathbf{w})\}\right\}. \tag{3}
$$

In this formulation function t represents the transaction costs incurred by moving to strategy $\mathbf{w}$ from current position $\mathbf{p}$, subject to costs $\mathbf{c}_b$, $\mathbf{c}_s$. The level of risk-aversion is determined by α which varies between $\alpha=0$ (risk-seeking) and $\alpha=1$ (risk-averse) and identifies the range of efficient investment strategies

In order to solve (3) we reformulate it as a quadratically constrained binary integer program, (P_w), presented below.

$$
\begin{aligned}
(P_w): \qquad & \min\ \alpha\nu-(1-\alpha)\mu \\
& \text{subject to} \\
& (\mathbf{w}-\mathbf{b}_k)^T\Lambda_i(\mathbf{w}-\mathbf{b}_k) \le \nu, \qquad i=1,\ldots,I,\ k=1,\ldots,K \\
& (\mathbf{w}-\mathbf{b}_k)^T\mathbf{r}_j-\tau \ge \mu, \qquad j=1,\ldots,J,\ k=1,\ldots,K \\
& \mathbf{p}+\mathbf{x}_b-\mathbf{x}_s = \mathbf{w}, \\
& \mathbf{c}_b^T\mathbf{x}_b+\mathbf{c}_s^T\mathbf{x}_s = \tau, \\
& \sum_{i=1}^{N} w_i = 1,
\end{aligned}
$$

$$\begin{aligned}
\sum_{i=1}^{N} z_i &= C, \\
l_i z_i \le w_i &\le u_i z_i, \qquad i = 1,\ldots,N \\
z_i &\in \{0,1\}, \qquad i = 1,\ldots,N \\
\mathbf{x}_b, \mathbf{x}_s &\ge \mathbf{0}.
\end{aligned}$$

It is worthwhile to mention that not all of the data above must be provided; this general problem is well-defined analogues for many subsets of the data. The problem size in terms of (continuous and binary) decision variables and (linear and quadratic) constraints depends on the number of assets, risk and return rival scenarios and benchmark portfolios considered.

3 General DC Programs and Solution Methods

Let $\Gamma_0(\mathbb{R}^n)$ denote the convex cone of all lower semi-continuous proper convex functions on $\mathbb{R}^n$. Consider the following primal DC program

$$(P_{dc}): \qquad \beta_p = \inf\{f(x) := g(x) - h(x) \mid x \in \mathbb{R}^n\}, \tag{4}$$

where $g, h \in \Gamma_0(\mathbb{R}^n)$. The function f is called a DC function. The difference between functions g and h, $g-h$, defines DC decomposition of f while convex functions g and h are DC components of f.

Let C be a convex set. Then problem

$$\inf\{f(x) := g(x) - h(x) \mid x \in C\}, \tag{5}$$

can be transformed to an unconstrained DC program by using the indicator function on C, i.e.,

$$\inf\{f(x) := g(x) - h(x) \mid x \in \mathbb{R}^n\}, \tag{6}$$

where $g := k + \chi_C$.

Let $g^*(y) := \sup\{\langle x,y\rangle - g(x) : x \in \mathbb{R}^n\}$ be the conjugate function of g. Then, the following program is called the dual program of P_{dc}:

$$(D_{dc}): \qquad \beta_d = \inf\{h^*(y) - g^*(y \mid y \in \mathbb{R}^n\}. \tag{7}$$

Under the natural convention in DC programming that is $+\infty - (+\infty) = +\infty$, and by using the fact that every function $h \in \Gamma_0(\mathbb{R}^n)$ is characterized as a pointwise supremum of a collection of affine functions, say

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

it can be proved that $\beta_p = \beta_d$. There is perfect symmetry between primal and dual DC programs; dual of (D_{dc}) gives (P_{dc}).

Recall that, for $\theta \in \Gamma_0(\mathbb{R}^n)$ and $x_0 \in dom\ \theta := \{x \in \mathbb{R}^n | \theta(x_0) < +\infty\}$, the sub-differential of θ at x_0, $\partial\theta(x_0)$, is defined as

$$\partial\theta(x_0) := \{y \in \mathbb{R}^n : \theta(x) \ge \theta(x_0) + \langle x - x_0, y\rangle, \forall x \in \mathbb{R}^n\} \tag{8}$$

which is a closed convex set in $\mathbb{R}^n$. It generalizes the derivative in the sense that θ is differentiable at x_0 if and only if $\partial\theta(x_0)$ is reduced to a singleton which is exactly $\nabla\theta(x_0)$.

The necessary local optimality condition for the primal DC program, (P_{dc}), is

$$\partial g(x^*) \supset \partial h(x^*). \tag{9}$$

A point that x^* verifies the condition $\partial h(x^*) \cap \partial g(x^*) \neq \emptyset$ is called a critical point of $g-h$. The condition (9) is also sufficient for many important classes of DC programs, for example, when function f is locally convex at x^* ([8,9,13]).

The transformation of global solutions between (P_{dc}) and (D_{dc}) is expressed by:

$$[\cup_{y^* \in \mathscr{D}} \partial g^*(y^*)] \subset \mathscr{P}, \quad [\cup_{x^* \in \mathscr{P}} \partial h(x^*)] \subset \mathscr{D}, \tag{10}$$

where $\mathscr{P}$ and $\mathscr{D}$ denote the solution sets of (P_{dc}) and (D_{dc}) respectively. Under certain technical conditions, this property also holds also for the local solutions of (P_{dc}) and (D_{dc}), for example see [7,9,13,14] for more information.

Based on local optimality conditions and duality in DC programming, the DC algorithm consists in the construction of two sequences $\{x^k\}$ and $\{y^k\}$, consisting of candidates to be optimal solutions of primal and dual programs, respectively, such that the sequences $\{g(x^k)-h(x^k)\}$ and $\{h^*(y^k)-g^*(y^k)\}$ are decreasing. In addition, $\{x^k\}$ and $\{y^k\}$) converge to a primal and dual feasible solutions $\widetilde{x}$ and $\widetilde{y}$, respectively. They verify local optimality conditions

$$\widetilde{x} \in \partial g^*(\widetilde{y}), \quad \widetilde{y} \in \partial h(\widetilde{x}). \tag{11}$$

The DC algorithm then yields the next scheme:

$$y^k \in \partial h(x^k); \quad x^{k+1} \in \partial g^*(y^k). \tag{12}$$

In other words, these two sequences $\{x^k\}$ and $\{y^k\}$ are determined in the way that x^{k+1} and y^k are a solution of the convex primal program (P_k) and dual program (D_k), respectively. These are defined as

$$(P_k): \qquad \inf\{g(x)-h(x^k)-\langle x-x^k, y^k\rangle \mid x \in \mathbb{R}^n\}, \tag{13}$$

$$(D_k): \quad \inf\{h^*(y)-g^*(y^{k-1})-\langle y-y^{k-1}, x^k\rangle \mid y \in \mathbb{R}^n\}. \tag{14}$$

At each iteration, the DC algorithm performs a double linearization with use of the subgradients of h and g^*. In fact, at each iteration, one replaces in the primal DC program, (P_{dc}), the second component h by its affine minorization $h_k(x) := h(x^k)+\langle x-x^k, y^k\rangle$ at a neighbourhood of x^k to construct the convex program (P_k) whose the solution set is nothing but $\partial g^*(y^k)$. Likewise, the second DC component g^* of the dual DC program, (D_{dc}), is replaced by its affine minorization $g_k^*(y) := g^*(y^k)+\langle y-y^k, x^{k+1}\rangle$ at a neighbourhood of y^k to obtain the convex program (D_k) whose $\partial h(x^{k+1})$ is the solution set. It is worth noting that the DC algorithm works with the convex DC components g and h but not the DC function f itself, for example, see [7,9,13,14].

Moreover, a DC function f has infinitely many DC decompositions which have crucial impacts on the performance of the DC algorithm in terms of speed of convergence, robustness, efficiency, and globality of computed solutions. Convergence properties of the DC algorithm and its theoretical basis are described in [7,9,13]. However, it is worthwhile to summarize the following properties for the sake of completeness;

– DCA is a descent method (*without line search*). The sequences $\{g(x^k)-h(x^k)\}$ and $\{h^*(y^k)-g^*(y^k)\}$ are decreasing such that

$$g(x^{k+1})-h(x^{k+1}) \leq h^*(y^k)-g^*(y^k) \leq g(x^k)-h(x^k).$$

- If $g(x^{k+1})-h(x^{k+1})=g(x^k)-h(x^k)$ then x^k is a critical point of $g-h$ and y^k is a critical point of h^*-g^*. In this case, the DC algorithm terminates at k^{th} iteration.

- If the optimal value α of problem (P_{dc}) is finite and the infinite sequences $\{x^k\}$ and $\{y^k\}$ are bounded, then every limit point $\widetilde{x}$ (resp. $\widetilde{y}$) of the sequence $\{x^k\}$ (resp. $\{y^k\}$) is a critical point of $g-h$ (resp. h^*-g^*).

- The DC algorithm has a linear convergence for DC programs. Especially, for polyhedral DC programs the sequences $\{x^k\}$ and $\{y^k\}$ contain finitely many elements and the algorithm convergences to the optimal solution in a finite number of iterations.

4 DC Programming Model for Worst-case Robust Investment Strategies

We consider a new approach based on DC program to solve quadratically constrained binary programming problem, (P_w), presented in the previous section. The DCA requires a reformulation of minimax problem so that the objective function represented by the difference of two convex functions. Then the original problem becomes a DC program in which the DC function is minimized subject to linear constraints. In this section, we first introduce a DC reformulation of the worst-case mean-variance portfolio optimization problem and then present an algorithm to solve the corresponding DC program of (P_w).

4.1 DC Program formulation for (P_w)

Let $X \subset \mathbb{R}^{4N+3}$ consist of all points $(\mathbf{w},\mathbf{x}_b,\mathbf{x}_s,\mathbf{z},\mu,\nu,\tau) \in \mathbb{R}^{4N+3}$ satisfying all constraints of the problem, (P_w), apart from the binary constraints. We define a concave function

$$f(\mathbf{w},\mathbf{x}_b,\mathbf{x}_s,\mathbf{z},\mu,\nu,\tau) := \sum_{i=1}^{N} z_i(1-z_i),$$

with nonnegative values on X. The feasible region of the problem (P_w) can be written as

$$\begin{aligned}\{(\mathbf{w},\mathbf{x}_b,\mathbf{x}_s,\mathbf{z},\mu,\nu,\tau) \in X,\ z_i \in \{0,1\},\ i=1,\ldots,N\}\\ = \{(\mathbf{w},\mathbf{x}_b,\mathbf{x}_s,\mathbf{z},\mu,\nu,\tau) \in X,\ f((\mathbf{w},\mathbf{x}_b,\mathbf{x}_s,\mathbf{z},\mu,\nu,\tau)=0\}\\ = \{(\mathbf{w},\mathbf{x}_b,\mathbf{x}_s,\mathbf{z},\mu,\nu,\tau) \in X,\ f(\mathbf{w},\mathbf{x}_b,\mathbf{x}_s,\mathbf{z},\mu,\nu,\tau) \le 0\}.\end{aligned}$$

Therefore, the problem (P_w) becomes

$$\min\{V(\mu,\nu) := \alpha\nu-(1-\alpha)\mu \mid (\mathbf{w},\mathbf{x}_b,\mathbf{x}_s,\mathbf{z},\mu,\nu,\tau) \in X, \quad f(\mathbf{w},\mathbf{x}_b,\mathbf{x}_s,\mathbf{z},\mu,\nu,\tau) \le 0\}. \tag{15}$$

Since the objective function $V(\mu,\nu)$ is convex and X is bounded polyhedral convex set, there is $\sigma_0 \ge 0$ such that for any $\sigma > \sigma_0$, the program (15) is equivalent to

$$\min\{F(\mathbf{w},\mathbf{x}_b,\mathbf{x}_s,\mathbf{z},\mu,\nu,\tau) := \alpha\nu-(1-\alpha)\mu+\sigma\sum_{i=1}^{N} z_i(1-z_i) \mid (\mathbf{w},\mathbf{x}_b,\mathbf{x}_s,\mathbf{z},\mu,\nu,\tau) \in X\}. \tag{16}$$

For the proof and more detailed information, the reader is referred to [10]. The function F is convex in the variables $\mathbf{w},\mathbf{x}_b,\mathbf{x}_s,\mu,\nu,\tau$ and concave in the variables $\mathbf{z}$. Consequently,

this shows that F is a DC function. Therefore, DC formulation of the problem (16) can be rewritten as

$$\min\{g(\mathbf{w},\mathbf{x}_b,\mathbf{x}_s,\mathbf{z},\mu,\nu,\tau)-h(\mathbf{w},\mathbf{x}_b,\mathbf{x}_s,\mathbf{z},\mu,\nu,\tau)|(\mathbf{w},\mathbf{x}_b,\mathbf{x}_s,\mathbf{z},\mu,\nu,\tau)\in \mathbb{R}^{4N+3}\}, \quad (17)$$

where

$$g(\mathbf{w},\mathbf{x}_b,\mathbf{x}_s,\mathbf{z},\mu,\nu,\tau) := \chi_A(\mathbf{w},\mathbf{x}_b,\mathbf{x}_s,\mathbf{z},\mu,\nu,\tau), \quad (18)$$

$$h(\mathbf{w},\mathbf{x}_b,\mathbf{x}_s,\mathbf{z},\mu,\nu,\tau) := (1-\alpha)\mu-\alpha\nu+\sigma\sum_{i=1}^{N} z_i(z_i-1), \quad (19)$$

and the indicator function on X, χ_A, is defined as

$$\chi_A(\mathbf{w},\mathbf{x}_b,\mathbf{x}_s,\mathbf{z},\mu,\nu,\tau) = \begin{cases} 0 & if \quad (\mathbf{w},\mathbf{x}_b,\mathbf{x}_s,\mathbf{z},\mu,\nu,\tau)\in X, \\ \infty & otherwise. \end{cases} \quad (20)$$

Note that 17 is a DC polyhedral program, because the function g given in 18 is polyhedral.

4.2 DC Algorithm for Worst-case Optimization Problems

We develop a new algorithm based on DC program for solving worst-case optimization problem presented in (16). This involves two main stages. In the first stage, we compute a sub-gradient of the function h defined in 19

From the definition of h, we have

$$\mathbf{u}^l \in \partial h(\mathbf{w}^l,\mathbf{x}_b^l,\mathbf{x}_s^l,\mathbf{z}^l,\mu^l,\nu^l,\tau^l),$$

and

$$u_i^l = \begin{cases} \sigma(2z_{i-3N}^l-1) & : if \quad i=3N+1,\ldots,4N, \\ (1-\alpha) & : if \quad i=4N+1, \\ -\alpha & : if \quad i=4N+2, \\ 0 & : otherwise. \end{cases}$$

The second stage involves computing an optimal solution $(\mathbf{w}^{l+1},\mathbf{x}_b^{l+1},\mathbf{x}_s^{l+1},\mathbf{z}^{l+1},\mu^{l+1},\nu^{l+1},\tau^{l+1})$ for the following linear program

$$\min\{-\langle(\mathbf{w},\mathbf{x}_b,\mathbf{x}_s,\mathbf{z},\mu,\nu,\tau),\mathbf{u}^l\rangle : (\mathbf{w},\mathbf{x}_b,\mathbf{x}_s,\mathbf{z},\mu,\nu,\tau)\in X\}. \quad (21)$$

The main steps of the DC algorithm for solving worst-case optimal portfolio allocation problem under discrete asset choice constraints is presented as follows;.

DC Algorithm (DCA)

– **Step 1: Initialization**:

Let ε be a sufficiently small positive number. Set $l=0$ and

$$(\mathbf{w}^0,\mathbf{x}_b^0,\mathbf{x}_s^0,\mathbf{z}^0,\mu^0,\nu^0,\tau^0)\in R^{4N+3}.$$

– **Step 2: Iteration**:
Set

$$u_i^l = \begin{cases} \sigma(2z_{i-3N}^l - 1) & : if \quad i = 3N+1,\ldots,4N, \\ (1-\alpha) & : if \quad i = 4N+1, \\ -\alpha & : if \quad i = 4N+2, \\ 0 & : otherwise. \end{cases}$$

Solve the following linear program

$$\min\{-\langle(\mathbf{w},\mathbf{x}_b,\mathbf{x}_s,\mathbf{z},\mu,\nu,\tau),\mathbf{u}^l\rangle \mid (\mathbf{w},\mathbf{x}_b,\mathbf{x}_s,\mathbf{z},\mu,\nu,\tau) \in X\}$$

and obtain $(\mathbf{w}^{l+1},\mathbf{x}_b^{l+1},\mathbf{x}_s^{l+1},\mathbf{z}^{l+1},\mu^{l+1},\nu^{l+1},\tau^{l+1})$.

– **Step 3: Termination**:
If

$$\| \mathbf{w}^{l+1} - \mathbf{w}^{l+1} \| + \| \mathbf{x}_b^{l+1} - \mathbf{x}_b^{l+1} \| + \| \mathbf{x}_s^{l+1} - \mathbf{x}_s^{l+1} \| + \| \mathbf{z}^{l+1} - \mathbf{z}^{l+1} \| + \| \mu^{l+1} - \mu^{l+1} \| + \| \nu^{l+1} - \nu^{l+1} \| + \| \tau^{l+1} - \tau^{l+1} \| \leq \varepsilon,$$

then terminate the algorithm by concluding that $(\mathbf{w}^{l+1},\mathbf{x}_b^{l+1},\mathbf{x}_s^{l+1},\mathbf{z}^{l+1},\mu^{l+1},\nu^{l+1},\tau^{l+1})$ is a solution.
Otherwise, set $l = l+1$ and go to Step 2.

5 Computational Experiments

5.1 Design of Experiments: Data and Parameters

The DC algorithm for solving the worst-case mean-variance portfolio allocation problem with discrete constraints is implemented in C++. The linear program relaxations are solved by the state-of-the solver CPLEX (Version 10.1). The performance of the DC algorithm is compared with the standard mixed integer quadratic program integrated in the CPLEX solver. All computational experiments were run on a Pentium IV $3GHz$ of $1.00GB$ RAM.

A good starting point plays an important role on the performance of the DC algorithm. Finding a good starting point is an open question and depends on the underlying application. In our computational experiments, the optimal solution of the relaxed problem for (P_w) is considered as an initial point for the DC algorithm. The relaxed problem of (P_w) which is basically obtained by replacing the binary constraints $z_i \in \{0,1\}$ by their continuous relaxations, $0 \leq z_i \leq 1$.

The DC algorithm also requires the parameter $\sigma_0 \geq 0$ such that for any $\sigma > \sigma_0$, the optimization problem (P_w) is equivalent to program (16). For our computational experiments, $\sigma = 1.0$ is selected.

For the worst-case mean-variance portfolio allocation model, single equally weighted portfolio is considered as a benchmark portfolio. Therefore, for $K = 1$, $b_i = 1/N$ is specified for all assets $i = 1,2,\cdots,N$. The current portfolio position defining the current holdings of the investor is assumed to be zero, that is $\mathbf{p} = 0$. Transaction costs for buying and selling is chosen as 0.1%.

We have randomly selected 98 stocks from Yahoo! finance page. The historical monthly price data from 1997 until 2007 is used to generate the rival risk and return scenarios. The specification of rival covariance matrices can be realized by observing the data during different periods in the past. Dividing a given observation period into a number of subperiods and

measuring volatility in each one is an effective way of estimating risk scenarios. It is well known that the estimates corresponding to each subperiod can be substantially different, and employing the worst-case scenario arising from this consideration yields a robust strategy.

Rival return scenarios are generated by clustering of randomized simulations. To parameterize the simulations, recent histories of each of the asset (classes) in question are fit to an exponential growth curve. The resulting mean growth rates γ form the central projection for the next time period. Matrix Λ is measured as the covariance of residuals from the fitted curves. A large number of quasi-random vectors are generated from a lognormal (γ, Λ) distribution using low-discrepancy Sobol sequences. The simulated scenarios are then clustered into the groups, and the most central scenario from clusters is identified as a return scenario. In the case of single return scenario, no simulation is done; the central projection γ is used as the only return scenario.

Another approach to generate return scenarios is based on the moment matching optimization. This approach chooses return values for the various scenarios which most closely match the statistics (first four moments) measured from historical data. For further details of both the simulation and optimization based approaches for generating scenario tree, the reader is referred to [2].

In our computational experiments, we generated three and ten rival return scenarios using simulation based method and three covariance matrices using time series of randomly selected assets.

5.2 Computational Results

The performance of the DCA presented in the previous section is measured by the CPU time taken to solve the mixed integer (binary) quadratic optimization problem and the worst-case characteristics of the local optimal risk-return efficient portfolios. Efficiency of the DCA is compared with state-of-the-art commercial solver, CPLEX. Time limit of 1200 seconds is set up to solve the problem with the CPLEX solver.

We first investigate the impact of the cardinality number on the performance of DCA. For this experiment, we are concerned with computing the worst-case risk-averse efficient portfolios ($\alpha = 1$). The cardinality number C varies between 5 and 20. These instances are selected since both DCA and the CPLEX solver have difficulty to find the solution for the worst-case portfolio allocation problem. The CPU time (in second) taken to solve the worst-case mean-variance optimization problem (P_w) by the CPLEX solver and the corresponding DC program using the DCA are presented in Table 2.

In this table, for each cardinality parameter C, the corresponding worst-case expected portfolio return (WC-Return) and risk (WC-Risk) provided by CPLEX and DCA are also presented.

These results show that the DCA finds the local optimal solution in between 0.6 and 1 seconds with the comparable quality of solutions in terms of worst-case risk and return with the CPLEX solver. However, for none of the cases, the CPLEX solver does prove the optimal solution in given time limit.

For the rest of our experiments, the cardinality number is fixed as ten. The risk aversion parameter, α, varies between 0 and 1. Recall that the highest worst-case risk-return portfolio is achieved when $\alpha = 0$ and the minimum worst-case risk-return efficient portfolio is obtained when $\alpha = 1$. Table 3 presents the results (in terms of CPU time and optimal values) obtained by solving (P_w) and the corresponding DC program with single and multi rival risk and return scenarios using the CPLEX solver and the DCA.

Table 2 Results obtained by CPLEX and DCA for different cardinality parameters

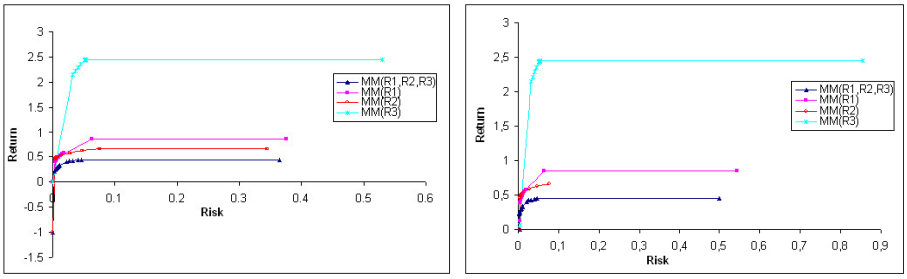
	CPLEX Solver			DCA		
C	TIME	WC-Risk	WC-Return	TIME	WC-Risk	WC-Return
5	1200.250	0.001105	-1.000	0.765	0.002725	0.023861
6	1200.062	0.000867	-1.000	0.781	0.004050	0.013534
7	1200.079	0.000725	0.000	0.985	0.003968	0.025722
8	1200.062	0.000590	-1.000	0.797	0.003824	0.040129
9	1200.062	0.000578	-1.000	0.891	0.003704	0.030346
10	1200.063	0.000412	0.000	0.796	0.003279	0.000029
11	1205.141	0.000455	-1.000	0.984	0.001906	0.014492
12	1206.547	0.000409	-1.000	0.875	0.001894	0.011593
13	1200.469	0.000348	0.000	0.922	0.001764	0.003485
14	1203.516	0.000244	-1.000	0.672	0.001585	0.003696
15	1204.359	0.000268	0.000	0.750	0.001551	-0.006994
16	1200.046	0.000200	-1.000	0.890	0.001052	0.016135
17	1201.062	0.000194	-1.000	0.765	0.000786	0.011584
18	1200.047	0.000225	-1.000	0.625	0.000448	-0.008982
19	1203.843	0.000200	-1.000	0.687	0.000395	-0.029241
20	1200.750	0.000191	-1.000	0.688	0.000619	-0.045734

Table 3 Results obtained with single and multi rival scenarios

	Single Rival Scenario				Multi Rival Scenarios			
	CPLEX Solver		DCA		CPLEX Solver		DCA	
α	OPT-VAL	TIME	OPT-VAL	TIME	OPT-VAL	TIME	OPT-VAL	TIME
1.0	0.000586	1200.063	0.003279	0.781	0.000532	1200.079	0.003279	0.782
0.96	-0.011740	126.297	-0.011813	0.625	-0.006169	71.954	-0.006112	0.703
0.95	-0.015946	0.500	-0.015946	0.532	-0.008485	3.859	-0.008479	0.516
0.94	-0.020428	0.438	-0.020428	0.625	-0.011030	0.485	-0.011069	0.547
0.93	-0.025249	0.859	-0.025249	0.609	-0.013859	0.922	-0.013859	0.656
0.92	-0.030411	0.453	-0.030411	0.625	-0.016877	0.531	-0.016877	0.562
0.91	-0.035917	0.844	-0.035917	0.453	-0.020105	0.593	-0.020105	0.703
0.9	-0.041749	0.906	-0.041749	0.484	-0.023393	0.625	-0.023478	0.594
0.8	-0.118714	0.203	-0.118714	0.422	-0.063321	0.563	-0.063321	0.375
0.7	-0.209630	0.172	-0.209630	0.375	-0.107344	0.672	-0.107344	0.391
0.6	-0.300572	0.172	-0.300572	0.343	-0.152804	0.203	-0.152804	0.453
0.5	-0.391536	0.485	-0.391536	0.282	-0.199755	0.234	-0.199755	0.328
0.4	-0.482551	0.188	-0.482551	0.344	-0.248653	0.203	-0.248653	0.343
0.3	-0.573571	0.469	-0.573571	0.297	-0.297840	0.203	-0.297840	0.344
0.2	-0.664618	0.172	-0.664618	0.281	-0.347028	0.203	-0.347028	0.359
0.1	-0.755666	0.187	-0.755666	0.281	-0.396215	0.203	-0.396215	0.313
0.0	-0.846713	0.187	0.846713	0.297	-0.445402	0.203	-0.445402	0.328
Time-range		0.1–1200		0.2–0.8		0.2–1200		0.3–0.8

These results show that the CPU time taken to solve the problems is very much dependent on the risk aversion parameter. When the risk aversion approaches to 1, the solution time increases. In cases where $\alpha = 0.96$ and 1.0, the DCA outperforms the CPLEX solver. However, the optimal values at these cases are quite close to each other. For $\alpha \leq 0.95$, the CPLEX solver taking to solve each problem varies in the range of 0.1 and 0.9 seconds, while

CPU time for the DCA is between 0.2 and 0.7 seconds. We are also concerned with the per-

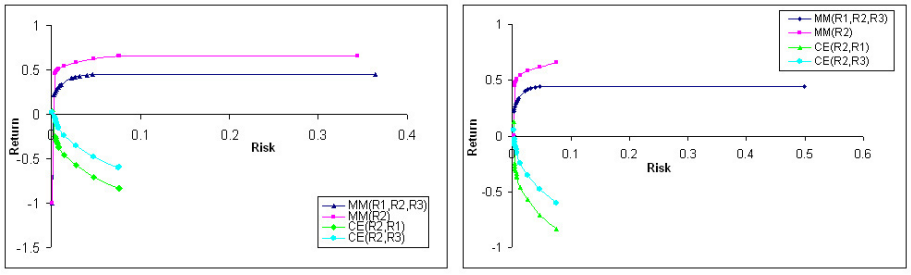


(a) CPLEX Solver (b) DCA

Fig. 1 Min-max strategies over single rival return scenario and multiple rival return scenarios

formance of the minimax strategies in terms of worst-case risk and return efficient frontiers. One of the important properties of minimax approach is presented in Figure 1. This shows that the min-max strategy in view of all rival scenarios creates a lower bound; i.e. any other scenario apart from min-max will improve the portfolio performance.

The worst-case efficient frontier at the bottom is obtained in view of all rival scenarios whereas the other min-max strategies consider a single scenario. This shows that minimax strategy in view of all rival scenarios creates a lower bound; i.e. any other scenario apart from minimax will improve the portfolio performance. In other words, the min-max strategy provides a guaranteed portfolio performance. In Figure 2, we present the cross evaluation of

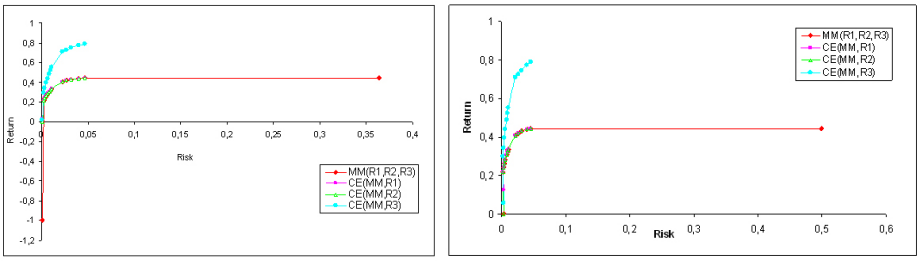


(a) CPLEX Solver (b) DCA

Fig. 2 Cross evaluation of single return scenario on others

a single rival return scenario over the other two scenarios. This shows that if an investment strategy based on a single rival scenario is decided to be implemented and any of other scenarios is materialized realized in the future, then the investor's portfolio position will be worse than the min-max strategy based on all rival scenarios.

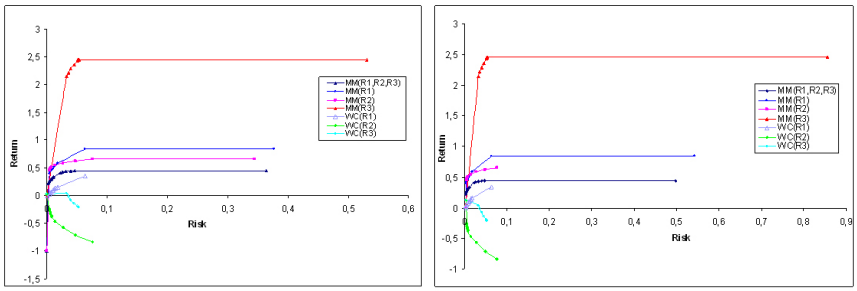
The non-inferiority of the min-max strategy is presented in Figure 3. As it can be easily seen from this Figure, if the minimax strategy is considered and any other scenario is realized in the future, then the portfolio return will be either better than or equal to the worst-case strategy itself. We also investigated what actually would happen if the worst-case of the rival return scenarios with respect to optimized portfolio was actually realized. Figure 4 displays



(a) CPLEX Solver (b) DCA

Fig. 3 Non-inferiority of min-max

the benefit of using the min-max optimization. The top three curves in Figure 4 represent the efficient frontiers obtained from optimizing with each of the rival return scenarios alone. The middle curve is the min-max strategy obtained by optimization models with all rival scenarios. The curves at the bottom show the worst-case of the rival return scenarios with respect to optimized portfolio. The min-max model constructs an optimal portfolio simul-



(a) CPLEX Solver (b) DCA

Fig. 4 Worst-case analysis

taneously with the worst-case scenario. Hence, the worst-case optimal investment strategy has the best lower bound performance which can only be improved when any scenario other than the worst-case is realized. Therefore, non-inferiority of min-max optimization ensures the robustness of the strategy.

6 Conclusion

In this paper, we present a new approach based on DC programming and DCA to solve worst-case single-period mean-variance optimization problem with discrete asset choice constraints in view of multiple rival risk and rival return scenarios. Our computational experiments show that DCA outperforms in cases where the standard solvers cannot provide any information in real-time. The issue of inaccuracy in asset return forecasting and risk estimation plays an important role on the portfolio performance in finance. Our computational experiments illustrate that rival scenarios need to be taken into account during investment

decision-making process. Worst-case analysis is an alternative approach to provide a robust mean-variance portfolio optimization framework. The advantage of worst-case analysis is to provide a guaranteed performance that will improve if any scenario other than the worst-case occurs.

References

1. Chang, T.J, N. Meade, J.B. Beasley and Y.M. Sharaiha, Heuristics for Cardinality Constrained Portfolio Optimization, Computers and Operations Research, **27**, 1271-1302 (2000).
2. Gulpinar, N., B. Rustem, and R. Settergren, Simulation and Optimisation Approaches to Scenario Generation, **28** (1-2), 1291-1315 (2004).
3. Gulpinar, N., B. Rustem, Worst-case Optimal Robust Decisions for Multi-period Portfolio Optimization, European Journal of Operational Research, **183** (3), 981-1000 (2007).
4. Gulpinar, N., B. Rustem, S. Zakovic, Stochastic Optimization and Worst-case Decisions, Eds. D. Grundel, R. Murphey, P. Pardalos, O. Prokopyev, Cooperative Systems Control and Optimization, Lecture Notes in Economics and Mathematical Systems, **588**, 317-338 (2007).
5. Fair, R. C., Specification Estimation and Analysis of Macroeconometric Models, Harvard University Press, Cambridge, MA, (1984).
6. Harrington J.E., B.F. Hobbs, J.S. Pang, A. Liu, G. Roch, Collusive game solutions via optimisation, **104** (1-2), 407-435 (2005).
7. Le Thi, H.A., Contribution à l'optimisation non convexe et l'optimisation globale: Théorie, Algorithmes et Applications, Habilitation à Diriger des Recherches, Université de Rouen, (1997).
8. Le Thi H.A. and Pham Dinh T., A continuous approach for globally solving linearly constrained quadratic zero-one programming problems, Optimization, **50**(1-2), 93-120 (2001).
9. Le Thi H.A. and Pham Dinh T., The DC (difference of convex functions) Programming and DCA revisited with DC models of real world non convex optimization problems, Annals of Operations Research, **133**, 23-46 (2005).
10. Le Thi H.A., Pham Dinh T., Huynh V.N., Exact Penalty Techniques in DC Programming, Research Report, LMI, National Institute for Applied Sciences - Rouen, France, July, (2005).
11. Markowitz H., Portfolio selection, Journal of Finance, **7**, 77–91 (1952).
12. Neumann J., Schnörr C., Steidl G., Combined SVM-based feature selection and classification, Machine Learning, **61**, 129-150 (2005).
13. Pham Dinh T. and Le Thi H.A, Convex analysis approach to d.c. programming: Theory, Algorithms and Applications, Acta Mathematica Vietnamica, dedicated to Professor Hoang Tuy on the occasion of his 70th birthday, **22**(1), 289-355 (1997).
14. Pham Dinh T. and Le Thi H.A, DC optimization algorithms for solving the trust region subproblem, SIAM J. Optimization, **8**, 476-505 (1998).
15. Rockafellar R.T., Convex Analysis Princeton University Press, Princeton, First edition, (1970).
16. Rustem, B., R. Becker, W. Marty, Robust Min-Max Portfolio strategies for Rival Forecast and Risk Scenarios, Journal of Economic Dynamics and Control, **24**, 1591-1623 (2000).
17. Stefan Weber, Christoph Schnörr, Thomas Schüle, Joachim Hornegger, Binary Tomography by Iterating Linear Programs, R. Klette, R. Kozera, L. Noakes and J. Weickert (Eds.), Computational Imaging and Vision - Geometric Properties from Incomplete Data Kluwer Academic Publishers, (2005).

Portfolio Selection Under Buy-In Threshold Constraints Using DC Programming and DCA*

Hoai An LE THI[1], **Mahdi MOEINI**[2]

[1] Université Paul Verlaine - Metz, France (lethi@univ-metz.fr)
[2]Université Paul Verlaine - Metz, France (moeini@univ-metz.fr)

ABSTRACT

In matter of Portfolio selection, we consider a generalization of the Markowitz Mean-Variance model which includes *buy-in threshold constraints*. These constraints limit the amount of capital to be invested in each asset and prevent very small investments in any asset. The new model can be converted into a NP-hard mixed integer quadratic programming problem. The purpose of this paper is to investigate a continuous approach based on DC programming and DCA (DC Algorithms) for solving this new model. DCA is a local continuous approach to solve a wide variety of nonconvex programs for which it provided quite often a global solution and proved to be more robust and efficient than standard methods. Preliminary comparative results of DCA and a classical Branch-and-Bound algorithm will be presented. These results show that DCA is an efficient and promising approach for the considered portfolio selection problem.
Keywords: Portfolio selection, DC programming, DCA, Branch-and-Bound.

1. Introduction

In the portfolio selection problem, given a set of available securities or assets, we want to find out the optimum way of investing a particular amount of money in these assets. Each way of the different ways to diversify this money between the several assets is called a portfolio [3]. For solving this portfolio problem, Markowitz [10, 11] has set up a quantitative frame-work. Markowitz's model which is called Mean-Variance model assumes that the return on a portfolio of assets can be completely described by the expected return and the variance of returns (risk) between these assets. For a particular universe of assets, the set of portfolios of assets that offer the minimum risk for a given level of return is the set of efficient portfolios. These portfolios can be found by convex quadratic programs (QP). But the Markowitz's standard model, does not contain some practical constraints. For example, the standard Mean-Variance model has not got any bounding constraints limiting the amount of money to be invested in each asset neither prevents very small amounts of investments in each asset. This kind of constraints is very useful in practice and is called *buy-in threshold constraints* [1]. In order to overcome these inconveniences, the standard model can be generalized to include these constraints.

In this paper we focus on solving the problem of portfolio selection under buy-in threshold constraints. We investigate a local deterministic approach based on DC (Difference of Convex functions) programming and DCA (DC Algorithms) that were introduced by Pham Dinh Tao in their preliminary form in 1985. They have been extensively developed since 1994 by Le Thi Hoai An and Pham Dinh Tao and become now classic and more and more popular (see e.g. [4], [6] - [8], [12, 13], [15] and references therein). DCA has been successfully applied to many large-scale (smooth or nonsmooth) nonconvex programs in various domains of applied sciences, for which it provided quite often a global solution and proved to be more robust and efficient than standard methods (see e.g. [4], [6] - [8], [12, 13], [15] and reference therein).

*1-4244-0451-7/06/$20.00 ©2006 IEEE

The existence of buy-in threshold constraints makes the corresponding portfolio selection problem nonconvex and so very difficult to solve by existing algorithms. By introducing the binary variables, we first express the buy-in threshold constraints as mixed zero-one linear constraints; then, using an exact penalty result, we reformulate the last problem in terms of a DC program. A so-called DC program is that of minimizing a DC function over a convex set. We then suggested using DC programming approach and DCA to solve this portfolio selection problem. For testing the efficiency of DCA we compare it with a Branch-and-Bound algorithm.

The paper is organized as follows. After the introduction, we present in section 2 the model of the portfolio selection problem under buy-in threshold constraints, and the reformulation in term of a DC program. Section 3 deals with DC programming and a special realization of DCA to the underlying portfolio problem. Section 4 is devoted to preliminary experimental results and some conclusions are reported in section 5.

2. Portfolio selection problem under buy-in threshold constraints

2.1. Problem formulation

First of all, as we introduce the notations that we are going to use in this paper, let us remind the well known Markowitz's [10, 11] Mean-Variance model for the portfolio selection problem. Let n be the number of available assets, r_i be the mean return of asset i, Q be an $n \times n$ Variance-Covariance (positive semidefinite) matrix such that its (i, j)-th element, that is $\sigma_{i,j}$ is the covariance between returns of assets i and j and its value is calculated by using the following formula:

$$\sigma_{ij} = (1/m) \sum_{k=1}^{m} ((r_{ik} - r_i)(r_{jk} - r_j)). \qquad (1)$$

Here r_{ik} is the (i, k)-th historical data and m is the number of periods that we have considered. Let R be the desired expected return and the decision variables y_i represent the

proportion ($0 \leq y_i \leq 1$) of capital to be invested in asset i and $y^T = (y_1, ..., y_n)$. Using this notation, the standard Markowitz's Mean-Variance model is ([1])

$$\min V(y) := y^T Qy \tag{2}$$

$$s.t : \left\{ \sum_{i=1}^{n} r_i y_i = R, \sum_{i=1}^{n} y_i = 1, y_i \geq 0,\ i = 1, \ldots, n \right\}.$$

By solving this problem, one minimizes the total variance (risk) associated with the portfolio by ensuring that the portfolio has an expected return R. In this paper no short-sale is allowed.
This formulation is a simple convex quadratic program for which efficient algorithms are available. By resolving the above QP for varying values of R, we can trace out the efficient frontier, a smooth non-decreasing curve that gives the best possible tradeoff of risk against return.

For generalizing the standard Markowitz model with the inclusion of *buy-in threshold* constraints, we will use some additional notations. Let a_i and b_i be, respectively, the lower and upper bounds for the proportion of capital to be invested in asset i, with $0 < a_i \leq b_i \leq 1$. The generalized Mean-Variance model for the portfolio selection problem under buy-in threshold constraints can be written as

$$\min V(y) := y^T Qy \tag{3}$$

s.t:

$$\left\{ \sum_{i=1}^{n} r_i y_i = R, \sum_{i=1}^{n} y_i = 1, y_i \in \{0\} \cup [a_i, b_i],\ i = 1, \ldots, n \right\}.$$

Due to the last constraints $y_i \in \{0\} \cup [a_i, b_i]$, this is a hard problem for which efficient algorithms are not available.

2.2. Reformulation

The later problem can be reformulated as a mixed integer quadratic problem by introducing the additional variables z_i such that

$$z_i = 1 \text{ iff } y_i \in [a_i, b_i], \text{ 0 otherwise.}$$

The new mixed integer quadratic programming formulation of the problem is

$$\min V(y) := y^T Qy \tag{4}$$

$$s.t : \quad \sum_{i=1}^{n} r_i y_i = R, \sum_{i=1}^{n} y_i = 1,$$

$$a_i z_i \leq y_i \leq b_i z_i, z_i \in \{0, 1\}, \quad i = 1, \ldots, n.$$

Using the exact penalty result presented in [9], we will formulate (4) in the form of a convex-concave minimization problem with linear constraints which is consequently a DC program. Let

$$A := \{(y, z) \in \mathbb{R}^n \times [0, 1]^n : \sum_{i=1}^{n} r_i y_i = R, \sum_{i=1}^{n} y_i = 1, a_i z_i \leq y_i \leq b_i z_i, \quad i = 1, \ldots, n\}.$$

Define the function

$$p(y, z) := \sum_{i=1}^{n} z_i (1 - z_i).$$

Clearly, p is a concave function with nonnegative values on A and the feasible region of (4) can be written as

$$\{(y, z) \in A : z_i \in \{0, 1\}\}$$
$$= \{(y, z) \in A : p(y, z) = 0\} = \{(y, z) \in A : p(y, z) \leq 0\}.$$

So, (4) can be expressed as

$$\min\{V(y) := y^T Qy : (y, z) \in A, \quad p(y, z) \leq 0\}. \tag{5}$$

Since the objective function V is convex and A is a bounded polyhedral convex set, according to [9], there is $t_0 \geq 0$ such that for any $t > t_0$, the program (5) is equivalent to

$$\min\{F(y, z) := y^T Qy + tp(y, z) : (y, z) \in A\}. \tag{6}$$

The function F is convex in variable y and concave in variable z. Consequently it is a DC function. A natural DC formulation of the problem (6) is

$$\min\{g(y, z) - h(y, z) : (y, z) \in \mathbb{R}^n \times \mathbb{R}^n\},$$

where

$$g(y, z) := y^T Qy + \chi_A(y, z),$$

and

$$h(y, z) := t \sum_{i=1}^{n} z_i (z_i - 1).$$

Here χ_A is the indicator function on A, i.e. $\chi_A(y, z) = 0$ if $(y, z) \in A$ and $+\infty$ otherwise.

3. Solution method via DC programming and DCA

3.1. DCA for general DC programs

Let $\Gamma_0(\mathbb{R}^n)$ denote the convex cone of all lower semicontinuous proper convex functions on $\mathbb{R}^n$, and consider the general DC program

$$(P_{dc}) \quad \alpha = \inf\{f(x) := g(x) - h(x) : x \in \mathbb{R}^n\} \tag{7}$$

where $g, h \in \Gamma_0(\mathbb{R}^n)$. Such a function f is called DC function, and $g - h$, DC decomposition of f while the convex functions g and h are DC components of f.
Let C be a convex set. The problem

$$\inf\{f(x) := k(x) - h(x) : x \in C\} \tag{8}$$

can be transformed to an unconstrained DC program by using the indicator function on C, i.e.,

$$\inf\{f(x) := g(x) - h(x) : x \in \mathbb{R}^n\} \tag{9}$$

where $g := k + \chi_C$.
Let $g^*(y) := \sup\{\langle x, y\rangle - g(x) : x \in \mathbb{R}^n\}$ be the conjugate function of g. Then, the following program is called the dual program of (P_{dc}):

$$(D_{dc}) \quad \alpha_D = \inf\{h^*(y) - g^*(y) : y \in \mathbb{R}^n\}. \tag{10}$$

Under the natural convention in DC programming that is $+\infty - (+\infty) = +\infty$, and by using the fact that every function $h \in \Gamma_0(\mathbb{R}^n)$ is characterized as a pointwise supremum of a collection of affine functions, say

$$h(x) := \sup\{\langle x, y\rangle - h^*(y) : y \in \mathbb{R}^n\},$$

one can prove that $\alpha = \alpha_D$. We observe the perfect symmetry between primal and dual DC programs: the dual to (D_{dc}) is exactly (P_{dc}).
Recall that, for $\theta \in \Gamma_0(\mathbb{R}^n)$ and $x_0 \in dom\ \theta := \{x \in \mathbb{R}^n : \theta(x_0) < +\infty\}$, $\partial\theta(x_0)$ denotes the subdifferential of θ at x_0, i.e., ([14])

$$\partial\theta(x_0) := \{y \in \mathbb{R}^n : \theta(x) \geq \theta(x_0) + \langle x - x_0, y\rangle, \forall x \in \mathbb{R}^n\}. \quad (11)$$

The subdifferential $\partial\theta(x_0)$ is a closed convex set in $\mathbb{R}^n$. It generalizes the derivative in the sense that θ is differentiable at x_0 if and only if $\partial\theta(x_0)$ is reduced to a singleton which is exactly$\{\theta'(x_0)\}$. The necessary local optimality condition for the primal DC program (P_{dc}) is:

$$\partial g(x^*) \supset \partial h(x^*). \quad (12)$$

A point x^* verifies the condition $\partial h(x^*) \cap \partial g(x^*) \neq \emptyset$ is called a critical point of $g - h$. The condition (12) is also sufficient for many important classes of DC programs, for example, in case of the function f is locally convex at x^* ([7, 8, 12]).

The transportation of global solutions between (P_{dc}) and (D_{dc}) is expressed by:

$$[\cup_{y^* \in \mathcal{D}} \partial g^*(y^*)] \subset \mathcal{P}, \quad [\cup_{x^* \in \mathcal{P}} \partial h(x^*)] \subset \mathcal{D} \quad (13)$$

where $\mathcal{P}$ and $\mathcal{D}$ denote the solution sets of (P_{dc}) and (D_{dc}) respectively. Under technical conditions, this transportation holds also for local solutions of (P_{dc}) and (D_{dc}) ([6, 8, 12, 13]).

Based on local optimality conditions and duality in DC programming, the DCA consists in the construction of two sequences $\{x^k\}$ and $\{y^k\}$, candidates to be optimal solutions of primal and dual programs respectively, such that the sequences $\{g(x^k) - h(x^k)\}$ and $\{h^*(y^k) - g^*(y^k)\}$ are decreasing, and $\{x^k\}$ (resp. $\{y^k\}$) converges to a primal feasible solution $\widetilde{x}$ (resp. a dual feasible solution $\widetilde{y}$) verifying local optimality conditions and

$$\widetilde{x} \in \partial g^*(\widetilde{y}), \quad \widetilde{y} \in \partial h(\widetilde{x}). \quad (14)$$

The DCA then yields the next scheme:

$$y^k \in \partial h(x^k); \quad x^{k+1} \in \partial g^*(y^k). \quad (15)$$

In other words, these two sequences $\{x^k\}$ and $\{y^k\}$ are determined in the way that x^{k+1} (resp. y^k) is a solution to the convex program (P_k) (resp. (D_k)) defined by

$$\inf\{g(x) - h(x^k) - \langle x - x^k, y^k\rangle : x \in \mathbb{R}^n\}, \quad (P_k)$$

$$\inf\{h^*(y) - g^*(y^{k-1}) - \langle y - y^{k-1}, x^k\rangle : y \in \mathbb{R}^n\} \quad (D_k).$$

In fact, at each iteration one replaces in the primal DC program (P_{dc}) the second component h by its affine minorization $h_k(x) := h(x^k) + \langle x - x^k, y^k\rangle$ at a neighbourhood of x^k to give birth to the convex program (P_k) whose the solution set is nothing but $\partial g^*(y^k)$. Likewise, the second DC component g^* of the dual DC program (D_{dc}) is replaced by its affine minorization $(g^*)_k(y) := g^*(y^k) + \langle y - y^k, x^{k+1}\rangle$ at a neighbourhood of y^k to obtain the convex program (D_k) whose $\partial h(x^{k+1})$ is the solution set. DCA performs so a double linearization with the help of the subgradients of h and g^*.

It is worth noting that ([6, 8, 12, 13] DCA works with the convex DC components g and h but not the DC function f itself. Moreover, a DC function f *has infinitely many DC decompositions which have crucial impacts on the qualities* (speed of convergence, robustness, efficiency, globality of computed solutions,...) of DCA.

Convergence properties of DCA and its theoretical basis can be found in [6, 8, 12], for instant it is important to mention that

- DCA is a descent method (the sequences $\{g(x^k) - h(x^k)\}$ and $\{h^*(y^k) - g^*(y^k)\}$ are decreasing) without linesearch;
- If the optimal value α of problem (P_{dc}) is finite and the infinite sequences $\{x^k\}$ and $\{y^k\}$ are bounded then every limit point $\widetilde{x}$ (resp. $\widetilde{y}$) of the sequence $\{x^k\}$ (resp. $\{y^k\}$) is a critical point of $g - h$ (resp. $h^* - g^*$).
- DCA has a linear convergence for general DC programs.

3.2. DCA for solving (6)

According to the general framework of DCA, we first need computing a sub-gradient of the function h defined by $h(y, z) := t\sum_{i=1}^{n} z_i(1 - z_i)$. From the definition of h we have

$$(u^k, v^k) \in \partial h(y^k, z^k) \Leftrightarrow u_i^k = 0, v_j^k = t(2z_j^k - 1), \quad (16)$$
$$i, j = 1, \ldots, n.$$

Secondly, we have to compute an optimal solution of the following convex quadratic program

$$\min\{y^T Q y - \langle (y, z), (u^k, v^k)\rangle : (y, z) \in A\} \quad (17)$$

that will be (y^{k+1}, z^{k+1}). To sum up, the DCA applied to (6) can be described as follows.

Algorithm DCA

1. **Initialization**: Let ε be a sufficiently small positive number, let $(y^0, z^0) \in R^n \times [0, 1]^n$, and set $k = 0$;
2. **Iteration**: $k = 0, 1, 2, ...$
   set $u_i^k = 0$ and $v_i^k = t(2z_i^k - 1)$ for $i = 1, ..., n$.
   Solve the following quadratic program
   $$\min\{y^T Q y - \langle (y, z), (u^k, v^k)\rangle : (y, z) \in A\} \quad (18)$$
   to obtain (y^{k+1}, z^{k+1}).
3. If $\| y^{k+1} - y^k \| + \| z^{k+1} - z^k \| \leq \varepsilon$, then stop, (y^{k+1}, z^{k+1}) is a solution, otherwise set $k = k + 1$ and go to step 2.

For evaluating the quality of solutions computed by DCA and by the way their globality, we solve the problem by a classical Branch-and-Bound algorithm for mixed zero-one programming (4). More precisely, the lower bound is computed by solving the classical relaxed problem of (4) (the binary constraints $z_i \in \{0, 1\}$ are replaced by $0 \leq z_i \leq 1$) which is a convex quadratic program, and the upper bound is updated when a better feasible solution to (4) is discovered. The subdivision is performed in the way that $z_i = 0$ or $z_i = 1$.

4. Computational experiments

We have tested the algorithms on two sets of data that have been already used in [2, 3, 5]. These data correspond to weekly prices from March 1992 to September 1997 and they come from the indices : Dax 100 in Germany and Nikkei 225 in Japan. The number n of different assets considered for each one of the test problems is 85 and 225, respectively. The mean returns and covariances between these returns have been calculated for the data. All the results presented here have been computed using the values $a_i = 0.05$ and $b_i = 1.0$ in (4). We have tested DCA and the classical Branch-and-Bound algorithm for different values of desired expected return R. The parameter t is taken the value 0.01 for the first set of data and 0.02 for the second one. The tolerance ε is equal to 10^{-7}.
The algorithms are coded in C++ and run on a Pentium $1.600GHz$ of 512 DDRAM.

Finding a good initial point for DCA.

In fact, one of the key questions in DCA is how to find a good initial solution for it. The question is still open. In this work, in order to find a good initial solution we first solve the relaxed problem of (4). In general the obtained solution is not necessarily integer and thus we have to modify it to get a feasible solution to (4). This new solution is taken as the initial point for DCA. The procedure can be summarized as follows:

1. **Solution of the relaxed problem**
   Solve the relaxed problem of (4) to obtain the optimal solution $(\widetilde{y}, \widetilde{z})$.
2. **Finding an integer solution**
   obtain an integer solution $\widehat{z}$ by rounding each nonzero value $\widetilde{z}_i$ to one.

The new solution $(\widetilde{y}, \widehat{z})$ may not be feasible to (6). We need just one iteration of DCA to obtain a feasible solution of (6), and all the other iterations of DCA will improve the solution.

We have tested DCA from different initial points:

- The point obtained by the above procedure;
- The optimal solution of the relaxed problem of (4);
- The optimal solution of the next problem

$$\min\left\{p(y,z) := \sum_{i=1}^{n} z_i(1-z_i) : (y,z) \in A\right\}.$$

In our experiments the initial point of DCA given by the first procedure is the best.

In Tables 1, 2, 3, and 4, we give the results for two considered data sets. In these tables, the number of iterations (iter), the computer time in seconds (CPU), and the solutions obtained by each of the algorithms are shown.

The computational results show that DCA gives a good approximation of the optimal solution within a very short time. The running time is less than 2 seconds and the number of iterations is at most 4 for computing each solution.

Tab. 1: Numerical results of Branch-and-Bound algorithm for the first set of data

R	Optimal value	iter	CPU
0.00001	0.000305	12	10.953
0.00002	0.000305	13	10.656
0.00003	0.000305	12	10.516
0.00004	0.000305	12	11.703
0.00005	0.000305	13	11.610
0.00006	0.000305	13	11.547
0.00007	0.000305	13	11.672
0.00008	0.000305	13	11.813
0.00009	0.000305	13	11.813
0.0001	0.000305	13	11.890
0.0002	0.000305	14	13.110
0.0003	0.000306	14	13.110
0.0004	0.000308	16	15.407
0.0005	0.000310	24	22.844
0.0006	0.000312	15	14.250
0.0007	0.000315	15	14.328
0.0008	0.000319	32	30.563
0.0009	0.000322	32	30.563
0.001	0.000326	30	29.265
0.002	0.000390	12	12.140
0.003	0.000517	11	11.657

Tab. 2: Numerical results of DCA for the first set of data

R	Optimal value	iter	CPU
0.00001	0.000306	2	1.594
0.00002	0.000306	2	1.609
0.00003	0.000306	2	1.594
0.00004	0.000306	2	1.625
0.00005	0.000306	2	1.609
0.00006	0.000306	2	1.578
0.00007	0.000306	2	1.610
0.00008	0.000306	2	1.594
0.00009	0.000306	2	1.609
0.0001	0.000306	2	1.703
0.0002	0.000305	2	1.750
0.0003	0.000307	2	1.719
0.0004	0.000310	2	1.781
0.0005	0.000311	2	1.735
0.0006	0.000314	2	1.719
0.0007	0.000316	2	1.719
0.0008	0.000322	2	1.781
0.0009	0.000324	2	1.687
0.001	0.000328	2	1.781
0.002	0.000391	2	1.828
0.003	0.000519	2	1.953

5. Conclusions

In this paper we present a new approach for solving the portfolio selection problem. Instead of the standard Markowitz mean-variance model, we have used an extension including buy-in threshold and bounding constraints. These constraints make the corresponding portfolio selec-

Tab. 3: Numerical results of Branch-and-Bound algorithm for the second set of data

R	Optimal value	iter	CPU
0.0001	0.000174	1348	147.953
0.0002	0.000170	718	77.343
0.0003	0.000167	491	53.328
0.0004	0.000164	549	59.313
0.0005	0.000162	671	72.625
0.0006	0.000159	788	86.500
0.0007	0.000158	1475	158.547
0.0008	0.000156	1648	175.828
0.0009	0.000154	1838	194.860
0.001	0.000153	1980	209.610
0.002	0.000141	204	22.062
0.003	0.000147	140	15.875
0.004	0.000170	129	14.406

Tab. 4: Numerical results of DCA for the second set of data

R	Optimal value	iter	CPU
0.0001	0.000186	2	0.235
0.0002	0.000189	2	0.234
0.0003	0.000193	2	0.218
0.0004	0.000182	3	0.266
0.0005	0.000174	3	0.266
0.0006	0.000173	4	0.312
0.0007	0.000170	4	0.313
0.0008	0.000167	3	0.266
0.0009	0.000167	4	0.313
0.001	0.000167	4	0.312
0.002	0.000156	2	0.219
0.003	0.000159	2	0.234
0.004	0.000207	2	0.203

tion problem nonconvex and so very difficult to solve by existing algorithms. We have transformed this problem into a mixed integer quadratic program and developed a deterministic approach based on DC programming and DCA. Preliminary numerical simulations show the efficiency of DCA, its inexpensiveness and its superiority with respect to standard branch-and-bound techniques. They suggest to us extending the numerical experiments in higher dimension, and combining DCA and Branch and Bound algorithms for globally solving the problem of portfolio selection Work in these directions is currently in progress.

REFERENCES

[1] M. Bartholomew-Biggs, "Nonlinear Optimization with Financial Applications", *Kluwer Academic Publishers*, First edition, 2005.

[2] Chang T.J., N. Meade, J.E. Beasley and, Y.M. Sharaiha, "Heuristics for cardinality constrained portfolio optimization", *Computers and Operations Research*, Vol. 27, pp1271-1302, 2000.

[3] Fernandez A., Y.M. Gomez, "Portfolio selection using neural networks", *Computers & Operations Research*, To appear.

[4] Harrington J.E., B.F. Hobbs, J.S. Pang, A. Liu, G. Roch, "Collusive game solutions via optimisation", *Math. Program. Ser. B*, Vol. 104, No. 1-2, pp407-435, 2005.

[5] Jobst N., M. Horniman, C. Lucas, G. Mitra, "Computational aspects of alternative portfolio selection models in the presence of discrete asset choice constraints", *Quantitative Finance*, Vol. 1, pp1-13, 2001.

[6] Le Thi, H.A., "Contribution à l'optimisation non convexe et l'optimisation globale: Théorie, Algorithmes et Applications", *Habilitation à Diriger des Recherches, Université de Rouen*, 1997.

[7] Le Thi H.A. and T. Pham Dinh, "A continuous approach for globally solving linearly constrained quadratic zero-one programming problems", *Optimization*, Vol. 50, No. 1-2, pp93-120, 2001.

[8] Le Thi H.A. and T. Pham Dinh, "The DC (difference of convex functions) Programming and DCA revisited with DC models of real world non convex optimization problems", *Annals of Operations Research*, Vol. 133, pp23-46, 2005.

[9] Le Thi H.A., T. Pham Dinh, V.N. Huynh, "Exact Penalty Techniques in DC Programming", *Research Report, LMI, National Institute for Applied Sciences - Rouen, France*, July, 2005.

[10] Markowitz, Harry M. "Portfolio Selection", *Journal of Finance*, Vol. 7, No. 1, pp77-91, 1952.

[11] Harry M. Markowitz, "Portfolio Selection", *John Wiley and Sons, New York*, First Edition, 1959.

[12] Pham Dinh T. and H.A. Le Thi, "Convex analysis approach to d.c. programming: Theory, Algorithms and Applications", *Acta Mathematica Vietnamica, dedicated to Professor Hoang Tuy on the occasion of his 70th birthday*, Vol. 22, No. 1, pp289-355, 1997.

[13] Pham Dinh T. and H.A. Le Thi, "DC optimization algorithms for solving the trust region subproblem", *SIAM J. Optimization*, Vol. 8, pp476-505, 1998.

[14] R.T. Rockafellar, "Convex Analysis", *Princeton University Press, Princeton*, First edition, 1970.

[15] Stefan Weber, Christoph Schnörr, Thomas Schüle, Joachim Hornegger, "Binary Tomography by Iterating Linear Programs", R. Klette, R. Kozera, L. Noakes and J. Weickert (Eds.), "Computational Imaging and Vision - Geometric Properties from Incomplete Data", *Kluwer Academic Publishers*, 2005.

Bibliographie

[1] Acerbi C. et D. Tasche, *On the Coherence of Expected Shortfall,* Journal of Banking and Finance, Vol. 43, No. 7, 1487–1503, 2002.

[2] Aftalion A., *La Nouvelle Finance et la Gestion des Portefeuilles,* Ed. Economica, Paris, 2003.

[3] Akoa F.B, *Approches de points intérieurs et de programmation DC en optimisation non convexe. Code et simulations numériques industrielles.* Thèse de Doctorat de l'Université de Rouen, 2005.

[4] Artzner P., F. Delbaen, J.M. Eber et D. Heath, *Coherent Measures of Risk,* Mathematical Finance, Vol. 9, No. 3, 203–228, 1999.

[5] Athearn J.L., *What is Risk ?* The Journal of Risk and Insurance, 639–645, 1971.

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

[7] Bartholomew-Biggs M.C., *Nonlinear Optimization with Financial Applications,* Kluwer Academic Publishers, United States of America, 2005.

[8] Bartholomew-Biggs M.C. et S.G. Bartholomew-Biggs, *A global optimization problem in portfolio selection,* Computational Management Science, Available online, 2007.

[9] Bawa V.S., *Optimal Rules for Ordering Uncertain Prospects,* Journal of Financial Economics, 2, 95–121, 1975.

[10] Bawa V.S., S.J. Brown, et R.W. Klein, *Estimation Risk and Optimal Portfolio Choice,* North-Holland, Amsterdam, Netherlands, 1979.

[11] Bawa V.S., *Stochastic Dominance : A Research Bibliography,* Management Science, 28, 698–712, 1982.

[12] Benson H.P., *Concave minimization : theory, applications and algorithms,* in Handbook of Global Optimisation, R. Horst and P. Padalos (Eds.), Kluwer Academic Publishers, 43–148, 1995.

[13] Benson H.P., *Deterministic algorithm for constrained concave minimization : a unified critical survey,* Naval Research Logistics, Vol.43, 765–795, 1996.

[14] Benson H.P. et R. Horst, *A branch and bound - outer approximation for concave minimization over a convex set,* Journal of Computers and Mathematics with applications, Vol.21, 67–76, 1991.

[15] Brusco M.J. et S. Stahl *Branch and Bound Applications in Combinatorial Data Analysis,* Springer, 2005.

[16] Chang T.J., N. Meade, J.E. Beasley et Y.M. Sharaiha, *Heuristics for cardinality constrained portfolio optimization,* Computers & Operations Research, 27, 1271–1302, 2000.

[17] Collobert R., F. Sinz, J. Weston et L. Bottou *Trading Convexity for Scalability,* Proceedings of the 23rd ICML, 2006.

[18] Cornuejols G. et R. Tutuncu, *Optimization Methods in Finance,* Cambridge University Press, 2007.

[19] Cottle R.W. et G.B. Dantzig *Complementary pivot theory of mathematical programming,* Linear algebra and its applications, Vol.1, 103–125, 1968.

[20] Crowe R.M. et R.C. Horn, *The Meaning of Risk,* The Journal of Risk and Insurance, No. 3, 459–474, 1957.

[21] Dantzig G.B., *Linear Programming and Extensions,* Princeton University Press, Princeton, New Jersey, 1963.

[22] Dantzig G.B., D.R. Fulkerson et S.M. Johnson *Solution of a large scale traveling salesman problem,* Operations Recheach, Vol.2, 393–410, 1954.

[23] Davis M.H.A. et A.R. Norman, *Portfolio selection with transaction costs,* Mathematics of Operations Research, Vol. 15, N. 4, 676–713, 1990.

[24] Demange G. et J.C Rochet *Méthodes mathématiques de la finance,* Economica, France, (3rd Edition) 2005.

[25] Eastman W.L., *Linear Programming with Pattern Constraints,* Ph.D. thesis, The computation Laboratory, Harvard University, 1958.

[26] Eastman W.L., *A Solution to the Travelling Salesman Problem,* Presented at the American Summer Meeting of the Econometric Society, Cambridge, Mass., 1958.

[27] Elton E.J., M.J. Grubner, S.J. Brown et W.N. Goetzmann, *Modern Portfolio Theory and Investment Analysis,* John Wiley & Sons, Inc., United States of America, 2003.

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

[29] Fernandez A. et S. Gomez, *Portfolio selection using neural networks,* Computers & Operations Research, 34, 1177–1191, 2007.

[30] Fishburn P.C., *Decision and Value Theory,* Wiley, New York, 1964.

[31] Fishburn P.C., *Mean-Risk Analysis with Risk Associated with Below-Target Returns,* The American Economic Review, Vol. 67, No. 2, 116–126, 1977.

[32] Fletcher R., *Pratical methods of Optimization,* John Wiley, New York, 1980.

[33] Gautam M., *A review of portfolio planning : models and systems,* S. Satchell et A. Scowcroft (Eds.), Advances in Portfolio Construction and Implementation, Elsevier, Burlington MA, 1–39, 2003.

[34] Grebeck M.J., *Applications of Stochastic Programming to asset liability Management,* Ph. D. thesis in Statistics and Applied Probability , University of California, Santa Barbara, 2006.

[35] Gulpinar N., H.A Le Thi et M. Moeini, *Robust Investment Strategies with Discrete Asset Choice Constraints Using DCA,* submitted to *Journal of Global Optimization.*

[36] Gulpinar N. et B. Rustem, *Continuous Min-Max Approach for Single Period Portfolio Selection Problem,* M. Breton and H. Ben-Ameur (Eds.), in Numerical Methods in Finance, Springer, 241–258, 2005.

[37] Gulpinar N. et B. Rustem, *Worst-case robust decisions for multi-period mean-variance portfolio optimization,* European Journal of Operational Research, Vol. 183, 981–1000, 2007.

[38] Gulpinar N., B. Rustem et R. Settergren, *Simulation and Optimization Approaches to Scenario tree Generation,* Journal of Economic Dynamics and Control, 28 , 1291–1315, 2004.

[39] Gupta O.K., *Branch and Bound Experiments in Nonlinear Integer Programming,* Ph.D. thesis, Purdue University, 1980.

[40] Hardy G.H. et J.E. Littlewood and G. Polya, *Inequalities,* Cambridge University Press, Cambridge, MA, 1934.

[41] Hanoch G. et H. Levy, *The Efficiency Analysis of Choices Involving Risk,* Rev. Economic Studies, Vol. 36, 335–346, 1969.

[42] Hiriart Urruty J.B. et C. Lemarechal *Convex Analysis and Minimization Algorithms,* Springer, Berlin, 1993.

[43] Holton G.A., *Defining Risk,* Financial Analysts Journal, Vol. 60, No. 6, 19–25, 2004.

[44] Horst R., *A general class of branch-and-bound methods in global optimization with some new approaches for concave minimization,* Journal of Optimization Theory and Application, 58, 11–37, 1988.

[45] Horst R., N.V. Thoai et J. De Vries, *On finding new vertices and redundant constraints in cutting plane algorithms for global optimization,* Operations Research Letters, 7, 85–90, 1988.

[46] Horst R., N.V Thoai et H. Tuy, *On a outer approximation concept in global optimisation,* Optimization, Vol.20, 255–264, 1989.

[47] Horst R., N.V Thoai et H.P Benson, *Concave minimization via conical partitions and polyhedral outer approximation,* Mathematical Programming, Vol.50, 259–276, 1991.

[48] Horst R., P.M Pardalos et N.V Thoai, *Introduction to Global Optimization,* Kluwer Academic Publishers, 1995.

[49] Horst R. et H. Tuy, *Global Optimization : Deterministic Approaches,* Third edition, Springer-Verlag, 1996.

[50] Horst R. et N.V Thoai, *DC Programming : Overview,* Journal of Optimization Theory and Applications, Vol.2103, 1–43, 1999.

[51] Jobst N., M. Horniman, C. Lucas et G. Mitra, *Computational aspects of alternative portfolio selection models in the presence of discrete asset choice constraints,* Quantitative Finance, 1, 1–13, 2001.

[52] Jorion P., *Value at Risk,* McGraw-Hill, New York, 2000.

[53] Kalantari B. et J.B. Rosen, *Algorithm for global minimization of linearly constrained concave quadratic functions,* Operations Research, Vol. 12, 544–561, 1987.

[54] Konno H., *Applications of Global Optimization to Portfolio Analysis,* C. Audet, P. Hansen and G. Savard (Eds.), Essays and Surveys in Global Optimization (First edn), Springer USA, 195–210, 2005.

[55] Konno H., K. Akishino et R. Yamamoto, *Optimization of a Long-Short Portfolio under Nonconvex Transaction Cost,* Computational Optimization and Applications, 32, 115–132, 2005.

[56] Konno H., T. Koshizuka et R. Yamamoto, *Mean-variance portfolio optimization problems under short sale opportunity,* to appear in Dynamics of Continuous, Discrete and Inpulsive System.

[57] Konno H. et A. Wijayanayake, *Mean-absolute deviation portfolio optimization model under transaction costs,* Journal of Operations Research Society of Japan, 42, 422–435, 1999.

[58] Konno H. et A. Wijayanayake, *Portfolio optimization problem under concave transaction costs and minimal transaction unit constraints,* Mathematical Programming Ser. B, 89, 233–250, 2001.

[59] Konno H. et A. Wijayanayake, *Portfolio optimization under DC transaction costs and minimal transaction unit constraints,* Journal of Global Optimization, 22, 137–154, 2002.

[60] Konno H. et R. Yamamoto, *Global optimization versus integer programming in portfolio optimization under nonconvex transaction costs,* Journal of Global Optimization, 32, 207–219, 2005.

[61] Konno H. et R. Yamamoto, *Integer programming approaches in mean-risk models,* Computational Management Science, 2, 339–351, 2005.

[62] Konno H. et H. Yamazaki, *Mean-absolute deviation portfolio optimization model and its applications to Tokyo stock market,* Management Science, 37, 519–531, 1991.

[63] Krause A., *Coherent risk measurement :an introduction,* Balance Sheet, Vol. 10, No. 4, 13–17, 2002.

[64] Krause N. et Y. Singer *Leveraging the margin more carefully,* International Conference on Machine Learning ICML, 2004.

[65] Kühn J., *Optimal Risk-Return Trade-off of Commercial Banks and the stability of Profitability Measures for Loan Portfolios,* Springer, Berlin Heidelberg, 2006.

[66] Land A.H. et A.G. Doig *An automatic method for solving discrete programming problems,* Econometrica, 28, 497–520, 1960.

[67] Laurent P.J., *Approximation et optimisation,* Paris : Hermann, 1972.

[68] Laurent P.J., *Approximation et Optimisation,* Kluwer Academic Publishers, Dordrecht, 1995.

[69] Lobo M.S., M. Fazel et S. Boyd, *Portfolio optimization with linear and fixed transaction costs,* Annals of Operations Research, 152, 341–365, 2007.

[70] Le Thi H.A., *Analyse numérique des algorithmes de l'optimisation DC. Approches locale et globale. Codes et simulations numériques en grande dimension. Applications.* Thèse de Doctorat de l'Université de Rouen, 1994.

[71] Le Thi H.A., *Contribution à l'optimisation non convexe et l'optimisation globale : Théorie, Algorithmes et Applications,* Habilitation à Diriger des Recherches, Université de Rouen, 1997.

[72] Le Thi H.A., *An efficient algorithm for globally minimizing a quadratic function under convex quadratic constraints,* Mathematical Programming, Ser. A, Vol.87, N°.3, 401–426, 2000.

[73] Le Thi H.A., *Solving large scale Molecular distance geometry problem by a smoothing technique via the Gaussian transform an DC programming,* Journal of Global Optimization, Vol.27, 375–397, 2003.

[74] Le Thi H.A., T. Belghiti et T. Pham Dinh, *A new efficient algorithm based on DC programming and DCA for Clustering,* In Press, Available July 2006, Journal of Global Optimization.

[75] Le Thi H.A., M. Le Hoai, T.P Nguyen et T. Pham Dinh, *Noisy Image Segmentation by Fuzzy C-Means Clustering based DCA,* Submitted, 2007.

[76] Le Thi H.A., M. Le Hoai et T. Pham Dinh, *Optimization based DC programming and DCA for Hierarchical Clustering,* In Press, Available Online June 2006, European Journal of Operational Research.

[77] Le Thi H.A., M. Le Hoai et T. Pham Dinh, *Une nouvelle approche basée sur la programmation DC et DCA pour la classification floue,* EGC 2007, RNTI, 703–714, 2007.

[78] Le Thi H.A., M. Moeini et T. Pham Dinh, *Portfolio Selection under Downside Risk Measures and Cardinality Constraints based on DC Programming and DCA,* to appear in *Computational Management Science,* 2008.

[79] Le Thi H.A., M. Moeini et T. Pham Dinh, *DC programming Approach for Portfolio Optimization under Step Increasing Transaction Costs,* to appear in *Optimization,* 2008.

[80] Le Thi H.A. et M. Moeini, *Optimization of a Long-Short Portfolio under Threshold Constraints using DC Programming and DCA,* submitted to *Operations Research.*

[81] Le Thi H.A. et M. Moeini, *Portfolio Selection Under Buy-In Threshold Constraints Using DC Programming and DCA,* Proceeding of third International Conference on Service Systems and Service Management (SSSM06/IEEE), Troyes, October 2006, pp. 296-300.

[82] Le Thi H.A., T.P. Nguyen et T. Pham Dinh, *A Continuous DC Programming Approach To The Strategic Supply Chain Design Problem From Qualified Partner Set,* In Press, Available Online June 2006, European Journal of Operational Research.

[83] Le Thi H.A. et T. Pham Dinh *Solving a class of linearly constrained indefinite quadratic problems by DC algorithms,* Journal of Global Optimization, Vol.11, 253–285, 1997.

[84] Le Thi H.A. et T. Pham Dinh, *A Branch-and-Bound method via DC Optimization Algorithm and Ellipsoidal techniques for Box Constrained Nonconvex Quadratic Programming Problems,* Journal of Global Optimization, Vol.13, 171–206, 1998.

[85] Le Thi H.A. et T. Pham Dinh, *DC programming approach for large-scale molecular optimization via the general distance geometry problem,* Nonconvex Optimization and Its Applications 40, in Optimization in Computational Chemistry and Molecular Biology : Local and Global Approaches, Kluwer Academic Publishers, 301–339, 2000.

[86] Le Thi H.A. et T. Pham Dinh, *Large Scale Molecular Conformation via the Exact Distance Geometry Problem,* In Optimization, Lecture Notes in Economics and Mathematical Systems, Vol.481, Heidelberg, Springer-Verlag, 260-277, 2000.

[87] Le Thi H.A. et T. Pham Dinh, *A continuous Approach for Globally Solving Linearly Constrained Quadratic Zero - One Programming Problems,* Optimization, Vol.50, 93–120, 2001.

[88] Le Thi H.A. et T. Pham Dinh, *DC optimization approaches via Markov models for restoration of signals (1-D) and (2-D),* Nonconvex Optimization and Its Applications 54 : In Advances in Convex Analysis and Global Optimization, Kluwer Academic Publishers, 300–317, 2001.

[89] Le Thi H.A. et T. Pham Dinh, *DC programming approach and solution algorithm to the multidimensional scaling problem,* Nonconvex Optimization and Its Applications 53 : In From Local to Global Optimization, Kluwer Academic Publishers, 231–276, 2001.

[90] Le Thi H.A. et T. Pham Dinh, *DC Programming Approach for Multicommodity Network Optimization Problems with Step Increasing Cost Functions,* Special Issue of Journal of Global Optimization (dedicated to Professor R. Horst on the occasion of his 60 th birthday), Vol.22, 204–233, 2002.

[91] Le Thi H.A. et T. Pham Dinh, *Dc Programming. Theory, Algorithms, Applications : The State of the Art,* First InternationalWorkshop on Global Constrained Optimization and Constraint Satisfaction, October 2-4, 2002, Valbonne-Sophia Antipolis, France, Research Report, Laboratory of Modeling, Optimization & Operations Research, Insa-Rouen, France, 2002.

[92] Le Thi H.A. et T. Pham Dinh, *Large scale molecular optimization from distances matrices by a DC optimization approach,* SIAM Journal of Optimization, Vol.14, N°.1, 77–116, 2003.

[93] Le Thi H.A. et T. Pham Dinh, *A new algorithm for solving large scale molecular distance geometry problems, Applied Optimization : in Hight Performance Algorithms and Software for Nonlinear Optimization,* Kluwer Academic Publishers, 276–296, 2003.

[94] Le Thi H.A. et T. Pham Dinh, *The DC (Difference of Convex functions) Programming and DCA Revisited with DC Models of Real World Nonconvex Optimization Problems,* Annals of Operations Research, Vol.133, 23–46, 2005.

[95] Le Thi H.A. et T. Pham Dinh, *A continuous approach for the concave cost supply problem via DC Programming and DCA,* to appear in Discrete Applied Mathematics.

[96] Le Thi H.A., T. Pham Dinh et H. Dinh Nho, *Towards Tikhonov regularization of nonlinear ill-posed problems : a DC programming approach,* C.R. Acad. Sci. Paris, Ser. I, Vol.335, 1073–1078, 2002.

[97] Le Thi H.A., T. Pham Dinh et H. Dinh Nho, *Solving inverse problems for an elliptic equations by DC (Difference of Convex functions) programming,* Journal of Global Optimization, Vol.25, 407–423, 2003.

[98] Le Thi H.A., T. Pham Dinh et H. Dinh Nho, *On the ill-posedness of the trust region Subproblem,* Journal of Inverse and Ill-posed Problems, Vol.11, 545–577, 2003.

[99] Le Thi H.A., T. Pham Dinh et N. Huynh Van, *Exact penalty techniques in DC Programming,* Submitted, 2004.

[100] Le Thi H.A., T. Pham Dinh et M. Le Dung, *Numerical solution for optimization over the efficient set by DC optimization algorithm,* Operations Research Letters, Vol.19, 117–128, 1996.

[101] HLe Thi H.A., T. Pham Dinh et M. Le Dung, *A combined DC Optimization : Ellipsoidal Branch-and-Bound Algorithm for Solving Nonconvex Quadratic Programming Problems,* Journal of Combinatorial Optimization, Vol.2, N°.1, 9–28, 1998.

[102] Le Thi H.A., T. Pham Dinh et M. Le Dung, *Exact Penalty in DC. Programming,* Vietnam Journal of Mathematics, Vol.27, N°.2, 169–178, 1999.

[103] Le Thi H.A., T. Pham Dinh et M. Le Dung, *Simplicially Constrained DC Optimization over the Efficient Set and Weakly Efficient Sets,* Journal of Optimization, Theory and Applications, Vol.117, N°.3, 503-531, 2003.

[104] Le Thi H.A., T. Pham Dinh et T. Nguyen Van, *Combination between Local and Global Methods for Solving an Optimization Problem over the Efficient Set,* European Journal of Operational Research, Vol.142, 257–270, 2002.

[105] Le Thi H.A., T. Pham Dinh et V.N. Huynh, *Exact Penalty Techniques in DC Programming,* LMI, National Institute for Applied Sciences - Rouen, France, July, (2005).

[106] Levenberg K., *A method for the solution or certain nonlinear problems in least quares,* Quart. Appl. Math., Vol.2, 1944.

[107] Levy H., *Stochastic dominance and expected utility : Survey and analysis,* Management Science, 38, 555–593, 1992.

[108] Levy H., *Stochastic Dominance Investment Decision Making Under Uncertainty,* Springer, United States of America, (2nd Ed.), 2006.

[109] Little J.D.C., K.G. Murty, D.W. Sweeney et C. Karel,*An Algorithme for Travelling Salesman Problem,* Operations Research, Vol. 11, 972–989, 1963.

[110] Liu Y., X. Shen et H. Doss *Multicategory ψ-Learning and Support Vector Machine : Computational Tools,* Journal of Computational and Graphical Statistics, Vol.14, 219–236, 2005.

[111] Liu Y. et X. Shen, *Multicategory ψ-Learning,* Journal of the American Statistical Association, Vol.101, 500–509, 2006.

[112] Markowitz H.M., *Portfolio Selection,* Journal of Finance, Vol. 7, 77–91, 1952.

[113] Markowitz H.M., *Portfolio Selection,* John Wiley & Sons, New York, 1959.

[114] Maringer D., *Portfolio Management With Heuristic Optimization,* Springer, Dordrecht, 2005.

[115] Marquardt D.W., *An algorithm for least squares estimation of nonlinear parameters,* SIAM J. Appl. Math., Vol.11, 1963.

[116] Moré J.J., *Recent developpements in algorithm and software for trust region methods,* Mathematique Programming, The state of the art, Springer-Verlag, Berlin, 258–287, 1983.

[117] Moré J.J. et D.C. Sorensen, *Computing a trust region step,* SIAM J. Sci. Statist. Comput., Vol.4, 553–572, 1983.

[118] Morgan/Reuters J.P., *Value at Risk,* RiskMetrics-Technical Document. J.P. Morgan, 4th edition 1996.

[119] Murty K.G., C. Karel et J.D.C. Little, *The traveling salesman problem : Solution by a method of ranking assignments,* Cleveland : Case Institute of Technology, 1962.

[120] K.G. Murty, *Operations research : Deterministic optimization models,* Englewood Cliffs, NJ : Prentice-Hall, 1995.

[121] Muu L.D., T.Q. Phong et T.D. Pham, *Decomposition methods for solving a class of nonconvex programming problems dealing with bilinear and quadratic functions,* Computational Optimization and Application, 4, 203–216, 1995.

[122] Nawrocki Q., *A brief history of Downside Risk Measures,* Technical Report, Villanova, Arcola, PA.

[123] Neumann J., C. Schnörr et G. Steidl, *SVM-based Feature Selection by Direct Objective Minimisation,* Pattern Recognition, Proc. of 26th DAGM Symposium, LNCS, Springer, August 2004.

[124] Neumann J., C. Schnörr et G. Steidl, *Combined SVM-Based Feature Selection and Classification Machine Learning,* in Press (published online) http ://www.cvgpr.uni-mannheim.de/Publications/ML-05.pdf.

[125] Ogryczak W. et A. Ruszczynski, *From stochastic dominance to mean-risk model : Semideviation as the risk measures,* European Journal of Operational Research, Vol. 116, 33–50, 1999.

[126] Ogryczak W. et A. Ruszczynski, *On consistency of stochastic dominance and mean- semideviation models,* Mathematical Programming, 89, 217–232, 2001.

[127] Pardalos P.M. et J.B. Rosen, *Constrained Global Optimization : Algorithms and Applications,* in Lecture Notes in Computer Science, G. Goos and J. Hartmanis eds., Springer, 1987.

[128] Peyrard J. et M. Peyrard, *Dictionnaire de finance,* Librairie Vuibert, 2001.

[129] Pham Dinh T., *Elements homoduaux relatifs à un couple de normes (φ, ψ). Applications au calcul de $S_{\varphi\psi}(A)$,* Technical Report, Grenoble, 1975.

[130] Pham Dinh T., *Calcul du maximum d'une forme quadratique définie positive sur la boule unité de la norme du max,* Technical Report, Grenoble, 1976.

[131] Pham Dinh T., *Contribution à la théorie de normes et ses applications à l'analyse numérique,* Thèse de Doctorat d'Etat Es Science, Université Joseph Fourier- Grenoble, 1981.

[132] Pham Dinh T., *Convergence of subgradient method for computing the bound norm of matrices,* Linear Alg. and Its Appl., Vol.62, 163–182, 1984.

[133] Pham Dinh T., *Algorithmes de calcul d'une forme quadratique sur la boule unité de la norme maximum,* Numer. Math., Vol.45, 377–440, 1985.

[134] Pham Dinh T., *Algorithms for solving a class of non convex optimization problems. Methods of subgradients,* Fermat days 85. Mathematics for Optimization, Elsevier Science Publishers B.V. North-Holland, 1986.

[135] Pham Dinh T., *Duality in DC (difference of convex functions) optimization. Subgradient methods,* Trends in Mathematical Optimization, International Series of Numer Math., Vol 84, 277–293, 1988.

[136] Pham Dinh T. et H.A Le Thi, *Stabilité de la dualité lagrangienne en optimisation DC (différence de deux fonctions convexes),* C.R. Acad. Paris, P.318, Série I, 379–384, 1994.

[137] Pham Dinh T. et H.A Le Thi, *Lagrangian stability and global optimality in nonconvex quadratic minimization over Euclidean balls and spheres,* Journal of Convex Analysis, Vol.2, 263–276, 1995.

[138] Pham Dinh T. et H.A Le Thi, *DC optimization algorithms for globally minimizing nonconvex quadratic forms on Euclideans balls and spheres,* Operations Research Letters, Vol.19, 207–216, 1996.

[139] Pham Dinh T. et H.A Le Thi, *Convex analysis approach to d.c. programming : Theory, Algorithms and Applications,* Acta Mathematica Vietnamica, dedicated to Professor Hoang Tuy on the occasion of his 70th birthday, Vol.22, N°.1, 289–355, 1997.

[140] Pham Dinh T. et H.A Le Thi, *DC optimization algorithms for solving the trust region subproblem,* SIAM Journal of Optimization, Vol.8, N°.2, 476–505, 1998.

[141] Phillips A.T. et J.B. Rosen, *A parallel algorithm for constrained concave quadratic global minimization,* Mathematical Programming, 42, 412–448, 1988.

[142] Phong T.Q., *Analyse Numerique des Méthodes d'Optimisation Globale,* Thèse de Doctoral, Université de Rouen, 1994.

[143] Phong T.Q., T. Pham Dinh et H.A Le Thi, *A New Method for Solving DC Programming Problems. Application to Fuel Mixture Nonconvex Optimization Problem,* Journal of Global Optimal, Vol.6, 87–105, 1995.

[144] Polyak B.T., *Introduction to optimization,* Inc., Publications Division, 1987.

[145] Potaptchik M., *Portfolio Selection under Nonsmooth Transaction Costs,* Ph.D. thesis, University of Waterloo, 2006.

[146] Rockafellar R.T., *Convex analysis,* Princeton University Press, 1970.

[147] Rockafellar R.T., *Monotone operators and the proximal point algorithm,* SIAM Journal on Control and Optimization, Vol.14, 877–898, 1976.

[148] Rockafellar R.T. et S. Uryasev, *Conditional Value-at-Risk for General Loss Distributions,* Journal of Banking and Finance, Vol. 26, No. 7, 1443–1471, 2002.

[149] Rustem B., R. Becker et W. Marty, *Robust Min-Max Portfolio strategies for Rival Forecast and Risk Scenarios,* Journal of Economic Dynamics and Control, 24, 1591–1623, 2000.

[150] Ronan C., S. Fabian, W. Jason et B. Léon, *Trading Convexity for Scalability,* International Conference on Machine Learning ICML, 2006.

[151] Rothschild M. et J.E. Stiglitz, *Increasing Risk. I. A Definition,* Journal of Economic Theory, 2, 225–243, 1970.

[152] Roy A.D., *Safety First And The Holding Of Assets,* Econometrica, 20, 431–449, 1952.

[153] Sadjadi S.J. et K. Ponnambalam, *Advances in trust region algorithms for constrained optimization,* Applied Numerical Mathematics, Vol.29, 423–443, 1999.

[154] Schaerf A., *Local search techniques for constrained portfolio selection problems,* Computational Economics, 20, 177–90, 2002.

[155] Scherer B. et M.R. Douglas, *Introduction to Modern Portfolio Optimization With NUOPT and S-PLUS,* Springer, USA, 2005.

[156] Schüle T., C. Schnörr, S. Weber et J. Hornegger, *Discrete Tomography by convex-concave regularization and DC programming,* Discrete Applied Mat., Vol.151, 229–243, 2005.

[157] Schüle T., S. Weber et C. Schnörr, *Adaptive Reconstruction of Discrete-Valued Objects from few Projections,* Electr. Notes in Discr. Math., Vol.20, 365–384, 2005.

[158] Shen X., *From large margin classification to ψ-learning,* School of Statistics, University of Minnesota.

[159] Shen X., G.C Tseng, X. Zhang, et W.H Wong, *On ψ-Learning,* Journal of American Statistical Association, Vol.98, 724–734, 2003.

[160] Simaan Y., *Estimation Risk in Portfolio Selection : The Mean Variance Model Versus the Mean Absolute Deviation Model,* Management Science, 43, 1437–1446, 1997.

[161] Sorensen D.C., *Newton's method with a model trust region modification,* SIAM J. Numer. Anal., Vol.19, N°.2, 409–426, 1982.

[162] Steihaug S., *The conjugate gradient method and trust region in large scale optimization,* SIAM J. Numer. Anal., Vol.20, 626–637, 1983.

[163] Stone B.K., *A General Class of Three Parameter Risk Measurs,* Journal of Finance, Vol. 28, 675–685, 1973.

[164] Thach P.T., *DC sets, DC functions and nonlinear equations,* Mathematical Programming, Vol.58, 415–428, 1993.

[165] Thoai N.V. et H. Tuy, *Convergent algorithms for minimizing a concave function,* Math. Operations Research, 5, 556–566, 1980.

[166] Tobin J., *Liquidity Preferences as Behavior Towards Risk,* Rev. Econ. Stud., Vol. 25, 65–85, 1958.

[167] Todd B.B., *Improved branch and bound algorithms for integer programming,* Ph.D. thesis, Rensselaer Polytechnic Institute, 1992.

[168] Toland J.F., *Duality in nonconvex optimization,* Journal of Mathematical Analysis and Applications, Vol.66, 399-415, 1978.

[169] Toland J.F., *On subdifferential calculus and duality in nonconvex optimization,* Bull. Soc. Math. France, Mémoire 60, 177–183, 1979.

[170] Toint P.L., *Towards an efficient sparsity exploiting Newton method for minimization,* Duff, I., ed., Sparse Matrices and Their Uses, 57–88, Academic Press, 1981.

[171] Tuy H., *Concave programming under linear constraints,* Translated Soviet Mathematics, Vol.5, 1437–1440, 1964.

[172] Tuy H., *On outer approximation methods for solving concave minimization problems,* Acta Mathematica Vietnamica, Vol.8, 3–34, 1983.

[173] Tuy H., *Global Minimization of a Difference of Two Convex Functions,* Mathematics Programming Study, Vol.30, 150–182, 1987.

[174] Tuy H., *Global Optimization : Deterministic Approaches,* 2nd revised edition, Springer-Verlag, Berlin, 1993.

[175] Tuy H., *DC Optimisation : Theory, Methods and Algorithms, Handbook of Global Optimisation,* Horst and Pardalos eds, Klaver Academic Publishers, 149–216, 1995.

[176] Tuy H., *Convex Analysis and Global Optimization,* Kluwer Academic Publishers, 1998.

[177] Tuy H., T.V. Thieu et N.Q. Thai, *A conical algorithm for globally minimizing a concave function over a convex set,* Math. Operations Research, 10, 498–514, 1985.

[178] Urruty J.B.H., *Generalized differentiability, duality and optimization for problem dealing with differences of convex functions,* Lecture Notes in Economics and Mathematical Systems, Vol.256, Heidelberg, Springer-Verlag, 260–277, 1985.

[179] Urruty J.B.H., *Conditions nécessaires et suffisantes d'optimalité globale en optimisation de différences de deux fonctions convexes,* CRAS, Vol.309, Série I, 459–462, 1989.

[180] von Neuman J. et O. Morgenstem, *Theory of Games and Economic Behavior,* Princeton University Press, Princeton, NJ (2nd Ed.), 1947 (3nd Ed.), 1953.

[181] Vinh N.V., *Méthodes exactes pour l'optimisation DC polyédrale en variables mixtes 0-1 basées sur DCA et des nouvelles coupes,* Thèse de Doctorat de l'Université de Rouen, 2006.

[182] Weber S., A. Nagy, T. Schüle, C. Schnörr et A. Kuba, *A Benchmark Evaluation of Large-Scale Optimization Approaches to Binary Tomography,* DGCI 2006, LNCS 4245, 146–156, 2006.

[183] Weber S., C. Schnörr, T. Schüle et J. Hornegger, *Binary Tomography by Iterating Linear Programs,* R. Klette, R. Kozera, L. Noakes and J. Weickert (Eds.), Computational Imaging and Vision - Geometric Properties from Incomplete Data, Kluwer Academic Press 2005.

[184] Weber S., T. Schüle, J. Hornegger et C. Schnörr, *Binary Tomography by Iterating Linear Programs from Noisy Projections,* IWCIA, LNCS 3322, 38–51, 2004.

[185] Weber S., T. Schüle, A. Kuba et C. Schnörr, *Binary Tomography with Deblurring,* IWCIA 2006, LNCS 4040, 375–388, 2006.

[186] Weber S., T. Schüle et C. Schnörr, *Prior Learning and Convex-Concave Regularization of Binary Tomography,* Electr. Notes in Discr. Math., Vol.20, 313–327, 2005.

[187] Rendl F. et H. Wolkovicz, *A semidefinite framework to trust region subproblems with application to large scale minimization,* CORR Report 94-32, University of Waterloo, 1994.

[188] Xue H.G., C.X. Xu et Z.X. Feng, *Mean–variance portfolio optimal problem under concave transaction cost,* Applied Mathematics and Computation, 174, 1–12, 2006.

[189] Yuille A.L. et A. Rangarajan, *The Convex Concave Procedure (CCCP),* Advances in Neural Information Processing System 14, Cambrige MA : MIT Press.

Résumé

Les travaux présentés dans cette thèse concernent les nouvelles techniques d'optimisation pour la résolution de certains problèmes importants issus de finance. Il s'agit des problèmes d'optimisation non convexe de grande dimension pour lesquels la recherche des bonnes méthodes de résolution est toujours d'actualité. Notre travail s'appuie principalement sur la programmation DC (Différence de fonctions Convexes) et DCA (DC Algorithmes). Cette démarche est motivée par la robustesse et la performance de la programmation DC et DCA comparée aux autres méthodes.
La thèse est divisée en deux parties et est composée de sept chapitres. Dans la première partie intitulée "Méthodologie" nous présentons des outils théoriques et algorithmiques servant des références aux autres. Le premier chapitre concerne la programmation DC et DCA tandis que le deuxième porte sur les algorithmes par séparation et évaluation. Dans la deuxième partie nous développons la programmation DC et DCA pour la résolution des problèmes en finance. Nous commençons par une introduction à la gestion de portefeuille (le Chapitre 3). Le Chapitre 4 est dédié aux généralisations du modèle moyenne-variance (MV) de Markowitz, où nous étudions le modèle MV sous les contraintes de seuil d'achat, de seuil et de cardinalité. Le Chapitre 5 est consacré à la mesure de risque de baisse et les contraintes de cardinalité. Le Chapitre 6 porte sur le problème de choix de portefeuille avec les fonctions des coûts de transaction en escalier. L'investissement robuste en gestion de portefeuille sous les contraintes de cardinalité est développé dans le dernier chapitre.

Abstract

The topics presented in this thesis are related to new optimization techniques for solving some challenging problems resulting from finance. They are large-scale non convex optimization problems for which finding efficient solving methods is currently the topic of numerous researches. Our work is based mainly on DC (Difference of Convex functions) programming and DCA (DC Algorithm). This approach is motivated by the robustness and efficiency of DC programming and DCA approaches in comparison to the other methods.
The thesis is divided into two parts and consists of seven chapters. In the first part entitled "Methodology" ; we present theoretical tools and algorithms that we are going to use in the thesis. The first chapter is about DC programming and DCA and the second focuses on branch and bound algorithms. In the second part we develop DC programming and DCA for solving some problems in finance. We begin with an introduction to the modern portfolio theory (The Chapter 3). The Chapter 4 is dedicated to the generalizations of the mean variance (MV) model of Markowitz, where we study the MV model under the buy-in threshold constraints, threshold constraints, and cardinality constraints. The Chapter 5 is devoted to the portfolio selection problem under downside risk measure and cardinality constraints. The Chapter 6 deals with the portfolio optimization under step increasing transaction costs functions. Finally, the robust investment strategies with discrete asset choice constraints are developed in the last chapter.